視窗程式設計期末報告

PC網路聊天室(TCP/IP)

組員：

沈政光　B11170008

林俊鋐　B11170025

分工：

沈政光　Word文書、圖表

林俊鋐　C#程式碼、PPT

# 第一章 緒論

## 1.1研究背景與動機

隨著網際網路與通訊協定技術的成熟，即時通訊（Instant Messaging, IM）已滲透於現代生活與工作之中。即時文字與圖片傳輸等功能，不僅促進了社交互動，也廣泛應用於企業內部通訊、客服系統與遠距協作工具。

在本專題中，我們選擇實作一款以 PC 為平台、採用 TCP/IP 協定的主從式 (Client-Server) 聊天室系統，結合文字即時傳輸、圖片分享與視覺互動功能，目的在於深化對底層網路傳輸機制與 Windows 應用開發整合的理解。不同於一般使用中心伺服器的多人聊天室，本系統強調雙方直接建立連線通訊的設計，有助於掌握點對點網路的特性與挑戰。

使用 C# 搭配 Windows Forms 技術進行開發，不僅能活用其在桌面應用中的穩定性與易用性，更能藉由 .NET 提供的多執行緒與資料串流支援，有效實作即時、圖文整合的網路聊天功能。

## 1.2研究目的

本專題旨在設計並開發一套基於 TCP/IP 的 PC 主從式聊天室，具體目標如下：

1. 理解 TCP/IP 通訊流程：學習建立與維持 TCP 連線的方法，掌握封包交換、訊息接收與發送的實作細節。
2. 實作簡易主從式模式：由一台主機擔任伺服器角色負責監聽，客戶端則與之建立直接的 TCP 連線進行通訊，藉此探討實作難點。
3. 實現圖文即時通訊功能：包含文字訊息、表情符號插入與圖片傳輸，並具備圖片預覽與儲存等附加功能。
4. 運用多執行緒技術：確保使用者介面操作不被封鎖，並正確處理併發網路事件。
5. 開發直覺式使用者介面：設計簡潔明確的聊天與圖片互動視窗，提升使用者體驗。

## 1.3問題陳述

在本系統的開發過程中，預期會遇到以下幾項技術挑戰：

1. 連線穩定性與握手確認：如何設計清楚的連線建立與確認流程，並處理連線超時或拒絕的狀況。
2. 圖片資料的有效傳輸與解析：圖片在網路傳輸前需進行壓縮、編碼與封裝，接收端需正確還原為可顯示圖片，過程涉及資料流與格式處理。
3. 訊息資料格式的區分與判讀：需設計可靠的封包標記方式，區分文字、表情或圖片等資料型態。
4. 非同步與執行緒切換：接收與發送操作多數需背景執行，避免阻塞主執行緒導致 UI 停滯。
5. 雙方圖文顯示同步與 UI 回應：確保訊息內容與圖片能即時出現在對方聊天室畫面，並保持介面的互動流暢。
6. 錯誤處理與使用者提示設計：設計使用者友善的錯誤提示與回饋機制，提升系統健壯性與操作容錯性。
7. 多用戶連線:

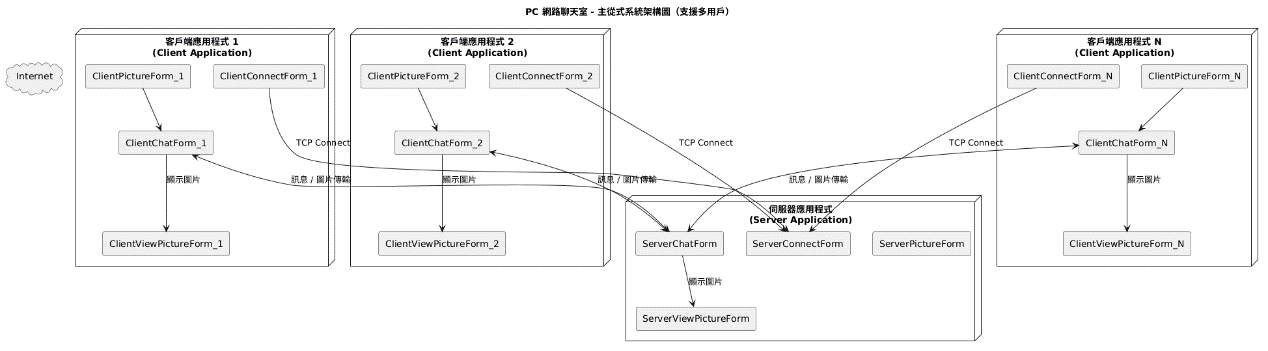
## 1.4預期貢獻

本專題完成後，預期可產出以下成果與價值：

1. 強化實務程式設計能力：熟悉 C# 在桌面應用中對 UI、網路與圖片處理的整合實作方式。
2. 建立可執行的主從式聊天室雛型：支援基本圖文傳輸功能，作為 TCP/IP 應用開發與教學範例。
3. 深化對 TCP 傳輸與資料封包的理解：實務演練 TCP 資料流的建立、封裝、解碼與處理。
4. 驗證圖片通訊的可行架構：包含圖片選取、轉碼、傳輸與互動檢視的完整流程設計。
5. 提供錯誤處理與使用者互動的實作經驗：可作為後續更大型網路應用設計的基礎參考。

# 第二章 系統設計

## 2.1系統架構圖

2.1.1架構概述

本系統採用**主從式 (Client-Server)** 架構，由一個伺服器端應用程式與多個客戶端應用程式構成。伺服器負責監聽並接受來自客戶端的連線請求，而客戶端則主動向伺服器發起連線。一旦連線建立，雙方即可進行一對一的即時文字與圖片通訊。此設計允許多個客戶端同時與伺服器建立獨立的聊天會話。

整體系統由以下主要模組構成：

1. 連線管理模組：負責建立與維護 TCP 連線，包括客戶端的連線請求與伺服器的監聽與接受。
2. 聊天通訊模組：提供圖形化文字訊息互動介面，支援訊息發送、接收與表情符號插入。
3. 圖片處理模組：處理本地圖片選取或圖片 URL 輸入，並將圖片編碼後傳送；同時負責圖片接收、顯示與儲存。
4. 圖像互動檢視模組：提供圖片縮放與平移功能，提升圖片互動性。
5. 使用者介面模組：整合多個表單控制元件與事件觸發，確保操作流程直觀順暢。

## 2.2 功能模組說明

### 2.2.1 連線管理模組（Connection Management）

伺服器端 (ServerConnectForm.cs)：

1. 啟動 TcpListener 並監聽指定通訊埠。
2. 當偵測到客戶端連線請求時，會彈出對話框供伺服器使用者選擇『接受』或『拒絕』 。
3. 接受連線後，會為該客戶端建立一個獨立的聊天視窗，同時伺服器會繼續監聽，準備接受下一個客戶端的連線請求，從而實現同時處理多個聊天會話的能力

客戶端 (ClientConnectForm.cs)：

1. 使用 TcpClient 嘗試連線至目標 IP 與埠號。
2. 顯示連線進度與等待時間，若逾時則提示錯誤。
3. 接收到 <ACCEPT\_CHAT> 即表示成功建立聊天室通訊。

### 2.2.2聊天通訊模組 (Chat Communication)

主要表單：ClientChatForm.cs, ServerChatForm.cs

功能細節：

1. 傳送訊息：將輸入文字透過網路串流送出（使用 Encoding.Unicode 編碼）。
2. 接收訊息：背景執行緒不斷監聽並處理資料流。
3. 表情符號：提供一個彈出式的表情符號選單，使用者點擊後會將對應的 **Unicode 原生表情符號 (例如 "😊", "😂")** 直接插入訊息輸入框中進行傳送。
4. 訊息類型判別：支援 <IMAGE> 標記判斷是否為圖片訊息。

### 2.2.3圖片傳輸模組 (Image Transfer)

檔案位置：ClientPictureForm.cs, ServerPictureForm.cs

功能：

1. 使用者可選取本地圖片（JPG、PNG、GIF 等）或輸入 URL。
2. 圖片會被編碼為 Base64 字串，並前綴 <IMAGE> 標記送出。
3. 傳送前支援圖片預覽，避免誤傳或格式錯誤。

### 2.2.4圖片預覽與儲存模組 (Image Preview & Save)

檔案位置：ClientViewPictureForm.cs, ServerViewPictureForm.cs

功能：

1. 顯示接收到的圖片（由聊天模組傳遞進來）。
2. 提供縮放（滑鼠滾輪）與平移（滑鼠拖曳）功能。
3. 可按鈕儲存圖片到本地資料夾，預設儲存格式為 PNG。

### 2.2.5使用者介面模組 (User Interface)

檔案位置：所有 .cs 表單程式與 .Designer.cs 自動產生檔案

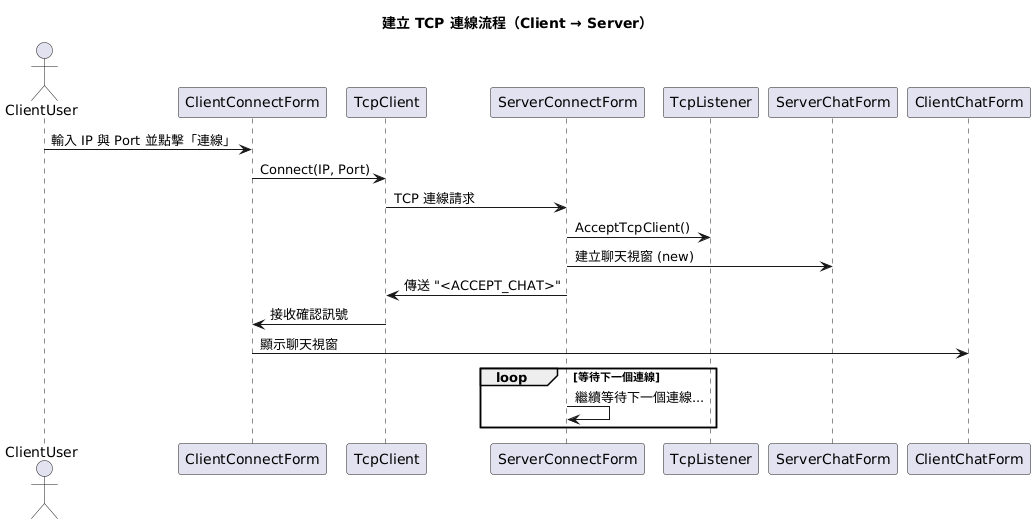
功能：

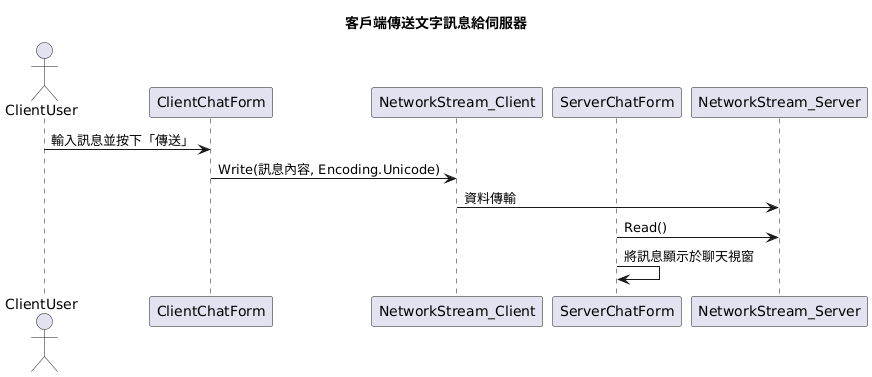
1. 管理主視窗與功能子視窗（連線、聊天、圖片處理等）。
2. 各控制項（按鈕、TextBox、ListBox、PictureBox）以事件驅動方式互動。
3. 支援動態顯示聊天記錄、圖片內容與系統提示訊息。

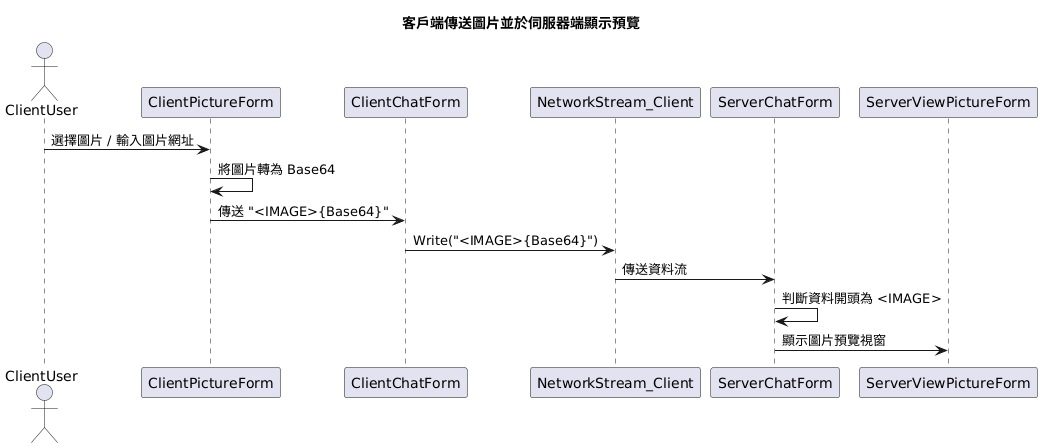
## 2.3類別圖、循序圖等 UML 圖表

### 2.3.1類別圖

### 2.3.2循序圖

客戶端發起連線流程

傳送文字訊息流程

傳送圖片流程

# 第三章 系統實現

## 3.1開發環境（Visual Studio 版本、.NET 版本等）

* 開發工具：Microsoft Visual Studio Community 2022 (64 位元)
* .NET 框架版本：Microsoft .NET Framework 4.8
* 程式語言：C#
* 執行平台：Windows 10 桌面應用程式
* UI 技術：Windows Forms (WinForms)

## 3.2核心功能代碼解析

本系統採用「事件驅動＋背景執行緒」架構來處理聊天訊息與圖片傳輸。以下為核心功能的實作概述：

### 3.2.1 TCP 連線建立

1. 客戶端使用 TcpClient 類別連接指定的 IP 與通訊埠。連線成功後會顯示聊天視窗。
2. 伺服器端透過 TcpListener 建立監聽器，等待來自遠端的連線請求，並在確認後回傳 <ACCEPT\_CHAT> 訊號表示允許通訊。

TcpClient client = new TcpClient();

await client.ConnectAsync(serverIP, serverPort); // Client 端連線

TcpListener listener = new TcpListener(IPAddress.Any, port);

listener.Start();

TcpClient acceptedClient = listener.AcceptTcpClient(); // Server 接受連線

### 3.2.2 訊息傳輸與接收

傳送訊息時，使用 NetworkStream.Write() 將編碼後的字串送出。

接收端以背景執行緒持續監聽 NetworkStream.Read()，接收到資料後依照內容格式解析。

byte[] buffer = Encoding.Unicode.GetBytes(message);

networkStream.Write(buffer, 0, buffer.Length); // 傳送訊息

byte[] receiveBuffer = new byte[8192];

int bytesRead = networkStream.Read(receiveBuffer, 0, receiveBuffer.Length);

string receivedText = Encoding.Unicode.GetString(receiveBuffer, 0, bytesRead); // 接收訊息

特定標記如 <IMAGE> 或 <emoji:名稱> 用於判斷訊息類型。

### 3.2.3 圖片處理與傳輸

選取圖片後，圖片會被轉為 Base64 字串並加上 <IMAGE> 標記再傳送。

接收端遇到該標記會觸發圖片預覽視窗。

Image img = Image.FromFile(filePath);

using (MemoryStream ms = new MemoryStream()) {

img.Save(ms, ImageFormat.Png);

string base64Image = Convert.ToBase64String(ms.ToArray());

SendMessage("<IMAGE>" + base64Image); // 傳送圖片

}

## 3.3關鍵技術實現說明

### 3.3.1 多執行緒與 UI 更新

為防止 UI 卡頓，資料接收與圖片處理等動作皆以背景執行緒執行，並使用 Invoke() 方法將結果回傳至 UI 執行緒。

private void ShowReceivedText(string text) {

if (InvokeRequired)

Invoke(new Action<string>(ShowReceivedText), text);

else

listBoxChat.Items.Add("對方：" + text);

}

### 3.3.2 UI 互動流程設計

主流程採用多表單模式，從 ConnectForm → ChatForm → PictureForm / ViewPictureForm 切換操作。

各表單間透過事件參數或共用欄位傳遞連線物件與訊息內容，保持資料一致性。

# 第四章 測試與驗證

## 4.1單元測試案例

雖然本系統主要為 GUI 應用程式，較難採用傳統單元測試工具自動測試，但仍可依模組設計手動進行逐一測試。以下列出主要測試案例：

|  |  |  |
| --- | --- | --- |
| 測試項目 | 測試說明 | 預期結果 |
| 伺服器啟動監聽 | 在 ServerConnectForm 中按下「啟動伺服器」 | 顯示伺服器 IP 與埠號，進入等待狀態 |
| 客戶端發送連線請求 | 在 ClientConnectForm 中輸入正確 IP 與埠 | 成功建立 TCP 連線，雙方顯示聊天視窗 |
| 傳送文字訊息 | 客戶端輸入訊息並按下傳送 | 對方正確接收到訊息，顯示於聊天框 |
| 插入表情符號 | 點擊表情按鈕可將原生表情符號插入輸入框 | 對方能正確接收並顯示該表情符號 |
| 傳送本地圖片 | 客戶端選取一張圖片並傳送 | 對方彈出圖片預覽視窗，顯示圖片內容 |
| 傳送圖片 URL | 客戶端輸入一個圖片網址並送出 | 對方能自動下載圖片並顯示預覽 |
| 縮放與平移圖片 | 在預覽視窗使用滑鼠操作圖片 | 支援放大/縮小與拖曳功能，無畫面錯誤 |
| 儲存圖片 | 點選「儲存圖片」按鈕 | 開啟存檔對話框並成功儲存圖片至本機 |

## 4.2系統整合測試

整合測試重點在於模組間互通、流程完整性與例外狀況應對：

1. 雙方連線流程測試：從啟動伺服器至建立連線、確認成功，驗證是否有逾時或訊號未傳達等問題。
2. 訊息收發連續性測試：連續收發多則文字與圖片訊息，檢查是否有漏訊、錯誤解碼或顯示異常。
3. 圖片處理流程整合測試：從選取、預覽、編碼、傳輸、接收、顯示、儲存等步驟完整執行並逐步驗證。
4. 多操作流程測試：快速切換表單、多圖重傳、多表情插入等操作測試 UI 穩定性與系統反應速度。
5. 例外流程測試：如：
   1. 傳送過大圖片
   2. 網路中斷或對方強制關閉
   3. 未接收到 <ACCEPT\_CHAT> 的處理行為

# 第五章 結論與未來工作

## 5.1研究成果總結

本專題以 C# 搭配 Windows Forms 技術，成功實作出一套基於 TCP/IP 協定的 PC 網路聊天室系統，具備以下核心功能：

1. 客戶端與伺服器間的雙向連線 。
2. 即時文字訊息收發與表情符號插入。
3. 支援本地圖片選取或圖片 URL 傳輸，並提供圖片預覽與儲存。
4. 使用者介面簡潔直覺，操作流程流暢。
5. 運用背景執行緒與非同步處理，確保 UI 響應不中斷。

在開發過程中，實際演練了 TCP 資料流處理、Base64 編碼解碼、WinForms 視窗互動控制與例外錯誤管理，深化了對網路通訊程式設計的整體理解。

## 5.2專題限制

雖已達成預期功能目標，系統仍存在部分限制與待強化之處：

1. 通訊模式限制：雖然伺服器可同時處理多個客戶端，但每次通訊皆為伺服器與單一客戶端的一對一模式，尚未支援多位使用者加入同一聊天室的群聊功能。
2. 安全性不足：資料未經加密，傳輸過程可能遭竊聽或篡改。
3. 錯誤處理仍簡化：如對方強制關閉、封包不完整等狀況，僅提供基本提示，尚未涵蓋完整異常情境。
4. 圖片傳輸效能有限：對於超過數 MB 的圖片傳輸，處理時間會明顯增加。

## 5.3未來改進方向

針對上述限制與潛在應用場景，未來可考慮以下擴充與優化方向：

1. 多使用者支援：加入伺服器端使用者列表與多人聊天功能，改為 Client–Server 架構或多 P2P 網狀拓樸。
2. 傳輸加密機制：導入 SSL/TLS 或 AES 對資料進行加密，強化安全性。
3. 訊息記錄與搜尋功能：新增聊天紀錄儲存、離線訊息快取與關鍵字搜尋等功能。
4. 語音與檔案支援：擴展至語音訊息傳送與一般檔案上傳下載，提高通訊豐富度。
5. 跨平台相容性：改寫為跨平台應用（如 .NET MAUI 或 Electron），支援 macOS、Linux 或 Web 應用。

# 附錄. 參考文獻

引用書籍、論文、網路資源

[**.NET Framework API 參考 version 4.8**](https://learn.microsoft.com/zh-tw/dotnet/api/?view=netframework-4.8&preserve-view=true)