

Escuela Superior de Cómputo

Proyecto 3: Chat Linux-Android

Application Development for Mobile Devices

Becerra Ramírez Luis Arturo

Islas Hernández Miguel Ángel

Martínez Méndez Eduardo Isai

3CM14

Profesor: Cifuentes Álvarez Alejandro Sigfrido

Fecha de entrega: 22 de junio del 2021

**INTRODUCCIÓN**

**Socket Multicast Java**

La operación de multicast consiste en enviar un único mensaje desde un proceso a cada uno de los miembros de un grupo de procesos, de modo que la pertenencia a un grupo sea transparente al emisor, es decir, el emisor no conoce el número de miembros del grupo ni sus direcciones IP.

Un grupo multicast está especificado por una dirección IP clase D y un puerto. Las direcciones IP clase D están en el rango 224.0.0.0 a 239.255.255.255, dentro de este rango existen direcciones reservadas, en concreto, la 224.0.0.1 y la 224.0.0.255. El resto de direcciones del rango pueden ser utilizadas por grupos temporales, los cuales deben ser creados antes de su uso y dejar de existir cuando todos los miembros lo hayan dejado.

Java proporciona una interfaz de datagramas para multicast IP a través de la clase MulticastSocket, que es una subclase de DatagramSocket, con la capacidad adicional de ser capaz de pertenecer a grupos multicast. La clase MulticastSocket proporciona dos constructores alternativos:

* **MulticastSocket():** que crea el socket en cualquiera de los puertos locales libres.
* **MulticastSocket(int port):** que crea el socket en el puerto local indicado.

Un proceso puede pertenecer a un grupo multicast invocando el método joinGroup(InetAddress mcastaddr) de su socket multicast. Así, el socket pertenecerá a un grupo de multidifusión en un puerto dado y recibirá los datagramas enviados por los procesos en otros computadores a ese grupo en ese puerto. Un proceso puede dejar un grupo dado invocando el método leaveGroup(InetAddress mcastaddr) de su socket multicast.

Para enviar datos a un grupo multicast se utiliza el método send(DatagramPacket p, byte ttl), este método es muy similar al de la clase DatagramSocket, la diferencia es que este datagrama será enviado a todos los miembros del grupo multicast. El parámetro TTL, Time-To-Live, lo pondremos siempre a 1, valor por defecto, para que sólo se difunda en la red local.

Para recibir datos de un grupo multicast se utiliza el método receive(DatagramPacket p) de la clase DatagramSocket superclase de MulticastSocket.

**DESARROLLO**

**Imagen que contiene Texto

Descripción generada automáticamente  
Figura 1. Servidor iniciado, en espera de conexión de clientes**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja**

**Figura 2. Pantalla inicial de la aplicación, solicita un nombre de usuario**

**Figura 3. Nombre de usuario “Lalo” introducido, conectado al chat grupal**

**Texto

Descripción generada automáticamente con confianza baja  
Figura 5. Cliente iniciado desde PC, solicitando nombre de usuario para iniciar**

**Imagen que contiene Texto

Descripción generada automáticamente  
Figura 6. Usuario “Miguel conectado”, muestra a usuario “Lalo” como conectado**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteImagen que contiene Calendario

Descripción generada automáticamenteTexto

Descripción generada automáticamente**

**Figura 9. Mensaje enviado “Como te va?”**

**Figura 8. Mensaje enviado “Hola Miguel!”, escribiendo mensaje “Como te va?”**

**Figura 7. Aplicación muestra a usuario “Miguel” como conectado**

**Texto

Descripción generada automáticamente con confianza baja**

**Figura 10. Mensajes “Hola Miguel!” y “Como te va?” recibidos por usuario “Miguel” en PC**

**Imagen que contiene Texto

Descripción generada automáticamente**

**Figura 11. Usuario “Miguel” envía mensaje “Bien, y a ti?”**

**Texto

Descripción generada automáticamente**

**Figura 12. Usuario “Lalo” recibe “Bien, y a ti?” de usuario “Miguel”**

**Texto

Descripción generada automáticamente con confianza baja**

**Figura 13. Información recibida por el servidor durante la transmisión del chat.**

**CONCLUSIONES**

**Generales:** A partir de este proyecto, podemos idealizar de una forma básica como funcionan las comunicaciones entre móviles, implementando servidores a nivel ordenador y permitiendo la transmisión de datos entre diversos cliente, ya sea de móvil o en PC.

**Becerra Ramírez Luis Arturo:** La comunicación es muy importante, con ella se logran muchas cosas, prueba de ello es la historia donde conflictos como guerras se resolvieron mucho más rápido con ayuda de ella comunicación, anteriormente habíamos logrado comunicar computadoras del mismo sistema operativo con la ayuda de cables ethernet pero esta vez lo hicimos comunicando un chat desde Linux y un chat de Android, es así como logramos encontrar otra forma de hacerlo, apoyándonos en otras materias para lograr que el proyecto funcione.

**Islas Hernández Miguel Ángel:** Con este proyecto pusimos en práctica un concepto super importante para nuestra carrera, la comunicación entre diferentes sistemas, comunicando este chat desde Linux con el mismo chat en Android, además aplicando conocimientos de otras materias de la carrera como son las redes.

**Martínez Méndez Eduardo Isai:** Pienso que este proyecto nos sirvió de ayuda para plantear que los conocimientos adquiridos en otras unidades de aprendizaje pueden ser aplicados a nivel móvil, en este caso, el uso de los sockets Multicast implementados en móvil, nos ayudan a tener un chat donde la información llega a todos los clientes, y en este caso, pueden existir mas de 2 dispositivos conectados, corriendo mas clientes desde PC o corriendo la aplicación en otros dispositivos móviles.

**BIBLIOGRAFIA**

1. <http://chuwiki.chuidiang.org/index.php?title=Socket_multicast_en_java>
2. <http://www.sc.ehu.es/acwlaalm/sdi/Laboratorio_UDP_IPcast.pdf>