Tutorial

Conexión de Tic Tac Toe 3D en Red Local (LAN)

1. Objetivo del proyecto

El objetivo de este proyecto fue permitir que dos computadoras ejecutaran el mismo programa de Tic Tac Toe 3D y pudieran jugar entre sí utilizando una conexión de red local (LAN), manteniendo la interfaz gráfica original hecha con Tkinter.

Para lograr esto, se implementó un modelo de comunicación cliente–servidor usando sockets en Python

2. ¿Por qué no basta con ejecutar el mismo archivo en dos computadoras?

Aunque ambas computadoras ejecuten el mismo archivo .py, cada programa se ejecuta de forma independiente.
Para que las acciones de un jugador se reflejen en la otra computadora, es necesario un medio de comunicación que permita enviar información entre ambos programas.

Por esta razón, se utilizó comunicación en red mediante sockets.

3. Modelo Cliente–Servidor

Se utilizó el modelo cliente–servidor porque:

- Permite que una computadora cree la partida (servidor)
- La otra computadora se conecte a ella (cliente)
- Se controla fácilmente el turno de cada jugador

Funcionamiento:

- El servidor inicia el juego y espera conexiones.
- El cliente se conecta usando la IP del servidor.
- Ambos intercambian jugadas durante la partida.

4. Uso de Sockets en Python

Los sockets permiten enviar y recibir datos entre computadoras conectadas a una red.

En este proyecto:

- Se utilizó socket.AF_INET para redes IPv4.
- Se utilizó socket.SOCK_STREAM para comunicación TCP (confiable).

Cada jugada se envía como un número entero que representa el botón presionado.

5. ¿Por qué fue necesario usar Threads?

Tkinter utiliza un bucle principal (mainloop) que controla la interfaz gráfica.
Si se ejecutan operaciones de red directamente en este bucle, la ventana se congela.

Por esta razón:

- Las operaciones de red (accept, recv) se ejecutan en **hilos secundarios** (threading.Thread)
- La interfaz gráfica permanece activa y responsiva

Esto evita bloqueos y cierres inesperados del programa.

6. Obtención de la IP local del servidor

Para que el cliente sepa a qué dirección conectarse, fue necesario mostrar la IP local del servidor.

Se creó una función que obtiene automáticamente la IP local del equipo servidor, evitando que el usuario tenga que buscarla manualmente.

Esto facilita la conexión y reduce errores.

7. Control de turnos

Como el juego es por turnos, fue necesario implementar un sistema que evite que ambos jugadores jueguen al mismo tiempo.

Se utilizó una variable de control (mi_turno) que:

- Permite jugar solo cuando es el turno del jugador local
- Bloquea la interacción cuando se espera una jugada remota
- Se actualiza automáticamente al enviar o recibir una jugada

Esto garantiza un juego sincronizado y justo.

8. Envío y recepción de jugadas

Cuando un jugador presiona un botón:

1. Se valida que sea su turno.
2. Se ejecuta la jugada localmente.
3. Se envía el índice del botón al otro jugador por red.
4. El otro programa recibe la jugada.

5. La jugada se ejecuta automáticamente en la segunda computadora.

Este proceso permite reutilizar la lógica original del juego sin duplicar código.

9. Uso del mismo archivo para cliente y servidor

Se decidió usar un solo archivo .py para ambos roles (cliente y servidor) para:

- Reducir duplicación de código
- Facilitar mantenimiento
- Simplificar la ejecución

Al iniciar el programa, el usuario selecciona si desea crear la partida (servidor) o conectarse a una existente (cliente).

10. Ventajas de la solución implementada

- Funciona en red local sin configuración avanzada
- Mantiene la interfaz gráfica original
- Usa conceptos reales de redes (cliente–servidor, sockets, threads)
- Es estable y escalable
- Cumple con buenas prácticas de programación

11. Conclusión

Mediante el uso de sockets, hilos y el modelo cliente–servidor, se logró convertir un juego local en un juego multijugador en red local.
La solución implementada permite comunicación eficiente entre dos computadoras, mantiene la estabilidad de la interfaz gráfica y garantiza el correcto control de turnos.

Este enfoque demuestra cómo un programa existente puede adaptarse para trabajar en red sin necesidad de reescribir su lógica principal.

TUTORIAL

Conexión de TicTacToe 3D entre computadoras en redes diferentes (WAN) usando ZeroTier

Problema a resolver

El juego **TicTacToe 3D** fue desarrollado originalmente para ejecutarse en **una sola computadora (offline)**.

Posteriormente se implementó una versión **LAN**, que permite jugar entre dos computadoras **en la misma red local**, usando sockets y direcciones IP privadas.

Sin embargo, **la conexión LAN no funciona** cuando las computadoras:

- Están en **casas distintas**
- Están en **redes diferentes**
- Están detrás de **NAT o routers**

Objetivo:
Permitir que dos computadoras en **cualquier lugar del mundo** jueguen entre sí **sin modificar la lógica del juego**.

¿Por qué no usar IP pública o port forwarding?

Opciones descartadas:

❌ Port forwarding
❌ IP pública directa
❌ ngrok TCP (requiere tarjeta)
❌ Cloudflare Tunnel (requiere dominio)

Estas soluciones:

- Son complejas
- Dependen del router
- No siempre están disponibles
- No son prácticas para un examen académico

¿Qué es ZeroTier y por qué usarlo?

ZeroTier es:

Una **VPN de capa 2 / 3** que crea una **red virtual privada** entre computadoras, como si estuvieran conectadas al mismo router.

Ventajas:

✅ Gratis
✅ No requiere abrir puertos
✅ No necesita IP fija
✅ Funciona detrás de NAT
✅ Compatible con sockets TCP normales

Conclusión:
ZeroTier permite tratar una conexión WAN como si fuera una LAN.

Instalación de ZeroTier

🔹 **Paso 1: Descargar**

Ir a:

https://www.zerotier.com/download/

Instalar en **ambas computadoras**:

- Servidor
- Cliente

🔹 **Paso 2: Crear red virtual**

1. Crear cuenta en ZeroTier
2. Entrar al panel web
3. Crear una red nueva
4. Copiar el **Network ID**
5. Ejemplo: 76fc96e4989124**

🔹 **Paso 3: Unir las computadoras**

En **ambas PCs**:

1. Abrir ZeroTier
2. “Join Network”
3. Pegar el Network ID

En el panel web:
Autorizar ambas computadoras

Obtener IPs virtuales (muy importante)

Cada computadora recibe una **IP privada virtual**, por ejemplo:

Equipo IP ZeroTier

Servidor 10.84.62.141

Cliente 10.84.62.50

Estas IPs:

- Son privadas
- Funcionan como LAN
- Pueden cambiar ocasionalmente
- Se usan para la conexión