國立交通大學

電機學院通訊與網路科技產業研發碩士班

碩士論文

終端服務的桌面影像壓縮

Desktop Image Compression in Terminal Service

研究生：李宗學

指導教授：張文鐘　教授

中華民國九十七年一月

終端服務的桌面影像壓縮
Desktop Image Compression in Terminal Service

研 究 生：李宗學　　Student：Tsung-Hsueh Lee

指導教授：張文鐘　　Advisor：Wen-Thong Chang

國 立 交 通 大 學
電機學院通訊與網路科技產業研發碩士班
碩 士 論 文

A Thesis
Submitted to College of Electrical and Computer Engineering
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Industrial Technology R & D Master Program on
Communication Engineering

January 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年一月

終端服務的桌面影像壓縮

研究生：李宗學　　指導教授：張文鐘博士

國立交通大學電機學院產業研發碩士班

摘要

隨著網路的發展，單純的多媒體串流已經不能滿足使用者的需求，除了影音的傳送，使用者需要應用程式的分享，利用 Client／Server 機制分享桌面，因此，終端服務的圖形壓縮為本論文的主題。

首先說明 VNC、X Window 以及 RDP 的差異，並分別比較其優缺點， RDP 的圖形傳送方式採取不同的畫面使用不同的指令，在文字部分傳送 1 bpp 的點陣字型，圖形部分則傳送參數由 Client 端繪製，複雜的圖片則傳送 bitmap，由於 RDP 以 T.128 為基礎，因此介紹 T.128（Multipoint Applocation Sharing Protocol）的標準，對於畫面的命令處理方式 T.128 有特別說明指令的定義及相關參數。

在實驗環境以 rdesktop 為主體，藉由程式碼的修改，進行畫面指令的分析，透過實驗畫面的結果瞭解桌面圖形壓縮方式，為了驗證 RDP 的圖形壓縮方式適合使用在 thin client 的架構中，最後透過 rdesktop 進行資料量的分析。

本論文的重點在於遠端圖形傳遞上資料壓縮的分析，除了影片播放外，Window 作業系統中的應用程式大多能順暢的操作，影片播放時由於畫面不斷更新，播放畫面越大資料量越多，忽略的畫面也越多，在 Client／Server 的架構中，除了影片播放外，其他的應用程式畫面傳送都可以利用 RDP 連線進行資料傳輸，最後藉由實驗結果推論出 RDP Server 的畫面切割原理。

Desktop Image Compression in Terminal Service

Student：Tsung-Hsueh Lee　　Advisor：Dr.Wen-Thong Chang

Industrial Technology R & D Master Program of
Electrical and Computer Engineering College
National Chiao Tung University

ABSTRACT

Along with the network development, multimedia sharing can not satisfy users. Except the video and music transmission, users need the application sharing. We can use client/server computing to share the application. This thesis is to discuss the desktop image compression.

First, we will explain the difference between VNC (Virtual Network Computing), X Window and RDP (Remote Desktop Protocol).For image transmission, RDP server according to different display uses different order. For text transmission, RDP server sends 1bpp bitmaps to RDP client. For simple graphics transmission, RDP server sends a parameter order to client. When client receive an order it will draw a graphic on the window. For the picture transmission, RDP server sends bitmaps to client. RDP Protocol is based on T.128 (Multipoint Applicaion Sharing Protocol), so we introduce the T.128 Protocol.

Our research is based on rdesktop program. We modify the source code, in order to analysis the display components. We use rdesktop to connect RDP server. During the transmission, we calculate the packets size from server.

Besides video playing, other applications can get a smooth transmission. We demonstrate that using RDP protocol to deliver application images can get high performance in data compression.Finally, make a conclusion of terminal desktop image compression.

致謝

碩士生涯這兩年，首先感謝的是我的指導教授 張文鐘 博士，感謝老師於學業上的教導，我才能夠順利完成碩士學位。同時也感謝 林大衛教授、黃仲陵教授及何文楨主任於口試時的指導，有了您的指導才使得這篇論文更趨完善。

另外，感謝實驗室學長姐，素仙、程翔在研究上給予的協助，同時也感謝實驗室同學們，文賢、明山、政達、峻權和振偉，感謝你們陪我度過兩年碩士生涯，與你們一同修課研究的歲月裡是值得珍惜及懷念。

最重要的要感謝我的父母，有你們的支持，讓我在經濟上、生活上沒有任何負擔，有你們的陪伴，我才能無後顧之憂的完成學位。感謝我的女朋友，在遇到挫折時給我的鼓勵。

最後，感謝這兩年曾給我協助、給我鼓勵的朋友們，謝謝你們。

誌於 2008.春 新竹。交大

宗學

目錄

圖目錄

表目錄

第一章 緒論

1.1 動機

隨著無線通訊技術的進步，從 3G 頻寬 2.4Mbps 到 WiMAX 頻寬 70Mbps，這樣的無線通訊技術是否可以應用在手機或是 PDA 上面呢?手機是現代人身上都有的通訊工具，利用手機的操控透過 WiMAX 連線到辦公室或是家中的電腦，不管何時何地（Anytime，Anywhere）都能處理事情，然而手機的處理速度有限以及電池的耗電量不宜過大，因此我們利用現今遠端桌面操控的技術實現在手機上，將手機視為 thin client 利用遠端桌面連線到終端伺服器（Terminal Server），一切資料處理、行動計算都在遠端的電腦完成，手機需負責兩件事情，一、接收處理後的圖形結果並顯示在螢幕上，二、透過手機操作將輸入封包傳遞給遠端的 Server。

由於手機上的操作沒有多餘的鍵盤輸入或是滑鼠，因此透過手機觸控螢幕的點選及手寫輸入應該是最適合的輸入方式，另外在顯示部分由於手機螢幕不像一般電腦螢幕那麼大，因此在文字及圖形顯示部分必須清晰，輪廓線條要清楚，最適合的方式就是現在的遠端桌面連線系統，透過彩度壓縮的方式傳送過來，也就是說傳送過來的圖形、文字或許彩度不高，但是其文字清晰不影響使用者操作，適合應用在手機的操作上。

相較於現行視訊串流（例如：VLC 或是 live555）[1]採用頻率（frequency doamin）壓縮方式保留了彩度卻犧牲了高頻變化的圖形部分，遠端桌面系統的畫面傳輸保留了高頻變化部分犧牲了彩度（space doamin），這樣的方式和視訊串流的壓縮方式明顯不同，遠端桌面的畫面傳輸壓縮部份較有利於使用在終端服務的圖形介面上。因此，將對遠端桌面的畫面更新傳輸進行探討,如果直接傳送完整 800x600 的畫面以 pixel 依序傳送，每張畫面的資料量達到 960K Btytes，造成龐大的傳輸量，本論文以探討畫面的傳輸編碼方式為目標，並完成畫面的資料分析以及影片播放的區塊定位。

1.2 研究背景

遠端桌面是一種讓使用者操控遠端系統的方式，一般來說，桌面系統的圖形具有以下特性：

i. 桌面的圖形結構上運用了大量的多邊形及線條。
ii. 桌面的文字需要清晰，使用者要能清楚辨別文字。
iii. 桌用的圖形具有重複性。

RDP 滿足了這方面應用的需求，在影像資料複雜的地方，RDP Server 將影像採取點陣圖的方式傳送到 Client，而背景單純的文字部分，就採用 1 bpp 的點陣字型將文字

傳遞過來，這樣的機制符合色彩壓縮的原則且保留了高頻的變化，另外在重複出現的點陣圖中，使用了 Cache 將資料暫存，因此本論文對微軟的 RDP 畫面傳送進行探討，將透過開放源碼 rdesktop[2]的程式設計，進行遠端桌面的畫面剖析。

rdesktop 為建立在 unix 系統上的 RDP Client，由於其程式的開放性，我們將利用 rdsktop 做為本論文的實驗環境，透過程式碼的修改編譯，提供我們分析遠端桌面的影像傳輸及快取暫存的機制。

1.3 論文架構

瞭解了本論文的動機和研究背景後，第二章將介紹現行的遠端桌面系統，其中較廣為使用的就是 VNC[3]、X-window 和 RDP，本章將進行比較分析;第三章將進行 RDP 原理分析，RDP 的設計建構於 T.128 協定（Multipoint application sharing），本章將說明 RDP Stack 及 RDP 協定分析，RDP 指令可分為 primary order 和 secondary order，這部份也將在第三章詳細說明。第四章將說明 RDP Client 實做以及 rdesktop 程式分析，針對每個指令的流程進行說明，並將畫面解析的結果記錄下來;第五章則是實驗數據結果及畫面指令分析，由 Cient 推測出 Server 的指令判斷機制;第六章為結論及未來發展。

第二章 遠端桌面介紹

現行遠端桌面連線系統有很多，常見的有 VNC、X-window 和微軟的 RDP（Remote Desktop Protocol），我們將針對這三種不同架構的遠端連線系統加以介紹，並於本章的最後一節討論這些架構的綜合比較。

2.1 VNC

Virtual Network Computing（VNC）是現行的遠端桌面連線系統之一，最初發展是在 1995 年由英國劍橋的 Olivetti & Oracle 實驗室發展出來，公開且免費的通訊協定，至今已發展多種版本，為跨平台的遠端桌面連線程式。

2.1.1 VNC 簡介

VNC 透過 Remote Frame Buffer (RFB) Protocol[4]控制遠端的電腦，VNC 為一套跨平台的應用程式，在操作端安裝 VNC Viewer 就是我們所說的 Client，在遠端的電腦安裝 VNC Server，可以多台 Clients 在同一個時間連接同一個 Server，另外，可透過 JAVA Applet 使用網路瀏覽器連接到安裝 VNC Server 的電腦。VNC 的架構可以分為 VNC Server 和 VNC Client（Viewer），彼此溝通的通訊協定為 RFB Protocol，如圖 1。

VNC Server：負責接收 Client 傳送過來的鍵盤或是滑鼠的輸入訊號後，並提供一套桌面分享機制，畫面改變的資料量透過 TCP/IP 傳送到 Client。

VNC Client：根據收到的資料，將 Client 端顯示 Server 的畫面，並根據接收到的資料做更新。

RFB Protocol：一個簡單的原則，將 Server 的點陣圖（bitmap）資料放到螢幕上指定的 X、Y 座標。

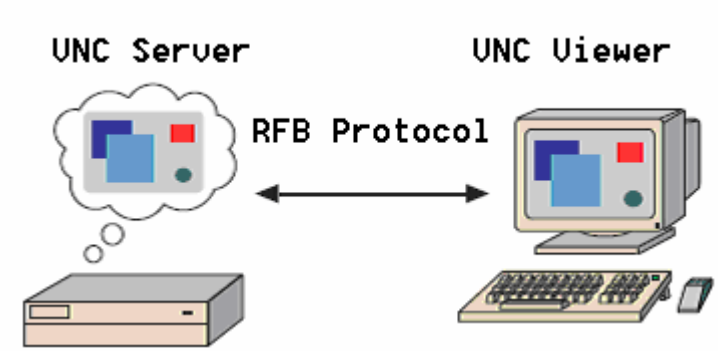


圖 1. VNC 系統架構

2.1.2 VNC 的圖形編碼

在圖形傳輸時，根據不同的網路頻寬，Client/Server 處理資料的速度，VNC Protocol 提供了多種編碼方式，主要有 Raw、Copy-Rect、RRE、Hextile 以及 ZRLE，接下來根據這幾種編碼方式說明。

- RAW：直接傳送圖形的資訊圖形資料不經過壓縮，傳送 pixel 資料由左至右依序傳送，任何版本的 VNC 都支援這樣的編碼方式，因此造成資料量驚人。

- Copy-Rect（Copy Rectangle encoding)：將 framebuffer 中的圖形位置由一塊區域複製到另一塊區域，是一種簡單有效率的方法，Client 端的 buffer 已經暫存了這塊區域的資料，Server 只要將 x、y 的座標告知 Client，只要複製 buffer 中的資料，便可以在畫面中顯示出來，可以節省網路頻寬。這樣的情形發生在移動桌面內的視窗時，由於視窗內的資料並沒有改變，只需要改變顯示的區域，透過這樣的方法可以降低頻寬的使用。

- RRE：這種編碼方式是將同樣的 pixel 壓縮成單一數值重複計算，在 VNC 中實現的方法就是將要傳送的矩型區域中的顏色資訊，找出分布最多的顏色當做背景色，然後根據其他顏色的分布定義出多個子區域（Sub-rectangle)並記錄該區域的顏色、位置以及大小，如圖 2.，這樣的方法在面積大顏色單純的區塊（例如：單色的桌布畫面），但是不適合於背景複雜的區塊。

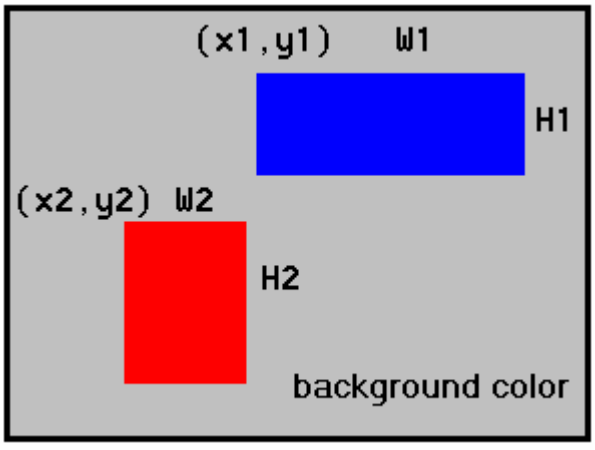


圖 2. VNC 的 RRE 編碼方式

- Hextile：Hextile 算是 RRE 原理變化，將一個矩形區域以 tile 為單位切割，每個 tile 為 16 x 16 pixels，每個 tiles 的順序由左至右、由上至下依序切割，如果一個矩形區域，寬不是 16 的倍數那麼最後一個 tile 的寬度就會窄一點，相同情形，如果高不能被 16 整除，那麼最後一個 tile 的高度就會短一點。每個 tile 可以使用 Raw 編碼或是 RRE 編碼，如果採用 Raw 編碼就直接記錄這個

tile 的 pixel 顏色資訊;如果採用 RRE 編碼，每個 tile 有自己的背景顏色，如果其背景顏色和前一個 tile 的背景顏色一樣，就不需要明確定義；另外尚需定義 tile 中，所有 subrectangle 前景顏色，並記錄下來，如果前景顏色都一致，只需要紀錄一次。每個 tile 中的 subrectangle 都需要清楚的定義前景顏色和 x、y 座標以及長、寬。

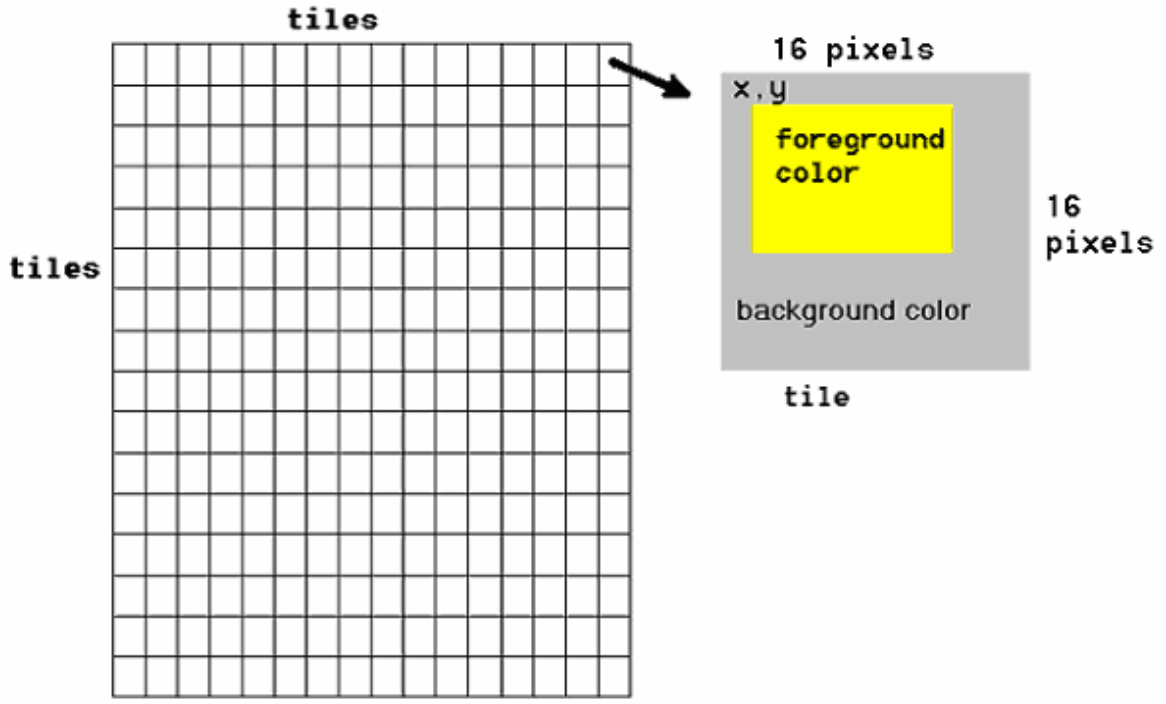


圖 3. VNC 的 Hextile 編碼

- ZRLE：Zib Run-Length Encoding（ZRLE），是結合了 zlib 壓縮、titling、調色盤和 run-length 編碼技術。傳輸的時候，矩形區域以 4 個 bytes 開始，緊接著是 zlib 壓縮資料，唯一的 zlib stream 傳送在 RFB protocol 連線上，因此 ZRLE 區域必須精確的依序編碼和解碼。Zlib 資料在還沒解壓縮之前，代表了 64 x 64 個 pixels，由左到右、從上到下，類似於 hextile 切割方式，如果高和寬不是 64 的整數倍，那麼最後一筆資料區域會小一點。

2.1.3 VNC 優缺點分析

VNC 最大的優點就是其跨平台的性質，透過其簡單的傳輸協定，在不同作業系統、應用程式都能夠使用，此外，其協定的開放及免費性質，任何想應用的實驗環境都可以實現或是自行設計。

VNC 在影像傳送有個缺點，假設我們要播放 640 x 480(pixels)的影片，VNC Server 就必需每秒傳送 30 張 16 位元的高彩圖片，每秒傳輸資料量為 17.58 MBytes，這樣的結果不利於在低頻寬的網路上使用，不適合播放動態影片，造成使用者觀賞影片時，Client 播放影片畫面有遲滯的現象發生。

2.2 X Window

X Window 系統（也常稱為 X11 或 X）是一種以點陣圖方式顯示的視窗系統。最初是 1984 年麻省理工學院的研究，之後變成 UNIX、Linux 等作業系統所一致適用的標準化軟體工具套件及顯示架構的運作協定，經過二十多年的演進，現今已成為工業標準。

2.2.1 X window 的架構

X Window 與一般的作業系統不同，在設計時就是以 Client/Server 為理念，整個 X Window 可以分為幾個部份：X Server、X Client、X Protocol 和 X Library，系統架構如下圖所示。

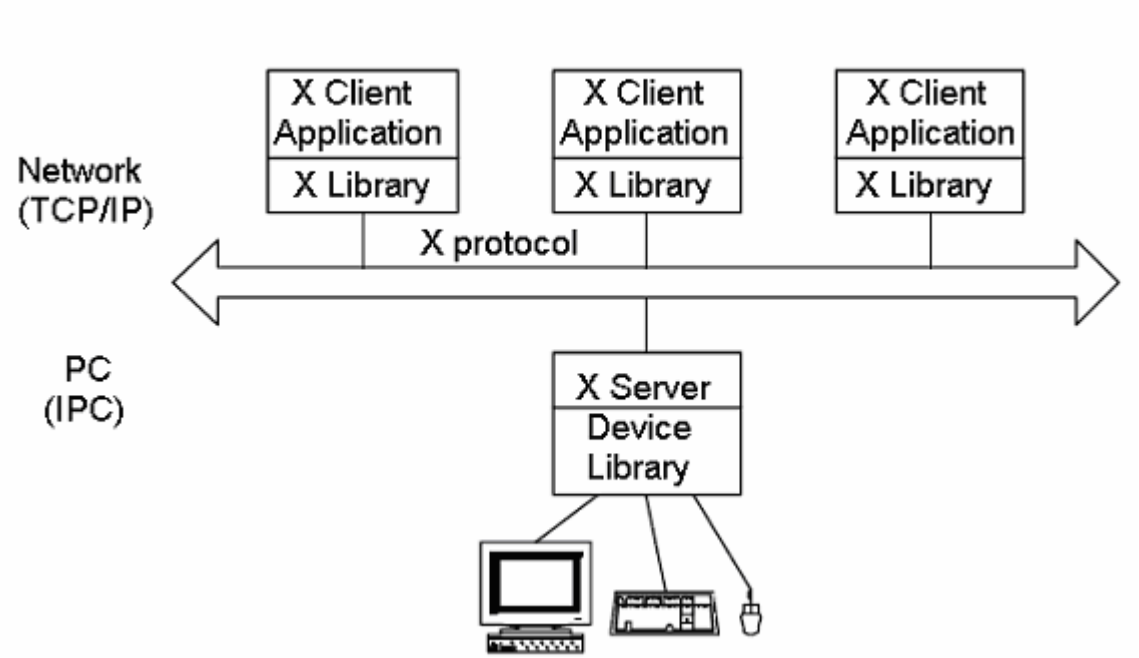


圖 4. X Window 系統架構

- X Server：負責控制顯示卡將影像畫在顯示器上，並且管理鍵盤和滑鼠的事件，產生視窗、對應視窗及刪除視窗，這部份要特別說明，與一般的 Client/Server 名詞定義有點差異，X Server 定義為 Display Server，而應用程式端稱為 X Client。

- X Client：在 X Window 下的應用程式，要求特定的 X Server 作特定的動作。主要的工作為：1、向 X Server 提出需求，2、接收來自 X Server 的事件訊息，3、接收來自 X Server 的錯誤訊息。

- X Protocol[5]：X Client 和 X Server 的通訊協定，定義 Requests、Reply、Error 和 Events，這部份將在 2.2.2 節詳細說明。

- X Library：簡稱 X Lib，大部分 X window 上的應用程式以 X Library 來建立 GUI 元件，例如：按鈕（botton）、目錄（menus）等等。

2.2.2 X Window 的網路透明性

由於 X Window 系統採用網路訊息協定作為應用程式（X Client）和顯示介面（X Server）的溝通管道，而不是一般作業系統常見的函式呼叫，X Client 和 X Server 之間就是夠過可靠的位元組資料流作為溝通。一般來說，如果 X Client 和 X Server 在同一台電腦，就以內部程序通訊（IPC，InterProcess Communication）方式溝通;如果 X Client 和 X Server 在不同的機器上，利用程式介面 X Lib 透過可靠的網路協定（例如：TCP/IP）進行溝通。X Protocol 定義了四種封包做為溝通的機制，如圖 5.。

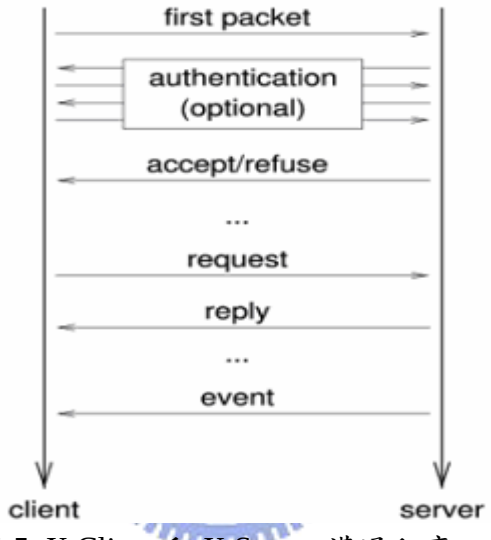


圖 5. X Client 和 X Server 溝通程序

- 請求（Request）：客戶端請求伺服器的資訊，或者請求伺服器執行一個動作。
- 回應（Respone）：伺服器回應請求。但是並非所有的請求都會產生回覆。
- 事件（Event）：伺服器傳送事件給客戶端，例如，鍵盤或滑鼠的輸入，或移動、調整視窗。
- 錯誤（Error）：如果請求無效時，伺服器會傳送一個錯誤封包。

當 X Client 要向 X Server 發出要求（Request）時，可透過網路向執行的 X Server 發出要求，然後交給 X Server 處裡。X Server 向 X Client 通知事件（Event）時，也可以透過網路傳送。Request 和 Reply 的封包沒有長度限制，而 Event 和 Error 則有固定 32 bytes 的長度限制。網路透明化的性質，就是不同機器上的應用程式可以在同一台電腦上顯示結果，這樣的特性乃是 X Window 不同於其他系統的一大特點。

2.2.3 X Client 和 X Server 的繪圖管道

瞭解了 X window 的架構及網路透明性，我們的重點還是影像畫面的繪製與傳遞，關於 Window 的繪圖功能以及 Client/Server 之間傳遞的影像資訊，X Window 提供了一些原則：

- X Server 將螢幕規劃為可重疊的視窗階層（window hierarchy），每個應用程式可以使用多個視窗，依實際需要，重新規劃大小、位置。每個 GUI 中，視窗就是一個頂層視窗(Top Level Window)，除了根視窗(Root Window)以外，每個視窗都有一個 Parent Window，也就是說，Client 建立一個視窗就是在現存的視窗下建立一個子視窗，所以 Client 所建立的視窗都是以樹狀結構去建立的，樹狀結構的根就是 root window。
- X Server 的繪圖是立即性的，X Server 並沒有儲存繪圖序列的動作，而是收到資料後立即繪出畫面。X Window 繪圖是以位元對映（bit-mapped）導向，所有繪圖動作均以視窗定位，因此 X Client 可以在視窗內畫出東西，而不用考慮視窗在哪個地方。
- X Window 顯示圖片（images），由於影像的資料量很大，因此 X Lib 提供了一個 XImage 結構，允許使用者操作影像。XImage 支持多種格式的影像資料並且操作影像中的 pixmaps。

在使用者圖形介面下，常會需要顯示一張圖片在畫面上，Xlib 提供了直接在 Client 和 Server 之間傳送影片的函數，這些函數都會使用 XImage 結構來作為函數的輸入參數，這裡提供一套顯示圖片的方法，這個方法也是本論文架構 rdesktop 使用的方法，以下進行說明：

1、宣告變數

XImage *image;

Pixmap bitmap;

2、呼叫 XCreatePixmap

bitmap=XCreatePixmap(display, drawable, width, height, depth)

建立一個 pixmap 的結構

3、呼叫 XCreateImage

image=XCreateImage(display, visual, depth, format, offset, data, width, height, bitmap_pad, bytes_per_line)

建立影像資料結構，並定義相關參數

4、XInitImage(image)

XImage 在使用前，必須先經過初始化的動作。

5、將建立的影像資料結構放入 bitmap 中

XPutImage(display, bitmap, GC, image, source X, source Y, destination X, destination Y, width, height)

X Server 利用 pixmap 支援影像功能，pixmap 實際是一塊存放 bits per pixel 的圖形儲存區，他的深度就是每一個 pixel 使用的位元數，pixmap 利用來存放影像，當我們在 pixmap 做圖時,就如同在 window 做圖一樣,只是沒有顯示在螢幕上,因此 X window 的可繪圖區（drawable）其實就是泛指視窗（window）和 pixmap。由於 pixmap 是系統資源，所以使用前必須創建它，當一個 pixmap 創立後，其內容是未被定義的，唯有透過 X Lib 的做圖函數在其中作圖，才可以設置它，在利用函數將其複製到視窗中就可以顯示了。瞭解了 X Window 做圖方式後，接下來說明 X Client 和 X Server 之間的交談方式，X Server 存放了資源（Resources）提供 X Client 使用識別碼（identifiers）操作，以下對各個資源做說明。

- 視窗（window）：這項資源是工作站上的矩型面積，使用者可利用視窗來觀看輸出結果,當應用程式產生需要顯示的圖形時,必須指定一個視窗來輸出。每個視窗都必須有一個根視窗（root window），整個螢幕就是一個根視窗。

- 繪圖環境（graphical contexts,GC）：這部份是用來追蹤圖形的屬性，例如：前景顏色、背景顏色、線條寬度、字型等等。每一個圖形輸出的請求必須參照 GC，X Window 提供 X Client 請求產生、改變或是刪除一個 GC。

- 字型（fonts）：這項資源包括了在視窗上顯示文字有關的資料，描述一組文字的大小及樣式，這部份 X Server 通常有字型可供選擇，X Client 說明字元大小、字元空間、字體，利用編碼（ASCII、ISO Latin-1 等等）由 X Server 顯示。

- 顏色對映表（color maps）：這項資源將應用程式所輸出的圖形資料轉換成視窗畫面上可見的顏色。

- 像數對映表（pixmaps）：這樣資源和視窗很相似，主要的差別在於使用者無法在畫面上看到，對於 X Client 和 X Server 之間複製資料相當有用。

當 X Client 向 X Server 請求建立某個資源的時候，同時也指定了一個識別碼給特定的資源，例如：建立一個視窗的時候，X Client 指定了視窗的屬性和識別碼，這個識別碼便和這個視窗關聯。X Client 為了避免衝突，這些資源不可有相同的識別碼，資源建立後，識別碼就成為 X Client 和 X Server 溝通的特定識別。當建立資源的 X Client 關閉和 X Server 的連線後，資源就會正常解除（destroy）。

這些繪圖的功能與處理程序都交由 X Library 來實現，X Library 是建立於 C 語言的函式，Client/Server 通訊協定都仰賴這個函式庫來完成，現行有些 X Window 高階的程式語言(例如：GTK+)，就是建構在 X Lib 函式庫的基礎上。

2.2.4 X Window 優缺點分析

X Window 是一套已經發展成熟且公開的標準，其中 X Client 和 X Server 又是以訊息溝通為主，X Server 屬於顯示部分，而 X Client 屬於運算處理部份，這樣的溝通方式可以降低頻寬的使用。但是，這樣的溝通方式造成冗長的 round trip time。round trip time 就是 X Client 和 X Server 溝通時，因為太多的 request 和 response 造成訊息往返時間長，如圖.6。這樣的情形造成影像傳送時，因為冗長的 round trip time 降低資料傳輸效益，可能 X Client 要求 X Server 做一個畫面的更新，卻需要雙方都傳遞訊息，造成過多的訊息往返時間。

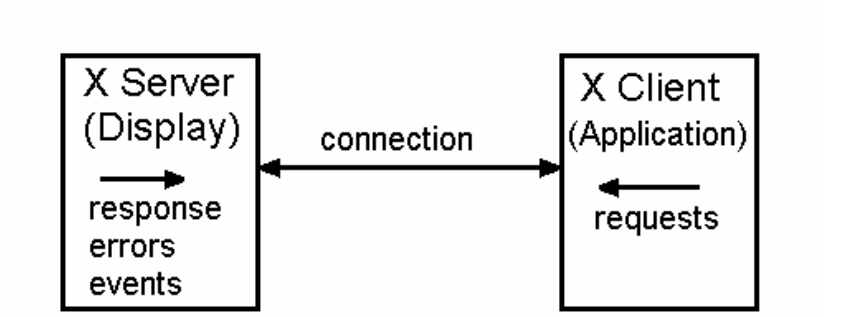


圖 6. X protocol round trip time

2.3 RDP

Remote Desktop Protocol，簡稱 RDP[6][7]。RDP 是微軟根據 ITU T.120 協議系列所制定的一套未公開發表的數據通訊協議。透過網路連接 RDP Server 將應用程式顯示畫面傳送到 RDP Client，RDP Client 將滑鼠、鍵盤等輸入訊息傳送給 RDP Server。

2.3.1 RDP 簡介

RDP 是 Windows 作業系統使用的 Terminal Service 通訊協定，其版本從 version 4.0 開始，演進到現在 version 6.0 版[8]，以下根據各個版本支援的作業系統與特色做比較，如下表。

表 1. RDP 版本比較

Version	Operating System	Feature
4.0	Windows NT 4.0 Server, Terminal Server Edition	支援 8 bpp bitmap cache clipboard mapping 複製區塊
5.0 (rdesktop)	Windows 2000 Professional	支援 16 bpp 列印到用戶端印表機 針對網路頻寬使用的改進 以 56 bit 或 128 bit 加密
5.1	Windows XP Professional	支援 24 bpp 支援聲音轉向播放

Version	Operating System	Feature
5.2	Windows Server 2003 Windows CE 5.0（用戶端) Windows CE 6.0（用戶端)	安全機制 SSL 用戶端支援取用
5.3	Windows Vista Windows Server “Longhorm”	支援 32 bpp 支援單一程式的分享

不管 RDP 版本如何演進，其依舊是商業產品，因此其通訊協定實際運作方式並未公開，本論文的架構建立於開放程式碼 rdesktop 的環境下，未來的章節都將以 rdesktop 作為實驗的基礎。

2.3.2 RDP Session

這一節，我們將介紹關於 RDP Session 建立流程及資料交換方式，在 Session 建立初期，RDP Client 根據 TCP/IP 位址連線到 RDP Server，RDP Client 告訴 RDP Server 自己支援的密碼和 MAC 算法，第二步 RDP Server 告訴 RDP Client 所選擇的密碼和 MAC 算法、Server 隨機碼和 RSA public key，第三步 RDP Client 發送用 RSA 演算法加密後的 Client 隨機碼給 RDP Server，雙方通訊的資料都經過 RC4 加密，以確保通訊安全。RDP Server 使用命令（order)控制 RDP Client 畫面的更新，而 RDP Client 將控制的 input 傳給 RDP Serer，整個 Session 通訊流程如下圖。

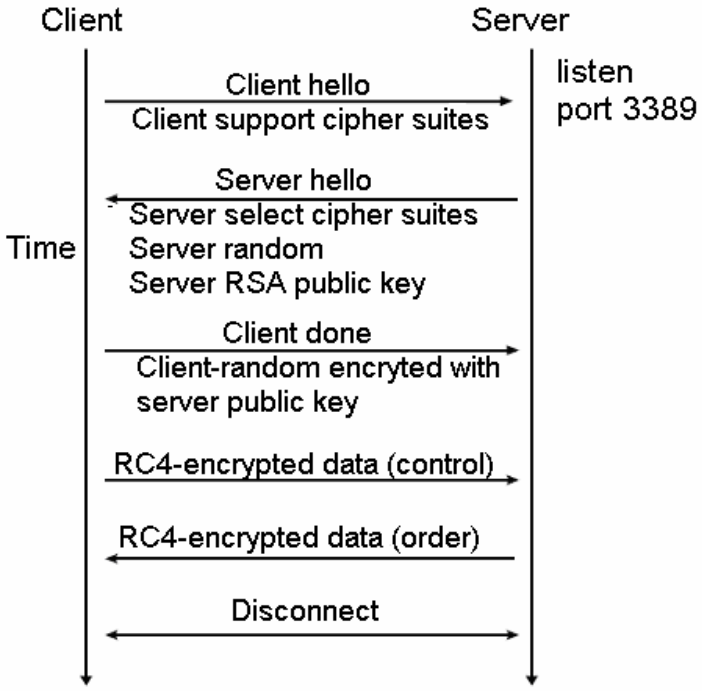


圖 7. RDP Session

關於 RDP 遠端桌面協定詳細的訊息交換方式以及 RDP Server 對於畫面更新的處理將在第三章作詳細的介紹，接下來將針對之前介紹的遠端桌面系統做個綜合性的比較。

2.3.3 RDP 操作原則

RDP 在畫面傳送上，以命令（order）為操作方式，可以分為 primary order 和 secondary order，Primary order 主要的工作在於處理線條、矩形或是出現過的點陣圖和文字，Secondary order 則是傳遞首次出現的 bitmap 和文字。在終端服務的桌面圖形傳輸上，畫面中 bitmap 的傳送，可以說是主要的資料量，RDP 的 bitmap 傳送方式有兩種方式：

一、 RDP Server 先利用 secondary order 的 bitmap cache 將 bitmap 傳送到 RDP Client 暫存起來，隨後馬上傳送一個 memory blt 將 bitmap cache 中的 bitmap 資料，依據 cache id 和 cache index 將指定的 bitmap 取出，將 bitmap 依據指定的位置顯示在螢幕上，通常主要的畫面傳送都是利用這樣的方法。

二、 RDP Server 直接傳送 bitmap 資料，並給予座標位置，RDP Client 收到後，直接將資料顯示在畫面上，傳遞的 bitmap 通常為視窗的外框或是工作列表邊緣線，在畫面傳送上，佔很小的比例，這部份的畫面更新方式稱為 bitmap updates。

2.4 遠端桌面系統的綜合比較

根據先前所介紹的遠端桌面系統，這章節將對系統特性以及更新方式做個比較。

2.4.1 畫面更新方式

Server 和 Client 之間的畫面更新方式可以分為以下幾種：

- RAW：所有的更新圖片，不經過壓縮編碼，每個 pixel 的資料依序傳送，可以想像資料量很大，假設傳送一張 640 x 480 大小的 16 位元高彩圖片，就需要 614.4K Bytes。VNC 支援這樣的編碼方式機制。

- 2D draw primitives：利用區塊填色的方法將圖形編碼，通常是一種非失真編碼的方式，VNC 的畫面編碼主要以這樣的方法。

- Low level graphic：除了圖形編碼的方法之外，還利用一些簡單的指令，例如：字型、多邊形、線條等等的指令呼叫 Client 繪圖，RDP 便是利用這樣的更新方式進行畫面更新。

- High level graphic：除了 2D draw primitives 和 Low level graphic，還支援視窗的建立和管理，X Window 所利用的 X Protocol 就是屬於這類。

2.4.2 Server/Client 訊息傳遞方式

Server 和 Client 的訊息傳遞方式主要可以分為兩種，Server Push 就是由 Server 主動決定何時傳送更新畫面到 Client；Client Pull 就是由 Client 向 Server 要求傳送更新，

使用 Server Push 的系統，其畫面更新較為平順但是相對資料量較大，而 Client Pull 則是只有在使用者輸入訊息時，才會通知 Server 作畫面更新的傳送，但是其資料量較少。

2.4.3 系統比較

瞭解了畫面編碼方式及系統更新方式，根據這些特點我們將針對這幾個系統做分析比較，說明如表 2.。

表 2. 遠端桌面系統比較

系統	畫面編碼方式	更新方式	壓縮方式	cache	license
VNC	Compressed Pixel Data	Client pull	RLE RAW	Client frame buffer	GPL
RDP	Low level graphics	Server push	2D RLE	YES	proprietary
X window	High level graphics	Server push	none	NO	GPL

第三章 RDP 通訊協定介紹

微軟的遠端桌面連線系統主要建立於 ITU-T T.120 系列，為了瞭解整個 RDP 的程序以及封包包裝程序，3.1 節我們將針對 RDP Protocol Stack 說明，了解 RDP Protocol 協議的結構；由於我們的研究重點在於遠端桌面畫面的傳送與剖析，因此 3.2 節我們將針對 ITU-T T.128（Multipoint Application Sharing）文件部分探討，對於畫面的傳送指令進行解析。

3.1 RDP Protocol Stack and Pack Format

3.1.1 RDP Protocol Stack

RDP Protocol 是基於 ITU-T T.120 協議系列的擴充[9]，做為多點通訊協議，可在不同虛擬通道中傳輸數據，並利用加密機制確保通訊安全，本身的協議層次結構如下圖 8.所示。

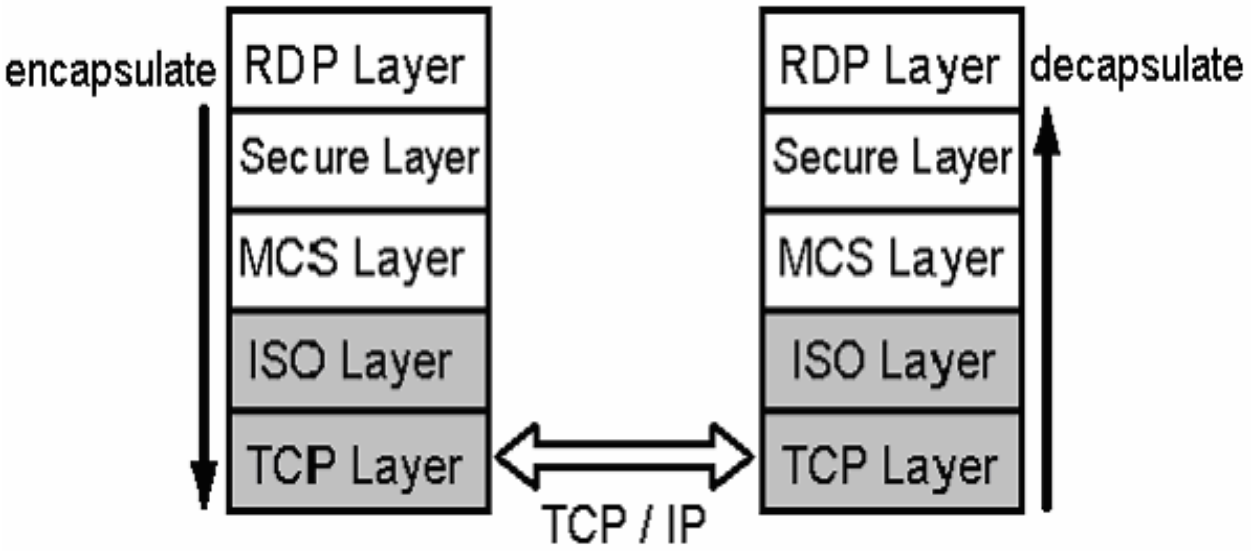


圖 8. RDP Protocol Stack

在一個網路上傳輸的封包建立在 TCP/IP 之上，因此最底層是 TCP/IP 層，TCP 層上面是一個與 RDP 協議無關的標準統一層 ISO 層，ISO 的標準就是 RFC 2126(Open System Interconnection over TCP)，RDP 通訊協定的特性即多點傳輸與多通道傳輸，因此 ISO 層上面是 MCS（Multipoint Communication Service）層，MCS 層上面是 Secure 層，這層完成加密套件（cipher suite）的協商、密碼的生成、MAC 的生成、數據加/解密以及 MAC 值得計算和驗證，最上面是 RDP 層，這層是以 T.128（ASP，Multipoint Application Sharing Protocol）為建立的基礎，因此一個完整的 RDP 封包其結構應該是如下頁圖.9。在 3.1.2 我們將針對 RDP 封包格式深入說明。

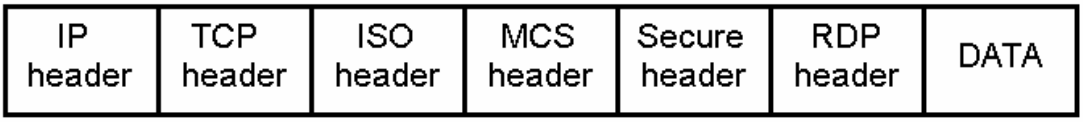


圖 9. RDP 結構

3.1.2 RDP 封包格式

一個完整的 RDP 封包包含了 IP header、TCP header、ISO header、MCS header、SEC header、RDP header 和資料。由於 TCP 層與 RDP 協議無關，因此本節針對其他四層的封包格式進行說明。

- ISO 資料格式：RFC2126 定義了 TPKT 封包 header，第一個 byte 定義了 ISO 版本，接下來一個 byte 為保留，然後封包總長度佔了 2 個 bytes，接下來為 1 byte 的 ISO header 長度，然後是 ISO_PDU_CODE 佔了 1 個 bytes 代表 TPDU 的類型，然後是 padding，接下來是資料。

- MCS 資料格式：T.125 針對 MCS 層的封包 header 定義，MCSPDU header 共 8 個 bytes，第一個 byte 是 MCSPDU type，接下來兩個 bytes 是 mcs_usrid，這個值是在建立連線時，由 RDP Server 分配，並在 MCSPDU 中作為資料回傳給 RDP Client，做為此後通訊時 RDP Client 的身分辨識。第 4、5 個 bytes 是 channel id，這個數值有兩種方式取得，一種是 RDP Client 向 RDP Server 發出請求，RDP Server 處理這個請求並分配一個 channel id，攜帶在 MCSPDU 的資料中傳給 RDP Client；另一種方式是由 RDP Client 自行指定，將感興趣頻道的 channel id 填寫於第 4、5 個 bytes 的欄位後，發出 MCSPDU。第 6 個 byte 是 flag，一般設定為 0x70 換成 01110000。第 7、8 個 bytes 是長度，這個長度不包括自己 MCSPDU header 長度，簡單說就是第 8 個 byte 之後到 MCSPDU 最後一位元的長度，整個 MCS 數據格式如下圖 10.

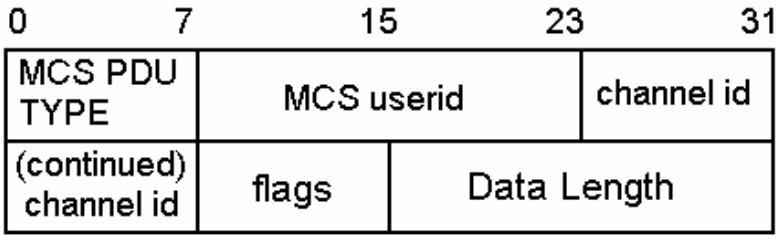


圖 10. MCS Data Format

- Security 資料格式：SEC 層是 RSA 加密算法而不是網路協議，因此沒有資料格式，但是 RDP 對資料加密時，也需要傳遞資料加/解密的訊息，例如：public

key，這個和網路分層包裝標頭（header）性質一樣，因此我們可以用資料格式來敘述。由於 RSA 的加解密流程已經在 2.3.2 RDP Session 小節說明過了，這裡就不多做說明。如果未獲得通訊協定許可那麼將傳送 4 個 bytes 的 sec_flags；如果是進行加密時，則傳送 4 個 bytes 的 sec_flags 和 8 個 bytes 的 public key；如果已經通過認證，那麼就不傳送這個欄位的資料。

- RDP 封包格式：

  RDP PDU header 格式可分為四個領域：

  1. 長度：定義包括 RDP PDU header 在內的整個封包的大小
  2. 固定領域：1 byte 的版本(version)、1 byte 的 RDP PDU TYPE、2 bytes 的 userid。
  3. 可變長度：根據 RDP PDU TYPE 的數值有兩種變化，當 RDP PDU TYPE 定義為資料型態（DATA）時，這個欄位的長度為 12 bytes（包含：4 bytes shareid、2 bytes streamid、2 bytes 的剩餘長度、1 byte 的 TYPE、1 byte 的壓縮標記、2 bytes 的壓縮長度）；當 RDP PDU TYPE 是其他三種型態（Demand Active(要求認證)、Confirm Active（認證許可）、Deactive(解除認證)）時，這個可變長度欄位的長度為 0。
  4. 資料領域：攜帶的資料是複雜的多媒體訊息，由 TYPE 欄位定義。

  RDP 封包格式分為以下兩類，一類是資料傳送封包，格式如圖 11.(a)、一類是驗證（請求、同意、解除）封包，格式如圖 11.(b)。

(a) RDP PDU TYPE 是 RDP PDU DATA 型態

(b) RDP PDU TYPE 是其他三種狀態

圖 11. 兩種不同 RDP PDU 的封包資料型態

在 RDP 資料傳輸時，資料由 RDP、SEC、MCS、ISO 往下到 TCP、IP 層包裝，便形成一個完整的 RDP 資料封包在網路上傳遞，其中值得注意的是在一個 RDP 連線中，連線前畫面更新封包都是以 RDP PDU DATA 的類型傳送，至於傳送的內容依據封包內 TYPE 欄位所決定。

3.2 ITU-T T.128 畫面更新指令探討

RDP 是以 T.128 為基礎，因此在介紹 RDP 之前先針對 T.128 指令更新做說明，這一節先了解 T.128 在 Application Sharing 方面做的規範，我們針對 ASP PDU DATA 中 TYPE 欄位為 ASP UPDATE PDU 的封包做探討，本論文的目標是針對遠端桌面傳送過來的畫面進行解析，因此將探討 UpdatePDU 也就是畫面更新指令的原理部份。在介紹指令內容前，我們先說明 ROP 這個參數，ROP 是 Raster Operation 的簡稱，中文稱為光柵操作參數，主要是進行點陣圖的處理就是點陣圖的 pixel 和背景的 pixel 做位元運算，可分為三種類型，如下表 3.。

表 3. ROP 類型介紹

ROP 類型	描述
ROP2	將筆或刷子與目標點陣圖進行結合。
ROP3	將刷子、源點陣圖和目標點陣圖依照不同參數內容進行結合
ROP4	使用二進位位元“遮罩”(mask)點陣圖結合前景 ROP3 和背景 ROP3。遮罩使用 0 和 1 標記使用 ROP3 的位元

在 RDP 中沒有使用 ROP4，而 ROP3 也都以 ROP2 完成，因此我們以 ROP2 作為說明，當 Windows 畫線條的時候，實際上執行畫筆（pen）和目標畫線區域做 pixel 的位元布林運算稱為「ROP」，也就是說決定最後顯示的顏色，舉例說明，一個黑色的背景畫上黑色的線條，那麼顯示結果就看不到黑色線條了，因此這部份 Windows GDI 定義 ROP 操作方式來決定顯示的線條顏色。由於畫筆和目標為兩種元素因此稱為「ROP2」，Windows 定義了 16 種 ROP2 的代碼，表示 Windows 組合畫筆 pixel 和目標

pixel 的方法，底下我們根據 rdesktop 所定義的類型，進行說明，以下 D 代表目標區域的 pixel 數值，P 代表畫筆的數值。

```
static int rop2_map[] = {
GXclear,          /* 0  不管畫筆和目標數值，顯示黑色線條*/
GXnor,            /* DPon  目標和畫筆先做 or，再做 not */
GXandInverted,    /* DPna  畫筆先做 not，再和目標做 and*/
GXcopyInverted,   /* Pn  畫筆做反向*/
GXandReverse,     /* PDna   目標做 not 後，再跟畫筆做 and*/
GXinvert,         /* Dn    目標做 not*/
GXxor,            /* DPx    目標和畫筆做 xor*/
GXnand,           /* DPan  目標和畫筆先做 and 再做 not*/
GXand,            /* DPa    目標和畫筆做 and*/
GXequiv,          /* DPxn    目標和畫筆做 xor  再做 not*/
GXnoop,           /* D  以目標為顏色，不考慮畫筆*/
GXorInverted,     /* DPno  畫筆做 not 後，和目標做 or */
GXcopy,           /* P  以畫筆為主(一般情況)*/
GXorReverse,      /* PDno  目標先 not，再和畫筆做 or */
GXor,             /* DPo  目標和畫筆做 or*/
GXset             /* 1  不管畫筆和目標的數值，顯示白色線條   */
};
```

RDP 定義 order 的操作時，只有 primary order 使用到 ROP 的參數，因為 primary order 是實際對畫面進行處理的命令，而 secondary order 則是用來進行影像或是文字的暫存，瞭解了 ROP 操作方式，繼續說明 T.128 的文件探討，T.128[10]定義了畫面更新的封包為 UpdatePDU orders，一般來說分成兩種類型：

- 主要命令（Primary orders）：主要是繪圖指令，將顯示的畫面命令發送給其他的 ASCE（Application Sharing Conference Entity），指的就是 RDP Client。
- 次要命令（Secondary orders）：提供資訊給 Primary orders，一般來說是 colormap 或是 bitmap cache 的資料。

在一個 RDP 連線中，如果傳送一張沒出現過的 bitmap，其傳送方式一定是先傳送一個 secondary order (bmpcache order)，接著傳送 primary order(memory order)方式傳送，接下來針對 primary order 和 secondary order 做說明。

3.2.1 Primary orders

Primary orders 主要是處理點陣圖及線條、矩形，接下來將針對幾個主要命令做說明。首先在介紹各個指令之前，我們先說明 Blt（Block Transfer）意思就是將某段記憶體區段的資料複製到另一個記憶體區段，從來源端（source）到目的端（destination），也就是將記憶體資料顯示在螢幕之意，在 Primary orders 中會有部分的命令使用這樣的方式。

3.2.1.1 Destination Blt

當 RDP Client 收到這個命令時，將桌面的指定矩形區域的畫面資料，根據不同的 ROP3 參數對 destination bitmap 做處理，大小受到指定邊界限制，Destination Blt 的參數及敘述如表 4.。

表 4. Destination Blt 參數及說明

參數	敘述
Primary Order Header	主要命令的 header 敘述
destLeft (Coordinate)	Destination Blt 所指定區域最左邊 X 座標
destTop (Coordinate)	Destination Blt 所指定區域最上邊 Y 座標
destWidh(Coordinate)	Destination Blt 所指定區域的寬度
destHeigth(Coordinate)	Destination Blt 所指定區域的高度
ROP3	指定的 ROP3 參數類型，在這個命令中必須參考目標區域的圖形資料，但不用參考來源區域的內容
nonStandardParameters	這個參數只能適用在 AS Protocol 的 base mode

3.2.1.2 Pattern Blt

當 Client 收到 Pattern Blt order 時，依據 ROP3 參數指定的動作，在畫面上進行目標區域（destination rectagle）和刷子（brush，就是指填滿區塊）的 raster operation，其區塊大小受到指定的邊界限制。這個指令在 RDP 應用中，主要是傳輸 1 byte 的圖樣依據命令內容在目標區塊中進行更新，這部份通常是 RDP 連線時，閃爍游標傳送就是依靠這個命令，Pattern Blt 的參數敘述，參考表.5

表 5. Pattern Blt 參數及說明

參數	敘述
Primary Order Header	主要命令的標頭敘述
destLeft (Coordinate)	所指示桌面繪圖區域最左邊 X 座標
destTop (Coordinate)	所指示桌面繪圖區域最上邊 Y 座標
destWidth(Coordinate)	所指示桌面繪圖區域矩形的寬度
destHeigth(Coordinate)	所指示桌面繪圖區域矩形的高度
ROP3	說明 ROP3 參數所指定的動作
backgroundColor	這個參數用來指定背景的顏色
foregroundColor	這個參數用來指定前景的顏色
brush	說明顯示的圖樣類型
nonStandardParameters	這個參數只能適用在 AS Protocol 的 base mode

3.2.1.3 Screen Blt

當 ASCE 收到 Screen Blt 命令時，依據 ROP3 參數內容，在桌面顯示上對來源區塊（source rectangles）和目標區塊（destination rectangle）做 raster operation，其大小依然受到指定的邊界所限制。

- 一個來源區域(source rectangle)的大小、位置，由 sourceX、sourceY 以及 destWidth、destHeigth 定義出來。
- 一個目標矩形(destination rectangle)由 destLeft、destTop、destWidth 及 destHeight 這幾個參數定義。
- destWidth 和 destHeight 兩個參數定義了來源區域和目標區域的寬度及高度，這樣可以避免，畫面複製的時候產生延展(stretching)。
- 來源(Source)矩形區域可能會和目標(destination)矩形區域發生重疊（overlap）。

由以上的敘述，我們可以利用這個指令來完成 RDP 的桌面視窗拖曳，透過這個指令避免拖曳視窗時，產生重繪畫面所產生的資料量，關於 Screen Blt 的參數，參考表.6。

表 6. Screen Blt 參數及說明

參數	敘述
Primary Order Header	主要命令的標頭敘述
destLeft (Coordinate)	在桌面視窗上定義出移動後影像位置的 X 座標
destTop (Coordinate)	在桌面視窗上定義出移動後影像位置的 Y 座標
destWidth (Coordinate)	移動後的影像，使用了幾個像素(pixel)
destHeigth (Coordinate)	移動後的影像，使用了幾個像素(pixel)
ROP3	ROP3 參數的使用定義，這個命令中需要參考來源（source）區域的資料，不用考慮 pattern 的資料。
sourceX (Coordinate)	移動前的影像位置，其左上角的 X 座標
sourceY (Coordinate)	移動前的影像位置，其左上角的 Y 座標
nonStandardParameters	只使用在 AS Protocol 的 base mode

RDP 連線中畫面開啟後，在視窗的移動上，利用 Screen Blt 做為視窗的移動指令，透過這樣的方式，節省資料量的傳送，RDP 在 Screen Blt 的參數定義與 T.128 的 Screen Blt 定義一致。

3.2.1.4 Memory Blt

傳送指令的 ASCE 需要先確認 bitmapCacheID，bitmapCacheIndex 和 colorTableCacheIndex 三個參數，這些資料是之前透過次要命令（secondary order）快取（cache）暫存下來的 bitmap 和 colortable 資訊。當 ASCE（RDP Client）收到從遠端 ASCE 傳來的 Memory Blt 命令時，會依據 ROP3 參數指定類型，對 cache bitmap 和目標的區域（destination rectangle）做 raster operation，這裡的目標區域指的就是螢幕的背景，Memory Blt order 的參數敘述如表 7.。

- 來源的 bitmap（暫存在 bitmap cache）是由 sourceX、sourceY 參數以及 destWidth、destHeight 所定義。
- 目的端區域是由 destLeft、destTop 以及 destWidth、destHeight 參數所定義。
- destWidth 和 destHeight 參數定義了暫存 bitmap 和目的區域寬、高，這樣可以避

免畫面的延伸。

- Cached bitmap bits 需要使用參照的 cached colortable 來比對自己的色彩。

表 7. Memory Blt order 參數說明

參數	敘述
Primary Order Header	主要命令的標頭(Primary Order Header)
colorTableCacheIndex	這個參數指定應該使用那一個 cached colortable
bitmapCacheID	這個參數結合 bitmapCacheIndex 參數，指定在 Memory Blt 中要用到哪個 bitmap
destLeft (Coordinate)	定義目標區域在桌面的 X 座標
destTop (Coordinate)	定義目標區域在桌面的 Y 座標
destWidth (Coordinate)	定義目標矩形的寬度，單位是 pixel
destHeight (Cooedinate)	定義目標矩形的高度，單位是 pixel
ROP3	定義 ROP3 的操作類型
sourceX (coordinate)	參考 cached bitmap 定義 source X 座標
sourceY (coordinate)	參考 cached bitmap 定義 source Y 座標
bitmapCacheIndex	結合 bitmapCacheID 參數，指定這個 Memory Blt 要用哪一個 cached bitmap
nonStandardParameters	這部份只使用在 AS protocol 的 base mode

在 RDP 協議中，傳送 memory blt order 之前必須先傳送 bitmap cache order 給 Client，將 bitmap 資料暫存於 Client cache 中，memoey blt order 功能就是將 Server 呼叫 Client 將 bitmap 取出後，根據座標顯示於畫面上，這部份將在第四章做說明，

3.2.1.5 Text

Text order 設定字體位置的方式，依據 Y 座標的位置定義字元格（cell）的底線（baseline）或是定義字元格的左上角（left，top），建議使用 baseline 定位，因為對於終端電腦要利用左上角定位出一個字體的正確位置、大小、字型是有難度的，當 ASCE 傳送 Text order 並同意使用底線定位法時，雙方就已經透過協議定位出字體的底線位置。這個指令中必須設定命令的 textFlags 參數中 BaseLine Start bit 為 true，並給予 startX 和 startY 參數值，以決定第一個字元格的底線啟始位置。反之，如果這個指令不是使用底線定位法，那麼 startX 和 startY 的座標位置就是說明了字元格的左上角位置。ASCE 收到 Text order 後，依據訊息在桌面上畫出字體，依據下來的規則：

- 字體將會畫在前景，如果背景混合模式（backMixMode）是不透明的，那麼字元格（character cell）背景就會畫在視窗背景上面。
- 當指令的 textFlags 參數的 Baseline Start 位元設定了，那麼 startX 和 startY 就

是定義出第一個字元格的左邊及底線 pixel 位置；反之，如果沒有設定 Baseline Start bit flag，那麼 startX 和 startY 就是定義第一個字元格的左邊上面 pixel 位置。

- 字元定位在任何 extraSpacing 向所有字元所請求，任何空白間隔是向空白的字元所請求。
- 在本地端畫出來的字型是由符合傳送的 ASCE FontID，任何附加的特質是由 textFlags 或是 fontWeight 參數所指示。
- 如果本地的字型是可變化大小的，字體是由所支援的 fontWidth 和 fontHeigth 所決定，否則將字型的高度寬度將由字體固定。
- 畫出的字體是由 fontWeight（字體粗細）、Italic（斜體）、Underline（底線）以及 StrikeOut（刪除，也就是畫中間線）所決定，這些參數都在 textFlags 定義。

關於 Text order 的相關參數及其敘述，如下表 8.。

表 8. Text order 參數說明

參數	敘述
Primary Order Header	主要命令的 header 敘述
backMixMode	這個參數說明文字與背景混合模式
startX (Coordinate)	這個參數定義了文字的起始座標，以左邊的 X 座標為基準
startY (Coordinate)	這個參數說明了文字在遠端桌面的起始 Y 座標。如果 Baseline Start bit flag 有設定，那麼 startY 座標就是 cell 的底部；反之，如果 Baseline Start bit flag 沒有設定，那麼 startY 座標就是 cell 的頂端 pixel
backgroundColor	說明使用哪個背景顏色
foregroundColor	說明所使用的前景顏色
extraSpacing	這個參數指定附加空間給適用的單獨字元，如果這個參數的值是 0，說明了沒有附加的空間
totalBreakSpacing	這個參數說明了所有附加空間可以被使用在空白字元（也就是空格），如果值是 0 就是說明了這個 Text 指令中沒有適用的空格
breakCount	這個參數是說明空白字元的數量，如果值是 0 就是說明了這個 Text 指令中沒有使用空格或是沒有空白字元
fontHeight	字型的大小，說明字型高的部份使用了多少 pixels
fontWidth	字型的大小，說明字型寬的部份使用了多少 pixels
fontWeight	指示字型的粗細，這個值介於 0 到 1000，大致上可分為細(light)≦400＜一般(normal) ≦700＜粗(bold)

參數	敘述
textFlags	說明了字型的附加特性，定義 bit flag 數值如下： ● 斜體（Italic） ● 底線（Underline） ● 刪除（StrikeOut） ● Baseline Start
fontID	傳送 ASCE 的 fontID，經過雙方協商後所定義。
codePointList	字型代號
nonStandardParameters	這部份只使用在 AS protocol 的 base mode，選擇性的列出 non-standard 參數

由於 T.128 是原則上的標準，在實際 RDP 在連線中有些參數定義與 T.128 不完全一樣，在 RDP 中並沒有定義 fontWidth 和 fontHeight 參數，實際上操作過程將在第四章說明，RDP 運用 text order 取出存放在 font cache 中的文字，透過這樣的方式，對於重複出現的文字，可以節省資料的傳輸。

3.2.1.6 Rectangle

RDP Client 收到 Rectangle 命令時，執行兩個動作。使用 ROP2 參數光柵操作(raster operation)，使用了刷子(brush)和背景混合模式(background mix mode)來完成矩形內部的顏色，範圍由邊界所限制。他也使用另外的 ROP2 參數光柵操作，就是筆(pen)和背景混合模式(background mix mode)，這樣完成只有邊框的矩形。

- 實體矩形的座標定義為 destLeft+1、destTOP+1、desyRigtht-1 和 destBottom-1。
- 矩形邊框線條則定義為座標點的序列（destLeft, destTop），(destRight, destTop)，(destRigh, destBottom)，(destLeft, destBottom)，(destLeft, destTop)。

這個指令在 RDP 協定上，用來傳送畫面中矩形的繪圖指令參數說明，如表 9.。

表 9. Rectangle order 參數說明

參數	敘述
Primary Order Header	主要指令的標頭定義
backMixMode	定義這個 Rectangle 指令的背景混合模式
destLeft (Coordinate)	這說明了虛擬桌面上的左邊 X 座標位置
destTop (Coordinate)	這說明了虛擬桌面上的上邊 Y 座標位置
destRight (Coordinate)	這說明了虛擬桌面上的右邊 X 座標位置
destBottom (Coordinate)	這說明了遠端桌面上的底部 Y 座標位置
backGroundColor	說明 Rectangle 指令所用的背景顏色
foreGroundColoe	說明 Rectangle 指令所用的前景顏色
brush(Group)	Rectangle 指令所用的刷子，於矩形內部填色
ROP2	說明這個 Rectangle 指令所使用的 ROP2 參數
pen(Group)	Rectangle 指令所用的筆，用於矩形外框線條
nonStandard Parameters	這個參數只使用在 AS Protocol 的 base mode

RDP 協議和 T.128 參數定義有些許差異，其中 T.128 定義了矩形的左上角和右下角座標位置，Remote Desktop Protocol 則是定義了矩形的左上角座標以及寬、高，而且在 RDP 中並沒有定義 ROP 參數，因為在 RDP 的連線中，Rectangle order 都是不透明的填充矩形，RDP 的實際操作過程將在第四章說明。RDP 畫面傳送中，對於單純背景中，RDP Server 將背景視為 rectangle，透過傳送 rect order 方次進行更新。

3.2.1.7 Line

ASCE（Client）接收到 Line 命令後，使用 ROP2 參數來完成指令，使用筆(pen)和背景混合模式(background mix mode)來劃線，在虛擬桌面上從起點(startX、startY)到終點(startX、startY)，線條的前景顏色由筆(Pen)決定，RDP 在畫面組成上使用 Line order 完成畫面上線條的描述。

表 10. Line order 參數說明

參數	敘述
Primary Order Header	定義 Primary Order Header
backMixMode	定義這個 Line 指令跟背景的混合情形
startX (Coordinate)	定義線條起始點的 X 座標
startY (Coordinate)	定義線條起始點的 Y 座標
endX (Coordinate)	定義線條終點的 X 座標
endY (Coordinate)	定義線條終點的 Y 座標
backgroundColor	定義背景顏色
ROP2	這個 Line 指令中所使用的 ROP2
pen(Group)	這個參數所使用的 pen
nonStandard Parameters	這個參數只使用在 AS Protocol 的 base mode

3.2.1.8 Desktop Save

視窗管理員(window manager)提供一個本地的操作來儲存或復原本地的桌面區域，不論是自行啟動的程式或是在應用程式控制下啟動的畫面。

- 使用者拉下了目錄，使得應用程式擋住一部分的主要視窗(area A)。
- 使用者選擇了一個視窗項目開啟了對話視窗擋住了一部分的應用程式視窗(area B)。
- 使用者選擇了對話按鈕，開啟了功能選項擋住了對話視窗或是擋住了應用程式視窗 (area C) 。
- 使用者關閉了所有的對話視窗。

視窗管理員提供了一個儲存/復原(save/restore)機制，area A 被儲存目錄拉下時，並且在關閉其他對話窗時復原。類似的情況下，area B 在對話視窗開啟後，就會被儲存，area C 在第二個對話視窗建立後，便被儲存。area B 和 area C 在對話視窗關閉後復原，有些區域可能會同時儲存，但是儲存和復原的原則是最後儲存先復原(last in，fast out)。

Desktop save 機制要求開啟 Client 的 desktop cache 以儲存保留的區域。當儲存區域發生變化在一般 ASCE 所連結的應用程式，ASCE 會傳送 Desktop Save 指令給所有連結的 ASCE，並說明那個區域該儲存，隨後 ASCEs 儲存桌面顯示上的指定區域資料到 desktop cache。Desktop cache 是有系統的組合 tiles，大小依據指定的 X、Y 決定。一般來說，ASCE 的效能根據相關儲存及復原的 tile 大小，以及 cache 所佔用的記憶體空間來決定。

Desktop Save 指令說明了儲存(save)或是復原(restore)的動作，並定義處理的區域(一般是 rectangle)和 cache 中 pixel 偏移量(offset)，傳送的 ASCE 計算 pixel offset 是根據 desktop cache 以及 X、Y 所定義的大小。如果這是儲存(save)操作，累積的 pixel offset 加入指定區域的 pixels 值。如果這是復原(restore)操作，累積的 pixel offset 減掉指定區域的 pixels 值，也就是將 desktop cache 暫存空間釋放出來。當傳送的 ASCE 偵測到 Cache 滿了以後，就不傳送 Desktop Save 指令除了 restore 的區域已經存在 desktop cache 中。那麼關於儲存或是復原的區域就需要重新繪製了，關於 Desktop Save 命令的參數，定義在表 11.。

表 11. Desktop Save order 參數說明

參數	敘述
Primary Order Header	主要指令的標頭
saveOffset	接收的 ASCE desktop ache 中的 pixel offset
desktopLeft	Desktop 區域的左邊 X 座標
desktopTop	Desktop 區域的上面 Y 座標
desktopRight	Desktop 區域的右邊 X 座標
desktopBottom	Desktop 區域的底部 Y 座標
action	定義在 Desktop Save 指令中，說明了處理動作是儲存(Save)還是復原(Restore)
nonStandardParmaters	這個參數只能適用在 AS Protocol 的 base mode

3.2.2 Secondary orders

在 T.128 協議中，Secondary orders 提供資訊 Primary orders 處理時所需要的資料，一般來說是 colormap 或是 bitmap 暫存（cache）的資料以下針對各個命令做說明。

3.2.2.1 Cache Bitmap

Server 應該要進一步確認 Cache Bitmap order 的參數應該反映出 Bitmap 的結果和 Bitmap Cache 的協議結果，說明如下：

- 當 Bitmap capability 數值說明了支援 bitmap 壓縮(Compressed)的時候，就應該傳送 Cache Bitmap (Compressed) order。
- bitmap BitsPerPixel 參數等於協議後 sendingBitsPerPixel 的值。
- CacheID 的參數需參考 bitmap cache 區域，一個 CacheID 至少指示一個項目。

- 在 bitmap cache 中 CacheIndex 參數(value)少於協商後項目（entry）的數量。
- Bitmap 的大小(在任何壓縮後)適合於商議後的 bitmap cache area 的最大 cell 尺寸。

Server 必須負責指定 cacheID 和 cacheIndex 參數，並且將 bitmap 資料傳送給 Client 的暫存中並且更新資料，因此需要了解先前傳送的 Cache Bitmap 命令來追蹤 Client 的 bitmap cache 使用程度。當 Client 收到 Server 送來的 Cache Bitmap 命令時，依照 CacheID 和 CacheIndex 將資料暫存，新的 bitmap 資料會取代原本的資料。RDP 協議中利用這樣的指令建立點陣圖快取（Bitmap Cache）可以有效降低相同點陣圖資料重複傳送，有效降低頻寬的使用，關於 Cache Bitmap order 的參數說明如下表 12.。

表 12. Cache Bitmap order 參數說明

參數	敘述
Secondary Order Header	由於 Cache Bitmap 指令(order)是 Secondary Order，因此這個參數定義 Secondary Order Header
cacheID	說明這個 bitmap 應該存放在哪個 cache 中
bitmapWidth	這個參數定義了 bitmap 的寬度，以 pixels 為單位
bitmapHeight	這個參數定義了 bitmap 的高度，以 pixels 為單位
bitmapsBitsPerPixel	這個參數定義了 bitmap 資料的 bits-per-pixel
cacheIndex	這個參數指示了在 CacheID 參數所指定的 cache 哪一個快取項目(cache entry)該使用，認可的 values 取決於在每一個 Bitmap cache 的 cell 商議後的結果
bitmapData	這個參數就是快取的 bitmap data，這個值可能是未經壓縮的
nonStandardParameters	這部份只使用在 AS protocol 的 base mode

3.2.2.2 Cache ColorTable

Server 傳送 ASCE 時，需要更進一步的確定 Cache ColorTable order 參數反映 Bitmap 的輸出以及 ColorTable Cache 商議結果。

- Colortable 項目的數量是根據商議後所傳送的 BPP (SendingBitsPerPixel)數值，也就是說 Colortable entries 數量等於 2 的 n 次方，其中 n 為 SendingBitsPerPixel 數值。
- 在 Colortable cache 中的 CacheIndex 數值小於 negotiated entry 的數量。

當 Bitmap.sendingBitsPerPixel 為 1 的時候，ASCE 將不傳送 Cache ColorTable 指令，舉例來說，AS protocol 定義了 Colortable，說明數值 0 是黑、1 是白。發送命令的 ASCE(Server)負責分配 CacheIndex 參數以及更新遠端 ASCE(Client)的 Colortable

cache，根據事先傳送的 Cache ColorTable 指令(orders)，ASCE（Server）需要追蹤維護所屬的 Colortable cache。

ASCE 收到 Cache ColorTable order 後，便會將所支援的 Colortable 放到它的 entry CacheIndex 的 Colortable Cache area，並替換這個 slot 中原有的資料。一個 Colortable 包含了 16 或是 256 組 RGB 的色彩值，Colortable 中色彩的排列是重要的，而且是依 Colortable 的順序從 0 到 15 或是 0 到 255，數值的多寡由傳送的 BPP(sendingBitsPerPixel)決定。關於 Cache ColorTable 的參數說明如表 13.。

表 13. Cache Colortable order 參數說明

參數	敘述
Secondary Order Header	這個參數定義 Secondary Order Header
cacheIndex	這個參數說明了該使用哪個 colortable cache 中的 cache entry，認可的 values 取決於在每一個 Bitmap cache 的 cell 大小商議後的結果
colorTable	這個參數列出了色彩值，由暫存的 colortable 所組成。在 AS Protocol base mode 下，colotable 有可能包含選擇性的色彩訊息
nonStandardParameters	這部份只使用在 AS protocol 的 base mode，選擇性的列出 non-standard 參數

在實際 RDP 連線中，Colortable 只有在顯示為 256 色時才會使用，對於本論文的操作環境為 16 位元的顯示並沒有使用到 Cache Colortable 的命令。

3.3 點陣圖更新

當收到一個 UpdatePDU（更新的資料封包）包含點陣圖資料（bitmap data），收到的 ASCE（Client）經過解壓縮後，必須從來源 bitmap 複製到虛擬桌面上的目標矩形，大小為目標矩形的寬、高，如果 bitmap 的大小超過目標的寬、高，那麼接收的 ASCE 將會裁切 bitmap 大小到目標矩形上，ASCE 會根據收到的 UpdatePDU 調色盤轉換點陣圖 pixel 的數值，相關的點陣圖更新（Bitmap updates）參數如表 14.

表 14. 點陣圖更新封包參數

參數	敘述
ShareData Header	ShareData 的標頭檔
destLeft	定義點陣圖要顯示在桌面上的左上角 X 座標
destTop	定義點陣圖要顯示在桌面上的左上角 Y 座標
destRight	定義點陣圖要顯示在桌面上的右下角 Y 座標
destBottom	定義點陣圖要顯示在桌面上的右下角 Y 座標

參數	敘述
width	說明點陣圖(bitmap)的寬度(以 pixels 來計算)，點陣圖的寬度至少要大於或等於目標矩形的寬度
height	說明點陣圖(bitmap)的高度(以 pixels 來計算)，點陣圖的高度至少要大於或等於目標矩形的高度
BitsPerPixel	每個 pixel 使用 bits 數量
compressedFlag	這個參數說明了 bitmap 資料有沒有壓縮
bitmapData	這個參數是 bitmap 的資料，可以是壓縮或是沒壓縮
nonStandardParameters	這個參數僅同意在 AS Protocol 的 base mode

點陣圖更新在 RDP 的協定上，通常為視窗外框的繪製或是桌面工作列的邊緣線，在一個視窗系統中，使用 bitmap updates 更新的比例不高。

瞭解了整個 ITU-T T.128 的指令說明以及畫面的點陣圖更新後，由於 T.128 只是原則上的應用標準，在 RDP 實際操作上，我們仍需要依靠 rdesktop 進行實驗，推論一個 RDP 連線的程序以及每個 order 的參數架構。第四章我們將利用 rdesktop 這個程式進行畫面的解析，也是本論文的實驗系統架構，從實驗中，我們可以了解遠端桌連線的畫面更新分為影像、文字和圖形，在影像的傳送上先傳送一個 bmpcache order 將影像儲存在 client 的 bmpcache 中，之後 RDP Server 傳送 memory blt 到 Client，由 Client 依據 Cache id and Cache index 取出 bitmap，在依據指定的座標位置將影像顯示上去，因此 RDP 的影像傳送便是先傳送 secondary order 再傳送一個 primary order，除了重複出現的 bitmap 只要傳送 memory blt 外，都是依據影像暫存的機制進行。文字部分，RDP Server 也是先傳送一個 fontcache 傳送點陣字型到 Client，由 Client 暫存到 fontcache 中，之後 RDP Server 傳送 text order 取出文字，並顯示在指定位置。而圖形的方式都是 RDP Server 傳送參數指令到 Client，由 Client 呼叫繪圖函數處理，整個 RDP 的運作原則就是如此，我們將在第四章詳細說明實做過程。

第四章 RDP Client 實做與程式分析

本章說明 RDP Client 的程式運作流程，由於 RDP 的 Server 和 Client 都是微軟發展出來的商業產品，因此本研究的程式架構以 rdesktop 為實驗環境，接收畫面的命令並透過程式的修改，針對不同指令將畫面的各項特徵顯示出來。

4.1 rdesktop

rdesktop 是一個開放源碼的 RDP Client，其操作平台在 unix、linux 作業系統下，主要是參照 RDP 協定經過不斷嘗試，進行反組譯完成的程式。

Rdesktop 建立了 TCP_connect，建立網路連接以及網路控制碼 sock，從最底層的 tcp layer 建立 sock，利用 tcp_recv（）接收來自 TCP 的封包，去掉 TCP header 將 TCP 封包中的 Payload 取出，之後資料經過 iso_recv（）處理後變成標準傳輸格式，再交由 mcs_recv（）整合成多點通訊（multipoint communication）數據格式，由 sec_recv（）得到密碼 sec flag 解密之後就是標準 RDP 封包格式，依據 rdp_recv（）接收來自底層 RDP 封包由於，我們的重點在於畫面更新及圖形傳遞方式，因此接下來的章節，我們著重在 rdesktop 處理 order 的實作部份，在處理 Server 傳過來的資料時，rdp_recv 一次接收一個 RDP 封包的資料量，封包長度不固定，隨著 order 一筆一筆處理，處理完之後，再透過 tcp_recv 接收下一個 TCP 封包，重覆相同的步驟將 RDP Packet 一筆筆從底層讀取出來，進行處理。RDP 封包透過 TCP/IP 進行資料傳輸，RDP 封包在上層，之後一層層往下封裝，每一層在資料的前端加上一段 header，方便對資料（data）、命令(order)進行分類，實際上對於每一層的 header，rdesktop 定義了不同的結構進行解析。

在畫面傳遞上 RDP Server 將多個 order 包裝在 RDP Packet 傳送出去，將 RDP Packet 存入 TCP 的 Payload，假設 RDP 資料量超過最大尺寸，TCP 便將 RDP 封包分開傳送，之後由 RDP Client TCP Layer 接收重組。Rdesktop 對於網路封包的結構定義如下：

```
Typedef struct stream
{
unsigned char *p;    //定義指標變數，用於指出現在的資料位置
unsigned char *end; //資料結束位置
unsigned char *data;//TCP/IP 資料的起始位置，開啟一段記憶體區塊，將資料放入區
unsigned int size;    //通常為資料大小
unsigned char *iso_hdr; //iso layer 的 header 指標
unsigned char *mcs_hdr;//mcs layer 的 header 指標
unsigned char *sec_hdr; //sec layer 的 header 指標
unsigned char *rdp_hdr; //rdp layer 的 header 指標
}*STREAM;
```

rdesktop 利用這樣方式，解開每一層 header 讀取資料，對於資料處理和未處理的定位都由 p 指標來完成。

RDP 的連線處理精神在於畫面更新時，絕大部分的 bitmap 傳送過來後，都會暫存在 Client 的 bitmap cache 中，這部份的資料都是依靠 bmpcache order 傳送，bmpcache order 屬於 secondary order，接著 RDP Server 傳送 memory order 到 Client，將暫存的 bitmap 取出後，根據相關資訊將 bitmap 顯示在畫面上，memory order 屬於 primary order，RDP 連線時，大部分的畫面都是依靠這樣的方式傳送，對於重複畫面的顯示是可以不用傳送傳送點陣圖的，只要利用 memory order 將重複出現的 bitmap 從 cache 中取出，因此點陣圖的暫存是 RDP 節省傳輸資料的主要原因，接下來 4.2 節說明 RDP order 的處理程序。

4.2 RDP order 處理流程

一個 rdesktop 的主要函數由 rdesktop.c 中的 main 開始，整個函數架構如下，細節部份將再依序說明。

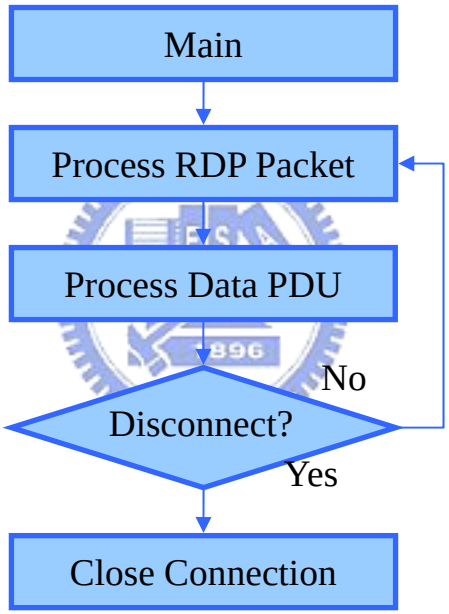


圖 12. rdesktop 處理流程

main 函數一開始便先根據使用者連線前的設定值與 RDP Server 進行協議，網路連線建立後，便呼叫 rdp_main_loop 進行資料的處理，如果連線中斷了，就交由 rdp_disconnect()結束連線，rdp_main_loop 只做一件事情就是呼叫 rdp_loop，當 rdp 連線持續建立時迴圈繼續呼叫 rdp_loop 函數，如果 rdp 連線中斷了就跳出迴圈中斷程式，rdp_loop 呼叫 rdp_recv 函數進行 rdp_packet 資料的讀取，由 rdp_recv 中判斷收到的封包類型，針對不同封包類型呼叫函數進行處理，rdesktop 程式處理流程在於 RDP Layer 對 RDP 封包處理，從接收封包到判斷封包類型是屬於哪種 type 的封包，這裡的 type 代表 rdp 封包類型，程序依下頁圖 13.的函式呼叫來處理。

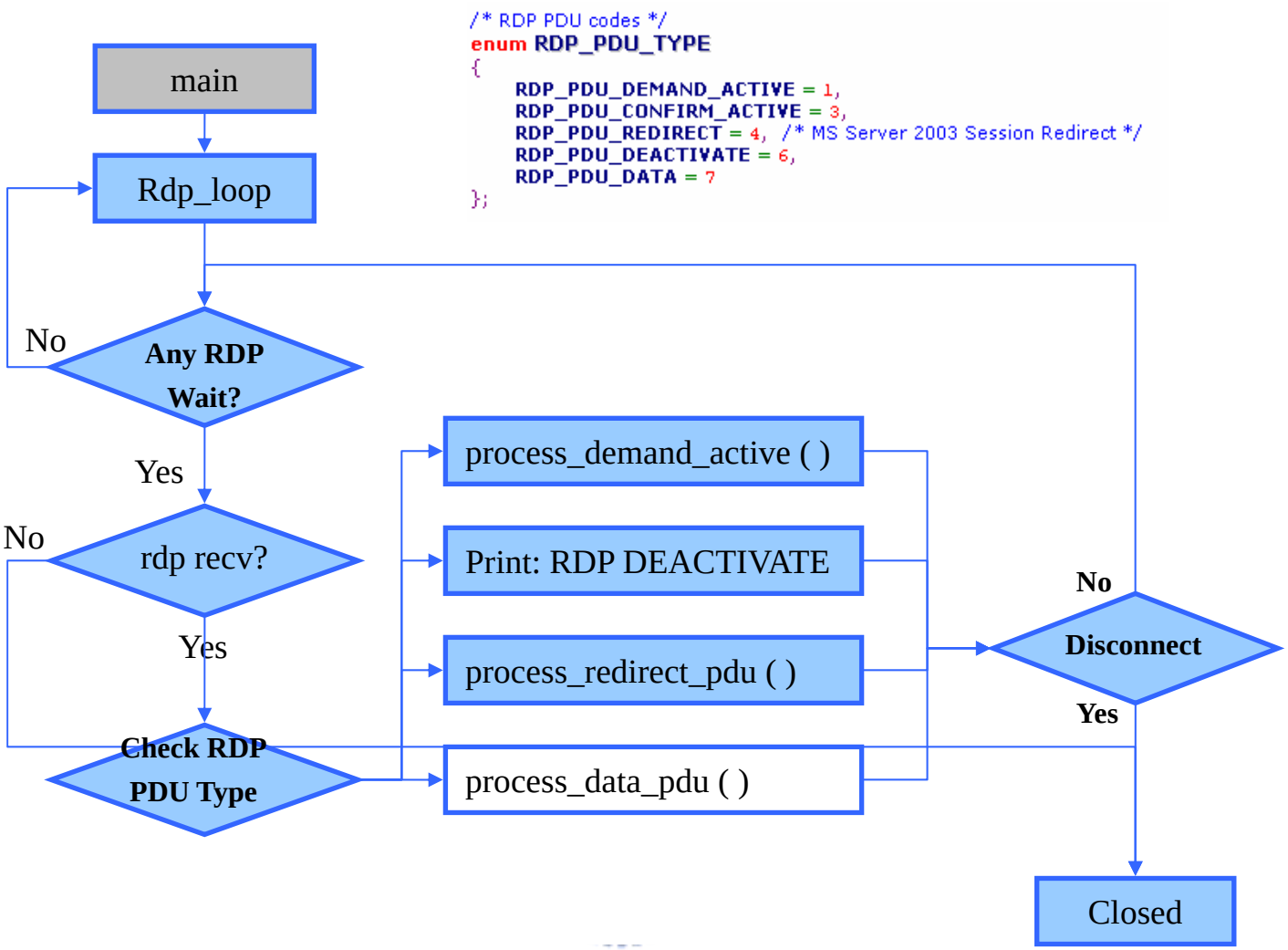


圖 13. RDP 封包類型判斷

圖 13.中經過 rdp_recv 函數後，rdesktop 利用 rdp_recv 處理 rdp 封包後，依據封包類型（type）呼叫函數，這裡指的封包類型就是第三章所說的 ASP PDU TYPE，封包的格式如下圖。

圖 14. RDP header 中 RDP PDU Type 欄位

RDP PDU TYPE 可以分成下面幾類：

- **RDP_PDU_DEMAND_ACTIVE**：process_demand_active 處理 RDP Server 驗證要求
- **RDP_PDU_DEACTIVE**：為解除驗證，rdesktop 僅做訊息的接收，沒有呼叫函數來處理
- **RDP_PDU_REDIRECT**：呼叫 process_redirect_pdu 為處理資料 redirect 封包（例如：聲音的轉向）
- **RDP_PDU_DATA**：process_data_pdu 為處理資料封包
- **0**：回到 rdp_recv 函數繼續處理 RDP Packet，正常連線時，type 回傳值都是 0
- **default**：例外情況 report an unimplemented protocol feature

RDP Server 和 RDP Client 完成連線後，假設 type 回傳值為 0 則 rdp 封包處理交由 rdp_recv 函數，在後面的篇幅我們將對 rdp_recv 函數流程進行說明。

Process_data_pdu 收到資料後，接下來判斷 data pdu type，依據其資料封包類型（data pdu type）呼叫不同的函數處理，程式處理程序如下圖。

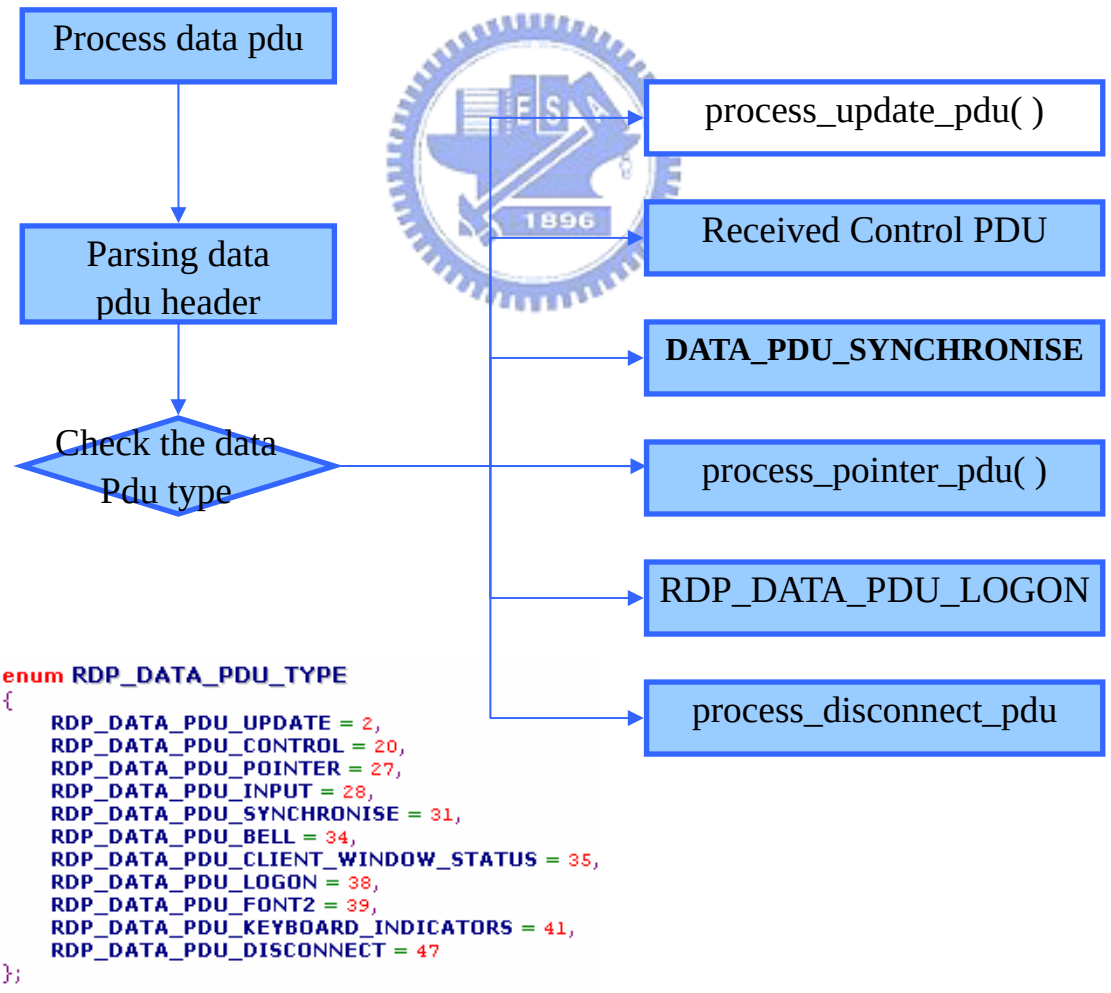


圖 15. RDP Process Data PDU

收到一個 RDP data Pdu 之後，判斷 data pdu type，分為以下八種類型：

- **RDP_DATA_PDU_UPDATE**：呼叫 process_update_pdu，這部份為資料更新，例如：update orders、update palette 等，這支函數是畫面更新的重點。
- **RDP_DATA_PDU_CONTROL**：接收到控制單元，rdesktop 不呼叫函數進行處理，僅標示 “received control PDU”
- **RDP_DATA_PDU_SYNCHRONISE**：接收到同步訊息
- **RDP_DATA_PDU_POINTER**：呼叫 process_pointer_pdu，處理 pointer color 或是 pointer cached 等。
- **RDP_DATA_PDU_BELL**：警告鈴聲的處理
- **RDP_DATA_PDU_LOGON**：登入遠端連線後，都會收到這個登入成功的訊息。
- **RDP_DATA_PDU_DISCONNECT**：呼叫 process_disconnect_pdu 進行處理，使用 rdesktop 和 Windows XP RDP5.1 進行 RDP 連線，連線中斷時 rdeskop 是不會收到這個訊息的；但是，與 Windows Vista RDP5.3 連線，在連線中斷會收到 dissconnect pdu。
- **Default**：未預期的 data PDU 類型，也就是上面類型的例外發生時通知使用者。

RDP Data PDU Type 在 RDP PDU header 中的欄位如下圖，TYPE 欄位灰色部分。

圖 16. RDP Data PDU Type 欄位說明

一個畫面更新的封包為 update pdu，因此接下來呼叫 process update pdu 函數進行處理，處理更新的封包（process update pdu）可以分為四種類型，指令（order）更新部分、點陣圖（BITMAP）更新部份、調色盤（PALETTE）更新以及同步更新，利用 process update pdu 函數中判斷 update_type 的類型，rdesktop 定義的形式如下：

- **RDP_UPDATE_ORDERS**：進行命令的更新，這裡指的是更新 order 格式及定義。
- **RDP_UPDATE_BITMAP**：更新 bitmap，呼叫 process_bitmap_update 處理。
- **RDP_UPDATE_PALETTE**：呼叫 process_palette，進行 palette 的更新。

- **RDP_UPDATE_SYNCHRONIZE**：不呼叫任何函數，單純做同步的動作。
- **Default**：其他未預期的 update_type。

Update PDU Type 包裝於 Update PDU 資料包的前兩個 bytes，包裝情形如下圖。RDP Data 部分通常是畫面更新的資料，可以是 RDP order 或是 RDP bitmap updates 的資料，下面以 RDP order 為例子。

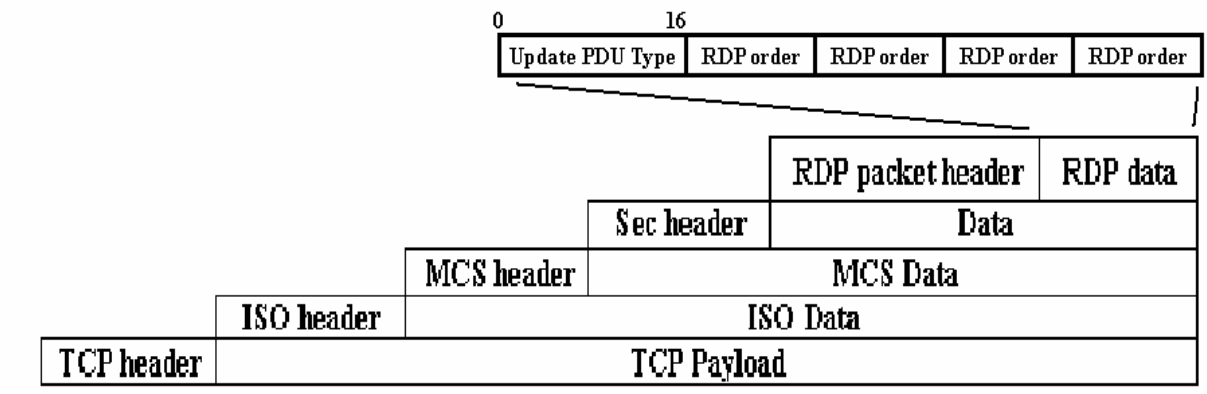


圖 17. RDP Data 內的 Update PDU type 欄位

Process_update_pdu 函數根據不同的 case 呼叫相對應的函數。

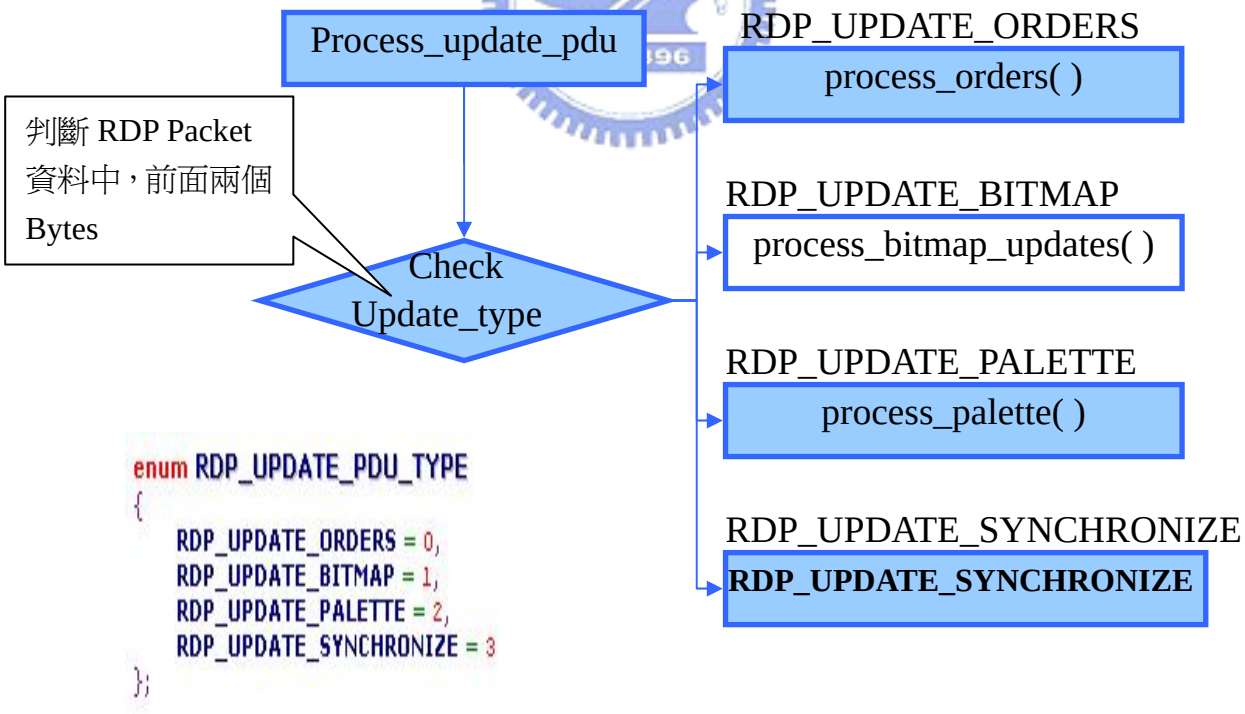


圖 18. 處理 update pdu 函數呼叫

process_bitmap_updates 函數中主要是處理來自 RDP Server 傳送的 bitmap updates，這部份的 bitmap 是直接傳送座標訊息、圖片長寬及 bitmap 的位元資料，process_bitmap_updates 接收後，透過 ui_paint_bitmap 呼叫 X Lib 的 XCreateImage 將點陣圖繪製出來，在利用 XPutImage 函數將圖形顯示在指定的畫面上。

瞭解 bitmap updates 方式之後，我們將針對另外一種畫面傳遞方式 process order 進行說明，在 RDP 封包處理上，在連線建立，使用者帳號密碼驗證成功後，主要的封包都是由 rdp_recv 處理，rdp5_process 是配合 RDP 5.0 Server 處理函數，對於 RDP date pdu 其資料來自 security layer 的資料，將封包交由 rdp5_process 函數進行 RDP 資料的讀取，在 rdp5_process 中依據不同的資料類型（type）總共分成 10 種類型，介紹如下：

- **update orders**：呼叫 process_orders 處理畫面命令
- **update bitmap**：利用 process_bitmap_updates 進行 bitmap 的更新
- **update palette**：呼叫 process_palette 函數，處理顏色對應表的處理
- **update synchronize**：不做任何處理，跳出 rdp5_process 函數，作為同步之用
- **null pointer**：設定滑鼠游標類型為 null
- **default pointer**：預測的游標類型
- **pointer position**：定義游標的位置，呼叫 ui_move_pointer 移動游標
- **color pointer**：process_colour_pointer_pdu 定義游標的顏色
- **cached pointer**：process_cached_pointer_pdu 處理暫存的游標類型
- **default**：未處理過的例外情形，結束函數。

我們的重點在於畫面的指令更新，因此將重點放在 process_orders，處理畫面中各種指令。當收到 rdp 資料後處理指令，主要是根據 order_flags 作判斷，流程說明如下：

1. 判斷指令封包是否發生錯誤（RDP_order_standard）
2. 判斷是否為 secondary order
3. 判斷 primary order 中 order_type，針對不同類型的命令呼叫不同的函數

讀取 order 的流程如下圖，其中 order function 由灰色的 block 依據 order type 定義，呼叫不同的函數來處理，以 case 方式來處理 primary order 和 secondary order 的程序。

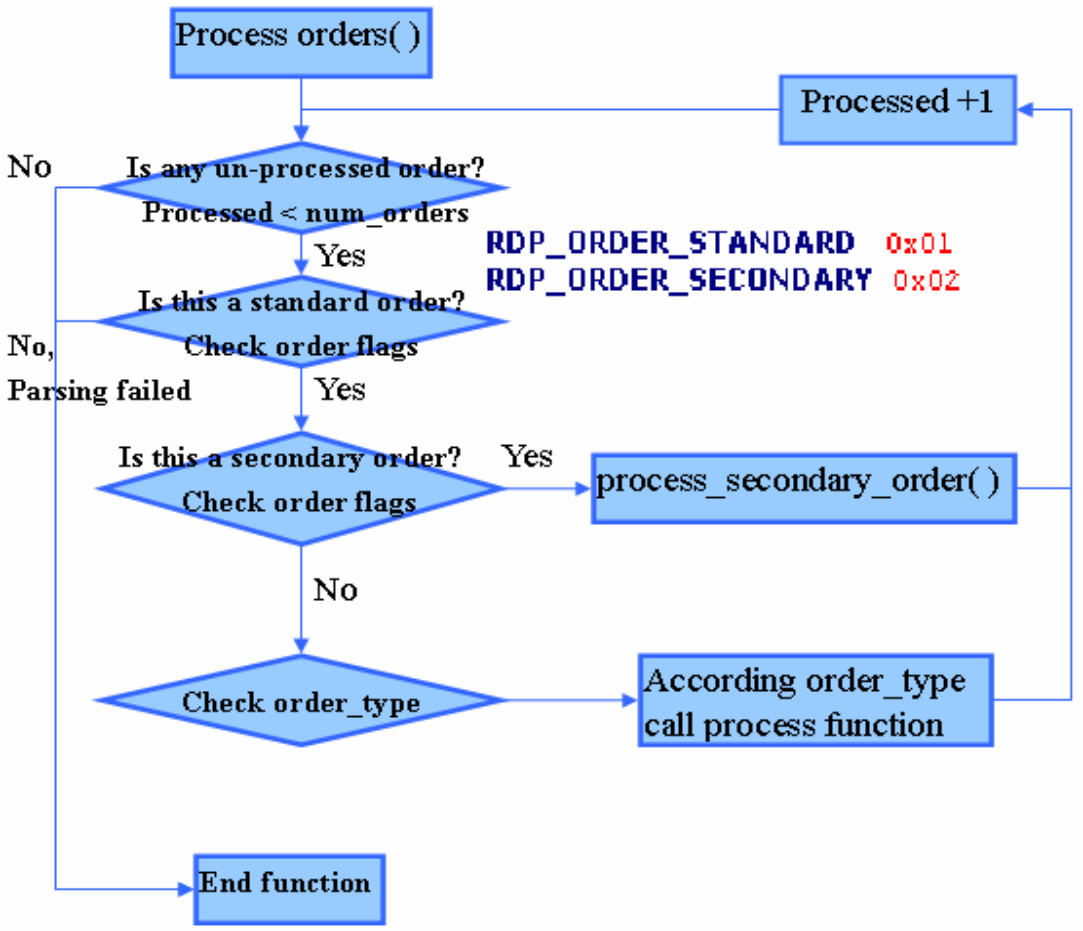


圖 19. RDP order 處理流程

由上述的程式流程，我們將 secondary order 和 primary order 分別呼叫不同的函數來處理，接下來針對函式的處理程序進行說明。這裡我們先針對 primary order 做說明，由於 process_orders 函數已經將 primary order 中的指令，根據 order type 分類出來，依據 Server 所定義不同的 order_type 類型，rdesktop 進行不同的處理方式，primary order 主要處理出現過的畫面（例如：文字、影像）以及繪製圖形（多邊形、矩形、線條），secondary order 則是處理首次出現的文字及圖形。下面的章節，分別針對 primary order 處理程序以及 secondary order 的流程進行探討，並將 rdesktop 收到的資料依據不同的命令將實驗結果顯示出來。

4.3 Primary order 的處理

在 rdesktop 的 orders.c 檔案中，process_orders 函數依據 order type 將不同類型的 primary order 分類出來，如下圖，接下來分別呼叫對應的函數，為了研究 RDP Server 對於畫面的傳送方式，經過程式碼的修改，我們可以看到畫面中指令的組成。4.3 節，將針對 rdesktop 收到命令後的實作過程進行說明，在正常的情況下，rdesktop 呈現出的遠端桌面如同一般的桌面，但是這樣的畫面我們無法了解 RDP Server 是傳送哪些指令來組成一個畫面，因此，我們將進行程式碼的修改，並重新編譯，透過修改後的程式了解 RDP Server 如何傳送畫面，並將實驗結果呈現出來。

```
enum RDP_ORDER_TYPE
{
    RDP_ORDER_DESTBLT = 0,
    RDP_ORDER_PATBLT = 1,
    RDP_ORDER_SCREENBLT = 2,
    RDP_ORDER_LINE = 9,
    RDP_ORDER_RECT = 10,
    RDP_ORDER_DESKSAVE = 11,
    RDP_ORDER_MEMBLT = 13,
    RDP_ORDER_TRIBLT = 14,
    RDP_ORDER_POLYGON = 20,
    RDP_ORDER_POLYGON2 = 21,
    RDP_ORDER_POLYLINE = 22,
    RDP_ORDER_ELLIPSE = 25,
    RDP_ORDER_ELLIPSE2 = 26,
    RDP_ORDER_TEXT2 = 27
};
```

圖 20. RDP Primary order 定義的 TYPE 類型

4.3.1 text order

RDP Server 透過 text order 傳送點陣字型，rdesktop 定義了 process_text2 函數進行處理，並且定義了 TEXT2_ORDER 的資料結構，如圖 21.，結構中包含了字型、mixmode、前景、背景、文字區塊的大小、區塊的座標等等參數。

```
typedef struct _TEXT2_ORDER
{
    uint8 font;         /*協議的font id*/
    uint8 flags;        /*文字型態分為兩種，IMPLICIT和 VERTICAL*/
    uint8 opcode;       /*ROP，實際上 沒用到*/
    uint8 mixmode;      /*文字和 背景混模式，分為透明和 不透明*/
    uint32 bgcolour;    /*背景顏色*/
    uint32 fgcolour;    /*前景顏色*/
    sint16 clipleft;    /*文字區塊座標*/
    sint16 cliptop;     /*文字區塊座標*/
    sint16 clipright;   /*文字區塊座標*/
    sint16 clipbottom;/*文字區塊座標*/
    sint16 boxleft;     /*通常為0*/
    sint16 boxtop;      /*通常為0*/
    sint16 boxright;    /*通常為0*/
    sint16 boxbottom;/*通常為0*/
    BRUSH brush;        /*Brush結構*/
    sint16 x;           /*文字的座標位置 X*/
    sint16 y;           /*文字的座標位置 Y*/
    uint8 length;       /*一 個text order的長度*/
    uint8 text[MAX_TEXT];/*文字的代號*/

} ? end _TEXT2_ORDER ?
TEXT2_ORDER;
```

圖 21. TEXT2 ORDER 的結構

text order 的處理過程，首先從封包中讀取出各項參數（例如：座標、前景顏色、混合模式）將各項參數定義到 TEXT2_ORDER 的結構中，之後將結構中的參數傳送給 ui_draw_text，並從 font cache 中取出暫存的文字資料，呼叫 DO_GLYPH 將文字顯示在螢幕上，由於 rdesktop 是應用在 X window，因此，最後是依靠 X Lib 的函數將文字顯示在畫面上，其流程如下圖 22.。

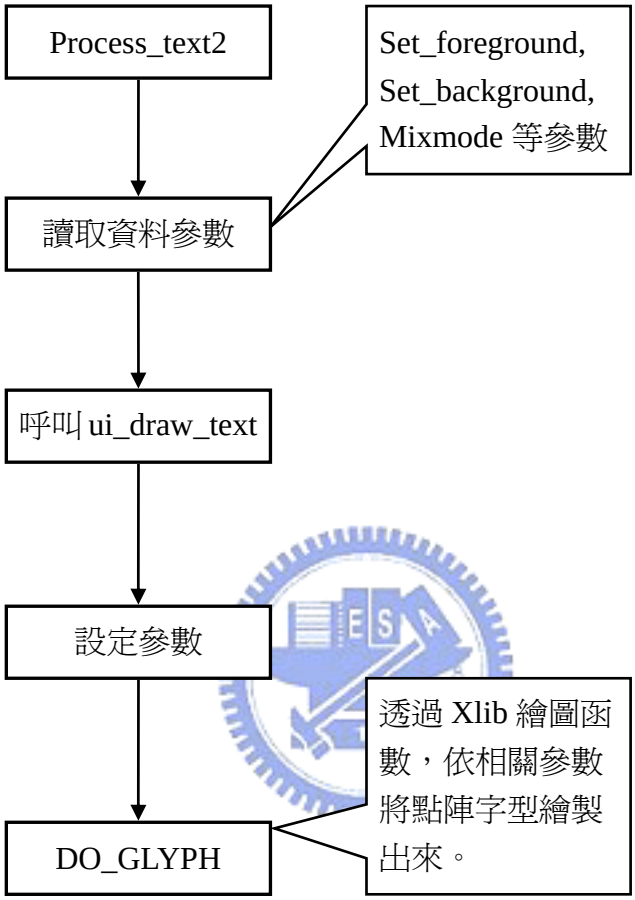


圖 22. text order 函數呼叫流程

RDP Server 所傳送的字型其實就是 1 bpp 的單色點矩陣圖形，由於 X window 中並沒有微軟 Windows 的 Win 32 平台無法使用 ROP 運算來達到目的，因此 rdesktop 定義了 DO_GLYPH 函數。

DO_GLYPH 函數可以說是 rdesktop 和 X window 的訊息傳遞函數，首先從 font cahce 中讀取出字型資料，透過呼叫 X Lib 中 XSetStipple 及 XSetTSOrigin 函數，XSetStipple 依造 Cache 中暫存的位元點陣字型，從指定的 font Cache 中取出點陣字型，接著透過 XSetTSOrigin 將文字依照相關位置繪製出來。

由 rdesktop 程式碼了解 text order 的流程後，在呼叫 ui_draw_text 之前修改文字的前景顏色（fgcolour），並重新編譯修改後的 rdesktop 程式碼，重新連線後，只要畫面中 RDP Server 透過 text order 所傳送過來的文字就以紅色顯示，如果以影像（pixmap）傳送過來，那麼就是正常的顏色。我們將遠端登入畫面、桌面、Command line、Notepad、

網頁以及 Word 實驗結果顯示出來，可以看到 RDP Server 針對不同文字畫面，有不同處理方式。

在圖 23.中我們可以看到，利用修改後的 rdesktop 在登入遠端畫面時，使用者名稱、密碼是以 text order 方式傳送給 RDP Client 也就是以單色點陣字型傳送過來，其他的文字部份則是以點陣圖方式傳送。

圖 23. 利用 rdesktop 登入遠端畫面

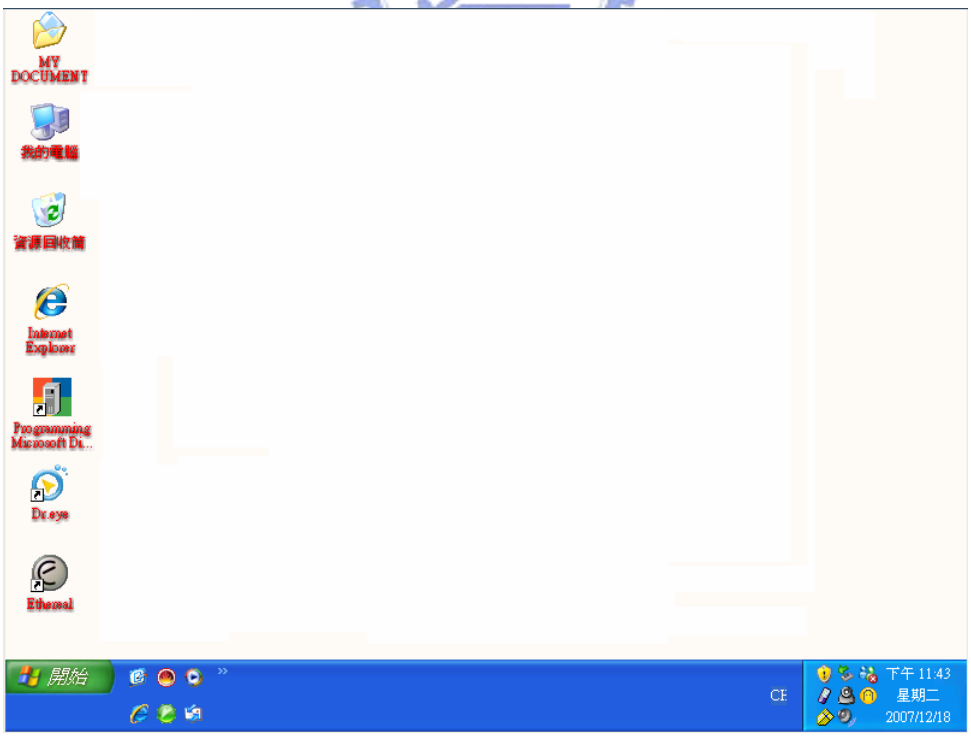


圖 24. 利用 rdesktop 連線到遠端桌面

在 Command Line（圖 25.）和 Notepad（圖 26.）這樣背景單純的畫面下，文字也都是以 text order 方式傳送，因為背景簡單，文字部分只需要傳送 1 bpp 的點陣字型，在定義其座標位置，RDP Client 根據 RDP Server 所傳送的顏色及座標位置，將文字顯示在畫面上。

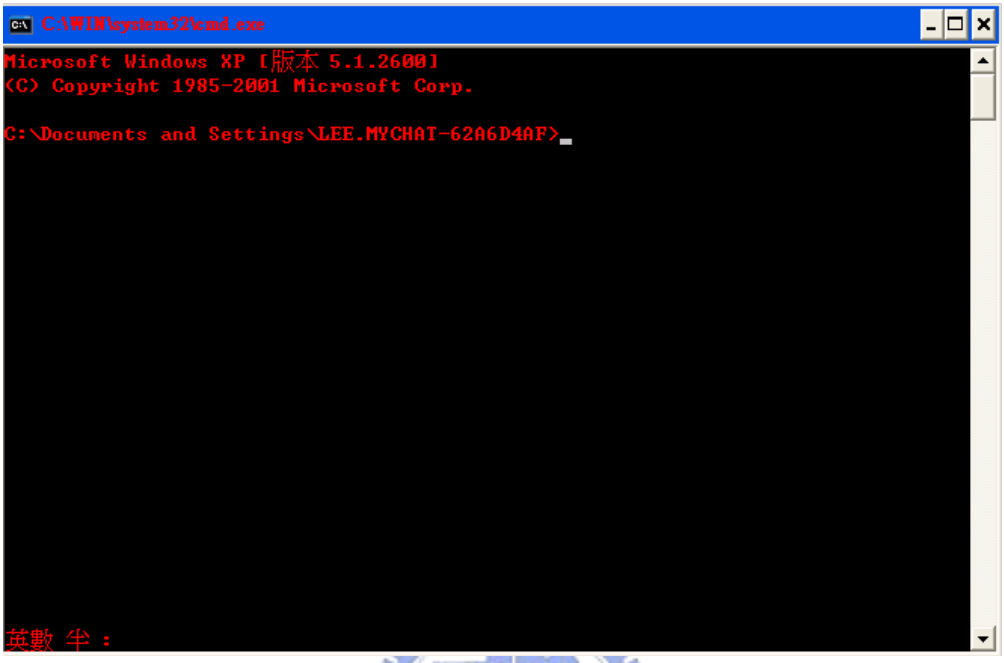


圖 25. 利用 rdesktop 連線到遠端 Command Line

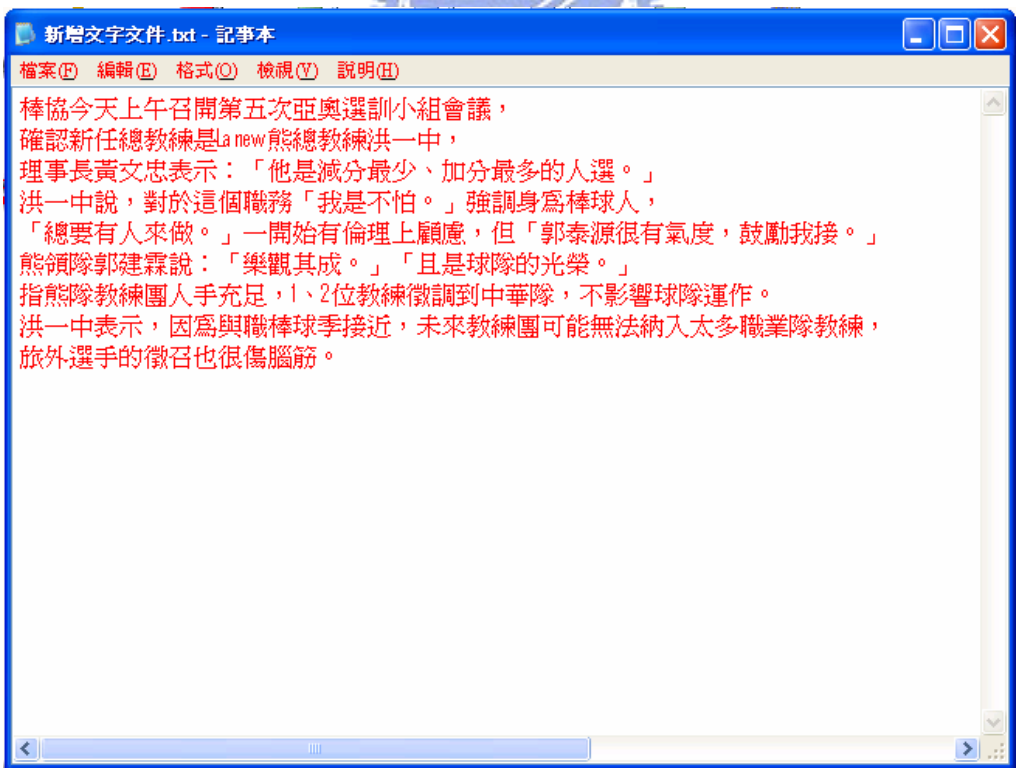


圖 26. 利用 rdesktop 開啟遠端 Notepad

在視窗系統中文字隨處可見，由上述實驗結果，我們可以看出在遠端桌面連線中，桌面的圖示部分是以 text order 方式傳送。值得注意的是一個視窗畫面系統中，並非所

有我們看到的文字都是使用 text order 方式傳送，在網頁的顯示上，Internet Explorer 會判斷畫面的文字部分，以 text order 用 1 bpp 點陣字型傳送，如圖 27.(a)；視窗變動後，網頁的內容以 bitmap 指令做更新，如圖 27.(b)。

圖 27.(a) rdesktop 遠端連線 Internet Explorer 開啟網頁（變動前）

圖 27.(b) rdesktop 遠端連線 Internet Explorer 開啟網頁（變動後）

除了網頁中的文字可能會透過 bitmap 的方式傳送外，我們特別針對 Microsoft Word 文書處理程式進行畫面的解析，發現了當 Microsoft Word 在檔案開啟時畫面以點

陣圖來傳送，但是當我們拖曳視窗後，視窗內文字的部份就是以 text order 的方式傳送，如圖 28.(a)。

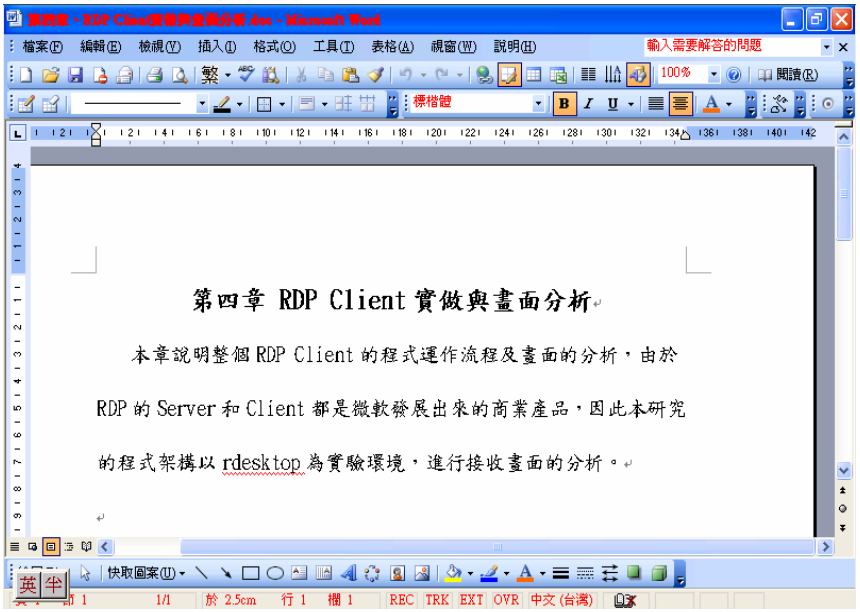


圖 28.(a) 利用 rdesktop 開啟遠端 Word（剛開啟）

Word 經過拖曳後，RDP Server 判斷其為背景單純的文字後，傳送 text order 做為畫面的更新如下頁圖 28.(b)，圖中檔案文字部分變成紅色，表示 RDP Server 重新傳送 text order 至 rdesktop，如此可以降低頻寬的使用，RDP Server 的判斷機制依不同的應用程式而有所不同。

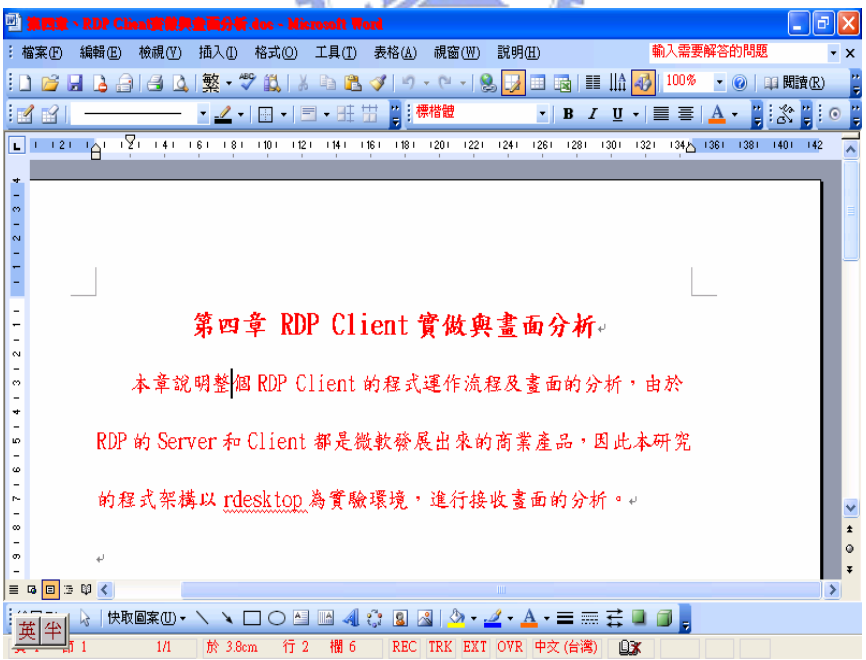


圖 28.(b) 利用 rdesktop 開啟遠端 Word（拖曳後）

一個 text order 中要從 fontcache 取出的文字數目，一個 order 可以從 fontcache 取出一個或是多個文字，也就是說一個 text order 一次可以傳送多個文字的資訊。

4.3.2 圖形指令

圖形指令在 RDP order 中，可以分為四種：矩形、多邊形、橢圓以及線條，多邊形及橢圓主要是應用在 power point 中的快取圖形，矩形主要是用在組合視窗的結構，線條則是用在文字的底線部分。

- 矩形（rectangle）

在遠端桌面連線中，常見到背景單純的畫面或是視窗框架的部份，RDP Server 利用傳送 rectangle 來呼叫 RDP Client 進行畫面的繪製，關於這個指令，我們定義一個 RECT_ORDER 的資料結構，如下圖。

```
typedef struct _RECT_ORDER
{
    sint16 x;  // x座標
    sint16 y;  // y坐標
    sint16 cx;  // 矩形寬度
    sint16 cy;  // 矩形高度
    uint32 colour;  // 顏色

}
RECT_ORDER;
```

圖 29. RECT ORDER 的結構

rdesktop 中建立一個 process_rect 的函數，將封包中的參數資料放入 RECT_ORDER 結構中，接著呼叫 ui_rect 然後引用 RECT_ORDER 的參數來設定矩形的顏色並將矩形座標位置及長寬參數傳送給 FILL_RECTANGLE，在 RECTANGLE 中呼叫 X Lib 的 XFillRectangle 將矩形繪製出來，RDP Client 對於 RECTANGLE 的處理流程如下圖 30.。

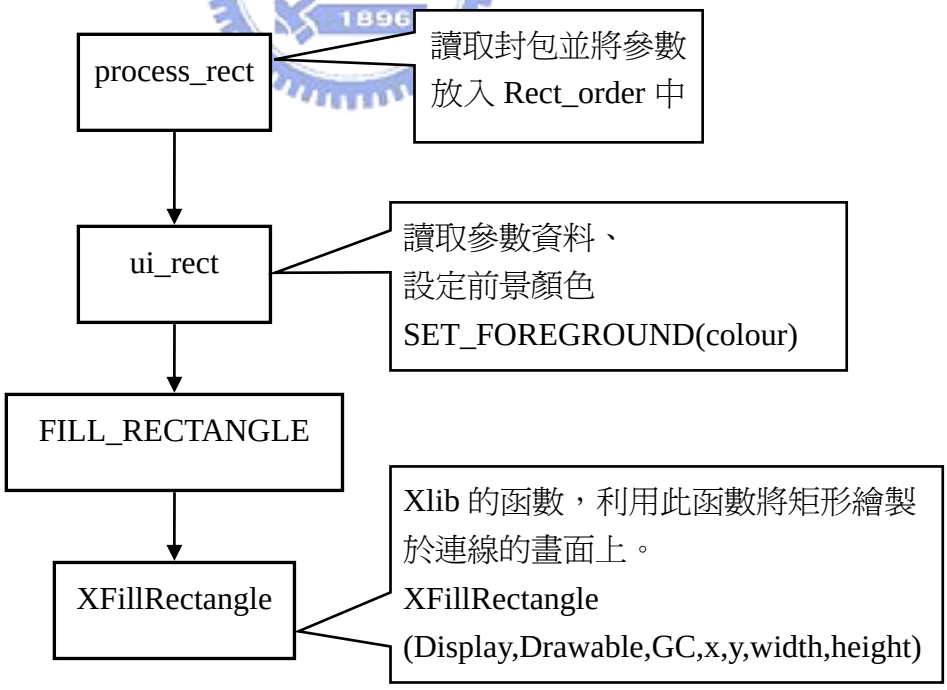


圖 30. rect order 處理流程

進行畫面分析之前，先說明 XFillRectangle 函數的重要性，由於 rdesktop 是建立在 X Window 中，XFillRectangle 是 rdesktop 和 X Window 的溝通函數。XFillRectangle

根據 RDP Server 所傳遞的座標（X、Y）和大小（長、寬）在指定的繪圖區，也就是 rdesktop 程式視窗進行繪製。

瞭解 RECTANGLE 處理過程後，我們進行 rdesktop 程式碼的修改，將傳送過來的 RECTANGLE，前景顏色都設定為綠色，在呼叫 ui_rect 之前，設定 colour 為指定的綠色，RDP Server 傳送過來的 rect order 以綠色矩形來顯示，幫助我們了解畫面的組成中哪些是利用 RECTANGLE 來繪製畫面，遠端桌面系統中，很多背景都是單調的顏色，RDP Server 利用 RECTANGLE 傳送參數後，由 RDP Client 進行繪製，可以大幅降低點陣圖的傳送，在微軟 Windows 中利用 GDI（繪圖方面的 API）來處理；在 X window 中，rdesktop 則是利用 X Lib 的函數進行繪製，以下我們針對常使用的畫面進行分析。

圖 31. 正常的遠端登入畫面

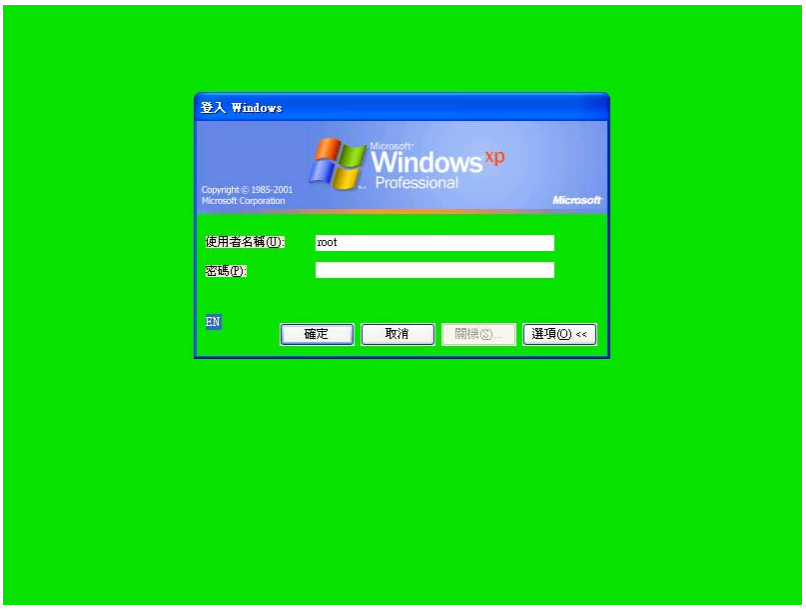


圖 32. 利用修改後的 rdesktop 進行遠端登入

由上述比較，原本黑色的背景及灰白色的視窗中，利用修改後的 rdesktop 連線登入，我們發現 RDP Server 在背景單純的畫面中，使用大量的 RECTANGLE 來繪製背景，透過這樣的傳送方式可以有效節省頻寬的使用。

在桌面背景中，RDP Server 也傳遞 RECTANGLE 的命令，將畫面的背景交由 RDP Client 也就是使用者端來繪製，如圖 33.所示。透過 rect order 的方式傳送桌面背景，參數化的指令其資料量遠小於傳送整張 bitmap 畫面的資料量。

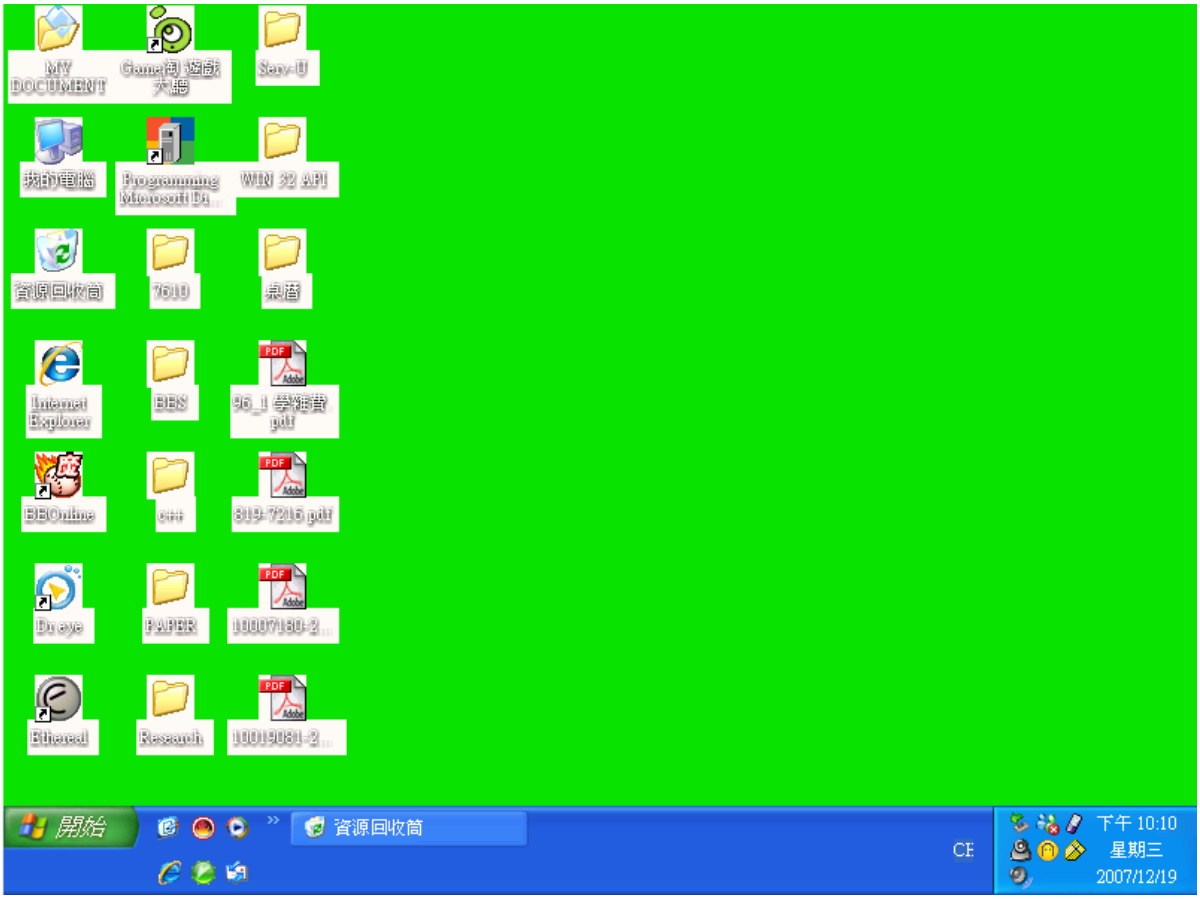


圖 33. 利用修改後的 rdesktop 顯示遠端桌面

瞭解了桌面的畫面後，我們對 Internet Explorer 開啟網頁的畫面以及 PowerPoint 畫面進行分析，在網頁畫面的傳送中，剛開啟的網頁背景單純的畫面，會使用 RECTANGLE 的命令傳送給 RDP Client，可以看出畫面背景為綠色矩形，如圖 34.(a)；但是當改變視窗大小或是拖曳視窗後，RDP Server 會以 bitmap 命令進行更新，除了視窗的框架外，網頁的畫面以點陣圖（bitmap）方式傳送，如圖 34.(b)。

圖 34.(a) rdesktop 顯示遠端網頁矩形特徵（剛開啟）

圖 34.(b) rdesktop 顯示遠端網頁矩形特徵（變動後）

在 PowerPoint 中，背景以及視窗的內部框架都是使用 rect order 來顯示畫面，而較複雜的圖片則以點陣圖的方式傳送，如圖 35.。

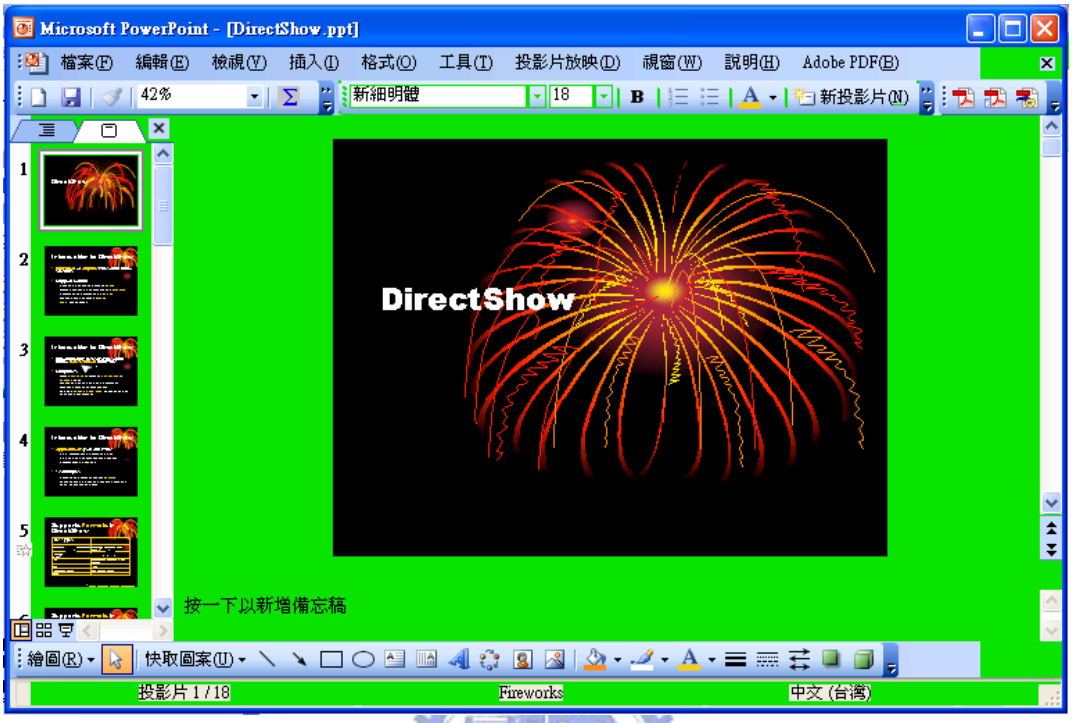


圖 35. 利用修改後的 rdesktop 開啟遠端 PowerPoint

- 多邊形（polygon）和橢圓（ellipse）

這部份的指令大多是使用在 PowerPoint 的快取圖案上，rdesktop 對於 polygone 和 ellipse 的處理，分為不透明跟透明，接下來，我們說明 rdesktop 對多邊形和橢圓命令的實作流程。

多邊形：

在接收多邊形命令前，我們先定義多邊形的結構，基本上的架構包含了座標、顏色、多邊形的樣式等等，多邊形其結構如下圖 36.。

```
typedef struct _POLYGON_ORDER
{
    sint16 x; //圖形x座標
    sint16 y; //圖形y座標
    uint8 opcode; //ROP
    uint8 fillmode; //圖形類型
    uint32 bgcolour; //矩形背景顏色，如果是不透明多邊形不定義此參數
    uint32 fgcolour; //矩形前景顏色
    BRUSH brush; //填色樣式，如果是不透明多邊形不定義此參數
    uint8 npoints;
    uint8 datasize;
    uint8 data[MAX_DATA];

}
POLYGON_ORDER;
```

圖 36. POLYGON ORDER 結構

接收到 polygon 的封包後，由 process_polygon 將資料資料從封包中讀取出來，呼叫 ui_polygon 進行處理，在根據 fillmode 決定多邊形形狀，依定義的 brush 類型，設定參數並呼叫 FILL_POLYGON 函數，FILL_POLYGON 則是呼叫 X Lib 中 XFIllPolygon 將圖形顯示在畫面上。程式流程方塊圖如下：

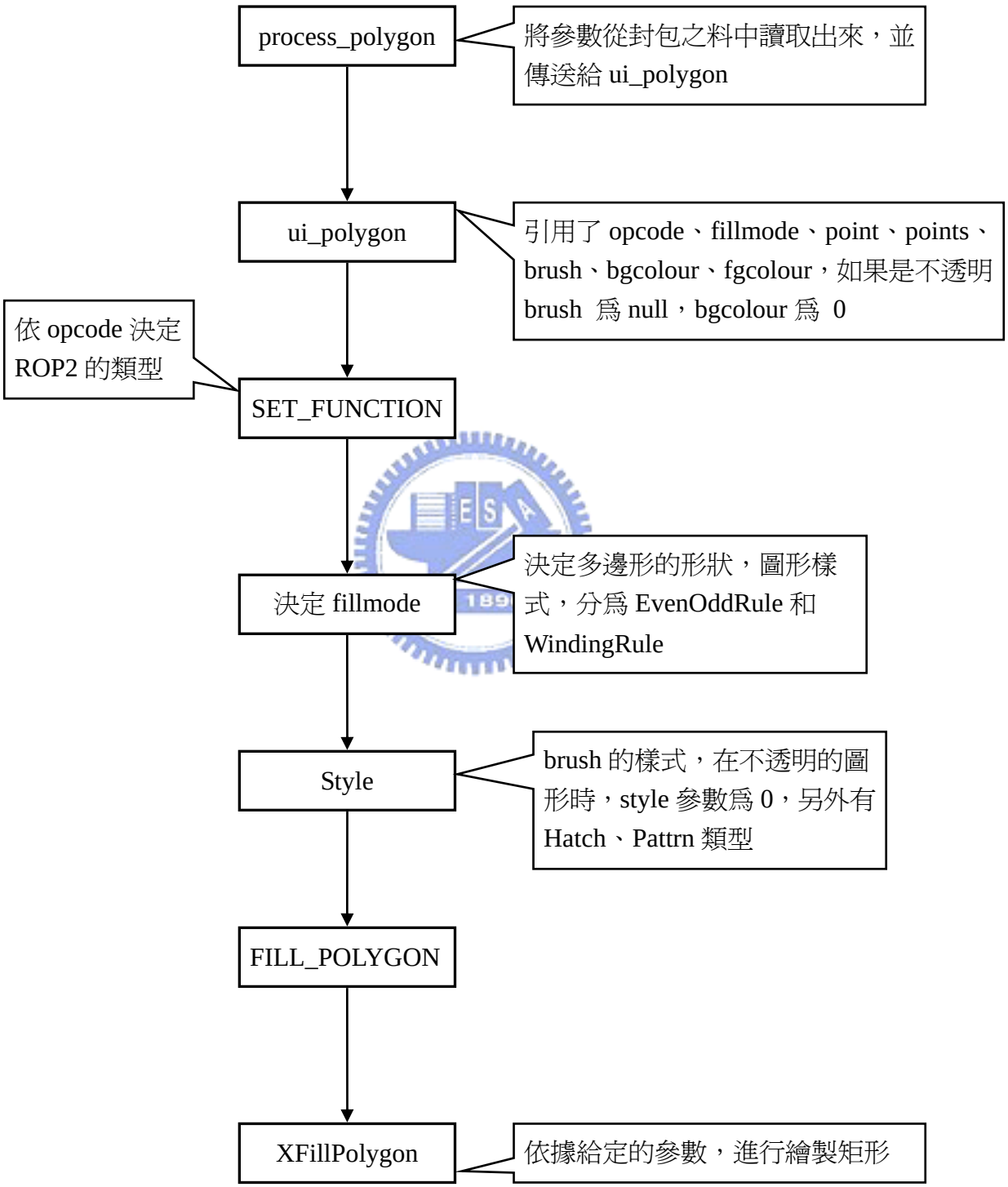


圖 37. 多邊形處理流程

在圖 37.中，提到 fillmode 決定了多邊形形狀，這個部分我們根據 X window 的繪圖規則[11]說明，當我們決定多邊形各頂點時如圖 38.，無法確切決定樣式，根據 X window 的填充規則，分成兩種 EvenOddRule 和 WindingRule。

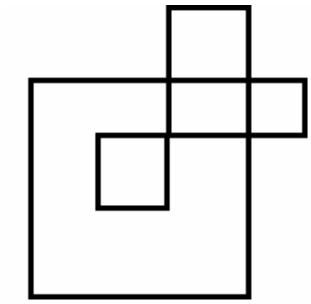


圖 38. 多邊形外框

EvenOddRule 就是利用一條直線，從某一點到多邊形外邊的點，如果這條直線和多邊形交界次數是奇數的話，此點才是多邊形內部的點，也就是填充的部份，如圖 39.。

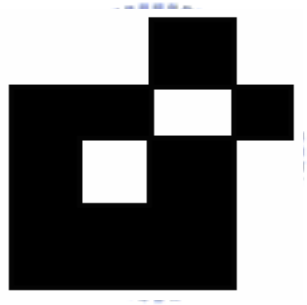


圖 39. 多邊形 EvenOddRule

WindingRule 凡是重疊的部份不管交叉幾次都是填充的部份，如圖 40.。

圖 40. 多邊形 WindingRule

瞭解了多邊形的填充模式（fillmode）後，我們針對 XFillPolygon 函數進行說明，由於 rdesktop 建立在 X Window 平台上，因此最後需要呼叫 X Lib 的函數顯示在畫面上，而 XFillPolygon 就是扮演這樣的角色。

XFillPolygon 的形式如下：

XFillPolygon （ display, drawable, gc, points, npoints, shape, mode)

Display：X Window 的顯示架構。

Drawable：可以是 window 或是 pixmap，差異在 window 為顯示的視窗，pixmap 可以是一個繪圖的區塊，不一定會顯示出來。

GC：一個繪圖的環境，指定繪圖時必須使用的屬性，屬性包含顏色、寬度等。

Points：各端點的陣列。

nPoints：點的數目。

Shape：代表了圖形的形狀，分為：Convex（外凸）、NonConvex、Complex。rdesktop 將 Shape 設定為 Complex，也就是說不知道多邊形的邊是否交叉，交由 X Window 去處理，花費較長，但是最後的圖形是正確的。

Mode：為各端點座標的位置，分為 CoordModeOrigin 和 CoorModePrevious。CoorModeOrigin 表示所有的端點座標都是相對於視窗的原點；CoorModePrevious 則表示除了第一點相對於視窗的原點外，其餘各點相對於前一點的座標，rdesktop 是使用 CoorModePrevious 的設定。

Rdesktop 就是依 RDP Server 所傳遞的訊息，呼叫 XFillPolygon 將多邊形繪製於 rdesktop 連線視窗上。

橢圓形：

首先定義 Ellipse 的參數結構，如圖 41.。

```
typedef struct _ELLIPSE_ORDER
{
    sint16 left;
    sint16 top;
    sint16 right;
    sint16 bottom;
    uint8 opcode;
    uint8 fillmode;
    BRUSH brush; // 在不透明的情況下，不定義此參數
    uint32 bgcolour;// 在不透明的情況下，不定義此參數
    uint32 fgcolour;

}
ELLIPSE_ORDER;
```

圖 41. ELLIPSE ORDER 參數結構

由 process_ellipse 處理封包的參數，並將參數放入 ELLIPSE ORDER 結構中，接著呼叫 ui_ellipse 處理，依據 opcode 決定 ROP2 參數，接著根據 brush 決定與背景的混合模式（Solid、Hatch、Pattern），之後呼叫 DRAW_ELLIPSE，由於 rdesktop 應用在 X Window 系統中所以 DRAW_ELLIPSE 則是呼叫 X Lib 的 XDrawArc 將圖形繪出。

XDrawArc 是 X Lib 中畫弧形的函數，rdesktop 利用此函數進行橢圓的繪製，在角

度部份定義為 360 度，XDrawArc 的函數形式如下：

XDrawArc (display, drawable, gc, x, y, width, height,start_angle,stop_angle)

Display：X Window 的顯示架構。

Drawable：可以是 window 或是 pixmap，差異在 window 為顯示的視窗，pixmap 可以是一個繪圖的區塊，不一定會顯示出來。

GC：一個繪圖的環境，指定繪圖時必須使用的屬性，屬性包含顏色、寬度等。

X,Y：是圖形的左上角座標。

Width,Height：分別是橢圓中心的兩個軸的長度。

Start_angle：為弧線的起始角度，rdesktop 設為 0*。

Stop_angle：為弧線的停止角度，rdesktop 畫橢圓時設為 360*64。(*64 是 X Window 畫弧的角度計算）函式說明如圖 42.。

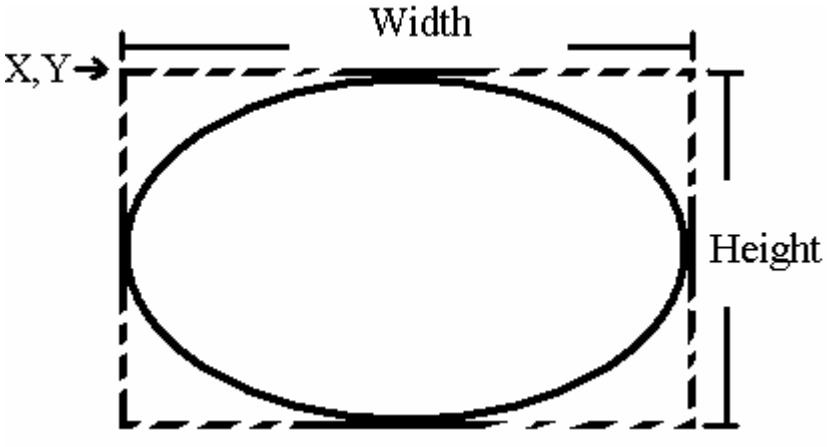


圖 42. rdesktop 橢圓形的定義

- 線條（Line）

主要是應用在文字的底線，首先我們依據 Line 的參數定義出 LINE_ORDER 的結構，如下圖。

```
typedef struct _LINE_ORDER
{
    uint16 mixmode; /*和 背景的混合模式*/
    sint16 startx;  /*起始點的X座標*/
    sint16 starty;  /*起始點的Y座標*/
    sint16 endx;    /*終 點的X座標*/
    sint16 endy;    /*終 點的Y座標*/
    uint32 bgcolour; /*背景顏色*/
    uint8 opcode;   /*ROP類型*/
    PEN pen;        /*線條類型*/

}
LINE_ORDER;
```

圖 43. LINE ORDER 參數結構

當收到一個 Line 命令時，rdesktop 呼叫 process_line 處理，這個函數先解開封包

內的參數，並將參數置入 LINE_ORDER 結構中，之後呼叫 rdp_parse_pen，將線條類型（style）、寬度（width）、顏色（colour）根據收到的資料欄位設定參數，在呼叫 ui_line 進行繪製，而 ui_line 最後仍然呼叫 X Lib 的 XDrawLine 繪製線條。XDrawLine 函數形式如下：

XDrawLine (Display, Drawable, GC, X1, Y1, X2, Y2)

XDrawArc (display, drawable, gc, x, y, width, height,start_angle,stop_angle)

Display：X Window 的顯示架構。

Drawable：可以是 window 或是 pixmap，差異在 window 為顯示的視窗，pixmap 可以是一個繪圖的區塊，不一定會顯示出來。

X1、Y1：為起始點座標

X2、Y2：為終點座標

在圖形的繪製上，不論是矩形、多邊形或是線條，其實都是依據 RDP Server 傳送過來的封包，根據其定義的參數，參數不外乎座標、前景、背景及混合模式，之後呼叫 X window 中的 X Lib 進行繪製，也就是說 X Lib 中的繪圖函數就是 rdesktop 和 X Window 的中介函數，透過參數的傳送在 Client 繪圖，而非直接傳送 bitmap 資料，可以降低頻寬的使用。

4.3.3 其他命令

RDP 中 primary order 除了文字、圖形外，還有一些控制畫面更新的命令，例如：Memory Blt、Pattern Blt、DeskSave、Screen Blt 等，利用這些命令將 cache 中暫存的圖片顯示在畫面，這一章節，對於這些命令的實作程序及實驗過程做說明。

- Pattern Blt

主要是傳輸圖樣，依據 RDP Server 指定的前景（foreground）、背景（background）在指定的位置顯示圖樣，最常見的情況是文字輸入的空格游標，在處理命令之前先定義 PATBLT_ORDER 參數結構，如下圖。

```
typedef struct _PATBLT_ORDER
{
    sint16 x;  //目標x座標
    sint16 y;  //目標y座標
    sint16 cx;//寬
    sint16 cy;//高
    uint8 opcode; // ROP
    uint32 bgcolour;//背景
    uint32 fgcolour;//前景
    BRUSH brush;   //Brush結構，說明顯示的圖樣

}
PATBLT_ORDER;
```

圖 44. PATBLT ORDER 參數架構

收到 Pattern Blt 封包後，由 process_patblt 處理參數後，將封包資料解開後，將收到的參數放入 PATBLT_ORDER 架構中，呼叫 ui_patblt 並引用 PATBLT_ORDER 的參數，ui_patblt 依據 opcode 中 Brush 的 style 決定樣式，例如：實心（Solid）、影線（Hatch）、圖樣（Pattern），根據不同的樣式設定前景或是背景，呼叫 X Lib 的函數進行繪圖，不同的樣式呼叫不同的 X Lib 函數，最主要的還是呼叫 XFillRectangle 繪製出圖形。

- Screen Blt

在視窗桌面系統中，在視窗拖曳的過程中，視窗內的畫面是沒有改變的，變動的是視窗的位置，因此 RDP Server 利用 Screen Blt 傳送給 RDP Client，通知改變視窗位置，而不用傳送整個視窗的 bitmap 給 RDP Client。一個 Screen Blt 的參數包含如下圖。

```
typedef struct _SCREENBLT_ORDER
{
    sint16 x; //目標x座標
    sint16 y; //目標y座標
    sint16 cx; //寬
    sint16 cy; //高
    uint8 opcode;
    sint16 srcx; //來源座標x
    sint16 srcy; //來源座標y

}
SCREENBLT_ORDER;
```

圖 45. SCREENBLT ORDER 參數結構

rdesktop 接收到 Screen Blt 封包後，讀取封包參數後，將參數存入 SCREENBLT ORDER 結構中，呼叫 ui_screenblt()處理，ui_screenblt()函數呼叫 X Lib 的 XCopyArea()將來源端的影像複製到目標座標的位置，封包處理流程如下圖。

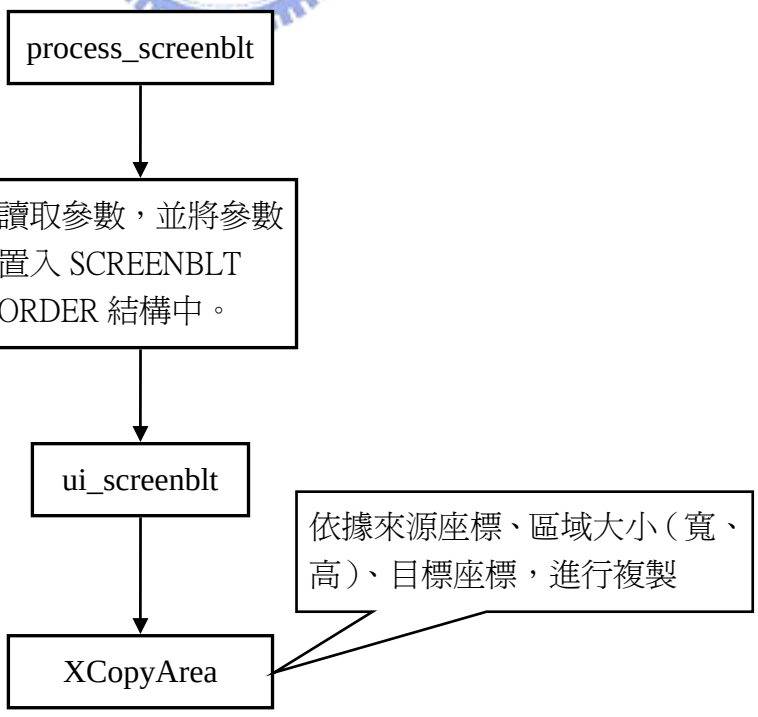


圖 46. screen blt 命令處理流程

- Desktop Save

  rdesktop 中建立了 DESKSAVE_ORDER 參數結構，如下圖。

```
typedef struct _DESKSAVE_ORDER
{
    uint32 offset; //記憶體位置 指標
    sint16 left;   //儲存區域的座標X
    sint16 top;    //儲存區域的座標Y
    sint16 right;  //儲存區域的座標X
    sint16 bottom; //儲存區域的座標Y
    uint8 action; //處理動作 : save or restore

}
DESKSAVE_ORDER;
```

圖 47. DESKSAVE ORDER 參數結構

rdesktop 程式中建立了 process_desksave()函數來處理這個指令，解析出 desktop 暫存區域的四個頂點後，計算出長、寬、offset 和 action(定義 save/restore)後，判斷 action 是 save 或是 restore，交給 ui_desktop_save 或是 ui_desktop_restore 處理。

假設是 save，交由 ui_desktop_save 處理並呼叫 cache_put_desktop 將資料存入 desktop cache；反之如果是 restore，就呼叫 ui_desktop_restore 處理，函式中呼叫 cache_get_desktop 將桌面圖檔從 dektop cache 中讀取出來。

這個命令是使用在視窗畫面中，常常可遇到的 menu 選單中，例如：Windows 視窗中的開始選單或是視窗選項中的工具列，由於選單延伸出來後往往會蓋住原本底下的畫面，但是當我們選擇完選項後，桌面被覆蓋的畫面又出現了，如果重複的畫面需要 RDP Server 重新傳送，將浪費頻寬，因此 RDP Server 在覆蓋前會判斷選單的出現，在畫面被覆蓋之前先傳送一個 desktop save 的命令，使得 RDP Client 將桌面的畫面暫存到 desktop cache 中，當選單結束，底下的桌面需要顯示時，RDP Server 便傳送一個 desktop restore 的命令給 RDP Client，RDP Client 便將暫存在 cache 中的畫面重現出來。

透過 RDP 這樣的機制可以有效節省網路頻寬的使用，為了瞭解 RDP Server 的判斷機制，下圖 48.系列就是我們實驗的畫面。圖 48.(a)是原始的畫面，還沒拉出選單，圖 48.(b)是桌面拉出「我最近的文件」選單後的畫面，圖 48.(c)是選單下的桌面畫面，也就是暫存在 desktop cache 中的畫面資料。

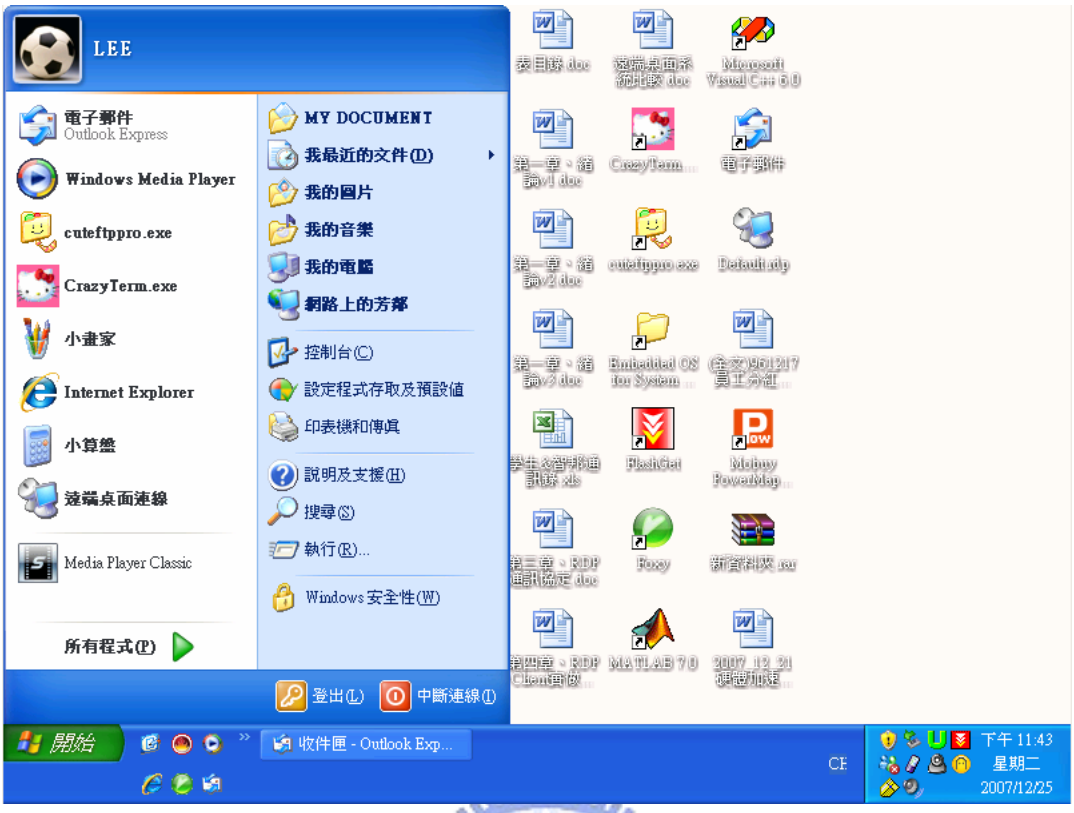


圖 48.(a) 原本的視窗畫面

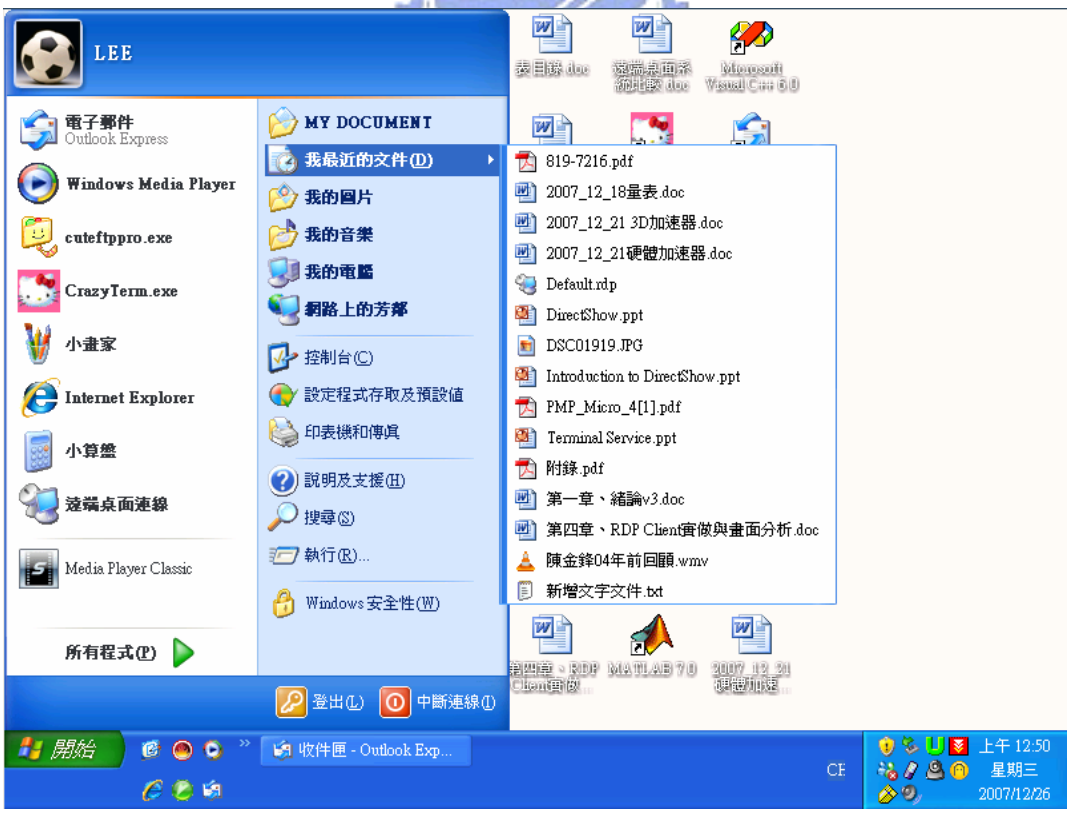


圖 48.(b) 拉出選單目錄後的畫面

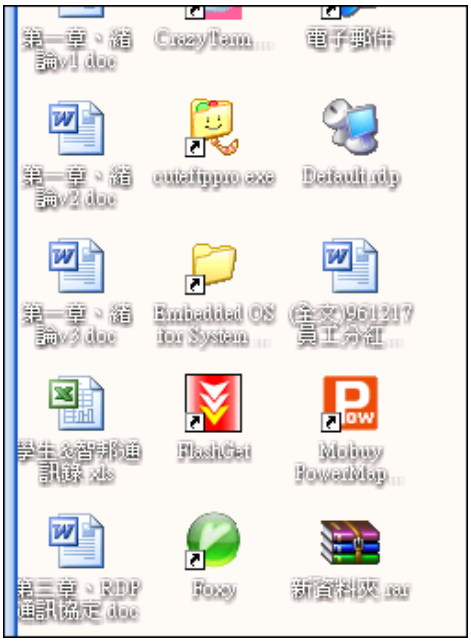


圖 48.(c) 暫存在 desktop cache 中的畫面資料

由圖 48.，我們可以看出(c)圖的資料，也就是(b)圖中選單拉出來後，所覆蓋的資料，因此在選單覆蓋的面積越大時，RDP 指令的效益越大。

- Memory Blt

在 RDP Server 傳送過來的 bitmap 都將暫存在 cache 中，每個 bitmap 都附予一個 cache id 和 cache index，在畫面顯示時，RDP Server 就傳送 Memory Blt 給 RDP Client，RDP Client 就根據 Memory Blt 所包含的參數，將 cache 中的圖片顯示在畫面上，簡單說 Memory Blt 是將 cache 中的 bitmap 顯示在畫面的特定位置。

一般來說，視窗中單一色調的底色，如果沒有使用圖形命令（例如：rectangle order）的話，就是利用單一個 bitmap（最大為 64x64），在不同的位置上進行重複的顯示。rdesktop 將 Memory Blt 的參數結構定義如下圖。

```
typedef struct _MEMBLT_ORDER
{
    uint8 colour_table; //colormap
    uint8 cache_id; // cache id
    sint16 x; //圖的座標位置 X
    sint16 y; // 圖的座標位置 Y
    sint16 cx; //width
    sint16 cy; //height
    uint8 opcode; //ROP
    sint16 srcx; //圖形的複製起始位置 X
    sint16 srcy; // 圖形的複製起始位置 Y
    uint16 cache_idx; // cache index

}
MEMBLT_ORDER;
```

圖 49. MEMBLT ORDER 結構

rdesktop 定義了 process_memblt 進行 memory blt 的封包處理並定義了 HBITMAP 變數，讀取 memory blt 參數後將資料放入 MEMBLT_ORDER 結構中，利用 cache_get_bitmap 函數依據 cache id 和 cache index 將 bitmap 從 cache 取出，並將讀取出來的 bitmap 資料放入 HBITMAP，之後使用 ui_memblt 進行更新，最後利用 X Lib 的 XCopyArea 將 bitmap 的資料複製到連線的畫面上，程式架構如下圖。

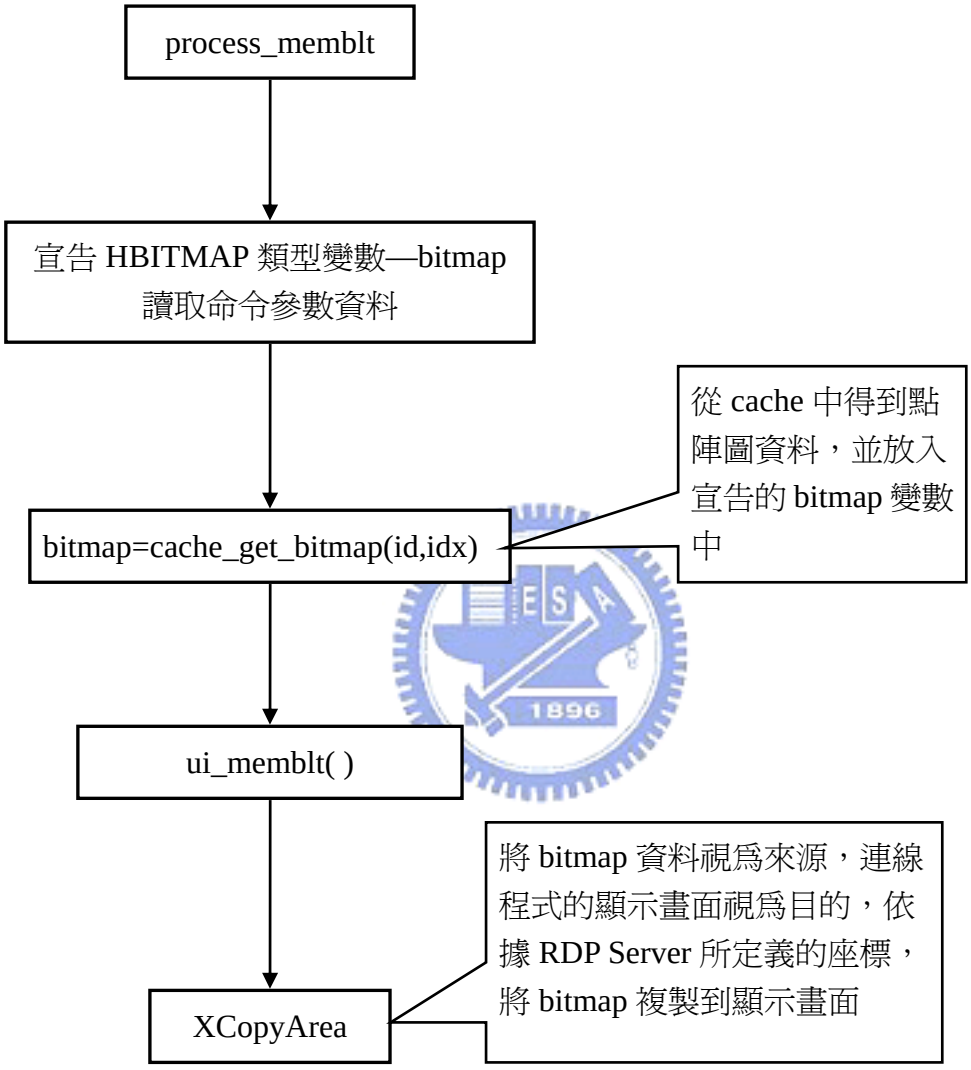


圖 50. memory blt 封包處理流程

透過 memory blt 指令的使用，在重複出現或是單一色調的區塊中，可以節省頻寬的使用，但是，當我們想要在遠端桌面進行影片播放時，假設影片大小為 800x600，因為影片的畫面每張 frame 都不一樣，RDP Server 不斷傳送 bitmap，那麼畫面中就是都用 bitmap 在更新畫面，那麼就沒有達到節省頻寬的目的，下頁圖 51.我們根據影片畫面中 memory blt 所提供的資訊，將傳送 memory blt 位置顯示出來，在第五章，我們將詳細說明 RDP 在影片播放上的瓶頸。

圖 51. RDP 連線中 Memory blt 顯示位置

可以看出，影片畫面為 320x240，寬的部份切為 64 pixel 五塊，高的部份為 64 pixel 三塊，剩餘一塊高度則為 48pixels。

4.4 Secondary Order 的處理

將 RDP Server 傳送過來的圖片(bitmap)、文字(text)及色彩對應表（color table）進行暫存，在遠端視窗桌面的連線上，文字、圖形會暫存在 cache 中，畫面中有重複出現的畫面利用 memblt 將畫面取出，如果是文字那麼就利用 text order 將 cache 中的點陣字型取出顯示在螢幕上。

rdesktop 中定義了 process_secondary_order 處理 secondary order 的封包，程式呼叫流程如下圖。

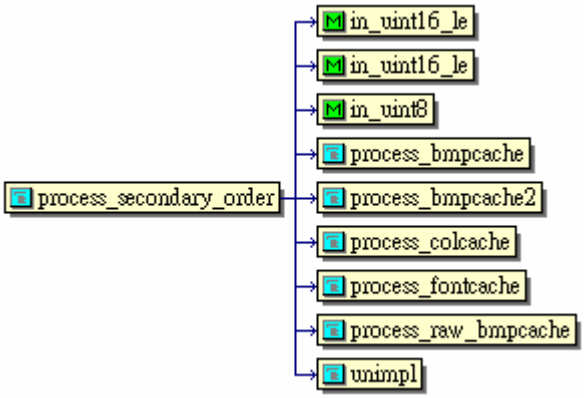
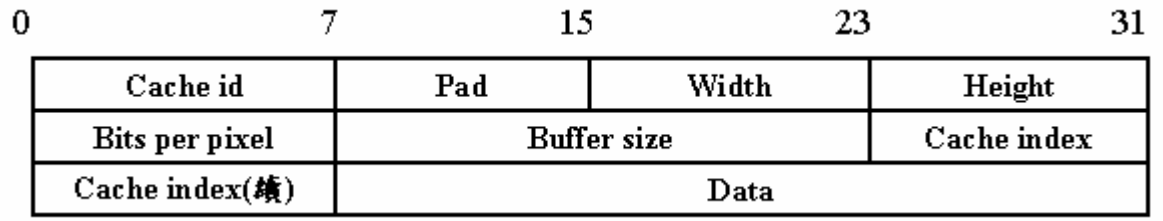


圖 52. process_secondary_order 函數呼叫流程

in_uint_le 分別解析出封包的長度及 flags，flags 用在 bmpcache2 中告知 RDP Client 關於 bitmap 的資訊(例如：是否為持續型點陣圖(persistent)或是 bitmap 的形狀)，in_uint8 則是讀取出封包類型(type)，用來決定下面所呼叫的 process 函數。

4.4.1 process_bmpcache

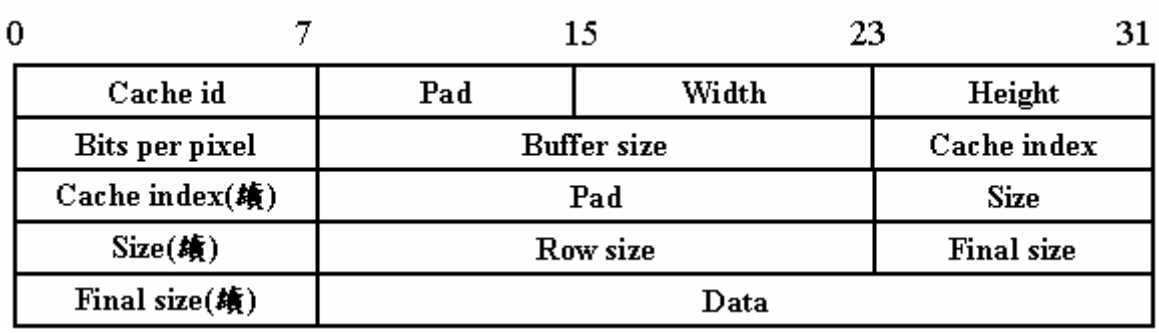
process_bmpcache：處理 bitmap cache 命令，根據傳送過來的 bitmap 封包，資料解壓縮後，依 cache id、cache index 將 bitmap 資料存入 bitmap cache 中，Bmpcache 在 rdesktop 分為兩種 header，關鍵在於是否為 rdp5，以下說明其封包格式。圖 53.為使用 rdp5 的格式，其中 Data 的資料量由 Buffer size 所決定。



0	7	15	23	31
Cache id	Pad	Width	Height	
Bits per pixel	Buffer size		Cache index	
Cache index(續)	Data			

圖 53. rdp5 process bmpcache 封包格式

如果不是 rdp5 的連線，其格式為圖 54.所示，封包中 Data 的最後大小即為 size 所說明的大小。



0	7	15	23	31
Cache id	Pad	Width	Height	
Bits per pixel	Buffer size		Cache index	
Cache index(續)	Pad		Size	
Size(續)	Row size		Final size	
Final size(續)	Data			

圖 54. process bmpcache 封包格式

process bmpcache 不管是哪一種封包格式，最後解壓縮後，由 ui_create_bitmap() 函數將 bitmap 繪出後，依 bmpcache 所指定的 cache id 和 cache index 將點陣圖儲存於 bitmap cache 中，在實驗進行的過程中，由 rdesktop 連線至 Microsoft XP RDP5 .1 並未使用這支函數，而是使用 process_bmpcache2()，因此接下來針對 process_bmpcache2 進行說明。

4.4.2 process_bmpcache2

process_bmpcache2(STREAM s, uint16 flags, BOOL compressed)，對 bitmap 處理可分為壓縮和無壓縮，並依據 flags 定義 bitmp 資訊，最後將 bitmap 放入 bmpcache 中，程式處理程序如下圖。

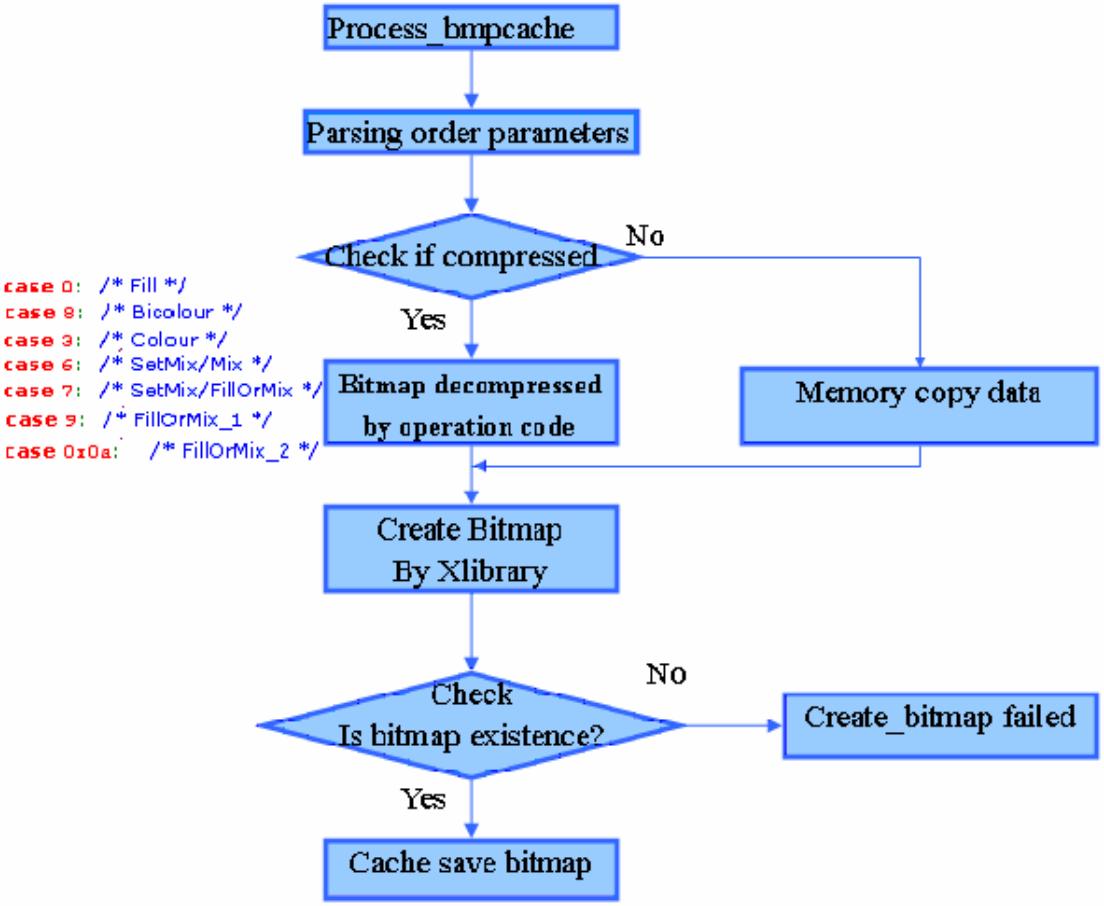


圖 55. process bmpcache2 程式處理流程

由 process_secondary_order 中的 flags 傳遞了以下資訊：

1. cache id
2. 決定暫存的 bitmap 是否存放在 persistent cache，如果存放 persistent cache 則另外給予一組 bitmap id
3. bitmap 是否為正方形（例如：64x64），如果是正方形長寬一樣。如果不是正方形，分別取出長寬資料
4. bits per pixel

接著定義出 buffer size 決定資料大小，由收到的封包中取出資料，如果沒有壓縮就直接將資料讀出來，由 ui_create_bitmap 繪出資料；如果 bitmap 有壓縮，利用 bitmap_compress()函數將資料解壓縮後，交由 ui_create_bitmap 繪出 bitmap，完成的 bitmap 由 cache_put_bitmap 存入 bitmap cache 中，如果是 persistent 則交由 pstcache_save_bitmap 函數，將 bitmap 存入硬碟暫存。

4.4.3 process_colcache

process_colcache：處理 colormap cache 命令，根據 cache id 呼叫 ui_set_colourmap 設定對應的 coloymap。在 256 色顯示時，才會利用到 colormap，在本研究中沒有使用到 colormap，這邊就不贅述。

4.4.4 process_fontcache

process_fontcache：處理 font cache 命令。接收點陣字型封包後，依 ui_create_glyph 將自行繪製出來，之後 cache_put_font 將點陣字型放入 font cache 中。一個 Font_Glyph 命令結構定義如下圖。

```
#define MAX_GLYPH 32

typedef struct _RDP_FONT_GLYPH
{
	uint16 character;
	uint16 unknown;
	uint16 baseline;
	uint16 width;
	uint16 height;
	uint8 data[MAX_GLYPH];

}
RDP_FONT_GLYPH;
```

圖 56. RDP Font Glyph 結構

收到一個 Font 點陣字型封包後，依據所定義的結構解開 header 後，將收到的 data 交由 ui_create_glyph()處理，將繪制好的 1 bpp 點陣字型存入 font cache 中，在文字部分，RDP Server 傳遞過來的資訊不經過壓縮編碼，而是一位元接著一位元依序傳送。在一個 process fontcache 的命令中，所傳送過來的文字是以 glyph 來定義的，一個文字就是一個 glyph，而一個 process fontcache 的 secondary order 包含的文字數目是一個或是多個的，rdesktop 收到後利用 ui_create_glyph 將每個文字建立出來，之後每個文字分別放入 fontcache 中。

4.4.5 process_raw_bmpcache

process_raw_bmpcache：處理 raw bitmap cache 命令。將未經處理壓縮的 bitmap 資料從封包中取出來，利用 memcpy 直接將封包資料複製到記憶體中，接著呼叫 ui_create_bitmap 將 bitmap 繪製出來，最後呼叫 cache_put_bitmap 將資料存入 cache 中。

由以上的敘述，可以得知 secondary order 的目的是處理暫存在 cache 中的資料，透過資料的暫存可以有效降低點陣圖資料的傳送，比起 VNC 連線方式，RDP 就是應用了大量的 cache 提升連線效益，但是在影片播放時，由於每個畫面幾乎是沒有重複的，因此 RDP Server 會傳送大量 process_bmpcache 命令，將 bitmap 暫存到 bitmap cache 中，造成影片播放時 frame 遺失。

4.5 Bitmap 編碼方式

在 4.3 節中，我們討論到 bitmap 的暫存機制，其中討論到 bitmap 有壓縮的機制，這一節我們說明 Bitmap 的編碼種類，主要是根據 operation code 的類別進行編碼，如下列所式：

- 0：Fill，填滿和之前相同的顏色
- 1：Mix，和之前的顏色混色，也就是進行位元的 AND 運算
- 3：Colour，設定某一個 pixel 為特定顏色
- 4：Copy，從每一區塊複製到另一個
- 8：BiColour 由兩種特定顏色，依 pixel 交叉填色
- 13：White，將一區塊填滿白色
- 14：Black，將一區塊填滿黑色

Bitmap 的編碼原則類似於 RLE（Run Length Encoding），屬於一種非失真壓縮，後由於變動長度編碼法對於資料的內容相當敏感，隨著影像複雜度的不同，其壓縮率也會大幅的變動，第五章我們將實際透過實驗的結果來瞭解其壓縮率。

4.6 RDP Cache 類別

由 rdesktop 的程式中，可以看出 RDP Cache 分為三種類別：Bitmap、Text 和 Colormap 因次這一節我們對於這三類的 Cache 進行說明。

- bitmap Cache

圖形快取在 RDP 連線時扮演重要的角色，圖形快取是降低資料的主要原因，由 rdesktop 程式中可以看出 RDP 提供三種 Cache Slot Types，這三種 Cache Slot Type 決定於 bitmap 的大小，最大的 bitmap 的長和寬是 64 pixels，如果是更大的 bitmap 將會被切割成這三種形式，下頁就是這三種 Cache：

表 15. bitmap cache 類別

Cache id	Slot 數量	Cache 大小(pixels)	bitmap size
Cache 1	600	256	16x16
Cache 2	300	1024	32x32
Cache 3	262	4096	64x64

利用 Bitmap Cache 的方法，比直接傳送 bitmap 複雜了點，但是卻可以節省很多頻寬，以避免相同的 bitmap 不斷重複傳送佔用頻寬。

- text Cache

在 RDP 連線中，由於 RDP Server 對於 RDP Client 所支援的字型並不知道，因此文字部分是以 1 bpp 的點陣字型在傳送，對於同一文字的粗細、斜體變化都必須傳送

不同的點陣字型。值得注意的是如果文字大小超過 64x64 就會被切割成 bitmap 傳送，以 bitmap order 傳送給 client。

- colormap Cache

對於每個 pixel 使用的位元數不夠多的時候，就必須使用 colormap，因此在傳送 bitmap 時，對於相同的 colormap 暫存起來，之後如果有重複的 colormap 透過 cache id 呼叫指定的 colormap。

第五章 實驗數據與畫面分析

這個章節，我們將討論實驗數據以及 RDP 連線畫面構成，我們將比較桌面視窗系統中，各種應用程式透過 RDP 傳送後，透過 RDP 連線的畫面資料量與原始畫面資料的比較。

實驗環境：

連線桌面大小：800x600 pixels

bits per pixel：16 bpp

RDP Server OS：Microsoft Windows XP

RDP Server：RDP 5.1

RDP Client OS：Linux Feodra core 5

RDP Client：rdesktop-1.5.0

我們從 rdesktop 程式中，擷取來自 RDP Server 的資料量並觀察畫面的命令（order）組成，目標在於分析畫面中命令的組成，進而統計由 RDP 連線畫面的傳輸量。

實驗的目標希望由 rdesktop 進行畫面分析，首先針對不同的命令（order），例如：rectangle order、line order、text order 等不同的畫面型態，進行 rdesktop 原始碼的修改，第一步驟將命令（order）在連線時，將畫面中的指令特徵顯示出來以進行畫面結構的分析，改變各 order 的顯示顏色，針對 RDP Server 傳送過來的畫面，進行指令分析。

第二步修改 rdesktop source code 中 tcp.c 的程式碼，由 tcp_recv 函數中建立 tcp.txt 文字檔案，根據收到的封包將封包資料量及時間訊息依序儲存在 tcp.txt 文字檔。

第三步修改 order.c 的程式碼，建立 order.txt 文字檔案，將 rdesktop 收到的 orders 依封包接收流程，將相關 order 名稱及 order 執行時間儲存於 order.txt 檔案中。

第四步由儲存的.txt 檔案計算畫面中 RDP 封包的資料量及傳送的 order，統計一個 800x600 16bpp 的遠端畫面所傳送的資料量，並分別針對不同的遠端應用程式（例如：Microsoft Word、Internet Explorer、PowerPoint）進行分析，在視窗桌面系統的環境中，傳輸資料量最大的時候就是畫面更新的瞬間，其他的使用上都不會造成太大的資料量，因此，除了影片播放外，其他的應用程式，我們都針對畫面的更新瞬間進行資料量統計，至於影片播放的動態畫面，分別就三種大小：800x600、640x480 以及 320x240 尺寸，統計其傳輸的總資料量(Bytes)、傳輸速率（bps）以及遺失的 frame 數量。

最後由實驗的數據證明 Remote Desktop Protocol 的圖形壓縮方式，除了播放影片外，其他遠端服務的應用程式都適合使用 RDP 的圖形壓縮方式，由於 RDP 的壓縮方式採取命令（order）方式的傳送，對於色彩豐富變化頻率高的部份採用 bitmap order 方式傳送，而色彩簡單、輪廓分明的圖形部分則使用 rect order、line order 方式傳送，最重要的訊息傳遞文字部分則使用 text order 方式傳送至使用者端，這樣的壓縮方式保留了文字邊緣清晰度並利用 bitmap cache 將 bitmap 部分將資料暫存，對於重複出現的畫面可以使用。

5.1 RDP 封包分析

在進行各應用程式畫面分析之前，我們先說明 RDP 封包、TCP 封包和 RDP order 的關係,我們藉由連線遠端桌面的資料進行說明,一個 RDP 封包內可能包含多個 order 或是單獨一個 order，RDP Server 包裝後將 RDP 封包交由 MAC Layer，如果封包大於 1500 Bytes，將切割為數個 TCP 封包進行傳送，關於 RDP order、RDP Packet 和 TCP 封包的關係如下圖。

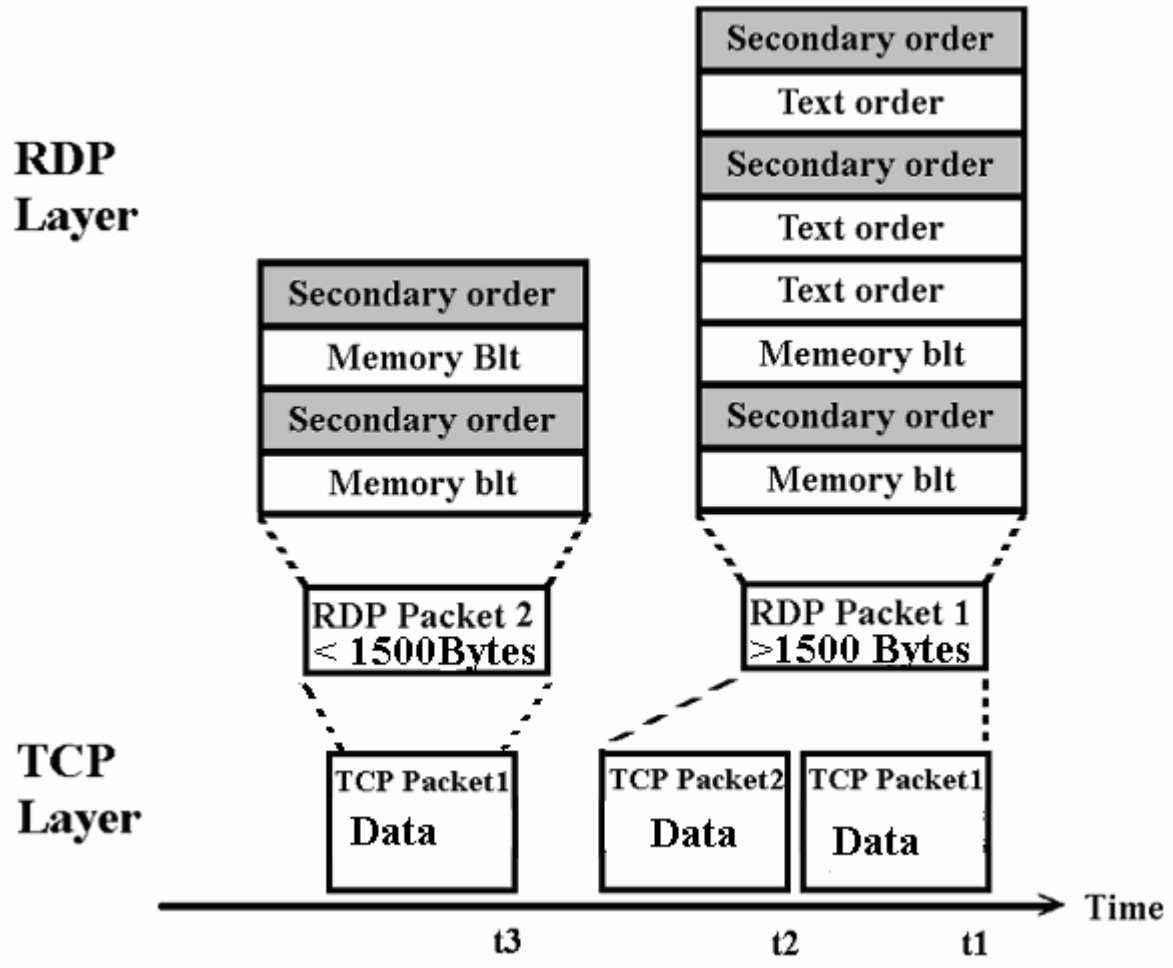


圖 57. RDP 封包與 TCP 封包關係圖

利用 rdesktop 進行實際上的封包擷取,圖 57.的情形是在畫面更新的瞬間所有的資料量改變的情況下，一個 RDP 封包將包含許多命令（order)，如果一個畫面沒有在改變，那麼也就不會有 order 的產生，另外一種情形，例如：命令輸入時的游標在閃爍時，通常 RDP Server 會傳送 Pattern Blt，此時將 RDP packet 包在 TCP 封包中 payload 傳送到 RDP Client，這樣的情況下就是一個 order 包在一個 RDP 封包，RDP 封包以一個 TCP 封包傳送。

RDP 畫面的更新分為 process order 與 bitmap updates，以時間軸上來看指令傳送，其程序如下圖。

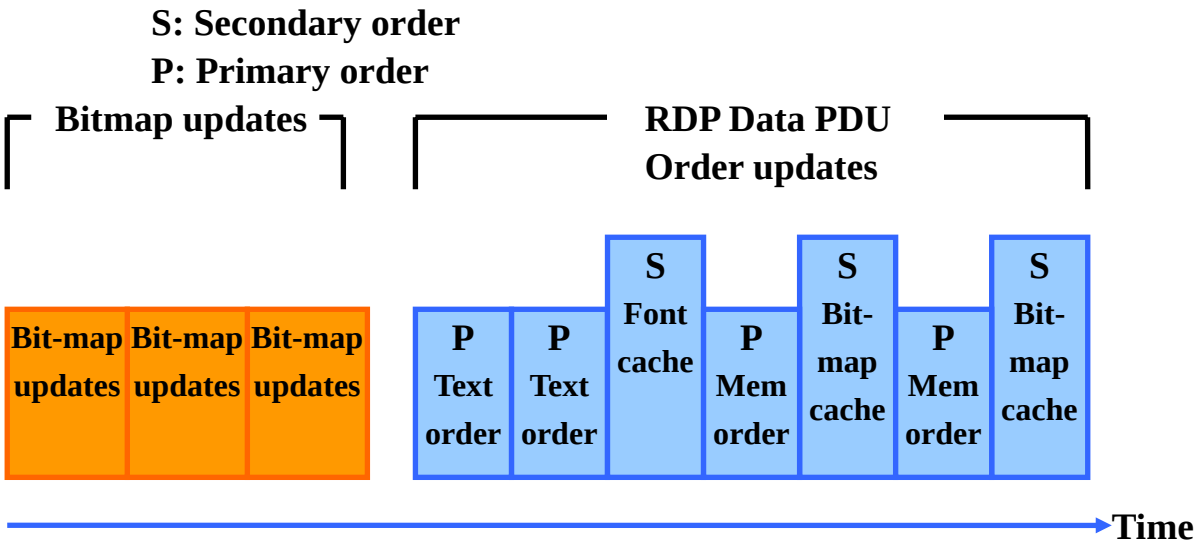


圖 58. 時間軸上的 RDP 指令程序

5.2 畫面命令結構分析

瞭解命令的傳送方式與 rdesktop 的操作流程，接下來以登入畫面為例，說明實際的命令分析，關於 bitmap cache、memory order、font cache 和 text order 的結構組成，如下圖所示，其中紅色框為 memory order 顯示的部份，綠色為 text order 部分。

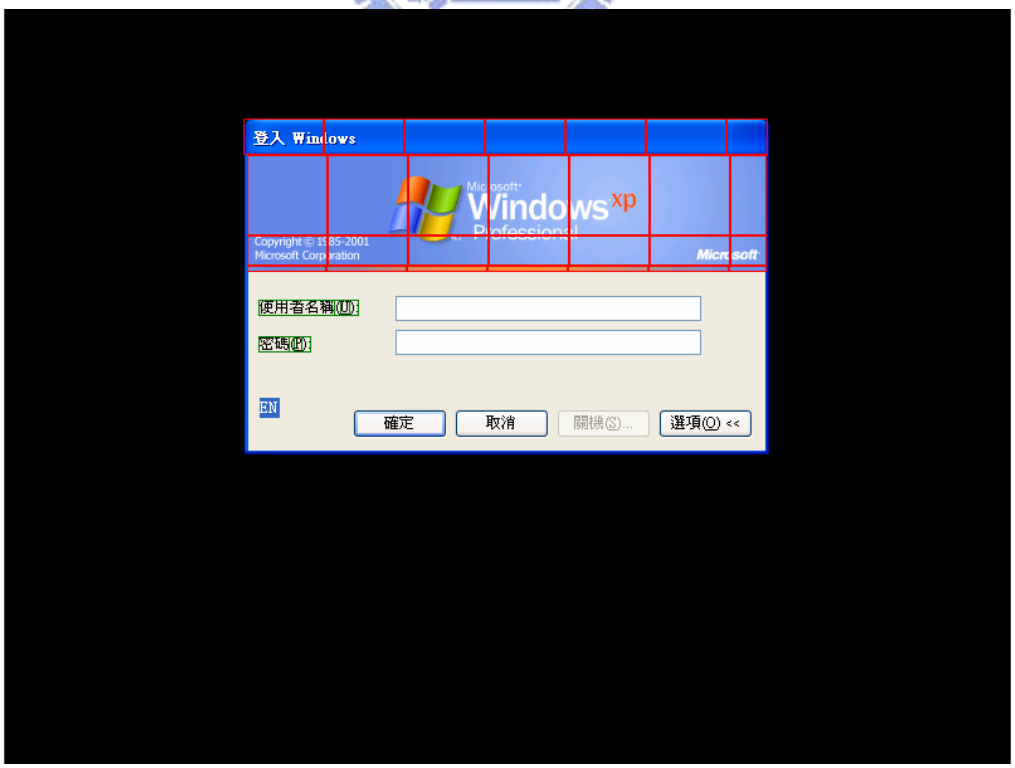


圖 59. 登入畫面指令組成

首先說明 bitmap cache 和 memory order 的處理程序，由 bitmap cache order 先將 bitmap 傳送到 client，之後 RDP Server 在傳送 memory order 將圖片從 bitmap cache 取出後，顯示在畫面上。瞭解了程序之後，實際依畫面的 bitmap cache 資料和 memory blt 資料說明，一個 bmpcache 定義了參數結構如下:

```
uint8       cache_id;
uint8       Bpp;
uint8       width;
uint8       height;
unit16      buffersize;
uint8       cache_index;
variable length data;
```

根據定義的參數開啟大小為 width*height*Bpp 的資料空間，將 data 寫入隨著指標一筆筆寫入，利用 XLib 函數將圖繪出，根據 cache_id 和 cache_index 存入 bmpcache 中，底下列出圖 55.紅色框的實際暫存在 bmpcache 資料

```
cx=64,cy=29,id=2,idx=0,Bpp=2,buffersize=1592
cx=64,cy=29,id=2,idx=1,Bpp=2,buffersize=420
cx=64,cy=29,id=2,idx=2,Bpp=2,buffersize=123
cx=64,cy=29,id=2,idx=3,Bpp=2,buffersize=97
cx=64,cy=29,id=2,idx=4,Bpp=2,buffersize=116
cx=64,cy=29,id=2,idx=5,Bpp=2,buffersize=162
cx=64,cy=29,id=2,idx=6,Bpp=2,buffersize=1553
cx=64,cy=5,id=1,idx=0,Bpp=2,buffersize=207
cx=64,cy=5,id=1,idx=1,Bpp=2,buffersize=363
cx=64,cy=5,id=1,idx=2,Bpp=2,buffersize=269
cx=64,cy=5,id=1,idx=3,Bpp=2,buffersize=106
cx=64,cy=5,id=1,idx=4,Bpp=2,buffersize=308
cx=64,cy=5,id=1,idx=5,Bpp=2,buffersize=330
cx=64,cy=5,id=1,idx=6,Bpp=2,buffersize=54
cx=64,cy=64,id=2,idx=7,Bpp=2,buffersize=25
cx=64,cy=64,id=2,idx=8,Bpp=2,buffersize=1698
cx=64,cy=64,id=2,idx=9,Bpp=2,buffersize=5258
cx=64,cy=64,id=2,idx=10,Bpp=2,buffersize=4720
cx=64,cy=64,id=2,idx=11,Bpp=2,buffersize=3663
cx=64,cy=64,id=2,idx=12,Bpp=2,buffersize=3411
cx=64,cy=64,id=2,idx=13,Bpp=2,buffersize=1271
cx=64,cy=24,id=2,idx=14,Bpp=2,buffersize=1574
cx=64,cy=24,id=2,idx=15,Bpp=2,buffersize=839
cx=64,cy=24,id=2,idx=16,Bpp=2,buffersize=1615
```

cx=64,cy=24,id=2,idx=17,Bpp=2,buffersize=1803

cx=64,cy=24,id=2,idx=18,Bpp=2,buffersize=685

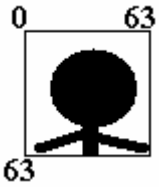
cx=64,cy=24,id=2,idx=19,Bpp=2,buffersize=1734

cx=64,cy=24,id=2,idx=20,Bpp=2,buffersize=757

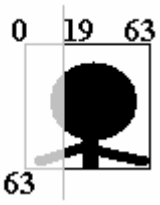
總共使用了 bmpcache1：7 個 bmpcache2：21 個，這部份共計有 28 個 bitmap 暫存在 client cache 中，瞭解 bitmap 暫存機制後，接下來看 memory blt 如何將資料取出並顯示在畫面上，首先看 memory blt 的參數結構

```
uint8 colour_table; //colormap
uint8 cache_id; // cache id
sint16 x; //圖的座標位置  X
sint16 y; //  圖的座標位置  Y
sint16 cx; //width
sint16 cy; //height
uint8 opcode; //ROP
sint16 srcx; //source X,複製圖片的 Block 內部起始位置 X
sint16 srcy; //source Y,複製圖片的 Block 內部起始位置 Y
uint16 cache_idx; // cache index
```

當 client 收到一個 memory blt，先利用 cache_id 和 cache_idx 將 bitmap 從 cache 中取出，之後根據 x 和 y 座標以及指定的長寬進行 ROP 操作將圖片顯示在指定的位置，這裡要特別說明 sourceX 和 sourceY。假設圖(a)是原本儲存於 cache 中的 64x64 bitmap，圖(b)是 bitmap 顯示於畫面的部份，灰色部份在螢幕上沒有顯示出來，則 memory blt 這部份的（sourceX,sourceY）=（19,0），就是說複製的 bitmap 從 block 內座標（19,0）開始，在一般情形下 block 沒有被切割，（sourceX,sourceY）都是（0,0）。



圖(a).儲存於 bmpcache 中 64x64 的 bitmap



圖(b).黑色為顯示於畫面的 bitmap 從 block 座標(19,0)開始

圖 60. memory blt 中 sourceX 和 sourceY 參數說明

瞭解了 memory blt 操作方式後，接下來將圖 59.紅色框線部分實際接收到的 memory blt 資料列出如下，

op=0xcc,x=191,y=87,cx=64,cy=29,id=2,idx=0
op=0xcc,x=255,y=87,cx=64,cy=29,id=2,idx=1
op=0xcc,x=319,y=87,cx=64,cy=29,id=2,idx=2
op=0xcc,x=383,y=87,cx=64,cy=29,id=2,idx=3
op=0xcc,x=447,y=87,cx=64,cy=29,id=2,idx=4
op=0xcc,x=511,y=87,cx=64,cy=29,id=2,idx=5
op=0xcc,x=575,y=87,cx=33,cy=29,id=2,idx=6
op=0xcc,x=194,y=204,cx=63,cy=5,id=1,idx=0
op=0xcc,x=257,y=204,cx=64,cy=5,id=1,idx=1
op=0xcc,x=321,y=204,cx=64,cy=5,id=1,idx=2
op=0xcc,x=385,y=204,cx=64,cy=5,id=1,idx=3
op=0xcc,x=449,y=204,cx=64,cy=5,id=1,idx=4
op=0xcc,x=513,y=204,cx=64,cy=5,id=1,idx=5
op=0xcc,x=577,y=204,cx=28,cy=5,id=1,idx=6
op=0xcc,x=194,y=116,cx=64,cy=64,id=2,idx=7
op=0xcc,x=258,y=116,cx=64,cy=64,id=2,idx=8
op=0xcc,x=322,y=116,cx=64,cy=64,id=2,idx=9
op=0xcc,x=386,y=116,cx=64,cy=64,id=2,idx=10
op=0xcc,x=450,y=116,cx=64,cy=64,id=2,idx=11
op=0xcc,x=514,y=116,cx=64,cy=64,id=2,idx=12
op=0xcc,x=578,y=116,cx=27,cy=64,id=2,idx=13
op=0xcc,x=194,y=180,cx=64,cy=24,id=2,idx=14
op=0xcc,x=258,y=180,cx=64,cy=24,id=2,idx=15
op=0xcc,x=322,y=180,cx=64,cy=24,id=2,idx=16
op=0xcc,x=386,y=180,cx=64,cy=24,id=2,idx=17
op=0xcc,x=450,y=180,cx=64,cy=24,id=2,idx=18
op=0xcc,x=514,y=180,cx=64,cy=24,id=2,idx=19
op=0xcc,x=578,y=180,cx=27,cy=24,id=2,idx=20

由 memory blt 和 bmpcache 兩筆資料可以看出，兩邊的 bitmap cache_id 和 cache_idx 完全對應。瞭解了 bitmap 的更新方式後，接下來說明文字的更新部份。第四章提到文字的更新部份是利用 fontcache 和 text order 來更新，我們先說明文字的暫存處理，一個 fontcache order 定義了以下參數結構，

uint8 font; //字型
uint8 nglyphs; //文字數目，glyph 數目
uint16 character; //fontcache 的 index
uint16 offset;

uint16 baseline; //定義文字是否為 baseline 型態

uint16 width; //文字的寬

uint16 height; //文字的高

variable length data;

當 client 收到 fontcache order 後，根據 width、height 及 data 指標，將 glyph 繪製出來，這裡的 glyph 就是點陣字型，一個文字都根據 font 和 character 分別儲存於 fontcache 中，這樣的程序為處理一個 glyph 的流程，一個 fontcache 處理 glyph 次數由 nglyph 參數決定，nglyph 就是一個 fontcache order 所傳送的文字數目。底下我們可以看到圖 59.實際傳送的 fontcache 資料。

FONTCACHE(font=6,n=1)　　//一個 fontcache order

font= 6,height=1 width=1,character=0//一個文字的長寬以及儲存在 fontcache 的位置

FONTCACHE(font=6,n=5)

font= 6,height=11 width=11,character=1

font= 6,height=11 width=10,character=2

font= 6,height=11 width=11,character=3

font= 6,height=11 width=10 glyph=5,character=4

font= 6,height=11 width=11 glyph=5,character=5

FONTCACHE(font=6,n=1)

font= 6,height=12 width=4 glyph=1,character=6

FONTCACHE(font=6,n=1)

font= 6,height=12 width=7 glyph=1,character=7

FONTCACHE(font=6,n=2)

font= 6,height=12 width=3 glyph=2,character=8

font= 6,height=12 width=2 glyph=2,character=9

FONTCACHE(font=6,n=2)

font= 6,height=11 width=11 glyph=2,character=10

font= 6,height=11 width=11 glyph=2,character=11

FONTCACHE(font=6,n=1)

font= 6,height=12 width=5 glyph=1,character=12

說明了 glyph 如何儲存於 fontcache 後，接下來說明 RDP Server 如何利用 text order 將文字從 fontcache 取出，並顯示在畫面上。一個 text order 的參數結構包含以下，

uint8 font;　　/*協議的 font id*/

uint8 flags;　　/*文字型態分為兩種，IMPLICIT 和 VERTICAL*/

uint8 opcode;　　/*ROP，實際上沒用到*/

uint8 mixmode;　　/*文字和背景混模式，分為透明和不透明*/

uint32 bgcolour;　　/*背景顏色*/

uint32 fgcolour;　　/*前景顏色*/

sint16 clipleft;　　/*文字區塊座標*/

```
sint16 cliptop;       /*文字區塊座標*/
sint16 clipright;     /*文字區塊座標*/
sint16 clipbottom;    /*文字區塊座標*/
sint16 boxleft;       /*通常為 0*/
sint16 boxtop;        /*通常為 0*/
sint16 boxright;      /*通常為 0*/
sint16 boxbottom;     /*通常為 0*/
BRUSH brush;          /*Brush 結構*/
sint16 x;             /*文字的座標位置 X*/
sint16 y;             /*文字的座標位置 Y*/
uint8 length;         /*一個 text order 的長度*/
uint8 text[MAX_TEXT];/*文字的代號*/
```

當 RDP Client 收到 text order 之後，根據 font 和 text[i]將文字從 fontcache 取出，透過 Xlib 函數將文字顯示在指定位置（x,y），一個 text order 可以取出多個文字，文字的數目由 length 參數決定，但是一個 text order 根據 font 參數只能從同一個 fontcache 中取出文字，如果是不同的 font 參數，必須傳送另一個 text order，以下列出圖 59.中，實際接收到的個 text order 資料結構，

```
x=203,y=241,cl=203,ct=231,cr=282,cb=243,bg=0xef5b,fg=0x0,font=6,fl=0x2,mix=1,n=2
x=203,y=241,cl=203,ct=231,cr=263,cb=243,bg=0xef5b,fg=0x0,font=6,fl=0x2,mix=0,n=13
x=263,y=241,cl=263,ct=231,cr=267,cb=243,bg=0xef5b,fg=0x0,font=6,fl=0x2,mix=0,n=2
x=267,y=241,cl=267,ct=231,cr=268,cb=243,bg=0xef5b,fg=0x0,font=6,fl=0x2,mix=0,n=2
x=267,y=241,cl=267,ct=231,cr=275,cb=243,bg=0xef5b,fg=0x0,font=6,fl=0x2,mix=0,n=2
x=275,y=241,cl=275,ct=231,cr=282,cb=243,bg=0xef5b,fg=0x0,font=6,fl=0x2,mix=0,n=4
x=203,y=270,cl=203,ct=260,cr=244,cb=272,bg=0xef5b,fg=0x0,font=6,fl=0x2,mix=1,n=2
x=203,y=270,cl=203,ct=260,cr=227,cb=272,bg=0xef5b,fg=0x0,font=6,fl=0x2,mix=0,n=4
x=227,y=270,cl=227,ct=260,cr=231,cb=272,bg=0xef5b,fg=0x0,font=6,fl=0x2,mix=0,n=2
x=231,y=270,cl=231,ct=260,cr=232,cb=272,bg=0xef5b,fg=0x0,font=6,fl=0x2,mix=0,n=2
x=231,y=270,cl=231,ct=260,cr=237,cb=272,bg=0xef5b,fg=0x0,font=6,fl=0x2,mix=0,n=2
x=237,y=270,cl=237,ct=260,cr=244,cb=272,bg=0xef5b,fg=0x0,font=6,fl=0x2,mix=0,n=4
```

n 為 length，bg 代表背景，fg 代表前景，由於 operation code 沒有用到，因此這裡不列出來，另外 boxtop、boxright 等座標都為 0，這裡也省略。

畫面中文字和 bitmap 的更新方式，主要都是利用 secondary order 傳送資料到 client 暫存，primary order 將文字或是點陣圖取出，這樣的方式進行資料的傳送與更新，但是文字和 bitmap 還是有點差異，在於 bitmapcache 和 memory order 一次只傳送一筆資料，而 fontcache 和 text order 一次可以傳送一個或是多個文字，說明了 text 和 bitmap 的更新方式後，接下來的章節進行資料量的分析。

5.3 桌面資料量分析

在遠端服務視窗系統中，最常使用的當然就是終端伺服器的桌面了，這節討論桌面畫面的組成，由於 rdesktop 的桌面連線，在連線前的預設值便設定不傳送遠端 RDP Server 的桌布背景，因此我們的實驗環境桌布背景部份是以白色背景取代，一般的 RDP 連線後的桌面如下圖，可以看出白色的背景佔了畫面很大的比例，其餘是桌面圖示以及說明文字部分。

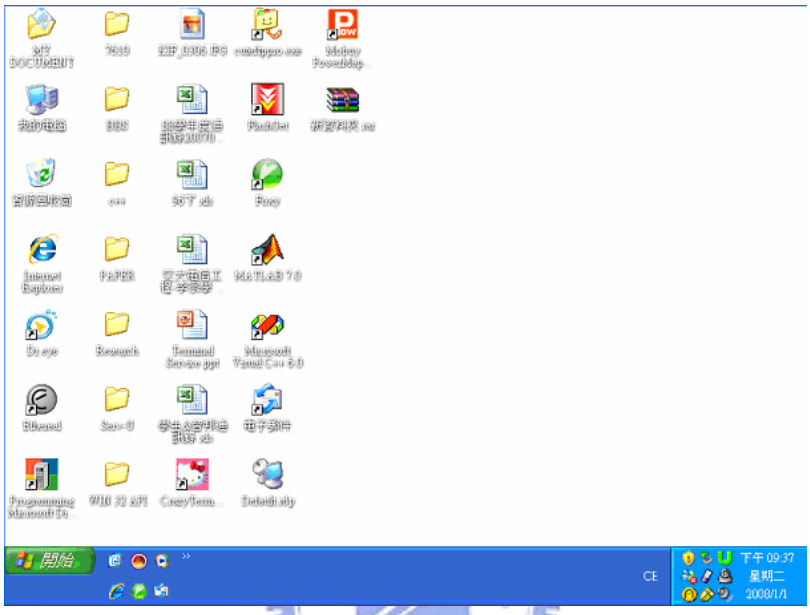


圖 61.(a) 一般 RDP 連線的桌面

利用修改後的 rdesktop 連線可看出畫面的命令組成如下圖，其中綠色部分為 rect order，紅色字體為 text order，點陣圖部分則維持不變。

圖 61.(b) 利用修改後 rdesktop 連線的遠端桌面

透過圖 61.我們了解到遠端桌面中，RDP Server 在背景部份運用了 rect order 傳送畫面，可以節省大量的頻寬使用，文字部分以 text order 方式傳送不僅保留了文字清晰度也因為傳送 1 bpp 的點陣字型而節省資料量，假設我們不是使用 RDP 連線來傳送畫面，那麼一個未經壓縮的 800x600 桌面以 16 bpp 點陣圖來計算其資料量高達 960K Bytes，其資料量比較如表 16.。在桌面的傳輸上，bitmap 的傳送分為兩種，桌面圖示 ico 由 bitmap updates 指令所傳送的，這部份的 bitmap 是沒有暫存在 client 的 bitmap cache。其餘的 bitmap 部份（例如：視窗下的工作列）則是透過 Secondary order 傳送 bitmap 資料到 Client cache 暫存後，RDP Server 接著傳送 memory blt 將 bitmap 取出，因此這部份的 bitmap 資料會暫存在 Client 的 bitmap cache 中。

表 16. RDP 連線的桌面資料量整理

	傳輸資料量
直接傳送 800x600 16 bpp 桌面資料	960K Bytes
rdesktop 接收 RDP 封包資料量	121.23K Bytes
節省的資料量	838.67K Bytes
資料壓縮率	12.63%

瞭解 RDP 所節省的資料量後，我們針對每個指令所傳送的資料量做個分析，計算一個畫面中，主要的傳輸資料量，另外可以看到 RDP 利用哪些機制節省畫面的資料量，其中，Sec、MCS、ISO 三層的資料量分析，為計算三層的標頭檔資料量乘以接收到的 RDP 封包數目，其資料量分析如下表。

表 17. RDP 連線的桌面命令分析

傳送的命令類別	傳輸資料量(KBytes)	佔傳輸比例	傳送次數
Bitmap cache (header and data)	69.83	57.67%	70
Bitmap updates(header and data)	36.08	29.8%	45
Sec、MCS、ISO、RDP header	8.27	6.83%	41
Text order	3.2	2.64%	95
Memory blt	2.1	1.73%	126
Font cache (header and data)	1.5	1.24%	41
Pattern blt	0.1	0.08%	5
Rectangle order	0.01	< 0.01%	2
Total transmit data size	121.09	99.99%	

由上表可以看出主要傳輸量還是來自於 Bitmap 的部份佔了 87%以上，接下來我們就 bitmap 的部份進行說明，在第四章我們有提到 RDP 的 bitmap 編碼原則，採用一種類似於 RLE 的非失真壓縮方式，下表說明 RDP 利用圖形壓縮及傳送命令的方式，透過這些原則所節省的資料量。

表 18. RDP 連線桌面資料壓縮

傳送的命令	壓縮方式	原始資料量	實際傳輸資料量	資料壓縮率
Bitmap cache	圖形編碼	221.75K Bytes	69.83K Bytes	31.49%
Bitmap updates	圖形編碼	69.06K Byes	36.08K Bytes	52.24%
Rectangle order	參數指令	844K Bytes (800x566 16 bpp)	12 Bytes	<< 0.01%

接下來，針對接受到的 41 個 RDP 封包中，其 RDP order 的詳細資料進行統計，下表為實際上接收到的 order 數目及順序。

表 19. RDP 連線遠端桌面資料統計

Order	RDP Packet Sequence
R、B+M、B+M、F+T、F+T、F+T、B+M、F+T	1
B+M、F+T、B+M、B+M	2
BU、BU、BU	3
F+T、F+T、F+T、F+TT	4
BU、BU、BU	5
B+M、F+TT、F+T、TTT	6
B+M、F+T、F+TTT、M、B+M、B+M	7
BU、BU、BU	8
B+M、B+M、MM	9
F+T、F+TT、M、B+M、B+M、F+T、F+TT、M	10
BU、BU、BU、BU	11
T、F+T、F+T、F+TT、F+T、M、B+M、B+M、F+T、F+TTT、MMM、	12
B+M、B+M、F+T、F+T、F+TT、F+T、M	13
BU、BU、BU、BU	14
F+TT、M、B+MT、M、B+M、TTT	15
B+M、B+M、B+M、B+M、B+M、TTT	16
F+T、M、B+M、F+T、M、B+M、TTT	17
TTT、M、B+M、B+M、F+TT、MMM	18
BU、BU、BU、BU	20
B+M、B+M、F+TTT、F+TT、B+M、F+T、F+TTT	21
B+M、B+M、T、F+T、F+TTT、B+M、TTT、B+M、TT、B+M、TTT	22
BU、BU、BU、BU	23
B+M、F+T、B+M、T、B+M、B+M	24
F+T、B+M、TTT	25

Order	RDP Packet Sequence
B+M、B+M、B+M、B+M	26
B+M、MM、B+M、M、B+M、MM、B+M	27
BU、BU、BU、BU、BU	28
B+M、MMM、B+M、B+M、B+M	29
B+M、B+M、B+M、M+M	30
BU、BU、BU、BU	31
B+M、B+M、TTT、MMM	32
B+M、B+M、B+M、MMM	33
BU、BU、BU	34
B+M、B+M、MMM	35
BU、BU、BU、BU	36
B+M、MMM、B+M、B+M、B+M	37
B+M、MM、MMM、	38
FT、M、B+M、B+M、F+TTT、MM、	39
BU、BU、BU、BU	40
M、M、M、M、M、B+M	41

在實驗進行過程中，當我們桌面中的 ico 圖示越少，傳輸的 bitmap 資料量就越少，當然節省的資料量也就越多，在 RDP 連線中最節省資料量的方式，還是依賴參數指令，交由 RDP Client 進行繪圖。

5.4 文書處理資料量統計

在 Windows 視窗系統中，主要的文書處理軟體為：Microsoft Word 和 NotePad，我們分別就這兩樣應用程式進行畫面的剖析和資料量的比較。

5.4.1 Microsoft Word

我們開啟一個存檔的 word 檔案，字型大小為 12 pt，字數約為 300 字，一般的連線畫面，如下頁圖 62。

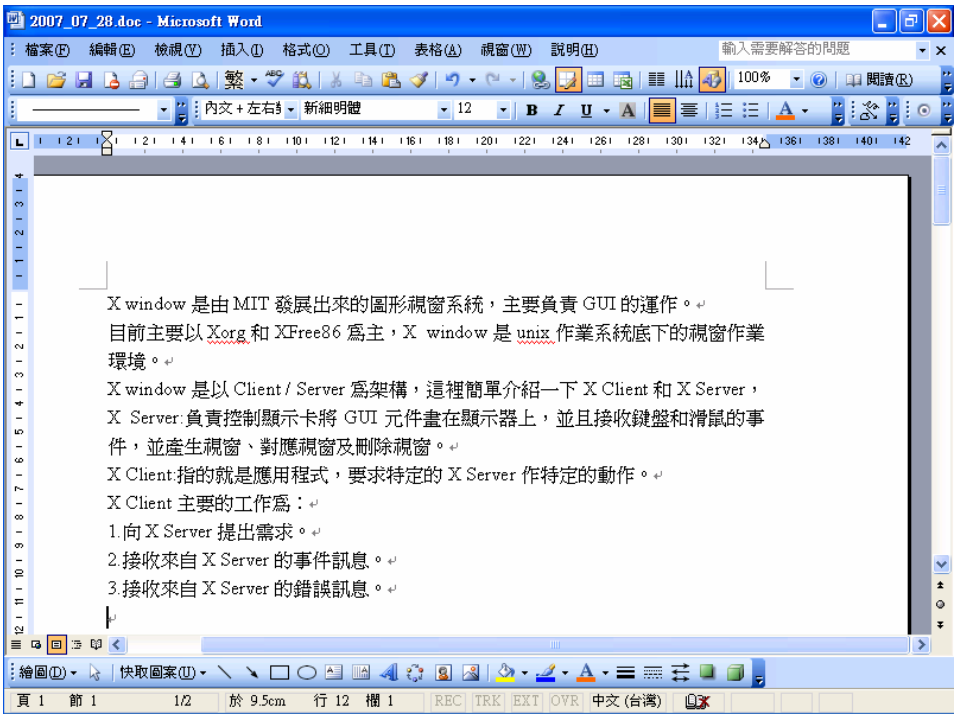


圖 62. 一般 RDP 連線開啟 Word

下圖由 rdesktop 進行連線，可以看出開啟後的檔案文字部分也是以點陣圖方式傳送，除了一些視窗的內框以 rect order 傳送將顯示為綠色矩形，以及部份選單中的文字以 text order 傳送顯示為紅色字體外，大多數的畫面以 bitmap 顯示，這部份用 memory blt 將暫存 bitmap 依座標位置貼於畫面上。

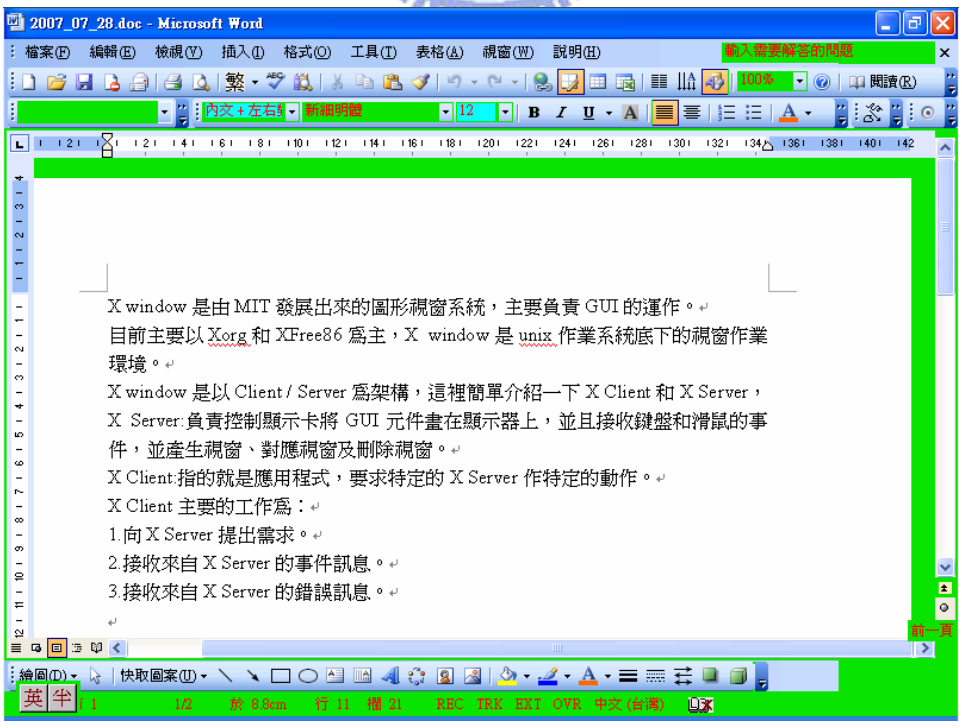


圖 63. 利用修改後 rdesktop 開啟遠端 Word

瞭解了 Word 畫面組成後，接下來進行資料量的比較，一般來說，Microsoft Word 的使用上除了開啟檔案的瞬間造成畫面資料的大幅更新，在進行文書處理時，畫面更新頻率取決文字輸入的速度，不論打字速度多快畫面的變化仍然有限，因此 Microsoft Word 的資料量擷取，我們測試的項目為檔案開啟瞬間至畫面更新完畢總共傳輸的資料量，資料量的比較如下表。

表 20. RDP 連線的 Word 資料量

	傳輸資料量
直接傳送 800x600 16bpp Microsoft Word 畫面	960K Bytes
rdesktop 接收 RDP 封包資料量	190K Bytes
節省的資料量	770K Bytes
資料壓縮率	19.8%

接下來分析畫面中各指令所佔傳輸資料量比例，在圖 63.可以看到畫面中少部份為矩形所繪製外，大部分的畫面都是 bitmap，Microsoft Word 畫面更新是比較特殊的，RDP 的兩種更新方式不斷互相更新，在檔案開啟後，Bitmap_updates 和 memory blt 重複更新畫面，更新的部份有重疊的情形，因此在資料量的分析上，我們以各命令（order）佔傳輸的比例來說明。

表 21. RDP 連線的 Word 命令分析

傳送的命令類別	傳輸資料量(KBytes)	佔傳輸比例
Bitmap cache (header and data)	89.9	47.40%
Bitmap updates(header and data)	76.4	40.28%
Sec、MCS、ISO、RDPlayer	3.5	1.85%
Text order	4.6	2.43%
Memory blt	5.8	3.06%
Font cache (header and data)	5.3	2.80%
Pattern blt	0.04	0.02%
Rectangle order	4.1	2.16%
Total transmit data size	189.64	100%

由於 Word 的畫面傳送，RDP Server 不斷傳送更新過來，其中 Bitmap cache 所傳送的 Bitmap 僅傳送一次，但是 Bitmap updates 的 Bitmap 仍然不斷傳送過來，這部份是由於 RDP 的特性為 Server Push，RDP Server 判斷畫面有更新便會傳送資料到 RDP Client，由於畫面有重複更新的情形，因此畫面的資料壓縮部份，僅說明傳送一次的 Bitmapcache 做分析，另外，一次完整畫面中相同的 bitmap 可以利用 memory blt 將 bitmap 從 cache 中取出，下表 22.為這兩項所節省的資料量。

表 22. RDP 連線桌面資料壓縮

傳送的命令	壓縮方式	原始資料量	實際傳輸資料量	壓縮率
Bitmap cache 傳送 bitmap	圖形編碼	831.84K Bytes	89.9K Bytes	10.87%
Memory blt 自 cache 取出 bitmap	Cache 暫存 & 參數指令	187.75K Bytes	0.74K Bytes (僅計算 Memory Blt 資料量)	0.38%

雖然 RDP 連線所傳送的 Word 畫面大部分以 bitmap 呈現,但是 RDP 使用了 Cache,因此畫面中背景部份，RDP Server 只要傳送 bitmap cache 將點陣圖暫存於 Client 的 cache 中，之後的相同的 bitmap 選擇傳送 memory blt ，依據不同座標重複顯示，由於傳送命令的資料量比直接傳送 bitmap 資料量小很多，因此可以降低連線的資料量，下表 23.為開啟 Word 畫面時，使用的存放 bitmap 的 cache 數量。

表 23. 開啟 Word 畫面所使用的 cache 數量

Cache id(Max bitmap size)	開啟 word 畫面使用的 cache 數量
16x16	173
32x32	3
64x64	3

5.4.2 Notepad

RDP 對於 Notepad 的畫面傳送方式不同於 Microsoft Word，在 Notepad 開啟副檔名為.txt 檔案後，RDP Server 將畫面中的文字以 text order 傳送，背景部份則是以 rect order 傳送，文字及背景的傳送方式和 Microsoft Word 有明顯差異，圖 64.為一般 RDP 連線開啟遠端 Notepad 畫面。

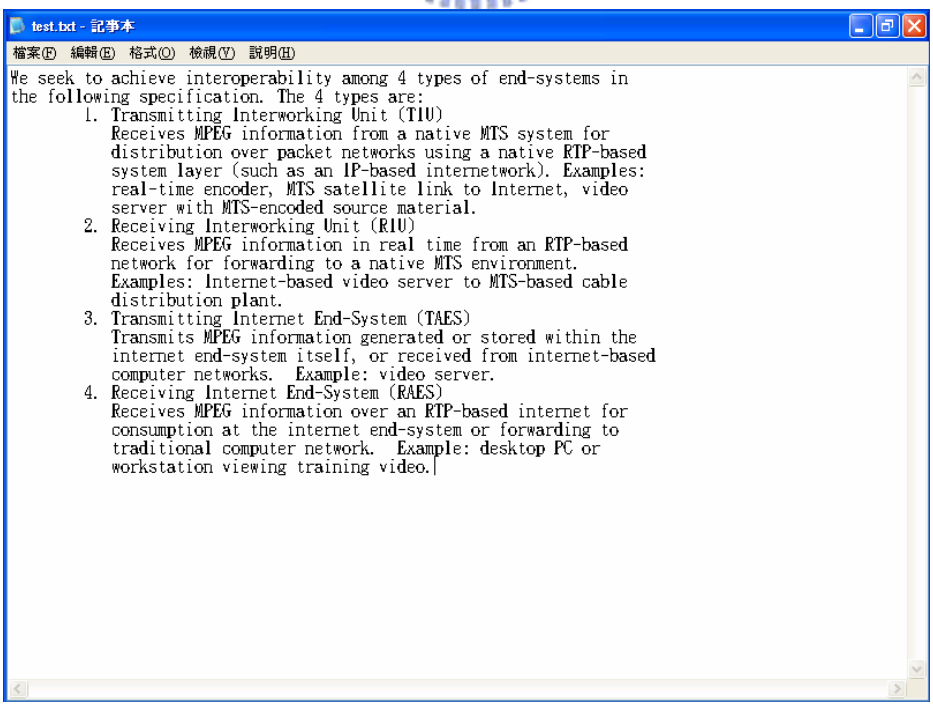


圖 64. 一般 RDP 連線開啟遠端 Notepad

下圖為利用修改後的 rdesktop 進行連線，可以看出文字部分以 text order 傳送，除了視窗外框，檔案中的背景，綠色部分以 rect order 傳送，另外在選項中的文字底線為藍色表示以 line order 傳送，文字前景設定為紅色，文字背景則設為淺藍色。

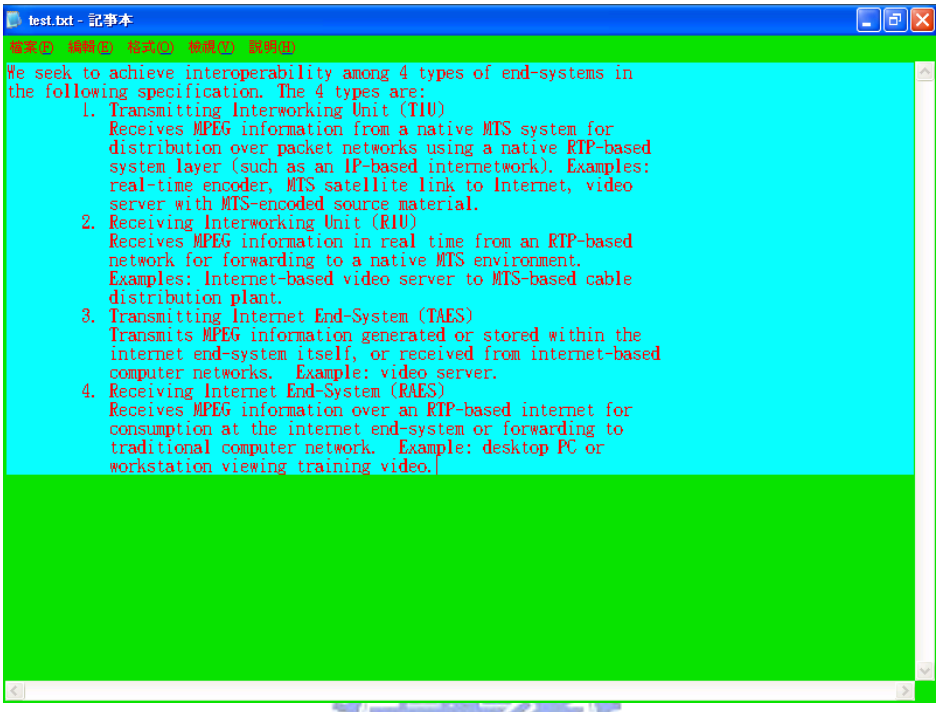


圖 65. 利用修改後 rdesktop 開啟遠端 Notepad

如同前幾節一個完整 800x600 的畫面，假設以 16 bpp bitmap 來傳送，那麼其資料量為 960K Bytes，相較於 Microsoft Word 的畫面複雜，Notepad 的畫面結構顯得檢單許多，由上圖可以看出檔案開啟後，除了視窗的外框使用 bitmap 方式傳送外，檔案內部畫面沒有 bitmap 的部份，完全是以 text order 以及 rect order 組成，因此可以想像 Notepad 開啟畫面的資料量應該不多，經過 rdesktop 連線擷取結果，當我們利用 RDP 傳送過來的資料量比較如下表 24.，實驗結果證明 RDP 的圖形壓縮方式非常適合我們應用在 thin client 的終端服務上面，利用這樣的壓縮方式，文字部分保留的清晰度，而且節省了資料量，如果僅以 800x600 16bpp 的畫面來看，傳輸量不到原來的百分之五，非常適合用於電子看板終端服務上。

表 24. RDP 連線的 Notepad 資料量

	傳輸資料量
直接傳送 800x600 16bpp Notepad 畫面	960K Bytes
rdesktop 接收 RDP 封包資料量	33.65K Bytes
節省的資料量	926.35K Bytes
資料壓縮率	3.5%

由圖 65.可以看出畫面中的組成命令為 rectangle order 和文字組成，利用這樣的方式大幅降低了畫面的資料量，由指令的資料來看，畫面的組成為背景部份填入一個 788x538 pixels 的矩形之後將文字填上去。

表 25. RDP 連線的 Notepad 命令分析

傳送的命令類別	傳輸資料量(KBytes)	佔傳輸比例
Bitmap cache (header and data)	9.61	29.33%
Bitmap updates(header and data)	9.7	29.6%
Sec、MCS、ISO、RDPlayer	0.81	2.47%
Text order	7.48	22.8%
Memory blt	2.8	8.5%
Font cache (header and data)	1.43	4.4%
Pattern blt	0.12	0.4%
Rectangle order	0.82	2.5%
Total transmit data size	32.77	100%

5.5 網頁畫面數據統計

網頁也是視窗系統中常見的畫面，這節我們分別對開啟交大首頁(http://www.nctu.edu.tw）及 google 網頁(http://www.google.com.tw)的畫面進行分析，網頁的內容如果圖片越多其資料量就越多，如果網頁中有動畫的內容，其資料量更是不斷的傳送更新，在網路的使用上，我們常常會開啟網路搜尋資料（例如：google），查完資料關閉視窗後不久又開啟網頁，相同的畫面又再度出現。因此在網頁的實驗中，我們加入了暫存於 cache 的資料量，由於 RDP 使用了 cache 的機制，RDP Server 對於畫面中的 bitmap 給予 cache id 和 cache index，將 RDP Server 傳送過來的 bitmap 存放於 Client 的 bitmap cache 中，也就是說對於 bitmap 暫存的機制是由 RDP Server 所管理的。對於 bitmap cache 資料量的擷取方式，是修改 order.c 中的 process_bmpcache2 函數，將傳送過來的 bitmap 大小記錄下來，待畫面更新完畢後，進行資料量的統計。

5.5.1 開啟交大首頁

一般 RDP 連線利用 Internet Explorer 開啟交大首頁，其內容如下圖 66.，可以看到網頁中圖片多，背景顏色豐富。觀察了一般 RDP 連線開啟交大首頁的畫面後，接下來我們利用修改後的 rdesktop 連線 RDP Server，並開啟相同網頁觀察畫面組成，其中紅色字體為 RDP Server 以 text order 傳送，顏色沒有改變的文字 RDP Server 則是以 bitmap 方式傳送過來，綠色部分則是以 rect order 方式傳送，另外，在畫面右邊的文字藍色底線則是以 line order 傳送參數由 RDP Client 繪製，其結果如下頁圖 67.。

圖 66. 一般 RDP 連線 RDP Server 開啟交大首頁

圖 67. 利用修改的 rdesktop 連線 RDP Server 開啟交大首頁

了解網頁畫面組成後，接下來進行資料量的分析，這裡我們分別對同一次連線，初次開啟網頁與第二次開啟網頁後的資料量比較，可以看出第二次連線因為 bitmap cache 已經將畫面暫存，因此傳輸的資料量降低很多，連線數據整理如表 26.。

表 26. RDP 連線開啟交大首頁資料量

	傳輸資料量	與 800x600 bitmap 比較減少的資料量	節省資料比率
800x600 16 bpp bitmap	960K Bytes	-	-
初次開啟網頁	354.4K Bytes	605.6K Bytes	36.94%
暫存於 Bitmap cache 的資料量	262.9K Bytes	-	-
第二次開啟網頁	151.6K Bytes	808.4K Bytes	15.8%

5.5.2 開啟 google 網頁

google 網頁的畫面明顯比交大首頁簡單，除了從畫面上直覺的判斷外，客觀的數據也可以看出資料量較小，下頁圖 68.為一般 RDP 連線透過 Internet Explorer 開啟 google 網頁的畫面。下頁圖 69.則為利用修改的 rdesktop 連線開啟 google 網頁的畫面，文字部分以 text order 傳送呈現出紅色字體，文字的底線也以 line order 傳送，最明顯的是網頁空白背景部份使用了 rect order 進行繪製，大幅降低了傳輸的資料量。

圖 68. 一般 RDP 連線開啟 google 網頁

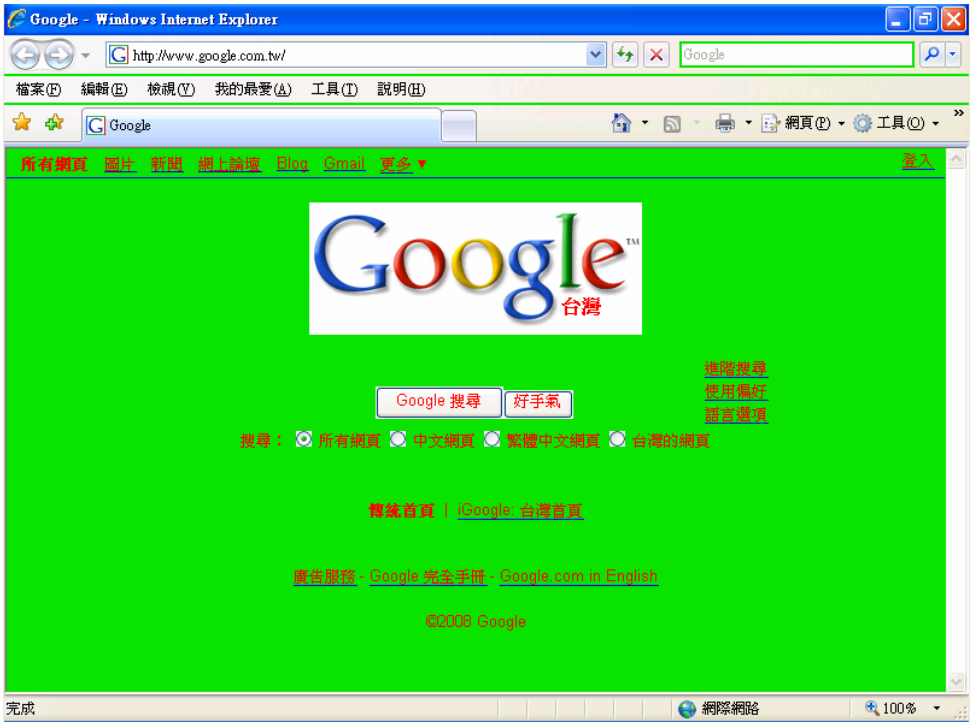


圖 69. 利用修改 rdesktop 連線 RDP Server 開啟 google 網頁

由於 google 的網頁簡單，沒有太多的圖片，因此 RDP Server 使用 rect order 傳送參數交由 RDP Client 繪製背景，表 27.整理出傳輸的資料量。

表 27. RDP 連線開啟 google 網頁資料量

	傳輸資料量	與 800x600 bitmap 比較減少的資料量	資料壓縮率
800x600 16 bpp bitmap	960K Bytes	-	-
初次開啟網頁	133.84K Bytes	826.16K Bytes	13.9%
暫存於 Bitmap cache 的資料量	74.57K Bytes	-	-
第二次開啟網頁	79.04K Bytes	880.96K Bytes	8.23%

由這兩個不同的網頁可以觀察出，網頁色彩越豐富、圖片越多，RDP Server 傳送的 bitmap 資料就越多，相對的傳輸的資料量也越多，如果網頁中有動畫或是影片資料量更會不斷傳送，另外，在同一連線中開啟相同的網頁，由於 bitmap cache 中已經暫存了 bitmap 資料，因此，我們將 Internet Explorer 關閉後，再次開啟相同網頁，僅需要傳送沒有暫存的畫面訊息，對於已經暫存的 bitmap 只需要傳用 memory blt 將資料從 bitmap cache 中取出，大幅降低傳輸資料量，但是當 RDP 連線中斷後 cache 的資料便清除。

5.6 Microsoft PowerPoint 畫面資料統計

Microsoft PowerPoint 是視窗作業系統中常使用的應用程式之一，視窗畫面中含大量的文字、圖片，這一節將進行 PowerPoint 的畫面及傳輸的資料量分析，透過兩個項

目來進行，分別是檔案開啟的畫面分析及投影片播放的畫面分析，下圖為一般的 RDP 連線所開啟檔案畫面。

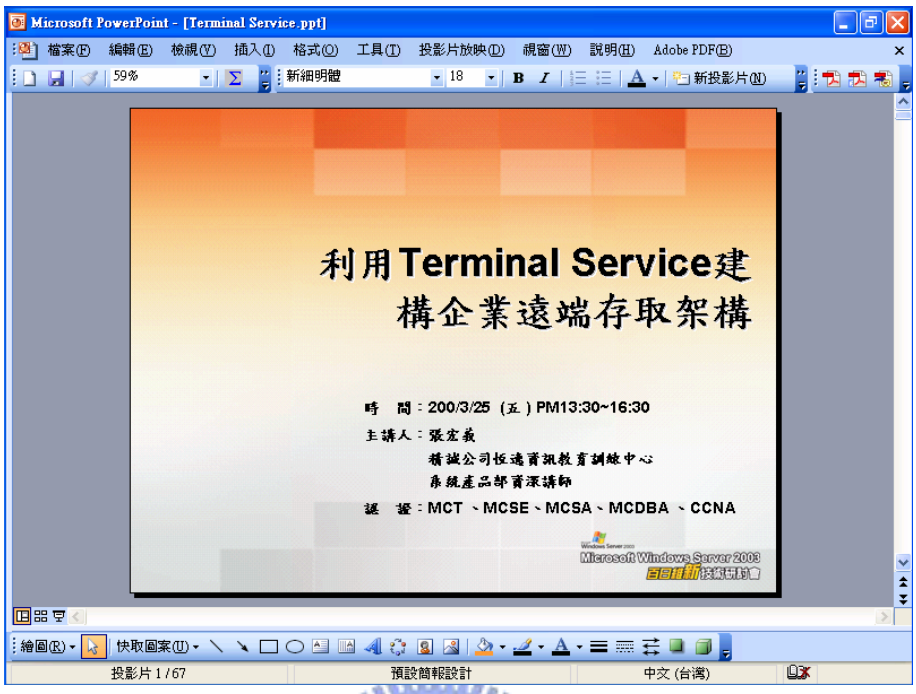


圖 70. 一般 RDP 連線開啟 PowerPoint

圖 71.為修改 rdesktop 程式碼後，連線至遠端伺服器（RDP Server）顯示出來的 PowerPoint 畫面，可以看出視窗內的內框用了 rect order，檔案的文字部分以 text order 傳送，bitmap 的組成則佔了畫面的一半以上。

圖 71. 修改的 rdesktop 連線開啟遠端 PowerPoint

傳送一個 800x600 16 位元高彩畫面最簡單的方法就是以直接傳送完整 bitmap，其資料量為 960K Bytes，比較透過 RDP 連線所節省的資料量，整理如表 28.，可以看出透過RDP連線所傳送的資料量，大約只有直接傳送800x600 16 bpp點陣圖資料的一半。

表 28. RDP 連線開啟 PowerPoint 資料量

	傳輸資料量
直接傳送 800x600 16bpp PowerPoint 畫面	960K Bytes
rdesktop 接收 RDP 封包資料量	490.7K Bytes
節省的資料量	469.3K Bytes
資料壓縮率	51.11%

由 PowerPoint 開啟 PowerPoint 檔案和播放投影片，在 RDP 處理畫面上有很大的差異，開啟檔案時，RDP Server 會判斷文字和 bitmap 的畫面，利用不同的命令(order)傳送不同的畫面資料，但是播放投影片時，RDP Server 將螢幕畫面當做 bitmap 來傳送，對於重複出現的背景，將相同的 bitmap 暫存在 Client cache 中，之後出現相同的背景畫面，利用 memory blt 將資料從 cache 讀取出來依指定位置顯示；而不同的畫面則是傳送 bitamp 過來，也是播放投影片時主要傳輸的資料量。

表 29. RDP 連線播放 PowerPoint 投影片

項目	數據
直接傳送七張 800x600 16 bpp 畫面資料量	6720K Bytes
由 RDP 連線播放投影片總資料量	1853.8K Bytes
總共減少資料量	4866.2K Bytes
總資料壓縮率	27.58%
第一張投影片資料量	419.51K Bytes
第一張投影片資料壓縮率	43.7%
平均每張資料量	264.83K Bbytes

表 29.可以看出第一張投影片的資料量明顯高出平均資料量許多，這是因為投影片播放時，每一張的背景都一樣，因此相同的 bitmap 暫存到 bitmap cache 中，之後的投影片只需要使用 memory blt 依 RDP Server 指定的座標位置，將相同的圖從 cache 取出後顯示在畫面上，這部份的資料量將少的許多。

透過 RDP 連線這樣的圖形壓縮方式，在 PowerPoint 播放時，由於畫面背景一樣，相同的 bitmap 暫存在 Client 的 cache 中，藉由這樣的圖形壓縮傳送方式，可以大幅降低傳輸的資料量。

5.7 影片播放

RDP 在影片播放時，因為畫面不斷更新，造成資料量暴增畫面更新不順，根據同一影片變更畫面大小來進行數據的擷取。分別以影片原始大小 320x240、影片放大兩倍為 640x480 及 rdesktop 預設視窗大小 800x600 進行實驗，以下為實驗環境的說明：

rdesktop 連線視窗大小：800x600
bits per pixel：16 bpp
RDP Server：Microsoft Windows XP，RDP 5.1
RDP Client：rdesktop-1.5.0
影片原始大小：320x240
影片播放程式：Windows Media Player 11.0
影片播放長度：一分鐘

實驗步驟由下列方式進行：
一、利用 Server Windows Media Player 統計略過的 frame 數量以及 Server Media Player 每秒播放的畫面數量（fps），由這兩項參數比較終端電腦播放影片時的流暢度。
二、由 rdesktop 程式統計接收到的 frames 數量，接收到的 frames 越多，證明影片播放越流暢，由於接收到的畫面經過切割，最大的 bitmap 為 64x64 pixels，單純計算封包的方式無法算出 frames 數量，bitmap 畫面更新的座標位置固定，因此根據 memory blt 重複出現的座標位置定義出來，接下來紀錄該座標的 memory blt 出現次數，統計一分鐘出現的次數，紀錄為 rdesktop 一分鐘播放的 frames 數量。
三、從 rdesktop 程式統計收到的 RDP 封包資料量，現行的網路頻寬是否能夠應付 RDP 連線影片播放的資料量。
由這三方面進行具體的數據分析，由處理的資料量、遺失的 frame 數量來驗證 RDP 連線在影片播放時產生巨大資料量引起網路雍塞的問題。

表 30. RDP 連線遠端播放影片的資料量

影片大小	800x600	640x480 (兩倍大小)	320x240 (原始大小)
Server Windows Media player 統計略過 frame 數量	972 frames	706 frames	60 frames
Server Windows Media player 每秒播放 frame 數目	14.4 fps	18.3 fps	29.8 fps
由 Windows Media player 計算畫面遺失率	52%	39%	0.67%
由 rdesktop 接收到的 frame 數量	880 frames	1116 frames	1778 frames

影片大小	800x600	640x480 (兩倍大小)	320x240 (原始大小)
Client 端畫面繪製速率	14.67 fps	18.6 fps	29.63 fps
RDP Client 一分鐘 接收的資料量	197.46M Bytes	189.17M Bytes	170.61M Bytes
平均資料量(bits/second)	26.33Mbps	25.2Mbps	22.75Mbps

由實驗數據可以看出，影片畫面大小影響了 frame 略過的情形，畫面越大相對的資料量越大，在網路頻寬無法負荷的情況下，略過的 frame 數量也越多，由上表可以看出最小傳輸率都有 20Mbps 以上，在一般網路頻寬下，目前最快的 ADSL 僅達 12Mbps，完全無法支援 RDP 進行影片播放，因此，我們設想利用 RDP 連線進行影片播放時，在影片畫面部份進行編碼是未來的目標。

第六章 結論

6.1 結論

由本論文利用開放程式碼 rdesktop 進行 RDP（Remote Desktop Protocol）連線的畫面分析及資料量比較，rdesktop 是應用在 RDP 連線的 Client，主要建立在 Linux 作業系統上，本論文主要的實驗環境便是以 rdesktop 為架構進行修改與封包資料量的分析。

由於 RDP 的圖形編碼由指令來分類，藉由指令的傳送來傳輸畫面，對於文字部分採用 1 bpp 的點陣字型傳送，保留了文字變化的清晰度；對於背景單調的話用則使用 rectangle order 傳送，透過參數的傳送大幅降低 bitmap 的資料量；對於複雜的畫面則使用 bitmap 來傳遞；由於視窗圖形介面，許多視窗畫面會重複出現，RDP 建立了 Cache 來暫存點陣圖資料，往後相同的畫面出現，只需要傳遞命令來完成畫面的顯示。

由於無線網路的發展，網路上的終端服務將蓬勃發展，終端服務的客戶端可能是手機、PDA、PMP（Protable Media Player）等，這類的裝置可能沒有大容量的硬碟或是高效能的運算處理單元（CPU），因此藉由遠端連線至伺服器，進行資料運算最後將結果顯示在使用者端。終端服務的架構在使用者端必需負責輸入訊號，伺服器將輸入訊號做處理後，將結果利用網路回傳給使用者端，由使用者端顯示出最後的結果。由於使用者端的計算機往往是 thin client，因此回傳的顯示畫面必須保留文字的清晰度且降低資料量，本論文探討的 RDP 連線方式便符合這樣的要求。

在第四章藉由 rdesktop 連線，分析傳送畫面的組成，分析出畫面中資料量最多的是哪部分，哪些命令（order）又可以節省頻寬。由第五章透過資料量的分析，了解傳送的資料量及資料壓縮率，這部份我們發現了在影片播放時，由於畫面不斷更新，造成資料量暴增，每秒播放的影片 frame 隨著影片尺寸的增加而減少，因此 RDP 連線無法順利播放影片。除了影片播放，經過實驗數據的證明其他應用程式畫面經由 RDP 傳送後都可以有效降低資料量，利用 RDP 連線這樣的圖形壓縮方式適合於 thin client 架構。利用 rdesktop 開發環境，成功解析出畫面中所有指令的組成，對於 RDP 連線的畫面圖形命令傳送準確分析，關於遠端播放影片的位置也能根據 memory blt 傳送資訊中，由重複出現的座標位置，判斷播放 frame 的數目，同時順利取得更新區域。

由實驗的結果，我們根據 rdesktop 所觀察到的情形，對於 RDP Server 指令包裝方式提出以下的看法。

- RDP Client 呼叫繪圖函數在 Window 中呼叫 GDI，本論文實驗環境 rdesktop 在 linux 底下呼叫 X Library 函數，對於多邊形、線條等顯示方式，不論是 Microsoft Windows GDI 或是 Linux X Lib 都有相對應的函數。
- Server 端的應用程式為了顯示畫面呼叫 GDI，RDP Server 接收到 GDI 函數呼叫後，透過 RDP 封包格式，將原本由 Server 端繪製的畫面轉由 Client 繪製，也就是說原本 Server 端應用程式呼叫 Server 端 GDI 函數，在 RDP 建立連線後，Server 便把相對應的 GDI 函數資訊，傳送給 RDP Client。

- 圖形（例如：rectangle、line 等）命令的傳送，RDP Server 對於應用程式呼叫 GDI 圖形函數包裝成 Recgangle order、Line order 等 RDP order 格式，封裝在 RDP Packet 傳送到 Client。
- 文字部分，RDP Server 利用 GDI 的文字函數呼叫，判斷顯示部分是否為文字，如果是文字且背景單純則利用點陣字型傳送，如果背景複雜，為了避免文字邊緣高頻變化區失真，則採用 bitmap cache 和 memory order 傳送。
- 點陣圖（bitmap）部分，本論文前面提到 RDP 連線在畫面更新有兩種方式，1、bitmap updates 2、bitmapcache 配合 memory order。這兩種畫面更新方式差異點在於呼叫程式的差異，根據實驗結果提出這樣的看法，bitmap updates 是桌面視窗管理員（Desktop Window Manager，Windows Vista 之前的版本稱為 Desktop Compositing Engine）所使用的 bitmap 畫面更新，例如：桌面圖示、視窗的外框建立，這部份的點陣圖更新藉由 bitmap updates 傳送給 RDP client 進行更新。至於應用程式的畫面更新則是使用 bitmapcache 配合 memory order 來傳送點陣圖。

由以上幾點來看，RDP Server 的畫面更新與應用程式相關，原因在於應用程式呼叫 Windows GDI 函數，RDP Server 根據呼叫函數傳送相對應的 RDP order 到 Client 端，將顯示畫面的工作呼叫 Client 端的 GDI 來完成。

6.2 未來發展與現況

隨著無線網路的發展及 WiMAX 技術的突破，未來透過無線網路連線的終端服務，我們對於這項研究的方向分為兩個部份。

6.2.1 現在 Windows RDP Client

由於 RDP 的通訊協定掌握在微軟公司手上，現存的 RDP Client 特性依網路速度來分，最低速度為 28.8Kbps RDP Client 一樣支援點陣圖快取（bitmap cache），如果將 bitmap cache 取消掉傳送過來的 bitmap 將不暫存，每個畫面都需要重送，第二等級為 56Kbps 除了支援基本的 bitmap cache 外，還支援主題顯示，也就是說配合 RDP Server 的桌面主題，第三等級為網路速度在 128Kbps 到 1.5Mbps 之間，除了支援前面兩項外，還顯示功能表及視窗動畫，並在拖曳時顯示視窗內容以及桌面的緩衝處理，也就是說頻寬越大 RDP Client 可以傳送的資料量就越多，最後是 10Mbps 以上的網路頻寬，除了桌面背景外還提供字形的平滑處裡。

6.2.2 透過編碼技術播放影片

由於 RDP 連線目前仍無法播放影片，但是 RDP Server 為微軟所掌握，無法取得其相關資訊，因此希望藉由代理伺服器（Proxy）取得畫面資料後，利用已完成的畫面分析，將影片區塊區分出來，將 frame 交由 H.264 或是 MPEG2 編碼後，將該區塊的資料利用 RDP 連線傳送出去，如此每個 block 的資料量降低，傳輸資料量也相對減少，藉由這樣的方式可以有效降低影片播放的資料量。

參考文獻

[1] 吳宗修，「串流伺服器特性剖析」，國立交通大學，碩士論文，民國 96 年。

[2] http://www.rdesktop.org/

[3] T Richardson, et al., ”Virtual Networking Computing”, IEEE Internet Computing, vol.2, pp.33, 1998

[4] Tristan Richardson, ”The RFB Protocol”, ORL/AT&T Labs Cambridge,June,2008

[5] Robert W.Scheifler, ”W Window System Protcol”, Laboratory for Computer Science, MIT,1994

[6] Microsoft Corporation, ”Understanding the Remote Desktop Protocol”, March 27,2007

[7] Microsoft Corpoation, ”Remote Desktop Protocol”, MSDN Library, November 1,2007

[8] 宋卓翰，「Thin-client 應用呈現平牌的最佳化設計與實作」，國立交通大學，碩士論文，民國 95 年。

[9] 杜天倉等，「基於 Linux 的 RDP 客戶端設計」，微電子學與計算機，第 22 卷 11 期，127~130 頁，2005 年 11 月。

[10]ITU-T Recommendation T.128, ”Multipoint Appplicationi Sharing” February 1998

[11]Oliver Jones 著，Introduction to the X Window System，黃豐隆譯，松崗電腦圖書資料股份有限公司，民國 82 年，台北

作者簡歷

李宗學，於民國七十年二月出生於台灣省桃園縣。民國九十二年六月畢業於中華大學電機系，同年八月入伍，軍種為憲兵。民國九十四年二月退伍，同年三月進入南亞科技擔任工程師。民國九十五年三月進入交通大學通訊與產業研發碩士班就讀。興趣為區域網路及影像串流，民國九十七年一月取得碩士學位，碩士論文題目為「終端服務的桌面影像壓縮」。