

義守大學資訊工程學系

TCP/IP與網路程式設計

專題研究報告

多人搶分系統

專題學生：方澤勳 11303124A

鄭煜薰 11303123A

李郁翔 11303120A

蔡承祐 11303068A

指導教授：高典良 老師

中華民國一一四年十二月十一日

摘要

本專題利用 Python 的 Socket 模組實作了一個基於 TCP/IP 協定的多人連線遊戲系統。有別於傳統單向的訊息傳輸，本系統設計了一個具備完整生命週期（等待、投票、進行、結算）的遊戲伺服器。系統支援多個客戶端同時連線進入「遊戲房間」，玩家能透過投票機制決定遊戲模式（自動抽獎或手動搶分）。伺服器端採用多執行緒（Multi-threading）技術，同時處理客戶端指令監聽與遊戲主迴圈的倒數計時，並維護一個有限且不重複的獎品池，即時廣播遊戲狀態給所有玩家，模擬真實且刺激的線上抽獎競賽。

目錄

摘要	2
第一章、專題簡介	4
1.1 功能特色	4
1.2 系統架構	5
1.3 協定設計：	5
1.4 安裝與執行	6
第二章、測試結果	7
1. Client 端連線時先輸入暱稱	7
2. 成功連線後等待人數到達	7
3. 人數到達時輸入/ready	7
4. 都準備就緒後，系統會開始統計投票要什麼模式，採多數決，若平手則隨機二選一	8
5. 遊戲進行中畫面(自動模式)	8
6. 遊戲進行中畫面(手動模式)	9
7. 遊戲結束顯示最終積分排行榜，輸入/replay 再玩一次，或/quit 離開。	10
第三章、未來改進方向	12
3.1 圖形化介面與動畫效果：	12
3.2 數據持久化與歷史戰績：	12
3.3 網路穩定性與作弊防護：	12
第四章、參考文獻	13
4.1 Source Code: server.py – Python Server Implementation with Game State Machine & Finite Prize Pool Algorithm.	13
4.2 Source Code: client.py – Python Client Implementation with Dual-threading for Real-time Broadcasting.	13
4.3 Python Software Foundation. "socket — Low-level networking interface." Python 3 Documentation. (用於基礎 TCP 連線、bind、listen、connect 實作)	13
4.4 Python Software Foundation. "threading — Thread-based parallelism." Python 3 Documentation. (用於 Server 併發處理、遊戲計時迴圈及 Client 接收執行緒實作)	13
4.5 Python Software Foundation. "json — JSON encoder and decoder." Python 3 Documentation. (用於 Client 與 Server 間結構化訊息協定交換)	13

第一章、專題簡介

前言

本專題旨在運用 TCP/IP 網路程式設計 課程所學知識，透過 Python 的 socket 模組建立一個高互動性的多人連線搶分遊戲系統。不同於傳統單向請求的抽獎程式，本系統導入了**「遊戲狀態機（Game State Machine）」**的概念，Server 端不僅負責處理連線，更掌控著從「等待玩家」、「全員準備」、「投票表決模式」到「遊戲進行」與「最終結算」的完整生命週期。

系統採用經典的 Client-Server 架構。Server 端負責維護一個有限且不重複的獎勵池（Finite Prize Pool）、處理併發連線、執行隨機或手動抽獎邏輯，並向所有參與者廣播即時戰況；Client 端則提供使用者參與投票、搶分與接收廣播的功能。專案著重於多執行緒（Threading）的實現，確保 Server 能同時監聽指令並執行遊戲倒數迴圈，並利用 JSON 格式 進行結構化資料傳輸，保證通訊的可靠性與擴展性。

1.1 功能特色

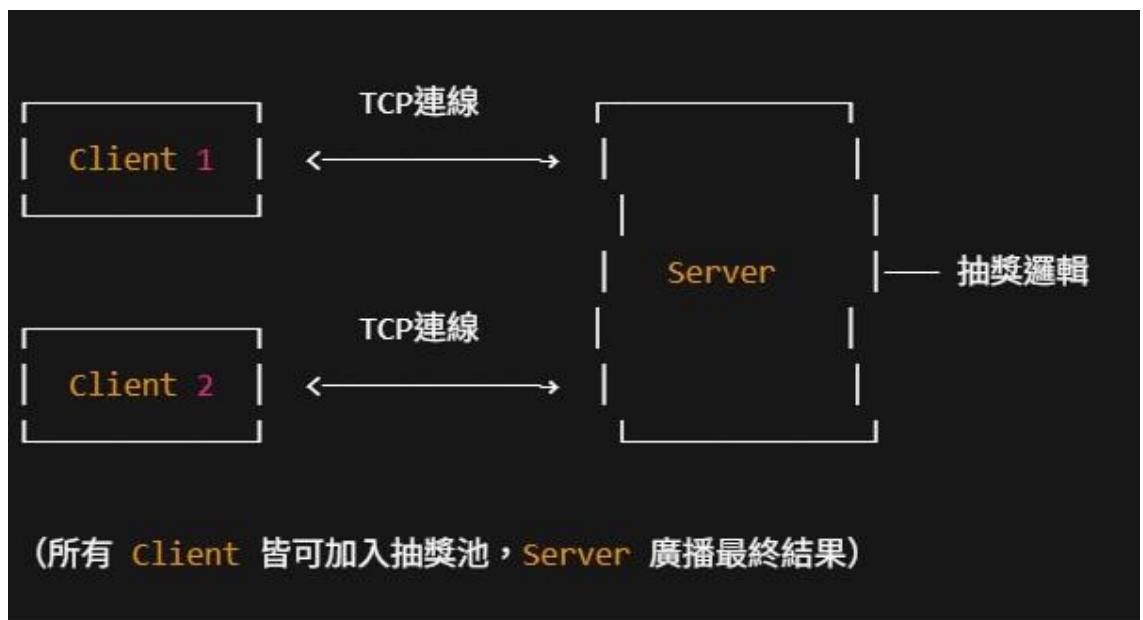
1. **遊戲狀態生命週期管理：** Server 端實作了完整的狀態控制，必須等待所有連線者輸入 /ready 才會進入下一階段，確保遊戲的公平性與同步性。
2. **投票與隨機裁決機制：** 創新導入投票系統，玩家可輸入 /auto 或 /manual 決定遊玩模式。系統採多數決判定；若票數相同，則由 Server 端的隨機演算法強制裁決，增加遊戲的不確定性與趣味。
3. **有限不重複獎池演算法：** Server 於遊戲開始時生成一份固定數量的獎品清單（人數 * n），包含稀有大獎與銘謝惠顧。採用 shuffle 洗牌與 pop 取出的方式，保證獎品不會重複被抽出，且抽完即止，模擬真實的資源搶奪戰。中獎廣播機制：抽獎結果產生後，Server 會即時將中獎者和獎品資訊廣播給所有連線中的 Client，實現即時互動體驗。
4. **雙模式遊戲體驗：**
自動模式（Auto Mode）：由 Server 每秒自動為所有玩家執行抽獎，適合運氣比拚。
手動模式（Manual Mode）：玩家須在 Client 端快速輸入 /draw 指令搶奪獎品，考驗手速與網路反應。
5. **即時廣播與結算系統：** 遊戲過程中的每一次抽獎結果、倒數計時以及最終的積分排行榜，皆透過 Server 的廣播機制即時推播至所有 Client 介面。
6. **離線與重玩機制：**支援 /quit 指令顯示感謝語並安全斷線；遊戲結束後支援 /replay 指令，可快速重置伺服器狀態，無須重啟程式即可進行下一局。

1.2 系統架構

系統採用標準的 Client – Server 架構。Server 端核心包含兩個主要執行緒：

1. Client Handler Thread：負責監聽並處理個別 Client 的 JSON 指令（如投票、搶分）。
2. Game Loop Thread：負責遊戲倒數計時、自動抽獎邏輯執行與廣播。

Client 端則採用雙執行緒設計，主執行緒負責讀取使用者輸入，背景執行緒負責接收 Server 的即時廣播訊息。



1.3 協定設計：

本專題採用 JSON (JavaScript Object Notation) 作為訊息格式，以提供結構化、易於擴展且易於解析的資料交換方式。所有訊息末尾均加上換行符 (\n) 作為訊息的定界符，確保 TCP 數據流可以被準確分割和解析。

Client → Server

動作	核心數據	目的
register	name: [暱稱]	向 Server 註冊身分並進入大廳。
ready	無	告知 Server 已準備就緒，等待遊戲開始。
vote	mode: ["auto" 或 "manual"]	投票選擇遊戲模式。
trigger_draw	無	(手動模式) 請求 Server 從獎池抽出一個獎品。
replay	無	請求 Server 重置遊戲狀態，重新開始。

Server → Client

狀態	核心數據	目的
welcome	message	連線成功的歡迎訊息。
info	message	系統廣播
draw_result	prize, points	回傳單次抽獎的結果與獲得分數。
auto_update	message	(自動模式) 廣播當前所有人的抽獎動態。
error	message	操作錯誤提示 (如：未輪到投票階段)。

1.4 安裝與執行

1. 需求：

Python 3.13

無需額外第三方函式庫，僅依賴 Python 標準庫 (socket, threading, json, random, time, sys)。

2. 安裝：

下載 server.py 和 client.py 檔案至同一資料夾。

3. 執行步驟：

1. 啟動 Server：在終端機運行 python server.py

2. 啟動 Client：開啟新的終端機運行 python client.py (可開啟多個視窗模擬多人)。

3. 遊玩流程：

輸入暱稱進入大廳。

等待人員到齊後，輸入 /ready。

輸入 /auto 或 /manual 進行投票。

遊戲開始後進行搶分或觀看自動抽獎。

第二章、測試結果

1. Client端連線時先輸入暱稱

The screenshot shows two terminal windows side-by-side. The left window is titled 'server' and the right is titled 'client'. Both windows have code editors at the top and a terminal output area below.

Server Terminal Output:

```
PS D:\TCP IP\server> & C:/Users/user/AppData/Local/Programs/Python/Python313/python.exe "d:/TCP IP/server/server.py"
Server 啟動於 0.0.0.0:65432, 需要 2 人開始...
[連線] ('127.0.0.1', 57532) 已連線
[加入] 測試1號 加入遊戲
[連線] ('127.0.0.1', 55936) 已連線
[加入] 測試2號 加入遊戲
```

Client Terminal Output:

```
PS D:\TCP IP\client> & C:/Users/user/AppData/Local/Programs/Python/Python313/python.exe "d:/TCP IP/client/client.py"
請輸入暱稱: 測試1號
指令: /ready, /auto, /manual, /draw, /replay, /quit, /help
✿ 歡迎！請輸入暱稱註冊。
> 測試1號 進場！
> 等待玩家... (1/2)
> 測試2號 進場！
> 人員到齊！請輸入 **/ready** 準備。
```

2. 成功連線後等待人數到達

The screenshot shows two terminal windows side-by-side. The left window is titled 'server' and the right is titled 'client'. Both windows have code editors at the top and a terminal output area below.

Server Terminal Output:

```
PS D:\TCP IP\server> & C:/Users/user/AppData/Local/Programs/Python/Python313/python.exe "d:/TCP IP/server/server.py"
Server 啟動於 0.0.0.0:65432, 需要 2 人開始...
[連線] ('127.0.0.1', 57532) 已連線
[加入] 測試1號 加入遊戲
[連線] ('127.0.0.1', 55936) 已連線
[加入] 測試2號 加入遊戲
```

Client Terminal Output:

```
PS D:\TCP IP\client> & C:/Users/user/AppData/Local/Programs/Python/Python313/python.exe "d:/TCP IP/client/client.py"
請輸入暱稱: 測試2號
指令: /ready, /auto, /manual, /draw, /replay, /quit, /help
✿ 歡迎！請輸入暱稱註冊。
> > 測試2號 進場！
> 人員到齊！請輸入 **/ready** 準備。
```

3. 人數到達時輸入/ready

The screenshot displays two terminal windows side-by-side, illustrating a TCP/IP communication session between a server and a client.

Left Terminal (Server):

- Path: D:\TCP IP\server> & C:/Users/user/AppData/Local/Programs/Python/Python313/python.exe "d:/TCP IP/server/server.py"
- Output:

```
Server 啟動於 0.0.0.0:65432, 需要 2 人開始...
[連線] ('127.0.0.1', 57532) 已連線
[加入] 測試1號 加入遊戲
[連線] ('127.0.0.1', 55936) 已連線
[加入] 測試2號 加入遊戲
```

Right Terminal (Client):

- Path: PS D:\TCP IP\client2> & C:/Users/user/AppData/Local/Programs/Python/Python313/python.exe "d:/TCP IP/client2/client.py"
- Output:

```
請輸入暱稱: 測試1號
指令: /ready, /auto, /manual, /draw, /replay, /quit, /help
> 歡迎! 請輸入暱稱註冊。
> 
> 🎉 測試1號 進場!
> 
> 🎉 人員到齊! 請輸入 **/ready** 準備。
> /ready
> 🎉 測試2號 進場!
> 
> 🎉 人員到齊! 請輸入 **/ready** 準備。
> /ready
> 🎉 測試1號 準備好了 (1/2)
> 
```

4. 都準備就緒後，系統會開始統計投票要什麼模式，採多數決，若平手則隨機二選一

5. 遊戲進行中畫面(自動模式)

The screenshot displays two instances of PyCharm running on a dual-monitor setup. Both instances have tabs for 'server.py' and 'client.py'. The left monitor shows the server side, and the right monitor shows the client side. Each instance has a terminal window at the bottom showing command-line interactions related to the game's logic and player management.

Server Side (Left Monitor):

```
PS D:\TCP\IP\server> & C:/Users/user/AppData/Local/Programs/Python/Python313/python.exe "d:/TCP/IP/server/server.py"
Server 啟動於 0.0.0.0:65432, 需要 2 人開始...
[連線] ('127.0.0.1', 57532) 已連線
[加入] 測試1號 加入遊戲
[連線] ('127.0.0.1', 55936) 已連線
[加入] 測試2號 加入遊戲
```

Client Side (Right Monitor):

```
> > 黃 黃 黃 測試2號 進場 !
> > 藍 藍 藍 人員到齊！請輸入 **/ready** 準備。
> /ready
> > 紅 紅 紅 測試1號 準備好了 (1/2)
> > 紅 紅 紅 測試2號 準備好了 (2/2)
> > > 全員準備就緒 !
> > > 游戲投票選擇模式 :
> > > 輸入 **/auto** (自動抽)
> > > 輸入 **/manual** (手動搶)
> > /auto
> > > 黃 測試1號 投給了 auto (1/2)
> >
```

Both clients show identical log output, indicating they are both ready and have selected the automatic mode for voting.

6. 遊戲進行中畫面(手動模式)

```

server.py
import socket
import threading
import json
import random
import time
import sys

def draw_one_prize():
    prizes = ["銘謝惠顧 (0分)", "銘謝惠顧 (50分)", "銘謝惠顧 (100分)"]
    return random.choice(prizes)

def handle_client(client_socket):
    while True:
        message = client_socket.recv(1024).decode('utf-8')
        if message == '/ready':
            client_socket.send("準備好了".encode('utf-8'))
        elif message == '/draw':
            prize = draw_one_prize()
            client_socket.send(prize.encode('utf-8'))
            print(f"[遊戲中] 測試2號 抽到了: {prize}")
        else:
            client_socket.close()
            break

if __name__ == '__main__':
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('127.0.0.1', 49152))
    server_socket.listen(2)
    print("Server 啟動於 0.0.0.0:65432, 需要 2 人開始...")
    while True:
        client_socket, address = server_socket.accept()
        threading.Thread(target=handle_client, args=(client_socket,)).start()

```

```

client.py
import socket

def receive_messages():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(('127.0.0.1', 49152))
    s.sendall('/ready'.encode('utf-8'))
    response = s.recv(1024).decode('utf-8')
    print(response)
    s.sendall('/draw'.encode('utf-8'))
    response = s.recv(1024).decode('utf-8')
    print(response)
    s.close()

if __name__ == '__main__':
    receive_messages()

```

7. 遊戲結束顯示最終積分排行榜，輸入/replay再玩一次，或/quit離開。

The screenshot shows two PyCharm instances running on a Windows desktop. The left instance is titled 'server' and contains the server.py code. The right instance is titled 'client' and contains the client.py code. Both instances show the game's log output in their terminal panes.

server.py:

```
server.py > draw_one_prize
1 import socket
2 import threading
3 import json
4 import random
5 import time
6 import sys
```

[遊戲中] 測試1號 抽到了: (10分)
[遊戲中] 測試2號 抽到了: 銘謝惠顧 (0分)
[遊戲中] 測試1號 抽到了: 銘謝惠顧 (0分)
[遊戲中] 測試2號 抽到了: 銘謝惠顧 (0分)
[遊戲中] 測試1號 抽到了: (100分)
[遊戲中] 測試2號 抽到了: 銘謝惠顧 (0分)
[遊戲中] 測試1號 抽到了: (10分)
[遊戲中] 測試2號 抽到了: (10分)
[遊戲中] 測試1號 抽到了: (10分)
[遊戲中] 測試2號 抽到了: (10分)
[遊戲中] 測試1號 抽到了: (50分)
[遊戲中] 測試2號 抽到了: (50分)
[遊戲中] 測試1號 抽到了: (50分)
[遊戲中] 測試2號 抽到了: (50分)
[遊戲中] 游戲時間結束，進行結算。
[系統] 結算結果:
--- 最終排行榜 ---
測試1號: 1850 分
測試2號: 1310 分
----- 輸入 /replay 再玩一次，或 /quit 離開。

client.py:

```
client.py > receive_messages
1 import socket
```

[遊戲中] 測試1號: 銘謝惠顧 (0分) | 測試2號: 銘謝惠顧 (0分)
> 0 10s | 測試1號: (100分) | 測試2號: 銘謝惠顧 (0分)
> 0 9s | 測試1號: (100分) | 測試2號: (10分)
> 0 8s | 測試1號: (100分) | 測試2號: (10分)
> 0 7s | 測試1號: (10分) | 測試2號: (10分)
> 0 6s | 測試1號: (10分) | 測試2號: (50分)
> 0 5s | 測試1號: (50分) | 測試2號: (50分)
> 0 4s | 測試1號: (50分) | 測試2號: 銘謝惠顧 (0分)
> 0 3s | 測試1號: 銘謝惠顧 (0分) | 測試2號: (10分)
> 0 2s | 測試1號: (50分) | 測試2號: (50分)
> 0 1s | 測試1號: (10分) | 測試2號: (10分)
--- 最終排行榜 ---
測試1號: 1850 分
測試2號: 1310 分
----- 輸入 /replay 再玩一次，或 /quit 離開。

[遊戲中] 測試1號 抽到了: 銘謝惠顧 (0分)
[遊戲中] 測試2號 抽到了: (10分)
[遊戲中] 測試1號 抽到了: (10分)
[遊戲中] 測試2號 抽到了: (10分)
[遊戲中] 測試1號 抽到了: (10分)
[遊戲中] 測試2號 抽到了: (10分)
[遊戲中] 測試1號 抽到了: (50分)
[遊戲中] 測試2號 抽到了: (50分)
[遊戲中] 測試1號 抽到了: (50分)
[遊戲中] 測試2號 抽到了: (50分)
[遊戲中] 游戲時間結束，進行結算。
[系統] 結算結果:
--- 最終排行榜 ---
測試1號: 1850 分
測試2號: 1310 分
----- 輸入 /replay 再玩一次，或 /quit 離開。

第三章、未來改進方向

3.1 圖形化介面與動畫效果：

目前系統使用 CLI (命令行介面)呈現，未來計畫改用 PyQt 或 Tkinter 模組為 Client 端建立 GUI。

1. 視覺化抽獎：將文字跳動改為類似「拉霸機」或「轉盤」的動畫效果。
2. 即時儀表板：用圖表動態顯示目前剩餘獎品數量與各玩家分數條。

3.2 數據持久化與歷史戰績：

目前所有分數暫存於記憶體中，Server 重啟即消失。未來將引入 SQLite 或 MySQL 資料庫。

1. 帳號系統：紀錄玩家的累積勝場與總得分。
2. 歷史戰績：可查詢過去所有場次的冠軍與獲得的稀有獎品紀錄。

3.3 網路穩定性與作弊防護：

1. 斷線重連：實作 Session 機制，允許在玩家在網路波動斷線後，於一定時間內重連回原房間並保留分數。
2. 防作弊機制：在手動模式下，限制 Client 端的請求頻率 (Rate Limiting)，防止使用「連點程式」進行惡意搶分。

第四章、參考文獻

- 4.1 Source Code: server.py – Python Server Implementation with Game State Machine & Finite Prize Pool Algorithm.
- 4.2 Source Code: client.py – Python Client Implementation with Dual-threading for Real-time Broadcasting.
- 4.3 Python Software Foundation. "socket — Low-level networking interface." Python 3 Documentation. (用於基礎 TCP 連線、bind、listen、connect 實作)
- 4.4 Python Software Foundation. "threading — Thread-based parallelism." Python 3 Documentation. (用於 Server 併發處理、遊戲計時迴圈及 Client 接收執行緒實作)
- 4.5 Python Software Foundation. "json — JSON encoder and decoder." Python 3 Documentation. (用於 Client 與 Server 間結構化訊息協定交換)

