Universidad Nacional Experimental del Táchira

Departamento de Ingeniería Informática

Comunicaciones II

**Informe del juego Ta Te Ti  
(Comunicación Multicast)**

Francisco Javier Santos Galviz C.I 19.033.692

Juan Diego Urbina Chuscano C.I 17.931.845

**Tecnología/ codificación:**

Para desarrollar el juego se utilizó sockets en el lenguaje de programación **java.**

**Métodos de conexión:**

Se realizó una conexión con un servidor y dos clientes para iniciar el juego.

* **ServidorUDP:** El Servidor Broadcast del juego se encontrará en la clase UDPServer.java la cual funciona de la siguiente manera:

Se toma la lista de interface de red de la máquina para seleccionar la dirección por la cual se hará el envío de invitación a la red del juego por medio de Broadcast. con los siguientes comandos:

NetworkInterface nif = NetworkInterface.getByName ("eth3");

List<InterfaceAddress> addressEnum = nif.getInterfaceAddresses();

InterfaceAddress address = addressEnum.get(0);

InetAddress localAddress = address.getAddress();

InetAddress broadCastAddress = address.getBroadcast();

Con esto obtenemos la dirección real de la maquina usando la red Ethernet, así como también la dirección del broadcast por donde se transmitirá el mensaje. Teniendo esto se crea el datagrama con el mensaje de la invitación al juego y se envía por dicha dirección broadcast, junto a un puerto especifico.  
Validaciones: En el mensaje se envía un token numérico para identificar la aplicación y que solo se conecten quienes tengan el mismo token.

Al Iniciar como servidor También se crea una instancia de un ServidorTCP en Unicast el cual se mantendrá escuchando, para cuando se conecten los clientes empezar el juego.

Problema: debe ingresarse el nombre de la interface de red por la cual se conecta la maquina donde se jugará ejemplo si se usa ethernet “eth3” o el numero q corresponda a su adaptador de red ethernet, si es por wifi, “wlan0” o el numero q corresponda a su adaptador de red wifi.  
solución: puede recorrerse el listado de interfaces y seleccionar cuál de ellas está corriendo y cual no es virtual para enviar por allí la invitación al juego sin necesidad de establecer en el método cual es nombre del adaptador que se está usando.

* **ClienteUDP**. El cliente Broadcast Se encuentra en una clase llamada UDPClient.java la cual funciona de la siguiente manera:

El cliente Crea un nuevo DatagramSocket por el mismo puerto en el cual está corriendo la aplicación del servidor. Para oír los mensajes enviados desde el servidor.

Luego se crea un DatagramPacket para obtener la información del mensaje. Hecho esto se procesa el mensaje obteniendo entre sus datos el token para validar q sea la misma aplicación y el puerto por el cual se conectará vía Unicast para empezar a jugar.

Hecho esto se cierra el socket UDP y se conecta por medio de TCP como cliente para empezar a jugar y se muestra la ventana del jugador y se espera por una siguiente conexión.

* **ServidorTCP:** El Servidor Unicast del juego se encontrará en la clase TCPServer.java la cual funciona de la siguiente manera:

Tiene un método para escuchar las conexiones de los jugadores en el cual se esperan que se conecten 2 clientes con ello empezando el funcionamiento del juego.se crea un socket tcp pa ra escuchar las conexiones entrantes. Para guardar las conexiones se utiliza LinkedList<Socket> usuarios = new LinkedList<Socket>();

Al existir 2 conexiones se cierra el ciclo de escucha y se crea una nueva instancia de este servidor TCP con un hilo y se corre el método run() con los datos de los dos clientes conectados.

En el método run se crea un DataInputStream y un DataOutputStream para el envío y recepción de las jugadas Realizadas por los clientes. Aquí se hace llamado al método que compara las jugadas para saber si son válidas y si existe un ganador. También envía a los clientes quien es el jugador de turno. Y envía los datos a ambos jugadores.

En este método se inicializa la matriz lógica del juego.

El mensaje enviado a los clientes lleva el turno del jugador, la matriz de jugada y la posición de la jugada actual para ser pintada.

* **ClienteTCP**. El cliente unicast Se encuentra en una clase llamada UDPClient.java la cual funciona de la siguiente manera:

En el método run se crea un DataInputStream y un DataOutputStream para el envío y recepción de las jugadas Realizadas por los clientes.

Se recibe el mensaje enviado desde el servidor con los datos de las jugadas y se procesan para pintarlos en la ventana y verificar si existe un ganador, si aún quedan movimientos, y modificar la matriz de jugadas.

al momento que un usuario interactúa con la parte grafica del juego y selecciona donde realizar un movimiento, se toman los datos se hacen las validaciones y se envía nuevamente los datos al servidor con la posición de la jugada actual. Y se repite el ciclo hasta que haya un ganador.

**Funcionamiento:**

Al estar emparentados ambos clientes por medio de un servidor **se** procederá de la siguiente manera:

* Empezará de forma aleatoria los turnos. Se tendrán 3 fichas por cada jugador,
* Cuando un jugador mueve una ficha se enviará un mensaje al servidor el cual contendrá los datos de la jugada, siendo ya verificado que sea permitido el movimiento.
* Si un jugador utiliza sus 3 fichas, en su siguiente movimiento deberá elegir una de las q ya existen en el tablero y moverla a alguna de las casillas que estén vacías y que el juego señale como un movimiento permitido. Para esto existe una ayuda grafica de los posibles movimientos a realizar.
* La aplicación Realizara todas las validaciones pertinentes en cada movimiento para determinar el ganador del mismo, así como la comprobación del turno de juego que mostrara gráficamente. Si alguno resulta ganador o no es aun su turno el juego mostrara una alerta con el mensaje indicado según sea el caso.

**Limitaciones**:

Por concepto de fallas en la obtención de los puertos y direcciones disponibles no se logró realizar una conexión multicast que funcionara en la red de los laboratorios pro eso se realizó la conexión del juego por medio de broadcast y no con la creación de grupos de multidifusión como debía haberse realizado.

No existe una serie de validaciones para comprobar que exista desconexión por parte de alguno de los clientes. Esto pudo haberse solucionado utilizando un mensaje cada intervalo de tiempo por parte de los clientes para decirle al servidor q se mantiene en línea y en caso de pasar un tiempo q no existan respuestas ni jugadas el servidor procediera a realizar la desconexión de dicho cliente y el reinicio del servicio ara esperar s que existan nuevamente dos jugadores.