國立交通大學

資訊科學與工程研究所

碩 士 論 文

Thin-client 應用呈現平台的最佳化設計與實作

Presentation Platform Optimization for Thin-client Applications

研 究 生：宋卓翰

指導教授：黃世昆 教授

中 華 民 國 九 十 五 年 六 月

摘　　要

爲了協助使用者更容易製作自己的內容加以分享，我們使用內建於 Windows XP 的遠端桌面傳輸協定(Remote Desktop Protocol, RDP)取得視窗桌面影像並分享給多個使用者。Windows XP 遠端桌面是 Thin Client / Server 系統之一，它的通訊協定 RDP 比起其它的系統在效能及資料量上都非常優秀。因爲它除了單純地將伺服器的桌面影像傳輸至遠端電腦上之外，還提供了圖形/文字快取的機制來降低資料量。

然而 RDP 尚有的一些特性讓它不利於直接使用於桌面分享。除了僅支援一對一的連線外，在處理複雜圖形影像時，RDP 的圖形壓縮率並不理想，在使用者瀏覽圖文併茂的網頁時產生可觀的資料量。這些都降低了使用者透過低頻寬的網際網路分享桌面的動力。

爲了解決此問題，我們根基於現有的 RDP proxy：xrdp 在伺服器與客戶端之間建立 proxy 機制，負責將自伺服器傳來的桌面影像傳輸給不同的使用者。爲了能進一步減少資料量，以符合網際網路的環境，我們針對以下三個項目進行處理：彈性壓縮圖形；預防資料暴增；處理資料暴增。

利用 proxy 搭配以上的方法，使用者將能輕易地透過網際網路傳輸自己的桌面影像，與他人分享真正屬於自己的數位內容。

Abstract

To help average users create their own contents, we propose the use the Remote Desktop Protocol (RDP) for broadcasting desktop screens to other people. RDP is one of the thin client / server protocols build in Windows XP systems, and provides better performance then other thin client systems, since it leverages bitmap and font caches to lower the data transfer rate.

Beside the good performance that RDP provides, there are still several situations that RDP might generate lots of data. For example, when dealing with complex bitmaps on web pages or in video, the lossless compression used by RDP performs badly, and the compress rate is extremely low which results in high data rates that makes it difficult for desktop sharing over the internet.

In our thesis, we modify the existing open source RDP proxy XRDP and use several ways to generate suitable data rates for the Internet.

目　　錄

表 格 目 錄

圖 片 目 錄

第 1 章 動機與目標

1-1 動機

網際網路的普及化，使得每一個家庭、甚至每個人花費越來越多的時間在網路上瀏覽、交友。網路的快速打破了國界的限制，而越來越多的網站提供使用者自己的空間，可以上傳圖片或影片與大眾分享。比較著名的 Google Video 與 youtube.com，標榜著 *Broadcast yourself*，提供一般大眾在網路上呈現自我的方式。然而無論是個人相片或影片都需要額外的硬體支援，並非所有的使用者都有這些設備。

我們覺得目前網路分享所欠缺，同時也是最簡單直覺的，就是分享個人的桌面。使用者可以透過分享桌面進行教學、照片分享、文件分享等等，比起其它靜態的分享空間，分享者可以主動提供導讀，引導參與者更能體會其中的精華，使「分享」更具有意義。

1-2 問題描述

分享桌面的方式很多，其中最直覺的方式是擷取全部的螢幕畫面並儲存爲影片檔或藉由串流散佈至網路上。這種方式有許多缺點，其中之一爲資料量非常龐大，以一個 1027*768 的全彩畫面(24bit)爲例，如果影片一秒鐘 15 格，在不壓縮的狀況下一秒鐘可以到達 4MB，縱使有經過壓縮，也很難在品質與檔案大小間做取捨。[1][2]

其實之前所述的方式並沒有考慮到螢幕操作的特性，它與一般影片是不一樣之處在於並不是隨時皆全畫面更新，而只是使用者點選的部分更新。因此比較合適的桌面分享方式是採用第二種：借用遠端桌面連線。

遠端桌面是一種讓使用者操作位於遠端的電腦的方式，屬於 Thin Client / Server 架構。這種系統也是傳輸桌面影像，不過乃是依據使用者的操作傳輸部分的更新畫面而已，因此相對地資料量就少得多。基於這個特點，我們的決定採用這種方式架構來達成桌面分享。

1-3 目標

桌面分享是一個複雜的系統，我們覺得與其從頭開始建立自己的通訊協定與系統，在效能上或許還不如修改目前既有者有效。

屬於 Thin Client / Server 的系統有很多種，經比較我們認爲內建於 Windows XP 的遠端桌面傳輸協定(RDP)較適合作爲我們的系統根基。目前具有開發程式碼的 RDP client 是 rdesktop，而 RDP server 則從缺。想要自

行製作 RDP server 並非不可能，但是限於時間我們決定轉而走向設計 proxy 的架構。目前開放程式碼的 RDP proxy 是 xrdp。

根基於 rdesktop 及 xrdp 的架構下，我們修改 RDP，以達成以下幾個方向：

(1) 目前的 RDP 被局限於 1 對 1 的遠端桌面分享，修改成可以能達成多個使用者同時分享一個桌面。

(2) 修改 RDP 以達進一步節省資料量的目的，包括

- 彈性壓縮圖形：原始的 RDP 採用非失真的壓縮方式處理所有的圖形，然而對於複雜圖形壓縮率低，因此檢查圖形並以失真方式壓縮其中較複雜的部分，同時以兩階遮罩負責保存圖形中重要的文字部分。

- 預防資料暴增：部分使用者的操作，例如拖曳滑鼠、捲動滑鼠等易造成資料量暴增，我們偵測這些情況並處理之。

- 處理資料暴增：利用 Queue 儲存較次要當資料量持續暴增時，利用增加壓縮率減少資料量的增加。

第 2 章 背景知識

本節中我們從傳統的螢幕擷取開始介紹 Thin Client 相關的背景，以及我們主要使用的 RDP 的相關說明。

2-1 傳統螢幕擷取方式

螢幕擷取可以在不同的層級下手。最單純的就是在 application layer 製作擷取程式，然而這種方式不但產生的資料量大，而且受限於硬碟及匯流排的資料傳輸速率，因此很難達成即時傳輸的目的。比較爲人熟知的軟體是 Hypercam 及 Windows Media Encoder。

另一種方式是在 driver layer 上擷取桌面影像。主要是將原本該傳至驅動程式的資料擷取下來轉變成串流。此種方式需要提供額外的硬體驅動程式，因此在軟硬體的搭配及設定上會受限制。同時當資料量很大時，這個驅動程式的可能無法負荷。此類型比較常見的軟體是 Screenwatch。

由此可見，擷取全螢幕所需的資料頻寬很大而且時常成爲程式的瓶頸。在幾篇論文裡提出利用改進的方法爲利用 Event Driven 的方式偵測在 repaint message 並取得其中所包含的畫面更新。這也許多是 Thin Client 系統所使用的方法。

2-2 Thin Client /Server 架構

在傳統的 Client / Server 架構中，Client 端往往負責的工作繁多，因此硬體要求高，需要的硬碟空間也大，以網路遊戲爲例，client 端需要 1GByte 以上的空間儲存地圖資訊、人物模組以及動畫及音效檔，同時也需要高檔的 3D 加速卡以運算 3D 的場景。這種 client 通常被稱爲 fat (thick) client，對於一個公司而言，使用 fat client 越來，帶來的硬體成本也越高。

因此另一種型態的 client 便產生了，它被稱爲 thin client。在 Thin Client / Server 架構中 Server 負責絕大部分的運算並將運算結果傳給 Client 顯示。因此 Client 端唯一的工作只是傳送使用者的輸入，並顯示伺服器回傳的螢幕更新圖形即可，因此不需要高檔的硬體與空間便能執行。[3]

另一個 Thin Client / Server 所強調的特色爲 stateless，也就是 client 不需記錄目前的狀態，想要斷線時就斷線，並可以稍後於不同的電腦硬體、不同的網路位址以及不同的解析度上重新連線。

2-3 各種 Thin Client / Server 系統

目前有許多不同的 Thin Client / Server 系統，有些是開放程式碼的，有些則是商用軟體，以下便對幾種常見的系統加以介紹。

2-3-1 VNC

Virtual Network Computing(以下簡稱 VNC)是一套 Thin client / server 的系統。它最初的發展約於 1995 年左右，隨著 Java 程式語言的興起，由位於英國劍橋大學的 ORL 實驗室所研發出利用 Java 語言撰寫的 VNC Viewer，讓使用者在任何一個作業系統下都執行並透過網際網路連接至遠端的 VNC server 執行程式。[4]

2-3-1-1 VNC 簡介

在 VNC 系統分爲 client 與 server 兩部分，其中 client 又被稱作 Viewer，可以是在 UNIX / Mac OS / Windows 下的應用程式，亦或是 Java Applet。VNC Viewer 的設計非常簡單，只是單純負責接收使用者所輸入的鍵盤或滑鼠訊號，之後將它包裝成 TCP 封包，透過網路傳輸至遠端的 server，因此它占的磁碟空間非常小，符合 "Thin client" 的概念。

VNC server 所在的位置可能是 UNIX / Mac OS 或 Windows，當它接收到由 client 傳來的輸入時產生對應的螢幕畫面改變。隨後 server 將改變的部分封裝爲 TCP 封包再透過網路傳至 client 顯示在操作者的螢幕上。

VNC server 與 client 間採用的通訊協定稱作 Remote Frame Buffer Protocol（RFB Protocol），可知它將 server 當作位於遠端的虛擬顯示卡，產生的螢幕畫面(framebuffer)不僅可在本機顯示，也可傳輸至遠端的 client 上

2-3-1-2 VNC 的圖形壓縮

VNC Protocol 提供了各種圖形編碼方式以達成較佳的傳輸效率。原始的 VNC 版本中提供的編碼方式包括「Raw」、「Copy-Rect」、「RRE」及「Hextile」等等。

1. Raw

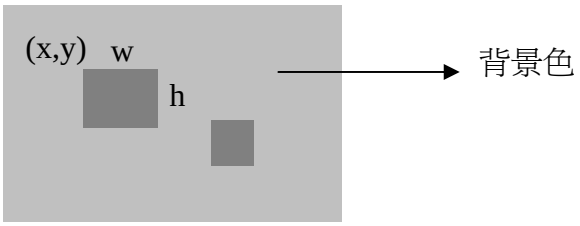
這種編碼是直接傳送圖形的資訊，而不加以任何處理。可想而知雖然每種 client 都會支援它，但所耗費的頻寬十分驚人。

2. Copy-Rect

Copy-Rect 編碼是將 framebuffer 中的圖形資訊由一塊區域複製至另一塊中，因此適用於當使用者移動視窗位置或是捲動視窗時。在 The RFB Protocol 文件中也提及，對於經常使用到的 Pattern，良好的 VNC server 應被設計爲先傳送該 pattern 至 client 端中，之後再運用 Copy-Rect 編碼複製至 framebuffer 中，以節省網路頻寬。

3. RRE

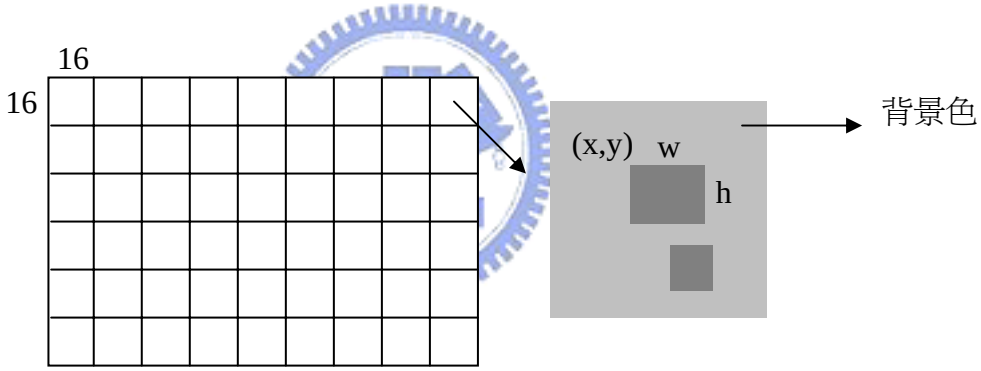
RRE 編碼是將傳輸的矩形區域定義爲多個子區域。首先它收集矩形區域內的顏色資訊，找出最多被使用的顏色當作「背景色（background）」，隨後再依據其它顏色的分佈情形定義出多個子區域，並記錄該區域的顏色、位置及大小。



圖形 1. VNC 的 RRE 編碼方式

4. Hextile

Hextile 編碼是由它將矩形區域以 16*16 圖素爲單位分作多個 Tile 而得名。在 Hextile 編碼中，每個 Tile 可以採用 Raw 或 RRE 編碼，如果是 Raw 會直接紀錄這個 Tile 的圖形資訊；如果是 RRE 則會找出其最多使用的顏色當作背景色，第二多的當作前景色，並記錄其下各子區域的資訊。

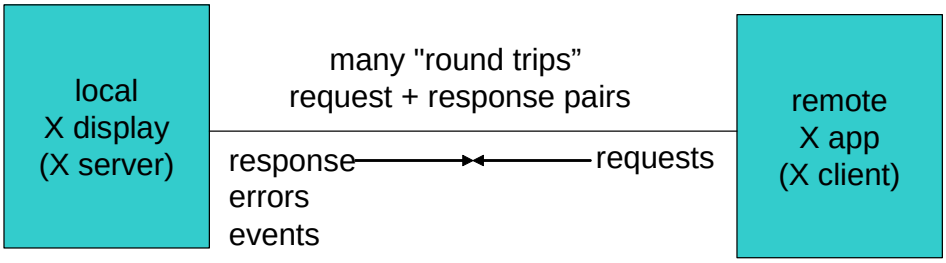


圖形 2. VNC 的 hextile 編碼方式

2-3-2 the X protocol

X protocol 爲 UNIX 作業系統上的 X Window 所使用。X-Windows 的設計上本來就是以 client / server 爲導向，在 X Protocol 中包含了許多與建立操控視窗相關的指令，可以透過網路在遠端顯示視窗。不過在名詞的定義上，X Protocol 將負責顯示畫面的部分稱爲 Display Server，而應用程式端則稱作 Client，與其它的 Thin Client / Server 相反。

由於 X protocol 規定了許多與視窗相關的指令，因此比起 VNC 以純圖形來表示更新的系統而言更爲「高階」。然而高階不代表它的效能比較好，因爲 X Protocol 的指令都需經過 client request 與 server response 的步驟，因此被冗長的 round trip time 所拖累。



圖形 3. X protocol 的 round trip time 過長

目前已有許多改善方式，包括 LBX 與 NX 兩種。[5]

2-3-2-1 LBX (Low bandwidth X)

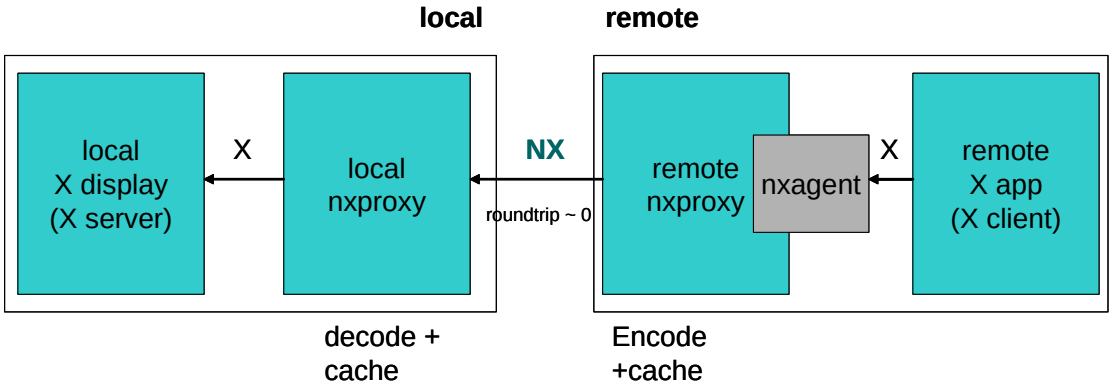
LBX 的設計是根基於 X Protocol 之上的更新，由於原本 X 只能在 LAN 運作，因此爲了讓使用者也能透過低頻寬的網路連線遠端的 application client。當使用者近端利用 display server 透過低頻寬的網路與遠端的 application client 連接時，這個連線會先透過遠端的 proxy 再連至各個 application client。

Short Circuiting 是 LBX 用於降低 round trip time 的機制。proxy 會攔截由 client 發出的 request，或由 server 發出之 response，當它發覺它們與 atoms, colorname/RGB mappings,與 AllocColor 等屬性設定有關時便不會再將這些 request/ response 傳下去，因爲這些都是在一個 session 中不會改變的資料，多送等於浪費額外的時間與頻寬。因此這種作法許多 request/response 變成只有單向的 request 而已。

其次 LBX 提供類似快取的架構。當 server 傳資料給 client，經由 proxy 時會將其中的 tag 及資料存於 proxy 中；當之後再次傳送時，如果與之前的資料相同者，則只會傳 tag 給 proxy，此時 proxy 會將對應的資料傳給 client。

2-3-2-2 NX

NX[6]的設計是分別在 server 及 client 端建立 proxy 的架構，如下圖所示。



NX 減少資料量的方式主要是利用 cache 機制。在 NX 中在兩個 proxy 言各建立 message store 儲存傳送/接收過的 message。傳送 message 那端的 proxy 會檢查待送 message，如果與在 message store 中者相符，則只傳送該 message 的位；如果該 message 是新的，則傳送至另一端，並分別加入其 message store 中。

由於一個 message 的變動性太大，例如指示滑鼠的座標位置，如果直接將整個 message 拿來與 message store 比較，cache hit 的比率是不會很高的。因此在 NX 中將一個 X protocol message 分成兩部分來處理：固定長度部分（稱作 identity）與資料部分。以 X Protocol 中的 X_PolySegment 爲例，它有以下幾部分：

表格 1 X Protocol 的 X_PolySegment Request

欄位	長度
reqType	1
pad	1
length	2
drawable	4
gc	4
data	54

因此 identity 部分占了 12 bytes，而 data 部分則是 54bytes。NX 中對這兩個部分有著不同的處理方式。

(1) Identity

同類型 Message 的 Identity 通常是不變的，因此以此來檢查能增加 cache hit 的比率，在 NX 中使用 message store 大約能減少 10:1 的資料量。

(2) 資料部分

這一部分的資料是時常會變動的，因此不適合作 cache。在 NX 中對該部分使用 Differential Encoding 的壓縮方法，大約能減少 5:1 左右的資料量。

2-3-3 THINC

THINC[7]這個系統是由美國哥倫比亞大學的網路計算實驗室所發展。它提供了多組指令處理不同的媒體，包括顯示畫面圖形的指令集、處理影片播放的指令與滑鼠指標的指令集。它的主要特色不在圖形的壓縮，而是在於指令的管理方面。

在 THINC 的系統中使用 Queuing 的方式來管理待傳送給 client 端的指令。THINC 建立了多個 queue，當指令產生時，並不是直接被傳送給 client，

而是先被加入 queue 中。每個 queue 負責儲存資料量在某範圍內的指令，而在同一個 queue 中的指令是以” Shortest remaining Size First”的規則來傳送。

當 THINC server 偵測出網路流量太大時，會暫時停止將 queue 中的內容送出。此時它會開始檢查 queue 中的畫面更新指令，如果發現有兩個更新指令的畫面彼此重疊時，會將它們分解成彼此不相交的新區域。待網路恢復正常後再重新開始送出指令。不過爲了即時性的需求，THINC 還有著 real time queue，負責處理例如滑鼠指標移動等等的指令。

2-3-4 Thin Client 綜合比較

除了之前介紹的 RDP、VNC、X、THINC 外，目前 Thin Client 尚有很多種，以下就幾個方面來歸納它們的特性。[8]

2-3-4-1 圖形編碼方式

Thin Client / Server 所使用的圖形編碼方式可以依據低階至高階分爲四種：

(1) RAW pixels

傳輸的圖形完全不經過編碼，可想而知資料量非常大。目前有 VNC 支援這個模式。

(2) 2D draw primitives

利用區塊填色的方式將圖形編碼，通常是非失真的壓縮方式。VNC 主要是使用這個模式。

(3) Low level graphics

除了(2)的方式外，還提供了字型、滑鼠指標、圖樣等等的支援。RDP 屬於這一類。

(4) High level graphics

除了(2)(3)外，還提供了視窗建立與管理的功能。像是 X Windows 使用的 X Protocol 就是此類。

一般而言，高階的不一定表現的效能也最好，如同前幾節所述，X Window 受累於 round trip time。相對地比較低階而單純的幾種 thin client 系統就能省去這些 overhead 而有較佳的表現。

2-3-4-2 Server Push / Client Pull Model

所謂 Server Push 的意思即由 server 主動決定何時傳送畫面更新至 client 端而 Client Pull 則是由 client 端去催促 server 傳送更新。使用前者的 thin client/server 系統所產生的畫面更新較爲平順，但同時資料量也會相對的多。前一節所介紹的 THINC 便是使用 server push 的策略。至於使用 Client Pull 的系統通常只有在使用者進行輸入時，才會通知 server 傳回當時產生的 update，因此它會比節省 server push 產生較少資料量。

可想而知，若使用者透過 Thin Client/ server 系統遠端播放影片時，使用 server push 的系統會產生龐大的資料量，而 client pull 的系統則會因爲使用者沒有任何輸入動作，而只看到停滯的畫面。因此基本上 thin client 系統並不十分適合播放影片。

2-3-4-3 Eager / Lazy Update

所謂的 Eager / lazy 是指 server 傳送 update 的頻率而言，若是 eager update，表示每當有 update 產生時 server 就立即加以送出，而 lazy update 的情況則是 server 會間隔一小段時間後再將 update 送出。

以播放影片爲例，如果影片是 15fps，表示畫面在一秒鐘內會有 15 次變化，屬於 Eager Update 的系統就會在這一秒鐘傳送 15 個更新畫面；相對地，如果屬於 lazy update 的系統可能是只在每一秒鐘傳送當時的畫面而已。因此 eager update 通常會讓 server 傳送過多的資料量。

在本節最後，以下表列出幾種常見的 Thin Client / Server 之比較。

表格 2 常見的 Thin Client / Server 之比較

系統	圖形等級	更新頻率	壓縮方式	cache 大小
VNC	2D primitive	client pull lazy	RLE	none
RDP	low level graphics	server push lazy	2D Hextile	1.5MB
X	high level graphics	server push eager	none	none
THINC	low level graphics	server push eager	PNG	none

2-4 遠端桌面通訊協定 RDP

Windows XP 的遠端桌面也是一種 Thin Client 系統，使用 RDP 作爲通訊協定，也是我們系統所根基的技術，因此特別提出來介紹與討論。

2-4-1 RDP 簡介

RDP 是 Microsoft Windows 作業系統的 Terminal Service 所使用的通訊協定。它從 4.0 版開始，總共經歷了多次的改版，如下表所列。

表格 3 RDP 各版本比較

版本	作業系統	特色
4.0 (xrdp)	windows NT 4.0 Terminal Server Edition	支援 8bpp bitmap cache clipboard mapping
5.0 (rdesktop)	Windows 2000 Professional	支援 16bpp printer redirection Persistent bitmap cache 以 56 bit 或 128 bit 加密
5.1	Windows XP Professional	支援 24bpp sound / drive redirection
5.2	Windows Server 2003	secure over SSL console mode conenction
6.0	Windows Vista	支援 32 bpp Remote Program WinFX support

由於 Windows Terminal Service 是商業產品，因此無從得知其通訊協定真實的運作過程。目前有公開程式碼的是在 Linux 上運作的 RDP client：rdesktop，也是經過反組譯而得到的成果，因此雖然大部分時刻能運作正常，但仍然充滿無法解析的 bug。以下幾節就是透過 rdesktop 來了解 RDP 大略運作的方式。

2-4-2 RDP 的 Order 類型

RDP 中除了基本的連線、同步功能外，重點還是在顯示畫面更新上。比起其它 Thin Client 系統中螢幕畫面皆以圖形來表示，RDP 中尚利用矩形、多邊形及文字來強化顯示效果。我們依循 rdesktop 程式碼中的稱法，將這些負責顯示畫面的指令稱作 orders，以下就分別對其中重要的指令加以介紹。

(1) Memory Blt

blt 有顯示在螢幕上之意。memory blt 即是將目前存於記憶體中的快取圖形顯示於螢幕上特定位置。同一份圖形可以顯示在不同位置，但是只需傳送一次即可，如此能降低資料量。在一般的 Windows 視窗中最常見的就是

全白的視窗背景，是以一個白色 64*64 大小的圖形分別進行 memory blt 顯示在畫面上的不同位置。

(2) Pattern Blt

這種指令是傳輸 1 bit 的圖樣，指定其前景及背景色後，在螢幕上的特定位置顯示。最常見的圖樣即是常我們拖曳視窗時所顯示的視窗外框，即時由單個圖樣在不同位置進行 pattern blt 所組成。

(3) Screen Blt

這種 Blt 的方式常見於拖曳視窗內容時，由於畫面完全一樣，只是在螢幕上的位置改變，因此 Server 只需透過這個指令讓該內容改變位置即可。

(4) Rectangle / Line / Polygon

在 RDP 中降低資料量的最大功臣即時能顯示基本的形狀，包括矩形、線段與多邊形等等，這些形狀看似簡單，卻能組成各種不同的視窗元件出來。矩形通常是組合成視窗本體，線段主要是爲字串加上底線，而多邊形則可見於在 Powerpoint 中加入的快取圖案。

(5) text

在視窗中隨處可見到文字，無論是功能表、標題列、網頁內容、Command Line 等等，以往在其它的 Thin Client 系統中全部都將文字轉化爲圖形表示，由於文字大多細長，在圖形中屬於高頻區，因此如果再配合失真壓縮，往往讓文字模糊難以辨認。因此 RDP server 讓部分背景單純的文字以點陣圖字型來顯示，而處於環境複雜的文字則仍以圖形表示。

以點陣圖形文字來表示的字串，RDP server 會指定其所使用的字型、字串快取的 id 及 index，以及它在螢幕上所示的位置等等。

(6) Cache

RDP 另一個特點就是快取的使用。大部分用過的文字、圖形都會存在於記憶體中的快取一陣子，以提供之後再次利用，如此 server 不需再次傳送相同資料至 client 端，達成節省傳輸量之目的。

2-4-3 RDP 圖形壓縮模式

RDP 內建的圖形壓縮模式多爲顏色上的編碼，如下所示。

- FILL 填滿和之前相同的顏色
- MIX 和之前出現的顏色混色

- BiColor 間隔填色
- Color 設定某一個 pixel 的顏色
- Copy 從一個區域 copy 至另一個
- Black 區域填黑色
- White 區域填白色

與 VNC 內建的壓縮方式相同，都是屬於非失真型的壓縮。觀查以上所列可以發現，它們大多屬於顏色的處理，因此若圖形中顏色越多，其壓縮的比率也越差。

2-4-4 RDP 的 Cache 機制

在 RDP 中提供多種快取的模式，想利用額外的快取空間來換取資料傳輸的時間。主要的快取模式有三種：圖形快取、磁碟快取及字型字串快取。

(1) 圖形快取

圖形快取是 RDP 用於降低資料傳輸量的主要機制。在 RDP 中大部分待顯示的圖形都會被先存入圖形快取內。圖形會依大小被分入三個 cache 之中，各個快取的數量及能容納的圖形尺寸大小是由 client 端所決定的。例如 rdesktop 的預設值如下表所列：

表格 4 RDP bmp cache 大小

	cache 數量	cache 大小
cache 1	0x258	0x100 * Bpp
cache 2	0x12c	0x400 * Bpp
cache 3	0x106	0x1000 * Bpp

在以上的情況下，圖形快取的尺寸最大不大於 64* 64 像素。如果該圖形的尺寸大於 64*64 像素，則 RDP server 會將該圖拆成多個 64*64 的小圖形，此乃爲了配合 RDP 的非失真壓縮無法有效壓縮大而複雜的圖形所致。

(2) 磁碟快取

磁碟快取主要是輔助圖形快取的機制。由於 client 端連線中斷後會自動清除記憶體中的快取，因此利用部分磁碟空間進行快取能節省下次連線時的資料傳輸量。

(3) 字型/字串快取

在 RDP 中字型是以單色的點陣圖表示，在 client 端可以指定其前景色以設定爲不同顏色的字型，然而對於不同的大小、粗斜體的字型 RDP server 皆會視爲不同而各自傳輸至 client 端。

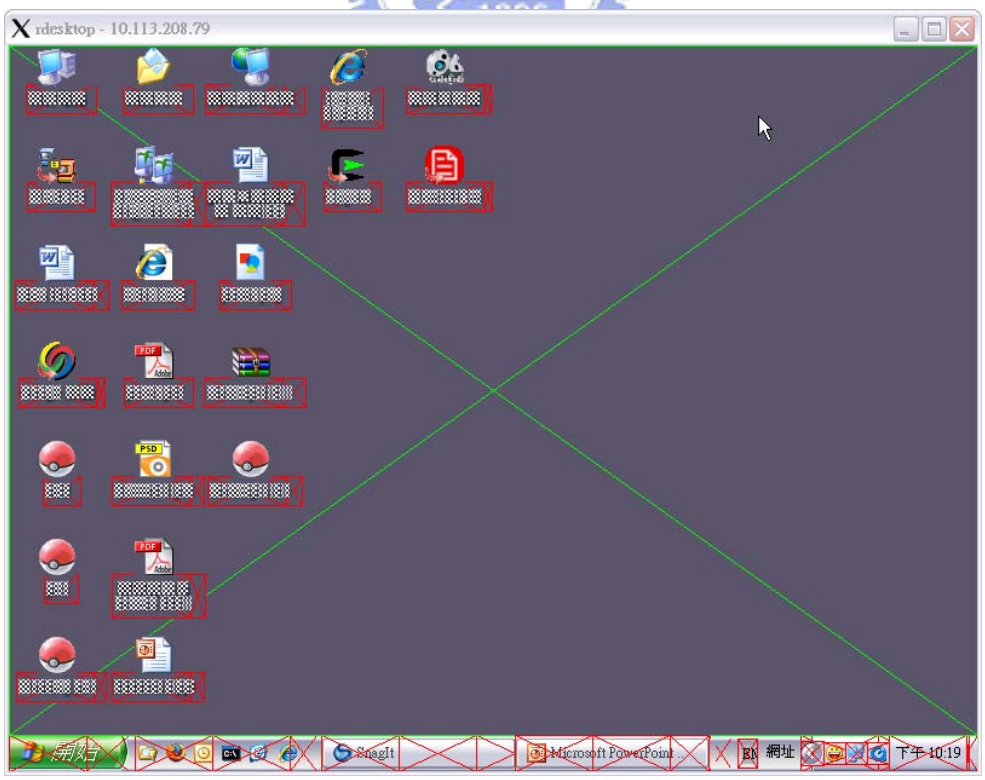
欲顯示某一字串時，RDP server 首先會傳輸需要的字型至 client 端的記憶體快取中，隨後再將該字串傳至 client。所有的字串皆以使用之字型快取所表示。字串本身也被暫存於記憶體之中，以供之後出現相同字串時快取之用。

2-4-5 RDP Server 視窗畫面分析

由於 RDP 中不僅僅以圖形表示畫面，尚有矩形、多邊形及文字來強化顯示效果，因此比起其它 Thin Client 系統更爲複雜。本節我們就桌面、視窗元件、網頁內容、文字編輯器、簡報軟體及 Command Line 等方面來分析 RDP 如果解析視窗畫面，試圖理解如何對這些模式進行最佳化。

(1) 桌面

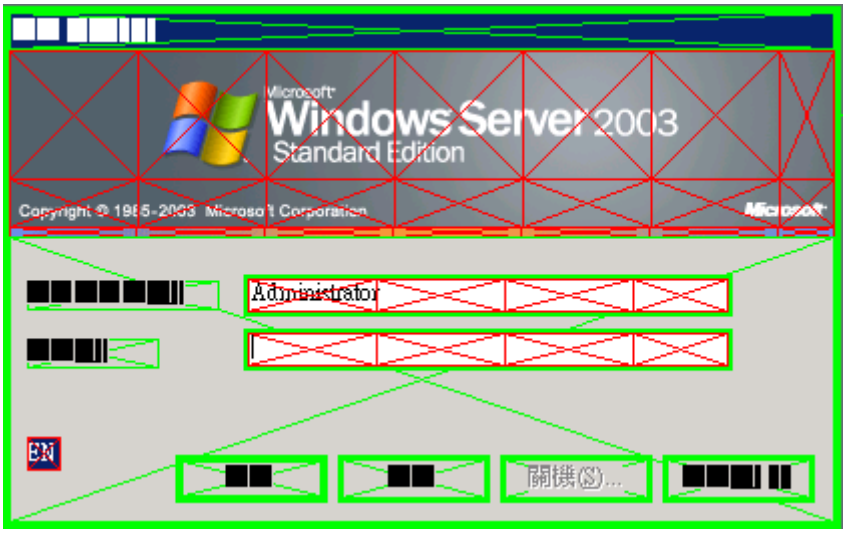
RDP Server 將視窗桌面設定爲單一矩形，桌面上的圖示是點陣圖，桌面上的文字則以純文字傳送。因爲進行遠端桌面時 RDP 不會顯示桌面背景，因此能節省許多傳送桌面圖案的時間。如下圖所示，其中綠色的框代表的是矩形的位置，而紅色的框代表的是圖形的位置。圖中的文字若能辨識，表示它們是以圖形的方式傳送，如果是純文字，則是灰色的圖樣表示。



圖形 4. RDP 顯示桌面的方式

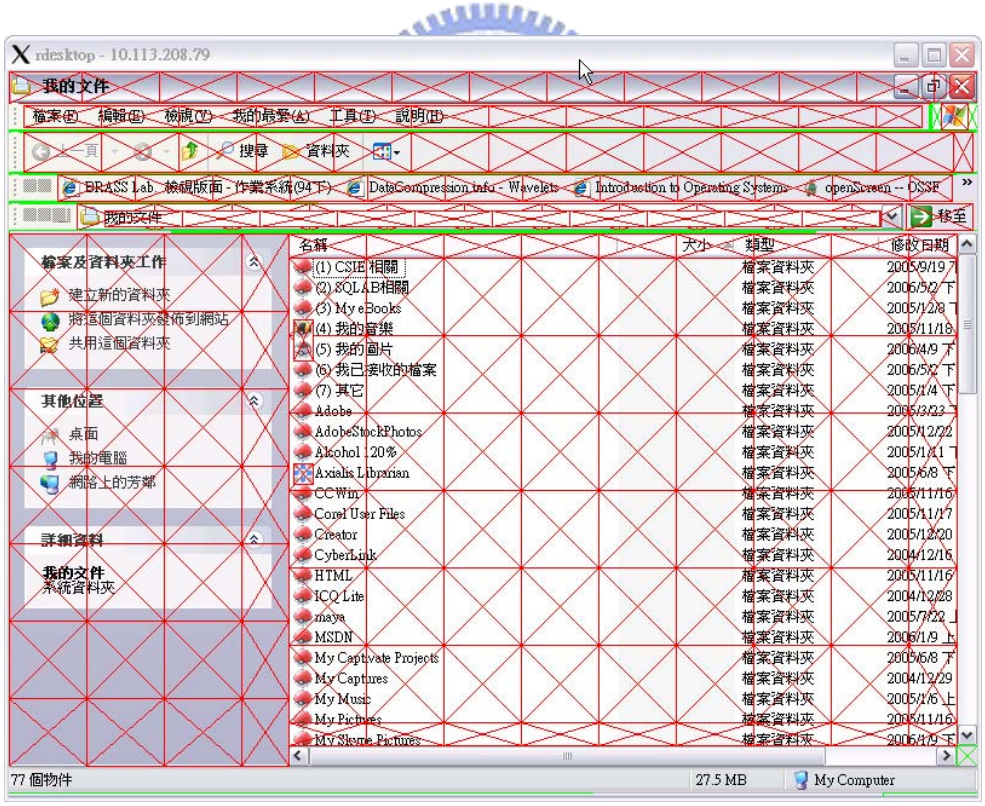
(2) 視窗元件

RDP 巧妙地將視窗與功能表分解爲由矩形及文字構成的圖案。



圖形 5. RDP 顯示視窗的方式

在 RDP server 中並沒有一定的圖形分割規則，並非所有的圖示都會單獨成爲一張圖形而傳送，由下圖便可看出。

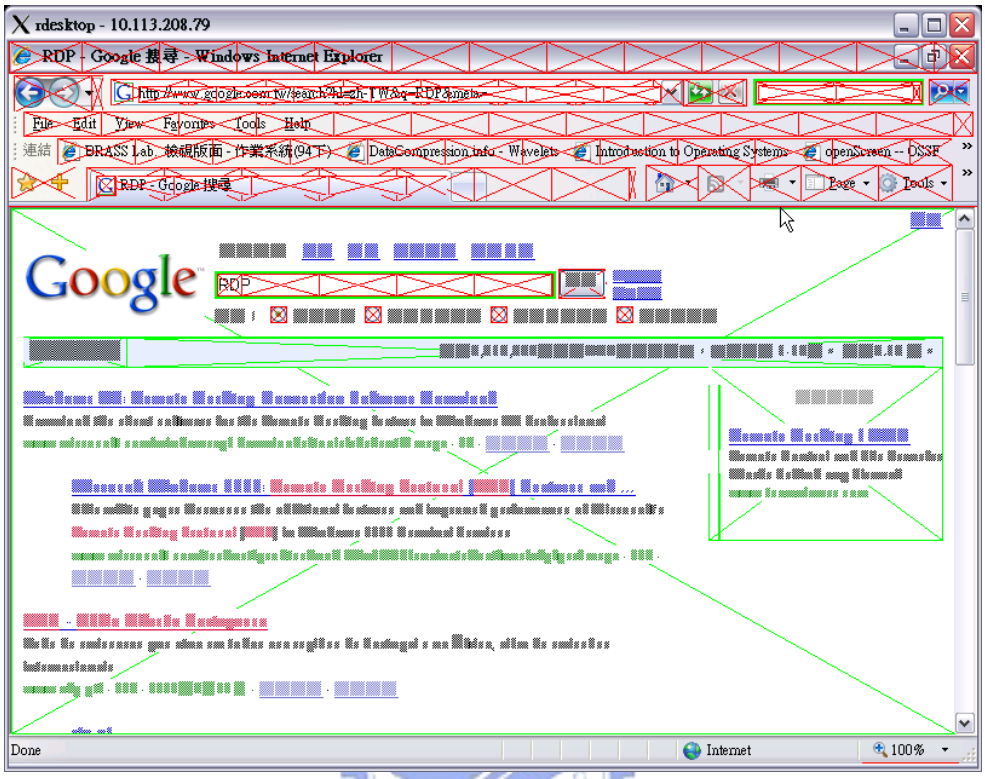


圖形 6. RDP 中部分文字以圖形方式傳送

(3) 網頁內容

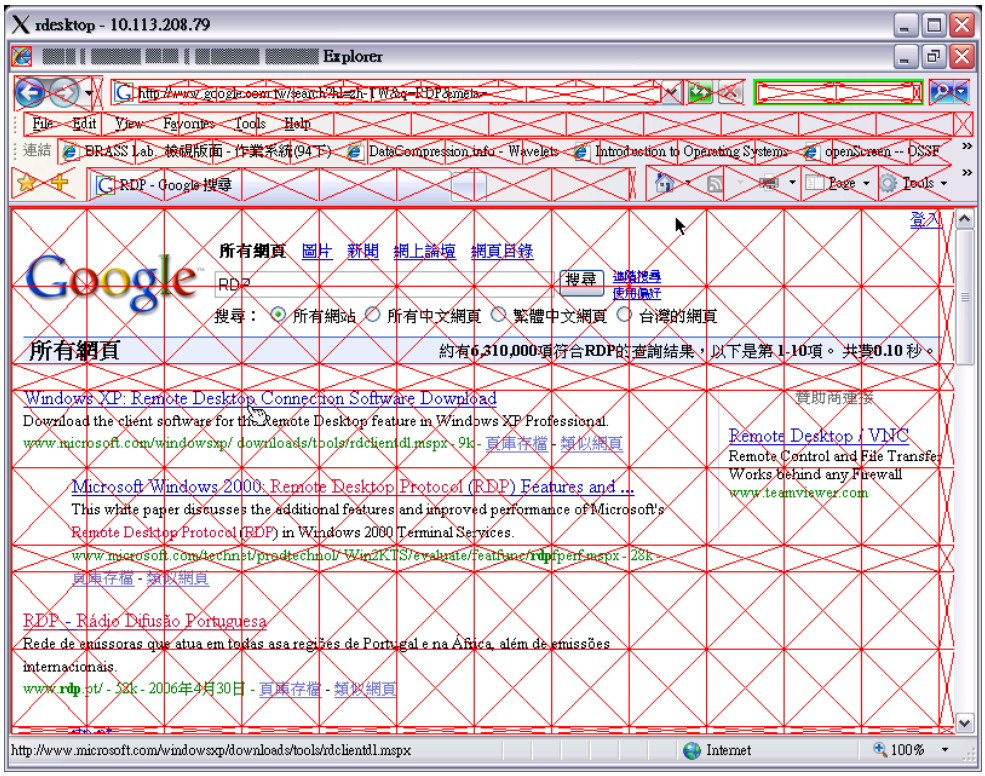
網頁依據其使用瀏覽器與內容不同，各有不同的解讀方式。以下便以目前主流的 Internet Explorer 與 Firefox 爲例。

對 Internet Explorer 而言，它會自動區分網頁中的圖形與文字，並純文字來傳送其中的文字部分。至於 flash 動畫等等則一律視爲圖形，因此如果不小心進入這類型網頁時往往會產生很大的資料量。



圖形 7. RDP 中 Internet Explorer 7 顯示網頁的方式 (1)

當我們捲動網頁時，在上下方新出現的網頁內容則會被 RDP server 儲存爲圖形，無論其中含有多少文字。因此在捲動時往往是讓使用者感覺操作最不順暢的時刻。



圖形 8. RDP 中 Internet Explorer 7 顯示網頁的方式 (2)

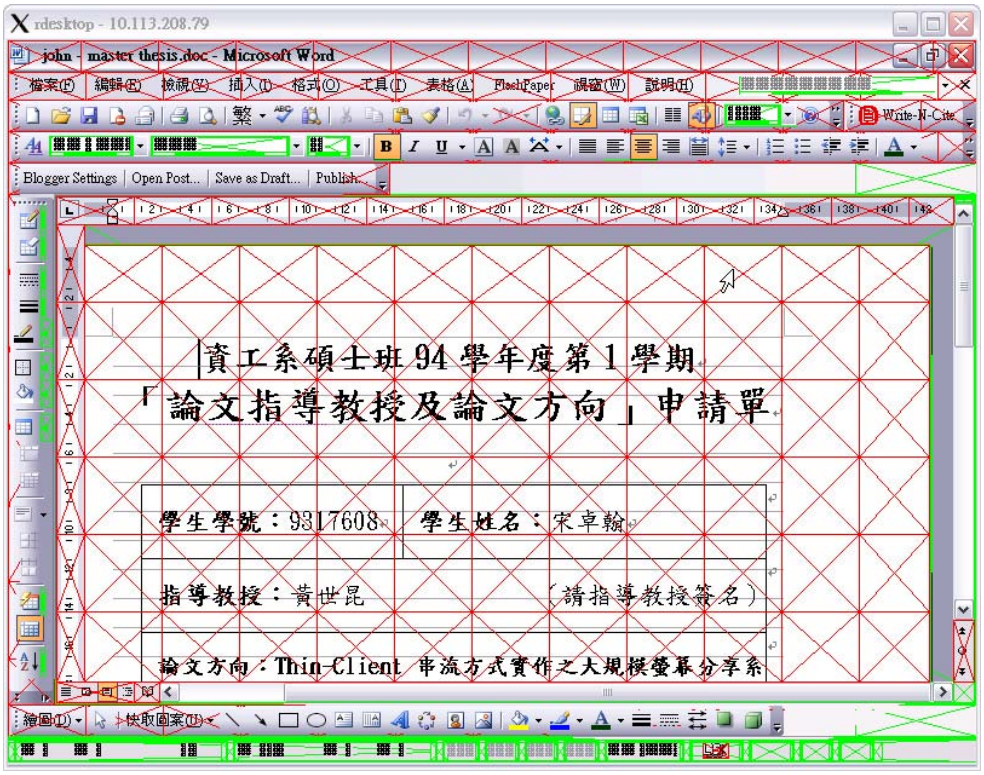
至於 Firefox 的顯示方式，則是全部以圖形的方式來傳送網頁內容

網頁是一個使用者無法控制的環境，目前網頁上充滿了圖片、動畫與影片。即便是單純地在網頁信箱收發郵件，也可能因爲旁邊一個不斷重複播放的動畫造成大量的傳輸資料。

(4) 文字編輯程式 Notepad / Word

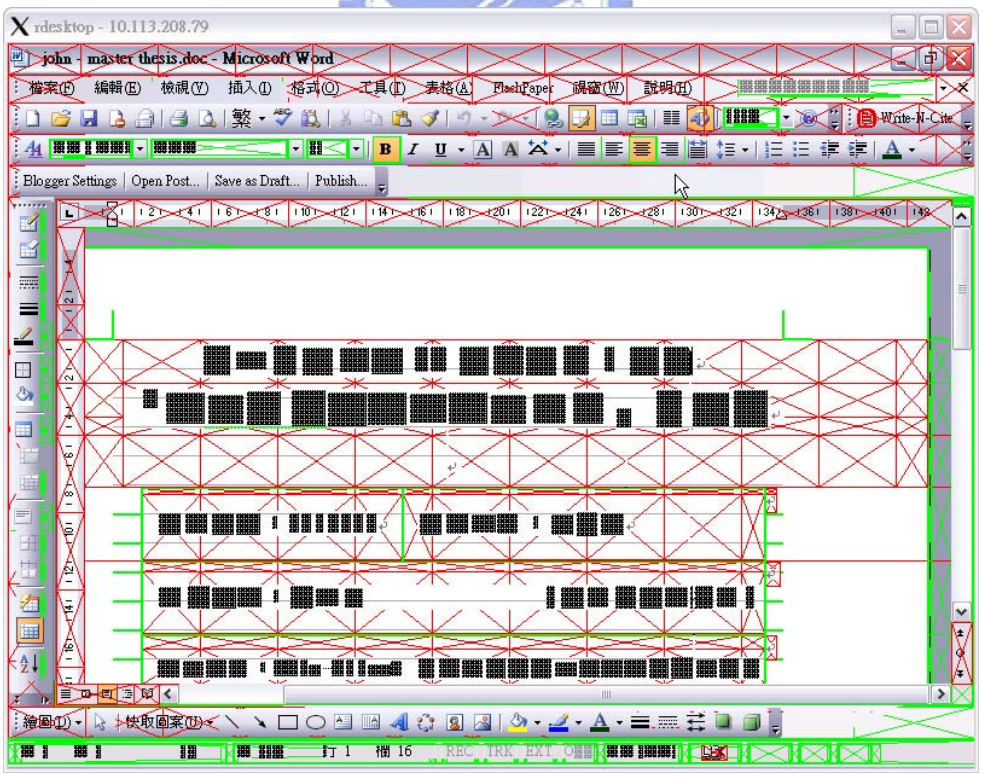
雖然都是文字編輯程式，但不同的程式在 RDP 內部運作部不相同。

在 Notepad 中輸入的文字皆以純文字方式傳送，在 Word 中則是輸入時以圖示方式顯示文字，如下圖。



圖形 9. RDP 中利用 Word 編輯文字 (1)

而之後捲動文字時這些文字才會變成以綁文字方式傳送，如下圖。

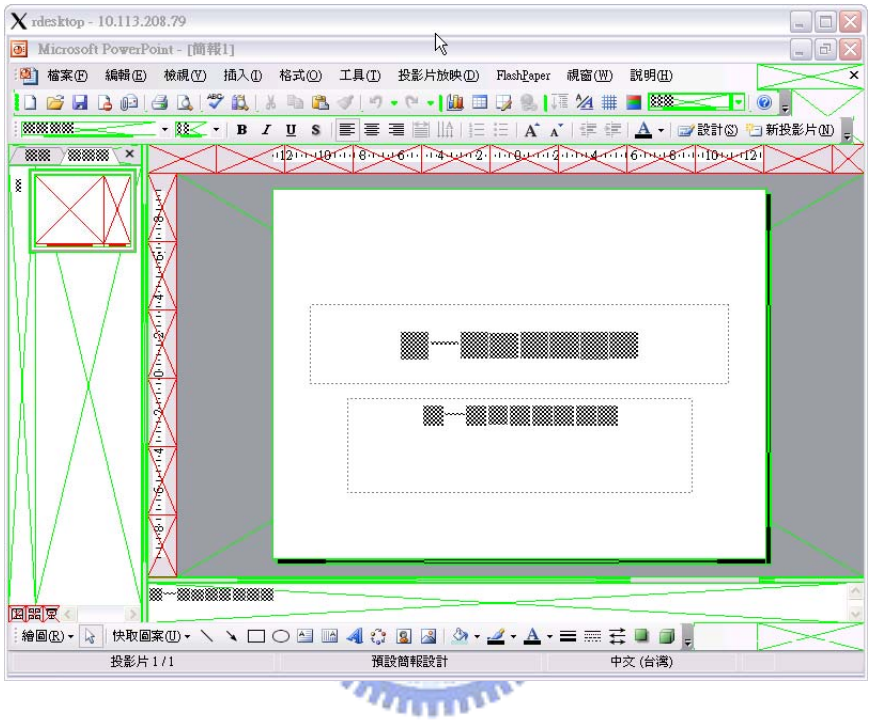


圖形 10. RDP 中利用 Word 編輯文字 (2)

由於文字編輯器的底色通常是白色爲主，因此資料量較不大，加上快取圖案通常都是由 RDP 的多邊形指令所表示，因此若非文件中塞滿了圖片，否則其資料量不大，而 cache hit ratio 也能維持在 90%以上。

(5) Powerpoint

在編輯模式時，左邊的縮圖是圖形，右邊的投影片編輯則是文字爲文字，圖形爲圖形，之後進行放映模式時，則是全部以圖形顯示。

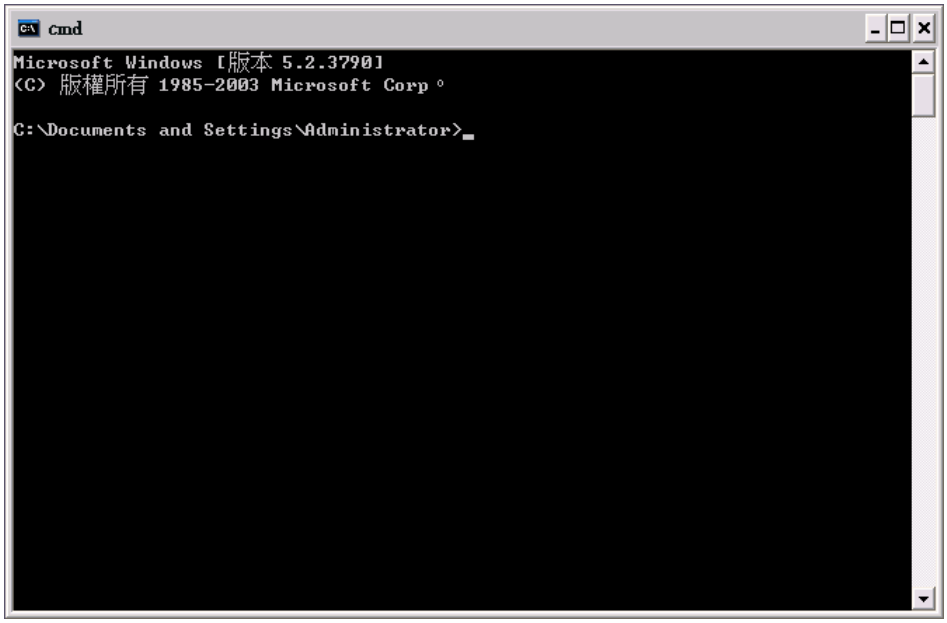


圖形 11. RDP 中 利用 Powerpoint 編輯簡報

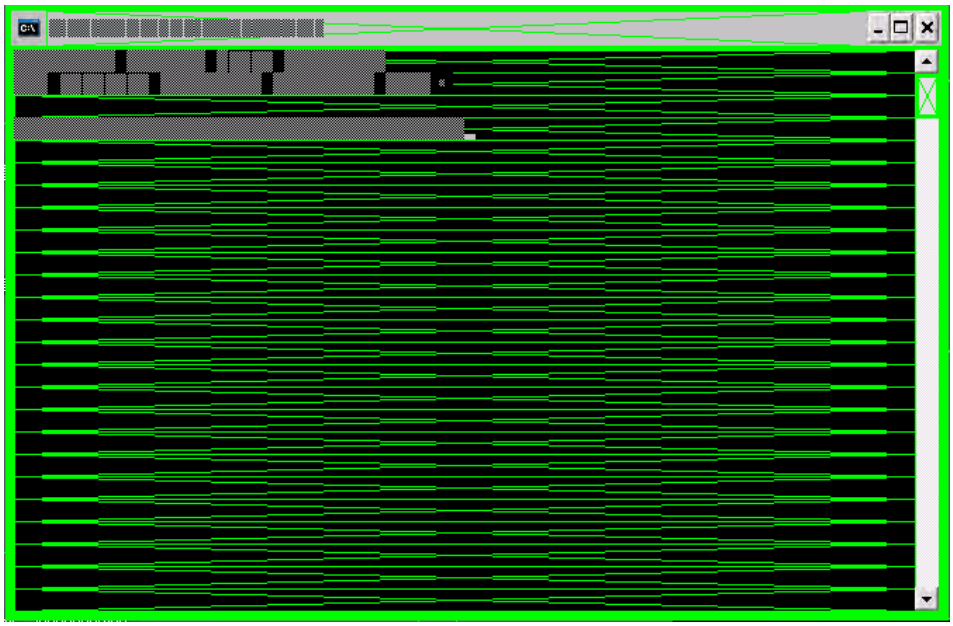
在播放模式時，由於播放速度取決於使用者，因此當使用者快速切換頁面時等同於在播放動畫一般，此時若背景又非純色，將造成大量資料量。

(6) Command Line 視窗

Command Line 主要是提供使用者指令輸入的環境，其視窗實際上是黑底的矩形的所構，而其中的文字則皆以文字來傳送。



圖形 12. 原始的 Command Line 視窗



圖形 13. RDP 中以黑色矩形及文字表達 Command Line

由於以上的特性，使得在實際運作時資料量僅限於字型/字串方面，並不會造成巨大的流量。即使是使用者下了如 dir/s 等會讓視窗內容快速捲動的指令，也會因爲都是使用相同的字型、不同的字串，使資料量不會暴增。相對地，如果是使用其它 Thin Client 系統時，則會因爲全是以圖形顯示，造成一股驚人的傳輸量。

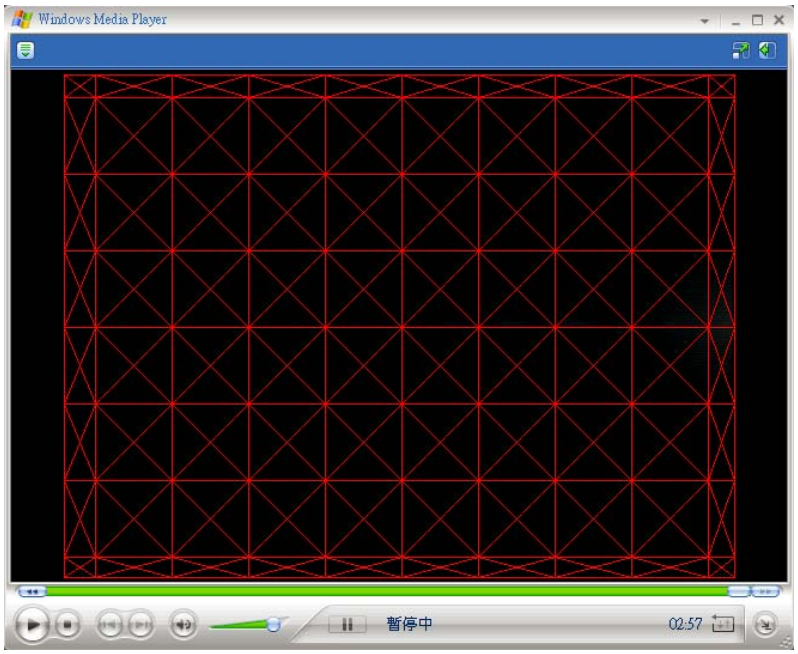
(7) 播放影片

基本上 RDP 系統並不適合播放影片。一般的影片都有經過特殊的壓縮過程，例如 MPEG 格式，有分 I、P、B 等影格，I Frame 是單獨的畫面，P

及 B 則會參考其它 I 或 P frame 補間出對應的畫面，這些措施能相對地降低傳輸影片時的資料量。

然而在 RDP 中播放影片時，RDP server 並不知道使用者正在播放影片，而僅只知道目前不停地畫面不停地變動，因此全部傳給連結的 client 端，如此反而會將資料量暴增許多倍。

另一方面，影片通常都是包含多種顏色形狀複雜的圖形所組成，因此例用 RDP 內建的非失真壓縮並無法有效地降低資料量，可以說是雪上加霜。有時 RDP server 甚至會決定直接傳輸未經壓縮的圖形，因爲影片圖形的壓縮率不高而且還會還帶來額外的 overhead。



圖形 14. 以 RDP 播放影片時傳送許多矩形

以下列出以上幾種操作模式時的平均資料量及圖形快取 hit ratio 比較表。

表格 5 RDP bmp cache ratio

	bytes/ minute	bitmap cache hit ratio	saved data ratio
網頁內容	1,392K Bytes	50%	25%
Word	105K Bytes	95%	95%
Command Line	582K Bytes	99%	95%
影片	12,511K bytes	1%	0%

2-5 Thin Client 系統的其它應用

2-5-1 VNC Session 的錄製、重播與檢索

由於 VNC server 及 client 是藉由 RFB Protocol 而運作，因此只要將這些用來溝通的 RFB Message 依發生之時間記錄成文字檔案，便能在日後藉由讀取文字檔案還原這些 Message，達到重覆播放的目的。[9]

2-5-1-1 錄製

爲了能達成任意播放影片的目的，在錄製 Session 時必須同時記錄其發生時間，作爲播放時的索引。在 Sheng Feng Li 等人所發展的 VNC 錄製系統中除了記錄 RFB 通訊協定的 message 外，還額外記錄了其它的資訊，總共可分爲四種型態。

(1). Intentional Annotations

這些是使用者特別標明的註記，只需在錄製時在系統提供的文字的視窗內輸入文字，client 就會傳送「client Cut Text」訊息給 Serve；另一方面，只要使用者利用滑鼠將螢幕上的文字加以標記，server 就會傳送「server Cut Text」事件。這些文字都會被記錄下來，除了在日後播放時會即時出現外，也可以作爲檢索之用。

(2). Side Effect Index

這些是使用者操作滑鼠、按下鍵盤或是 server 傳來 Bell Message 時所會記錄下的索引。

(3). Derived Index

這是利用分析程式分析畫面所自動產生的索引，例如當發現畫面有大幅度變動時，可能代表一個新的視窗被開啓了。因此也很適宜作爲索引。

(4). Post Hoc Index

這是使用者在播放錄製好的影片時所加入的額外註解。

2-5-1-2 重播

在 Sheng Feng Li 等人所發展的 VNC 播放系統中，系統會讀取之前產生的索引檔案還原錄製時產生的各個動作。利用索引值，可讓使用者能依需求反覆播放影片，或是跳到影片中的其它章節。

2-5-1-3 檢索

在 Sheng Feng Li 所發展的 VNC 播放系統中將錄製影片時所記錄下的索引與資料庫充分結合。它能利用 SQL 語法在索引檔案中找尋相符的結

果，例如「畫面改新超過 40%」或是「使用者輸入的註解中包含"VNC"」，並將對應的螢幕畫面轉換爲 Thumbnail，並於網頁瀏覽器中呈現。

2-5-1-4 整合同步與非同步合作

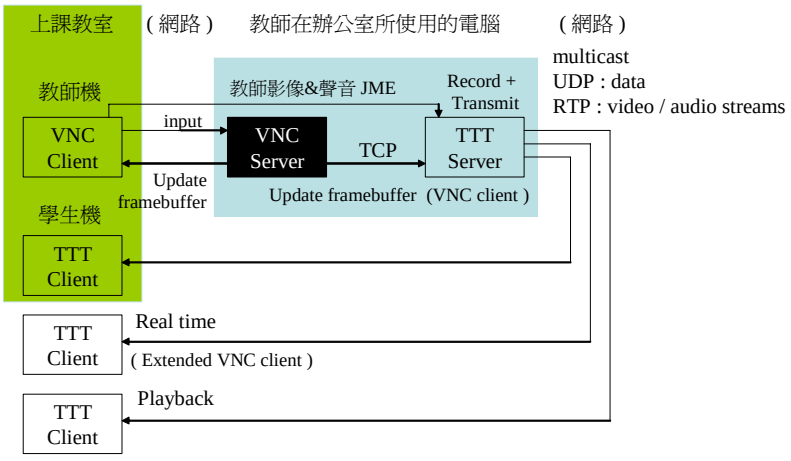
VNC 的 session 錄製功能可以擴展爲提供企業裡多人進行同步合作，或是對已錄製好的影片進行非同步合作模式。[10]

在這個系統中，負責錄製 VNC session 的不再是任一台 VNC client，而是一部它發展的 VNC proxy。這個 proxy 能同時連結多個 client 與一部 server，將 client 發出的 message 交由此 proxy 統一發送給 server。對 server 而言，proxy 會將其發出的 message multicast 給多個 client，因此每個 client 都能看到同樣的螢幕改變。

由於錄製工作交由 proxy，使得操作的 client 或負責產生畫面的 server 負擔都減輕。同時 proxy 的設計能達成多人合作的目的，也可依據實際需求只讓單一 client 進行輸入，不過全部 client 都能看到其實際執行結果。

2-5-2 TTT (transparent tele-teaching)

TTT 是一套利用 VNC 作爲擷取螢幕畫面的遠距教學工具[11]。它的架構如下圖所示：



圖形 15. TTT 的系統架構

(1) 設立 TTT Server

TTT server 是一個修改過的 VNC client，它與教師所使用的 VNC client 都同時連接至 VNC server，只是它只接收 VNC server 傳來的螢幕更新，而不傳送任何輸入。之後 TTT server 會將螢幕畫面轉送至其它的學生電腦上的 TTT client。

另一方面 TTT server 也負責錄下教學的桌面影像提供之後學生 on demand 的閱覽，方式如同前一節所述。

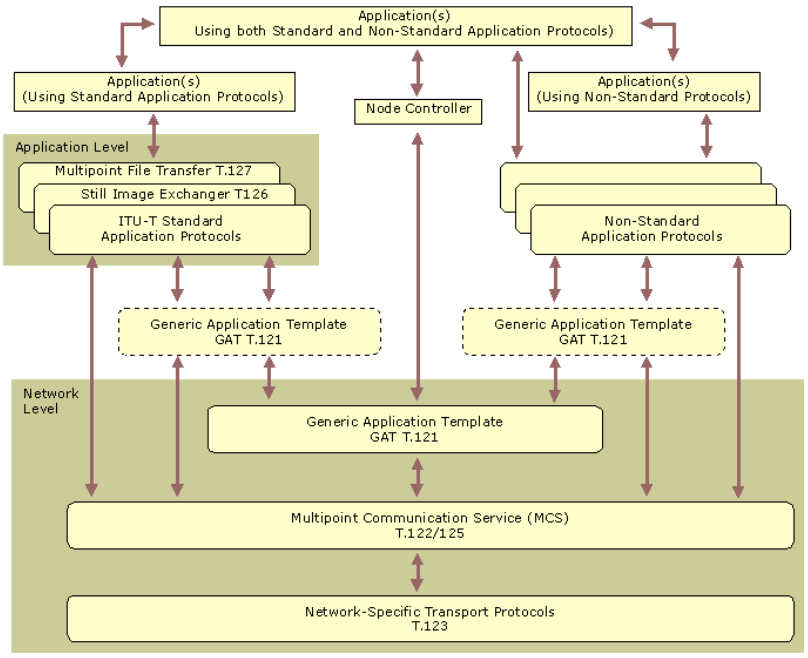
(2) TTT client

學生所使用的是經修改的 VNC client，與 TTT server 連接後接收螢幕畫面。爲了減輕 TTT server 的負擔，作者修改了 VNC 使用的 Protocol，從 TCP 改成 UDP 來傳輸資料。爲了防止 UDP 在傳送資料時遺失，以致 client 端的畫面出現不連續的現象，TTT server 每隔一分鐘就會傳送一次全螢幕畫面至 TTT client。

2-5-3 Net Meeting

Net Meeting 是由 Microsoft 所推出的視訊會議軟體，包括視訊、音訊與桌面的分享。視訊與音訊是採用 ITU 的 H.323 標準，而桌面與其它資料的傳輸則依靠 ITU T.120 系列的標準。

在桌面傳輸的部分，是由 T.120 中的 T.128 所負責進行程式的分享。T.128 是由 Microsoft 所提出並由 ITU-T 採用而成爲的標準，Microsoft 的 RDP 也是根基於 T.128 所發展，因此 Netmeeting 採用非失真的圖形壓縮方式。不過在效能上 Netmeeting 仍比使用 RDP 的 Windows 遠端桌面差。

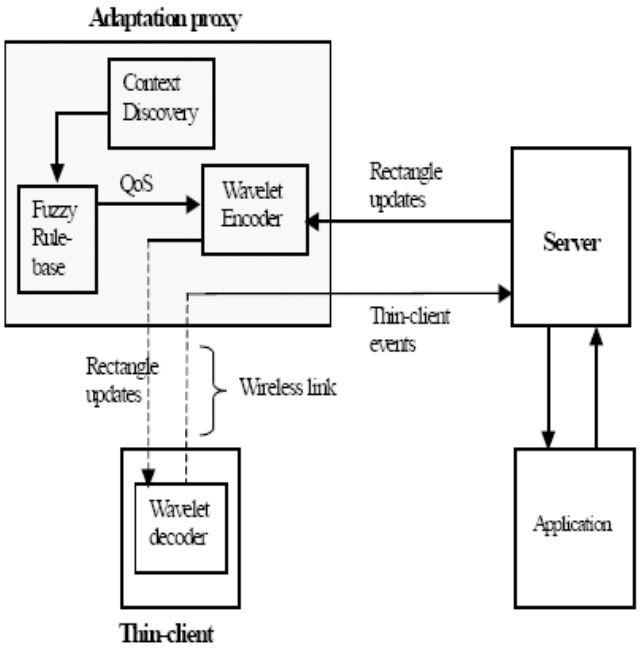


圖形 **16. Netmeeting** 的桌面傳輸架構

2-5-4 動態決定圖形壓縮率

大部分的 Thin Client 系統都是採用非失真或是固定壓縮率的方式來處理圖形。然而面對隨時變動的網路環境，有時使用者的操作會造成較多的畫面改變而產生較大的網路流量；有時則處於 idle 狀態，幾乎沒有畫面更新送出。面對這兩種極端，在 Thin Client 系統中若能加入決策機制，依據當時的網路環境決定不同的圖形壓縮比率，可以讓系統更具彈性並節省更多頻寬。

以 Fuzzy Rule 動態決定壓縮率的系統常是以設置 proxy 爲主，在不修改 server 的前提下，將由 server 送出的圖形以小波轉換（wavelet transform）方式進行壓縮後送至 client 端。系統架構如下[12]。



圖形 **17.** 以 **proxy** 方式動態決定壓縮比率

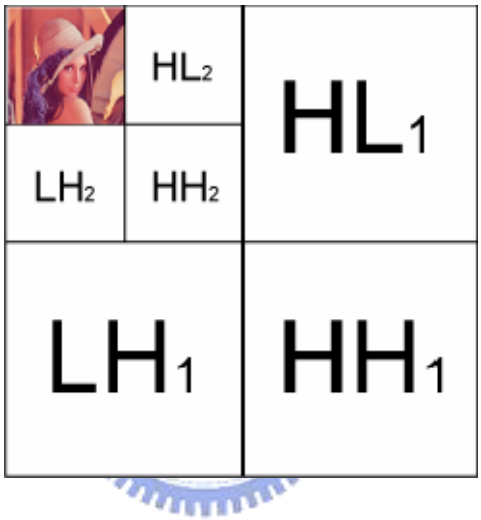
2-5-3-1 小波轉換的圖形壓縮原理

小波轉換能將圖形的高低頻部分解，例如在下圖中(a)經由對水平方向的轉換後得到(b)，再經由對垂直方向的轉換後得到(c)。其中的低頻部分(L)表示影像，而高頻部分(H)則表示邊緣。

(a) (b) (c)

圖形 18. wavelet 壓縮的階層特性

以上的過程可以重覆多次，得到一個階層式的金字塔架構，下圖即爲經由二階小波轉換後所得之架構。之後再經由常見的 zero-tree coding 及 arithmetic coding 可以達到高壓縮率及保存良好的品質。



圖形 19. 兩階的 wavelet 轉換

(a) (b) (c)

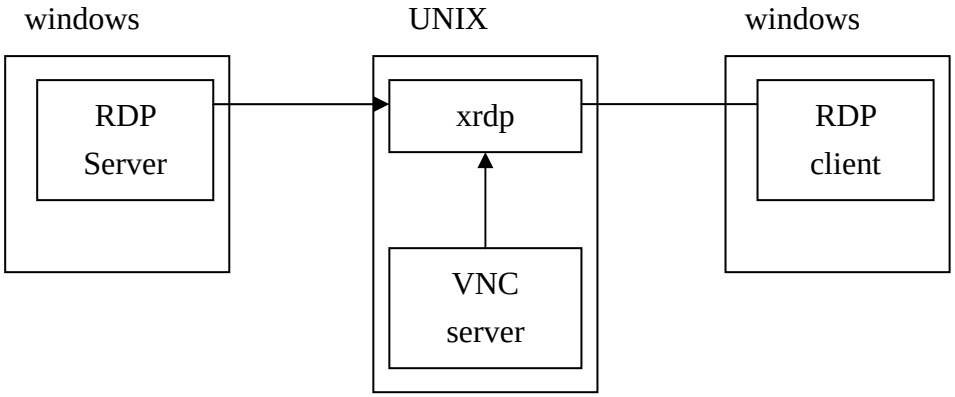
圖形 20. (a) 原圖 (b) JPEG 高度壓縮 (c) wavelet 壓縮

2-5-5 xrdp

上一小節[12]的 VNC proxy 系統，在 RDP 中也有類似的例子，就是 Xrdp。Xrdp 原本是爲了在 UNIX 作業系統上建立一個虛擬的 RDP server，

提供一般的 RDP client 進行連結，然而在背後它提供的是一個 proxy 機制，真正的 server 功能還是得靠其它的 server 提供。

目前它提供了連接至 VNC server 或 RDP server 的模組以取得實際的更新畫面。在 VNC Server 方面，VNC 允許 self looping，也就是 VNC viewer 可以連結至同一部電腦的 VNC server。但 RDP server 則不允許 RDP client 連結至它正在使用的桌面，因此若想讓安裝在 Windows XP 上的 xrdp 連結至 RDP server，該 server 必須位於其它機器上。



圖形 **21. xrdp** 的系統架構

2-6 圖形前景/背景分離技術

在影像處理的領域中，學者提出分離前景與背景的技術不可勝數。早先的研究是針對掃描文件中的文字進行辨識，近年來則發展至螢幕畫面的辨識上。有些是利用前景與背景之亮度作區分，有的則是利用圖形的頻率區分。

2-6-1 MRC 的概念

MRC 是 Mixed Raster Content 的簡寫[13]，是由 XEROX 公司所提出。最早是希望用於彩色傳真的圖形壓縮，然而後來彩色壓縮並沒有盛行，這項技術反倒是運用在反倒是常用於網路文件。

依據 MRC 的規畫，圖被分作三層：前景、遮罩與背景。前景常儲存的是文字與圖示的顏色，乃是單純的色彩。至於文字與圖示的位置則交由遮罩所決定，它是一個二階的影像，由於它的重要性大於其它圖層，因此使用非失真的方式來壓縮。背景則通常較爲複雜，因此可利用失真的方式壓縮。

(a) (b) (c) (d)

圖形 22. (a) background (b) mask (c) foreground (d) result

2-6-2 分離前景/背景的技術

關於前景與背景分離的技術有許多種，但主要可以歸納爲 top-down 與 bottom-up 兩種方式。

所謂的 top-down 方式，是將圖形切割成多個不同的區域，之後再辨識各區域爲文字或影像。這種方式較適合用於辨識兩階調影像或擁有單純背景色的影像。

至於 bottom-up 方式，則是利用某些特徵將文字很精確地萃取出來，例如頻率、文字走向、高度等等的資訊，比起前一種方式較爲複雜。

2-6-3 DjVu

DjVu[14]是由 AT&T 依據 MRC 架構所研發的技術，主要的目的是在減少在 Internet 中傳輸高品質與高解析度的掃描影像的資料量。它也是一種檔案格式，可以利用專門的 viewer 讀取利用 DjVu 壓縮的圖形檔。

DjVu 將一張圖形的遮罩利用非失真的 JBIG2 壓縮方式，背景利用失真的 wavelet 壓縮。

DjVu 的目的不是在找出文字精確的位置，而主要是在分離，因此它採用的是 top-down 的區分方式。它利用 K-mean 的方式將圖形拆成多個區塊，在各區塊中反覆地找尋正確的前景與背景值，並儘量將相鄰的區塊設定爲接近的前景與背景，隨後再依各像素與前景/背景相似的程度歸入其中。

2-6-4 應用於 Thin Client 上的圖層分離技術

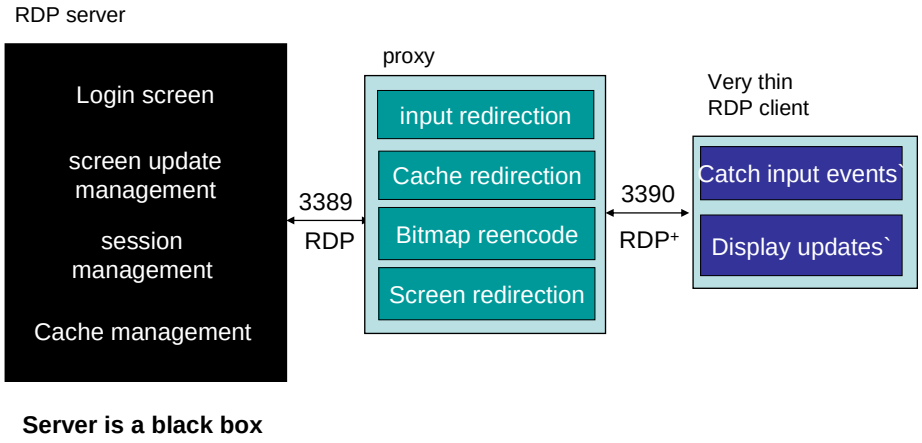
對於視窗桌面的影像，許多部分是文字與圖形混雜的，例如設定了複雜背景的桌面，若採用失真壓縮，易造成其中的文字模糊難以識別；若採用非失真的方式壓縮，則壓縮率不高，造成資料量龐大。

爲了解決此問題，可以將圖層分離的技術應用於壓縮視窗桌面圖形之上。然而由於 Thin Client 顯示的圖形有其即時性的需求，因此無法像 DjVu 等分析掃描文件的技術使用非常複雜的方式。

在[15]的 paper 中提出的方式是將整個圖形以 16*16 像素爲單位畫分，將它們分爲 Text/Graphic 或 Photo 兩類，前者以類似 VNC 的方式壓縮，後者以 Jpeg 的方式壓縮。首先，系統會檢查每格圖形的顏色數作爲初步的畫分依據，顏色數少於 32 色的歸爲 Text，而多於 32 色的則進行進一步分解動作，全部過程如下圖所示。

第 3 章 系統設計步驟

我們的系統主要是根基於 xrdp 與 rdesktop 之上，達成桌面分享並減少資料量之目的。系統功能如下圖，其中 RDP server 對我們而言是一個黑箱子，因爲沒有公開程式碼，因此無法了解其內部運作過程，也無法加以更改。



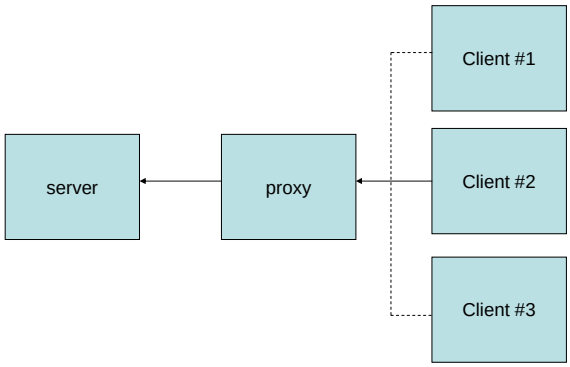
圖形 23. 我們的系統架構

3-1 提供多人連線的 RDP proxy

由於原始的 xrdp 只提供一個使用者的連線，因此第一步我們得修改 xrdp，使其能符合多人分享的目的。

3-1-1 輸入的控制

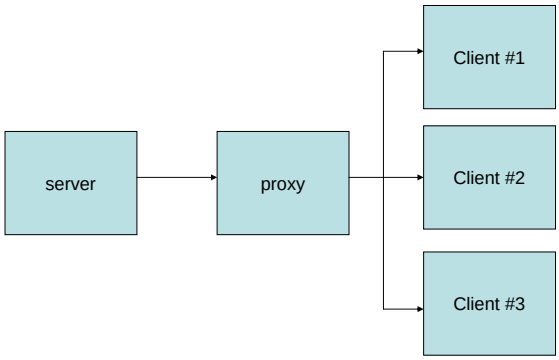
對於使用者的輸入而言，我們發覺如果允許多個使用者同時輸入，會讓情況變得混亂而難以控制。因此我們打算一次只讓一個使用者來控制。預設的情況下是第一個連線的 client 擁有控制權。而其它 client 傳入的 input 則會被忽視。



圖形 **24. rdp proxy** 進行輸入的控制

3-1-2 輸出畫面更新

proxy 必須負責讓由 RDP server 所產生的螢幕更新正確地傳至每個連線的 client。



圖形 **25. rdp proxy** 將畫面傳給多個客戶端

3-1-3 維護快取的一致性

由於 RDP client 大多數更新畫面時都是從快取中取得，因此如果 client 端的快取與 server 端的快取不一致時便會造成顯示的錯誤。

這種情況會發生於第二個以後的使用者與 proxy 連線。當 RDP server 通知各個 RDP client 存取快取中的圖形時，第二個連線中的快取可能還是空無一物。因此會在螢幕上產生不連續的區塊。

比較可行的做法有以下三種

1. 不理會所遇到的 cache miss 情況

使用這個策略的結果是讓沒有建立cache的client端產生多個不連續的區域。不過也是最省力的做法。

2. 先建立完整的快取

這種做法是在 proxy 傳送任何 orders 之前先將 proxy 端全部的 cache 資料傳送到 client 端。這種做法的缺點在於需傳送較多資料量，因爲以 rdesktop 爲例，一個完整的 cache #3 資料就有 262 筆資料，每筆資料若皆爲 1KB，則需傳送 262KB 的資料量。

3. 依據 client 的需求動態給予 cache 之值

這樣的做法表面看起來很合理，然而當 client 數變多時，相對地會讓 proxy 的負擔也加大。

4. proxy 只傳送部分的快取至剛建立的 client 端

這個做法是結合了(1)與(2)，考慮到 client 端實際上並不會用到那麼多的快取資料，因此只需傳送較常用的幾個即可。

3-2 混合非失真與失真壓縮

在 RDP 中內建的非失真壓縮雖然對文字畫面能達到良好的壓縮比率及清晰度，但是並不能有效地減少播放影片及瀏覽網頁時不時出現的動畫效果產生的龐大資料量，這些大資料量往往降低了操作的順暢，同時

3-2-1 RDP 壓縮的特性

RDP 使用的圖形壓縮方式屬於非失真壓縮，意即其解壓縮後的圖形與原圖一致，相較於其它失真壓縮時對於高頻的文字部分難以辨識的，RDP 非失真的壓縮讓帶有文字的圖形易於閱讀，同時其壓縮後的資料量也很小。這是因爲文字通常只有前景色與背景色。至於對於顏色/形狀複雜的圖形 RDP 的非失真壓縮的壓縮率不高，是它的缺點。

3-2-2 Wavelet 壓縮的特性

wavelet 的壓縮方式與其它的圖形壓縮方式一樣，是將圖形先由 RGB 色域轉 YUV 色域，接著分別對 YUV channel 進行壓縮。由於這個特性，我們可以對各個 channel 加予不同的壓縮率，例如表示亮度的 Y 較高，而表示彩度的 UV 較低。

3-2-3 RDP 圖形壓縮之資料量表

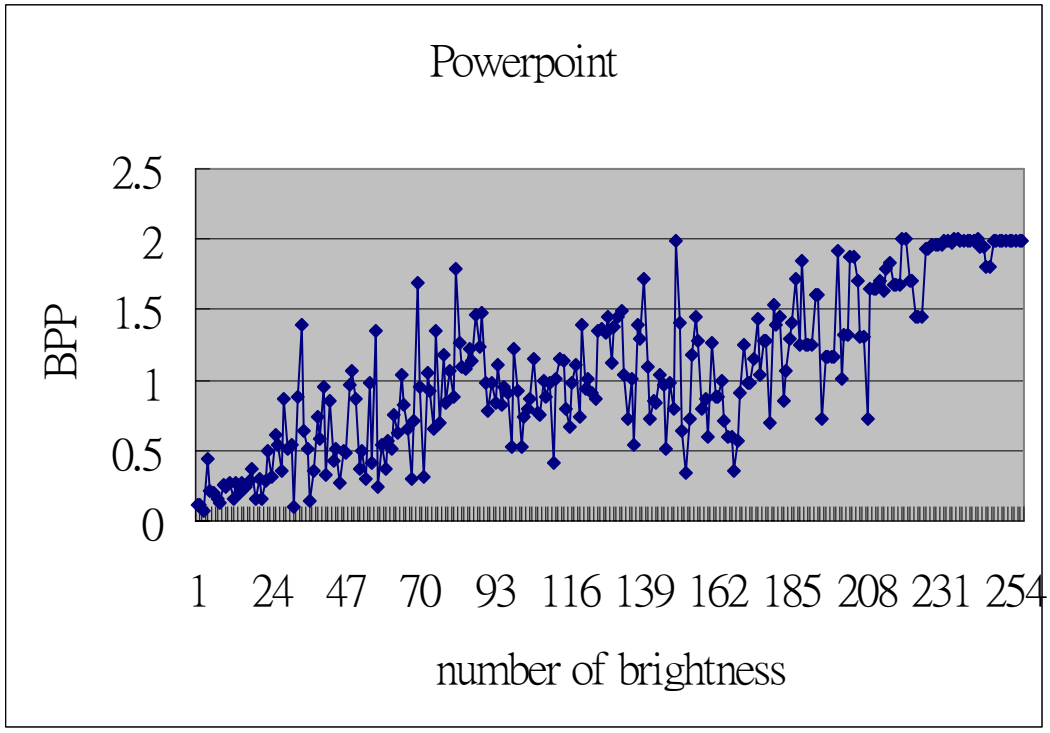
以一張大小爲 64 * 64 像素，一個像素以 16 Bit 來儲存的圖形而言，在未經壓縮前，其資料量爲 8192 Bytes，經由 RDP 壓縮後的資料量如下表所示。

表格 6 RDP 壓縮率比較

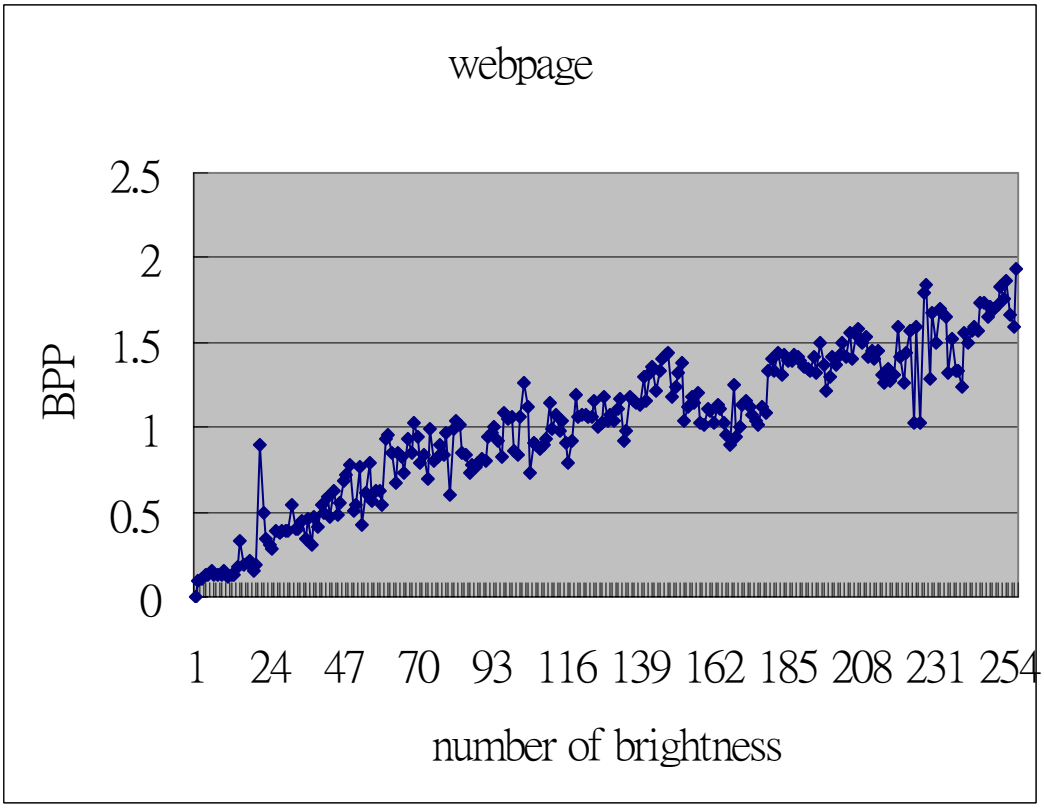
	2 色	4 色	8 色	16 色	32 色	256 色
						
資料量	25	29	36	50	142	895
Bpp	0.006	0.007	0.0085	0.01	0.03	0.213
壓縮率	99.7%	99.7%	99.6%	99.4%	98.3%	89%
						
資料量	869	5260	6452	7209	7341	8130
Bpp	0.21	1.25	1.53	1.71	1.75	1.93
壓縮率	89.4%	36%	21.3%	12%	10.4%	0.8%

上表顯示了兩個極端的例子，從中可以發現，RDP 處理單純色塊的圖形時能得到較佳的壓縮率，即使是 256 色時也可以壓縮 89%的資料量。然而對於複雜的圖形，RDP 即便是在 32 色以下資料量仍非常大。因此對於複雜圖形，應改爲採用失真的 wavelet 壓縮方式。

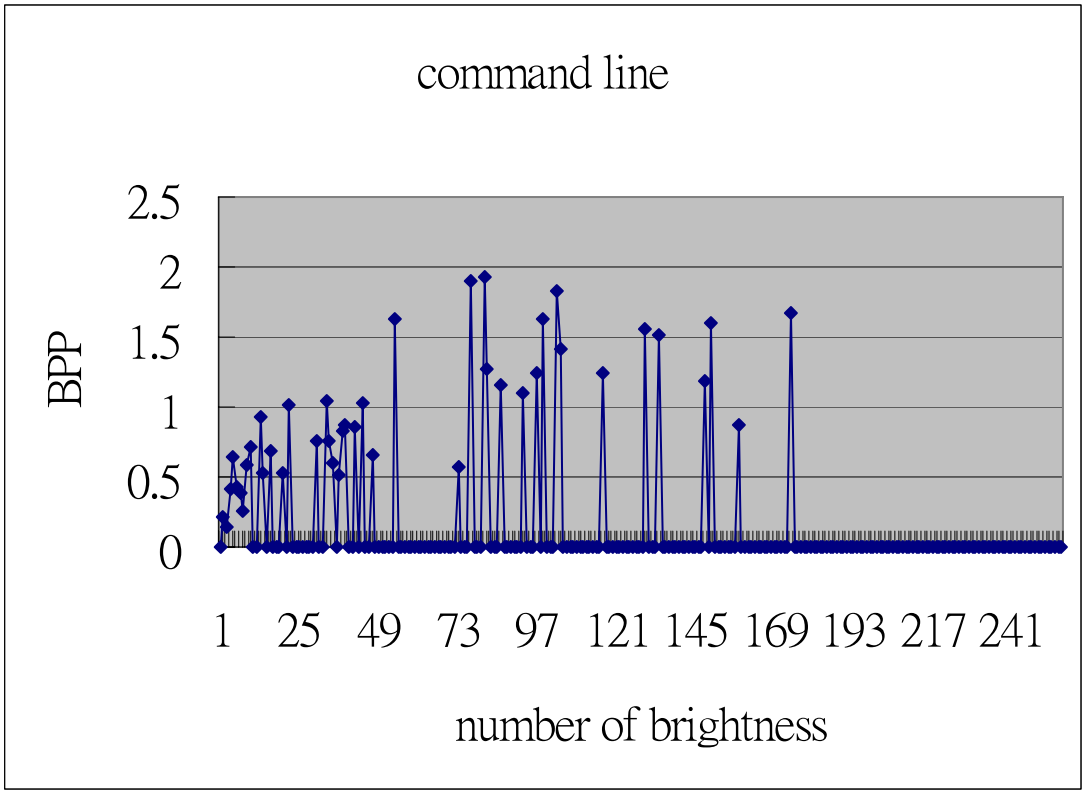
爲了能進一步了解在平均情況下 RDP 的顏色數目與資料量之間的關係，我們對不同情況做了以下的統計。爲了統計方便，我們不直接計算圖形的顏色數目，因爲這樣就有 2^24 種；我們先將圖形由 RGB color space 轉換至 YUV color space，之後計算 Y channel(亮度)的數目，因此在不失代表性的前提下能快速進行統計。



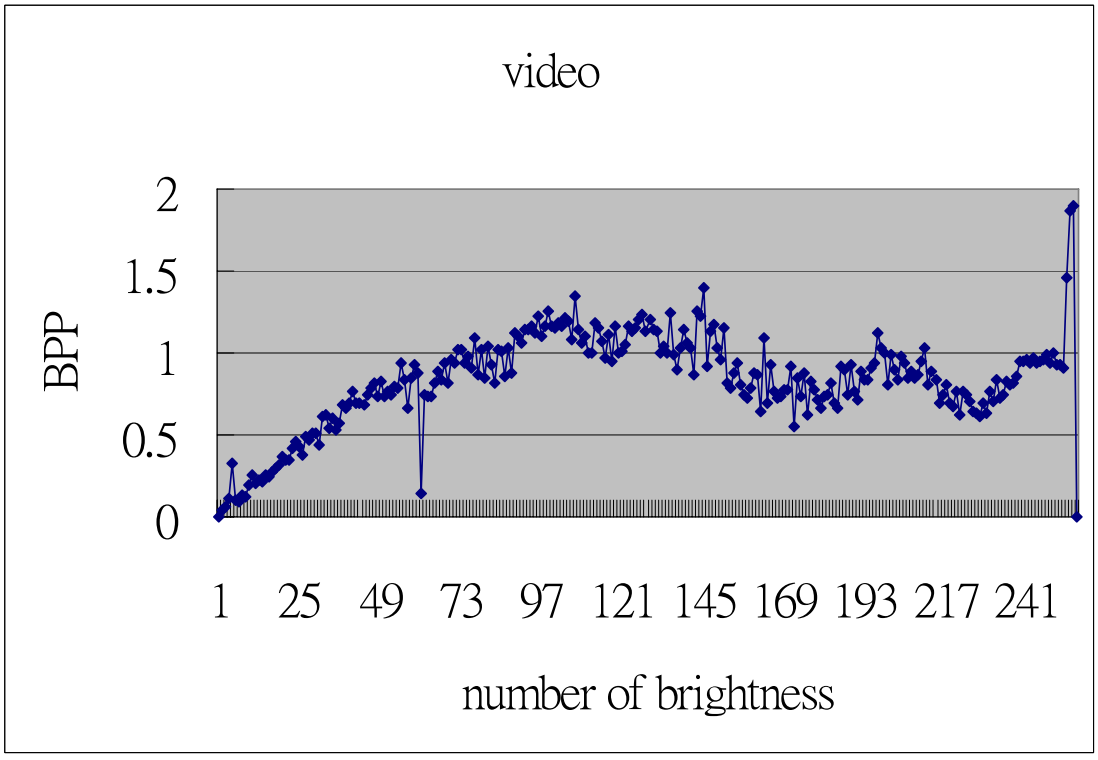
圖形 **26.** 播放 **powerpoint** 時的圖形亮度數與資料量關係



圖形 **27.** 瀏覽網頁時的圖形亮度數與資料量關係



圖形 **28.** 執行指令視窗時的圖形亮度數與資料量關係



圖形 **29.** 播放影片時的圖形亮度數與資料量關係

當螢幕畫面以 800*600 的圖形更新時，我們希望能在 192Kbps 的頻寬限制下傳輸，因此理想的 Bpp 爲 0.4Bpp，而以 64*64 像素爲大小的單位畫面更新之資料量則約爲 1676 bits，相當於 210Byte。在參考利用 RDP 的非失真壓縮所得的平均資料量後，我們將亮度低於 24 色者以 RDP 的失真方式加以壓縮，將高於 24 色者以分層式的方式加以壓縮，方式將於下一小節中討論。

3-2-4 圖形的分層

由前一小節可知，使用 RDP 內建的壓縮方式較合適於含有文字的圖形，而使用會失真的 wavelet 壓縮則適合色彩數較多的複雜圖形。

在我們的作法中，我們將圖形區分爲重要的部分與次要的部分。重要部分指的是圖形的輪廓，一旦變形將難以辨識，因此我們採用非失真壓縮來處理這個部分；至次要部分則可以利用失真壓縮加以處理。

我們的方法考慮到作爲 Presentation 的畫面特性，提出以下幾點作爲我們分割圖形的考量。

(1) 如何找出文字/圖示輪廓並保存之。

- 桌面視窗通常是單色背景搭配單色文字(前景)。
- 無論是找出前景色或儲存背景色都能達到保存輪廓的目的。
- 由於人眼於亮度的變化比較敏感，因此只需對 Y channel 作處理。

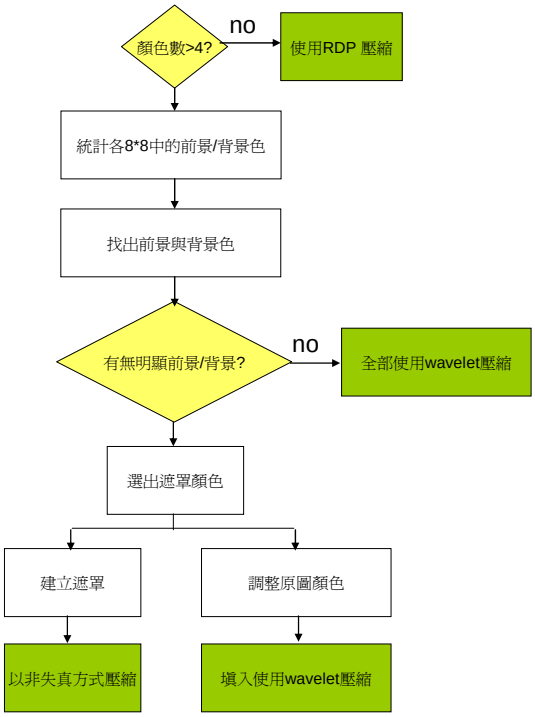
(2) 如何找出圖形中的前景色與背景色

- 背景通常是畫面中占最多顏色者，前景通常是占次多顏色。
- 文字通常只占圖形的一小部分，但它是所有作爲前景色中最多的。
- 爲避免找出錯誤的顏色，將圖形以 8*8 區隔，並統計出現最多的前景色與背景色。
- 黑色與白色是預設也是最常見的前景色與背景色，因此優先考慮。

(3) 如何進一步比較前景與背景色的重要性，選出最重要的作爲遮罩代表的顏色。

- 若圖形是前景文字橫跨多色背景，選擇前景色。
- 若圖形是由單一背景色與多個前景色組成，選擇背景色。

(4) 爲確保以低 Byte Limit 壓縮的次要圖形不會過度扭曲，將位於遮罩色位置的像素改填入其相鄰的色彩。



圖形 30. 決定圖形採用的壓縮方式流程圖

3-3 預覽模式

預覽模式的概念主要是源於使用者操作時通常會因爲突發的大量資料而讓使用者的輸入被 block 住。這種時候與其忠實地顯示原本的圖片，不如先快速地讓使用者了解它操作的反應狀態，如果發現是開啓了影片檔，能快速地關閉，以免讓影片播放吃光了所有的網路頻寬。

3-3-1 主要想法

預覽模式是在資料量變大時，暫時呈現粗糙的畫面，以維持一定程度的傳輸效率，待資料量減少時，則回復到原始的畫面品質。

3-3-2 預覽模式的啓用時機

(1) 拖曳滑鼠

拖曳滑鼠的動作即使用用按下滑鼠鍵不放並移動滑鼠指標的位置。

由於 RDP server 的設計，在大部分應用程式裡拖曳滑鼠只會顯示外框及滑鼠指標，因此對於資料量並沒有顯著影響，然而在少部分操作過程中，會隨滑鼠位置產生大量畫面更新。

(2) 捲動頁面

捲動頁面可以藉由在視窗上的捲動軸拖曳滑鼠，或是利用滑鼠中鍵的滾輪達成。

捲動頁頁時，頁面中央受利於 cache 的幫助，因此不會再傳送額外的圖形，然而頁面的上下部分則會依據捲動的程式產生多個矩形更新區域。

3-3-3 預覽模式的實作

預覽模式的實作主要需配合 Order Queue 的結構。並可以從兩個方面著手：(1) 圖形的壓縮比率(2) 圖形的色彩精確度 Order Queue 的概念與 THINC 中的 Command Queue 相仿，主要也是讓產生的 order 不會直接傳給 client 端，因此能依據情況進行流量的控制。

3-3-3-1 動態調節資料量

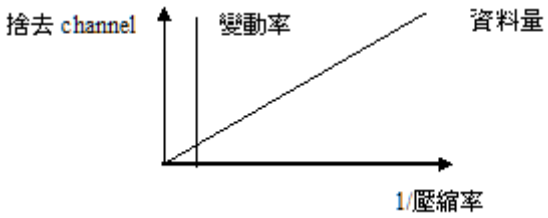
在我們的系統中，主要考慮改變以下兩個項目來調節資料量。

- **圖形的壓縮比率**

當資料量傳輸大時，可以藉由增加壓縮比率來減低暴增的資料量所帶來的衝擊。

- **圖形的色彩精確度**

一般圖形可以區分爲 Y U V 三個 channel，Y 代表亮度，而 U V 代表彩度。人眼對於亮度比彩度來得敏感，因此在資料過多時可以捨去 U V channel，保留 Y 即可。



圖形 31. 畫面變動率、產生資料量與壓縮率的關係

3-3-3-2 Queue 的運作模式

Queue 是一個 Linking List 的資料結構。在 xrdp 中建立兩個 queue，一個是 real time queue，也就是產生的 update 會直接被 flush out，另一個 queue 則儲存著次要的圖形彩度資料，每隔一分鐘傳送一次。每次 flush Queue 時得依循先進先出的規則，因此得從該 linked list 的開頭開始 flush 資料

將圖形加入 Order Queue 主要有以下兩種情況：

1. 本秒鐘資料量不超過上限

   將圖形的亮度與彩度資訊皆置入 real time queue 中

2. 本秒鐘資料量超過上限

   將圖形的亮度資訊置入 real time queue；彩度資訊置入 order queue。

當圖形要被 flush 出 Order Queue 時主要有以下兩種情況：

1. 上秒鐘未超過資料上限

   表示只是偶發性的資料量暴增，將 order queue 中的彩度資訊傳輸至客戶端。

2. 連續兩秒鐘的資料量皆超過上限

   表示是持續性的資料量暴增，例如使用者在播放電影，因此將彩度資訊自 Order Queue 中移除。

3-3-3-3 Order Queue 的管理

除了上節所述的情況外，系統需偵測當播放影片時，某個區域的畫面呈現在螢幕上不久後隨即被新的更新畫面所覆蓋的情況。

在 THINC 系統中將被覆蓋的情況會作部分覆蓋及完全覆蓋：如果是部分覆蓋則直接在 Queue 內將它們重新格式化爲不會相交的畫面更新；如果是完全覆蓋時則將舊的更新從 Queue 中去除。

鑑於分割更新可能會帶來額外的處理時間，以及對於之後 cache 的處理不一致的問題。在我們的系統中並不會進行分割的動作，如果遇到被覆蓋的更新時，無論它是屬於部分覆蓋或完全覆蓋，皆捨去被覆蓋者的 U V 頻道。

第 4 章 實驗數據與討論

在本章中我們比較了各個系統的傳輸資料量。由於 Netmeeting 只支援 8 bpp 的環境，而我們的系統（以 MRC 表示）則只支援 16bpp，因此分成兩個小節分開來進行比較。

4-1 直接擷取、Netmeeting、VNC 與 RDP 資料量比較

本節中先針對 Netmeeting、VNC 及 RDP 這三個系統的資料量進行比較。

4-1-1 測試環境

分享桌面：800*600，8 bpp

直接擷取 1：Microsoft Media Encoder 9 Screen Codec

直接擷取 1：Microsoft Media Encoder 9 Codec

RDP：Windows 2003，RDPv5.2

VNC：TightVNC 1.3 dev7，Hextile 壓縮

NetMeeting：Microsoft Netmeeting 3.01

4-1-2 測試項目

網頁：Windows 2003 作業系統中的說明及支援文件

Wordpad：輸入交大校歌

Command Line：連續執行 dir/s 10 次

Powerpoint：混合圖形及文字的投影片，每五秒切換

video：640*480 大小，長三分鐘

4-1-3 測試數據

表格 7 RDP、Netmeeting 及 VNC 資料量比較

網頁	Wordpad	Command Line	Powerpoint	Video

直接擷取 1	155 Kbps	16.8 K bps	137 Kbps	80 Kbps	222 Kbps
直接擷取 2	250 Kbps	250 Kbps	250 Kbps	250 Kbps	250 Kbps
RDP	52 Kbps	13 Kbps	113 Kbps	124 Kbps	1009 Kbps
VNC	291 Kbps	75 Kbps	207 Kbps	219 Kbps	272 Kbps
Netmeeting	202 Kbps	42 Kbps	179 Kbps	184 Kbps	442 Kbps

4-1-4 討論

雖然 Netmeeting 所使用的 T.128 通訊協定與 RDP 非常相似，但由數據可以看出雖然同在 8bpp 的環境下，Netmeeting 所產生的資料量比起 RDP 都大，這可能是因爲 Netmeeting 的 bitmap cache 很小，或是沒有使用 cache 之故，因此無法有效利用 bitmap cache 節省傳輸資料量的原意，資料量比起完全沒有利用 cache 之 VNC 而言相差無幾。

觀查 video 的項目中可以發現 RDP 的資料量比起 VNC 或 Netmeeting 反而大得多。這不是因爲它們的壓縮效率好，而是由於 VNC 屬於 client pull update，因此如果使用者不隨時移動滑鼠，畫面就無法更新；而 Netmeeting 的更新的頻率比 RDP 低得多，因此產生較低的資料量與畫質，造成斷斷續續的畫面。

由以上的數據可知，雖然 Netmeeting 使用與 RDP 類似的 T.128 進行桌面分享，但是其效能非常不理想，因此有改進的必要性。

4-2 RDP、WAVELET、MRC 模式資料量比較

本節中比較了使用原始的 RDP 進行圖形壓縮、完全以 wavelet 進行壓縮以及混合了兩者的 MRC 方式所產生的資料量比較。

4-2-1 測試環境

分享桌面：800*600，16 bpp

RDP Server：Windows 2003

RDP proxy：xrdp beta 2

RDP client：rdesktop 1.4

wavelet encoder：GWIC[16]，Y 250 Bytes，U 100 Bytes，V 50 Bytes

4-2-2 測試項目

網頁：Windows 2003 作業系統中的說明及支援文件

Wordpad：輸入交大校歌

Command Line：連續執行 dir/s 10 次

Powerpoint：混合圖形及文字的投影片，每五秒切換

video：640*480 大小，長三分鐘

4-2-3 測試數據

表格 8 **RDP、wavelet 及 MRC 資料量比較**

	網頁*	Wordpad	Command Line	Powerpoint	Video
RDP	174.5 Kbps	14 Kbps	73Kbps	329 Kbps	1255Kbps
wavetlet	70 K bps	15.5 Kbps	同 RDP	75 Kbps	306 Kbps
MRC	92.8K Bps	同 RDP	同 RDP	207 Kbps	261Kbps

4-2-4 瀏覽網頁效能分析

以表格 4-1 中的數據可以發現，利用 wavelet 能得到最佳的壓縮比，然而由於高度的壓縮，使得畫面變得難以判別，如下圖。

圖形 32. 全部以 wavelet 壓縮的圖形結果

相對地利用 MRC 的方法壓縮所得到的資料量是 RDP 的一半，同時亦能保持文字的清晰度，如下圖。

圖形 33. 以 MRC 的方式壓縮的圖形結果

4-2-5 Powerpoint Presentation 分析

在播放 Powerpoint 時，由於投影片中大部分是兩色的文字圖形，會以原本的 RDP 方式壓縮，使得表面上綜合的資料量 MRC 的方式並沒有顯注的資料量改善。然而若將投影片的各部分區分，則可拆成如下表三個部分。

表格 9 RDP、MRC 於 powerpoint 之資料量比較

元素	MRC 資料量	RDP 資料量
bi-level Mask	20 Kbps	169 Kbps
wavelet 背景	25 Kbps	
RDP 文字	162 Kbps	

由此可知利用 MRC 能將原本 160Kbps 的資料量減少爲 45Kbps，有顯著的改善。

4-2-6 影片效能分析

在播放影片的部分，可以明顯的發現利用 wavelet/MRC 的方法產生的資料量是 RDP 的壓縮方式的四分之一至五分之一。這是由於影片圖形的不規則性極不利於 RDP 內建的非失真壓縮，因此改用失真壓縮時資料量能大幅減少。

若觀查 wavelet 及 MRC 的壓縮方式，可發現 MRC 比 wavelet 產生的資料量更小，這主要是 MRC 的 mask 會將與其相近的亮度合併。這樣的做法也會在畫面上產生區塊的副作用，不過 thin client 本身並不合適進行影片播放，也不是作爲 presentation 的重點。

4-3 綜合比較

依據以上兩節所進行的比較，我們將各個系統的整體表現比較如下表。

表格 10 RDP、MRC 於 powerpoint 之資料量比較

	色彩表現	快取機能	節省資料量	流暢度	影片播放
直接擷取	***	無	*****	***	*****
VNC	*****	無	**	**	*
RDP	*****	*****	***	*****	*****
wavetlet	**	同 RDP	*****	同 RDP	同 RDP
MRC	***	同 RDP	****	同 RDP	同 RDP

由上表可以看出，RDP 在色彩表現及流暢度的方面都比其它機制優異，而資料量方面雖有快取機制輔助，但是受限於其內建的非失真壓縮，使

得資料量稍大。我們利用 wavelet 壓縮的方式降低其處理複雜圖形時所產生的資料量，然而卻讓部分的文字過於模糊難以辨識，爲此我們利用額外的資料量保存圖形中的重要外形。

利用以上的方式，我們的系統同時保存了 RDP 操作流暢的優點，同時又能在節省資料量的同時保存圖形文字的可辨識性。

第 5 章 結論

5-1 結論

即時的視窗桌面的可以用於數位內容的分享與軟體教學等等，這些都是目前網際網路的資源分享尚未發展成熟的型態。相較於其它現有的視窗桌面分享系統，我們提出的架構能節省更多的傳輸資料量，同時提供清晰的視窗畫面。

我們的論文重點集中在改進利用 RDP Thin Client/Server 系統的傳輸資料量，以提供在低頻寬的網路中進行桌面視窗分享的功能。

首先我們分析 RDP 的運作模式，得知其將圖形與文字分開傳送，由此我們得以在不影響文字顯示的條件下藉由非失真的 Wavelet 壓縮方式來壓縮圖形，一方面降低原本 RDP 壓縮處理複雜圖形所產生的龐大資料量，同時也保持文字的清晰。

其次我們做進一步的觀查時，發現在 RDP 中有部分的文字是以圖形的方式傳輸。此時若對它們加以高壓縮率的 Wavelet 圖形壓縮，會造成裡面的文字模糊難以辨識。針對這些圖形，我們利用 MRC 的圖層概念將圖形中的最重要的文字輪廓採以兩階影像儲存，並以非失真的方式壓縮後傳輸至客戶端，以減低利用 wavelet 對圖形高度壓縮時產生的變形與失真，讓圖形在資料量與辨識度等方面取得良好的平衡。

最後爲了能更進一步地節省資料量，我們利用人眼對亮度較爲敏感的特性，分別傳輸圖形亮度（Y）與彩度（UV）的資料。其中亮度資料會在產生後立即傳輸至客戶端，而彩度資料則會在資料量大時被置入 Queue 中固定時間間隔傳輸至客戶端；若兩個單位時間內資料量連續超過上限值時，將圖形彩度的資訊自 Queue 中捨棄，以達成節省資料量並保持整體操作的流暢度之目的。

我們的系統在畫面上重點文字外觀的保存，在犧牲部分次要圖形的畫質下達成精簡資料量之目的，以符合網際網路的低頻寬特性。採用我們改進的方式後，利用遠端桌面播放 Powerpoint 投影片時比起原本的 RDP 能將節省三分之一的資料量；瀏覽網頁時能節省二分之一的資料量；至於播放影片時則是五分之一的資料量，在 ADSL 256Kbps 的環境下能操作得更順暢。

5-2 未來發展

由於我們的時間與能力有限，無法將系統的每個部分做得盡善盡美，因此將它們列爲往後研究的方向，如下所示。

5-2-1 使用專爲兩階層圖形設計的壓縮方式

在 RDP 中傳送文字與我們的 MRC 架構中的 mask 皆是兩階層的圖形，目前無論是 RDP 或是我們的系統都未對這些 bi-level 的文字使用專門的壓縮。

在未來我們會研究各種常用的 bi-level 圖形壓縮方式，例如 JBIG2，以期能更進一步節省資料量。

5-2-2 增強系統的操作性

目前我們的 client 端系統是架構在 Cygwin/X 的環境下運作，在效能及使用上皆不適合一般使用者。

另一方面由於 RDP Server 的特性，使得 proxy 及 Server 無法架構在同一部機器上運作，未來預計讓 RDP proxy 能更有彈性地連結至本機的 VNC server 或是遠端的 RDP Server。

經由以上幾點可知，整個系統距離真正讓多使用者進行桌面分享尚有許多工作有待我們後續的努力。

5-2-3 更進一步利用 Wavelet Progressive 的特性

由於我們的系統的特性，當預覽模式啓動時所傳輸的圖形皆是經過高度壓縮，而且會被存於快取之中。稍後若再需要顯示該圖形時，RDP 會自快取中讀取這些高度失真的圖形，因此即使這時系統並不是在預覽模式下，畫面仍會顯示粗糙的畫面。

爲了解決這個問題，我們可以進一步利用 wavelet 壓縮的 progressive 特性，將置於 rdesktop 的 wavelet decoder 具備漸進解壓縮的特性，也就是

5-2-4 使用 P2P 的傳輸資料

目前我們的系統是由 proxy 負責將桌面影像傳給多個使用者。當使用者數增多時，對於 proxy 所在的電腦負荷也會增大，進而可能降低傳輸的效能。因此在未來可以改爲以 P2P 的傳輸方式傳輸桌面影像，讓不同的使用者彼此分享自己收到的桌面影像，進而減少 proxy 的負擔。

5-2-5 加入註解、聲音及影像

我們的論文主題主要是針對桌面影像傳輸所進行的最佳化，但作爲一個完整的桌面分享系統，需要更多的互動。因爲網路媒體的最大特性，就是使用者間能進行互動，一個缺乏互動的系統很難吸引人使用。

在未來我們規劃畫在系統上加入使用者間的影音互動，以及如同 powerpoint 的手寫註解功能，讓各個使用者能彼此分享自己的心得，使得整體的桌面分享動作更有意義。

參考文獻

[1] T. Y. Liu, Y. C. Huang and W. C. Chen, "A novel algorithm for real-time full screen capture system," *Multimedia Signal Processing, 2001 IEEE Fourth Workshop,* pp. 395-400, 2001.

[2] S. H. Chang, S. T. Lee and J. M. Ho, "AN EFFECTIVE APPLICATION-LAYER CONTROL FOR REAL-TIME SCREEN RECORDING,"

[3] J. Nieh, S. J. Yang and N. Novik, "A Comparison of Thin-Client Computing Architectures," *Network Computing Laboratory, Columbia University, Technical Report CUCS-022-00, November,* 2000.

[4] Richardson, "Virtual network computing," *IEEE Internet Computing,* vol. 2, pp. 33, 1998.

[5] M. Mauve, "Protocol Enhancement and Compression for X-Based Application Sharing,"

[6] G. F. Pinzari, "NX X Protocol Compression,"

[7] R. Baratto, J. Nieh and L. Kim, "THINC: A Remote Display Architecture for Thin-Client Computing,"

[8] Lai, "On the performance of wide-area thin-client computing," *ACM Transactions on Computer Systems,* vol. 24, pp. 175, 2006.

[9] S. F. Li, Q. Stafford-Fraser and A. Hopper, "Frame-buffer on Demand: Applications of Stateless Client Systems in Web-based Learning," *5th International Conference on Information Systems Analysis and Synthesis (ISAS'99),* 1999.

[10] S. F. Li, Q. Stafford-Fraser and A. Hopper, "Integrating synchronous and asynchronous collaboration with virtualnetwork computing," *Internet Computing, IEEE,* vol. 4, pp. 26-33, 2000.

[11] P. Ziewer and H. Seidl, "Transparent Teleteaching," *ASCILITE 2002: 749,* vol. 75, 2002.

[12] A. M. and H. A., "Intelligent adaptation framework for wireless thin-client environments," in 2004,

[13] R. L. de Queiroz, R. Buckley and M. Xu, "Mixed raster content (MRC) model for compound image compression," *Proc.IS&T/SPIE Symp.on Electronic Imaging, Visual Communications and Image Processing,* vol. 3653, pp. 1106–1117,

[14] L. Bottou, P. Haffner, P. G. Howard, P. Simard, Y. Bengio and Y. LeCun, "High quality document image compression with DjVu," *Journal of Electronic Imaging,* vol. 7, pp. 410–425, 1998.

[15] Lin, "Compound image compression for real-time computer screen image transmission," *IEEE Transactions on Image Processing,* vol. 14, pp. 993, 2005.

[16] GNU Wavelet Image Codec, http://www.jole.fi/research/gwic/