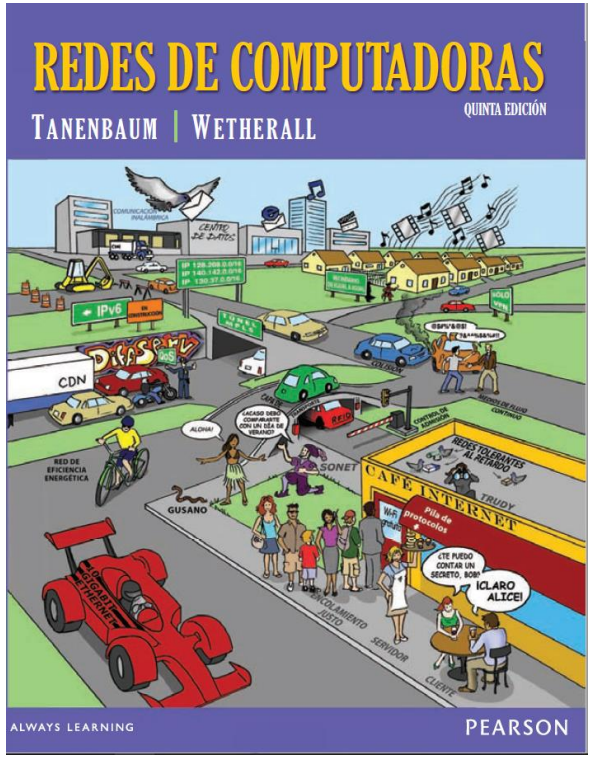
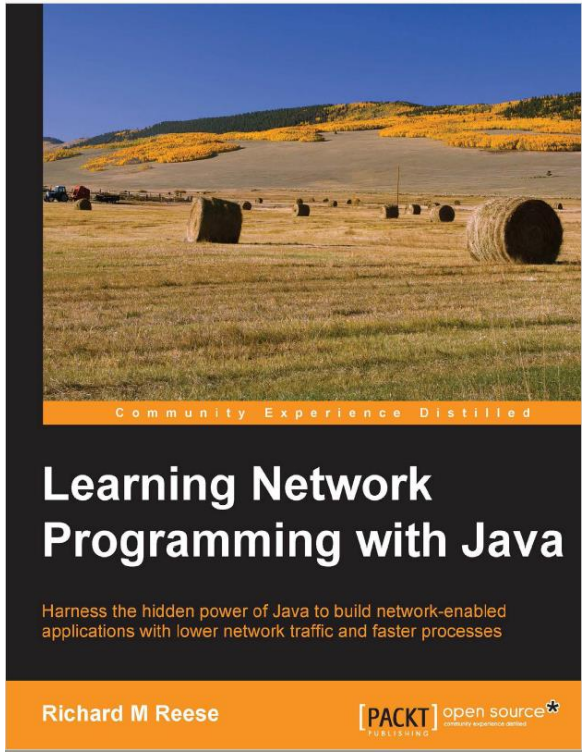
REDES DE COMPUTADORES Y LABORATORIO

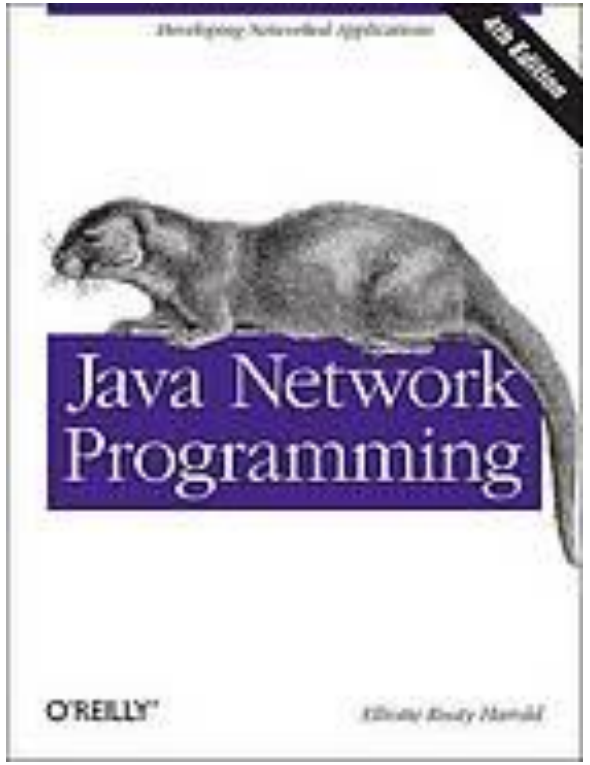
Christian Camilo Urcuqui López, MSc

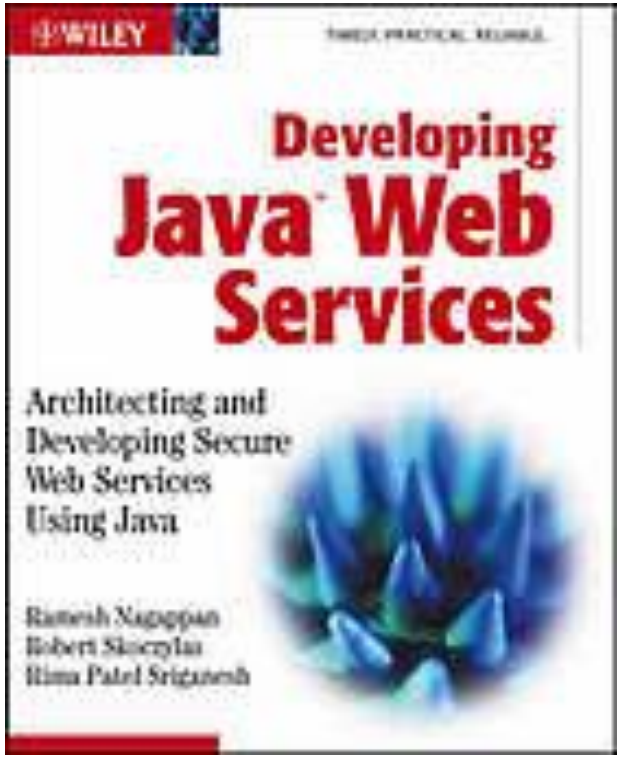
UNIVERSIDAD ICESI | A OTRO NIVEL

BIBLIOGRAFÍA









COMPETENCIAS

- Aplicar la API de Java para el desarrollo de servicios no orientados a la conexión.
- Describir la aplicación de datagramas.

DATAGRAMAS

- Como hemos visto, los datagramas hacen parte de un servicio no orientado a la conexión.
- Un datagrama es un mensaje independiente y auto-contenido, que se envía a través de la red, y cuyo arribo, tiempo de arribo y contenido no están garantizados.
- Las clases **DatagramPacket y DatagramSocket** del paquete java.net implementan la comunicación mediante datagramas independientes del sistema, utilizando UDP.
- **Los números de puerto de TCP y UDP son totalmente independientes**, es decir, ser puede tener un servidor UDP y un servidor TCP escuchando por el mismo número de puerto.

JAVA.NET

- ¡Las cosas cambian acá!
  - Ya no se deben utilizar las clases de flujos ya que no tenemos una conexión confiable entre los dos host. Para ello se debe "empaquetar" la información que se va a enviar a través de la clase **DatagramPacket.**
  - El concepto de Socket se conserva, la diferencia radica en que se utiliza la clase **DatagramSocket** para especificar el tipo de protocolo.
  - DatagramSocket y DatagramPacket son utilizados tanto en el cliente y en el servidor, pero, sus parámetros cambian dependiendo de si la acción es enviar o recibir información.
- La aplicación puede enviar DatagramPacket a través de un DatagramSocket.

DATAGRAMPACKET

- Permite "empaquetar" los mensajes, para poder hablar de unidades de información auto-contenidas, es decir, unidades de información con todos los datos necesarios para que el mensaje pueda llegar a su destino final y una vez allí, el receptor pueda extraer la información necesaria para procesarlo y enviar la respuesta al cliente que lo generó.

1. **DatagramPacket (byte buf[], int lontigud)**. Se especifica el buffer de bytes y una longitud.
2. **DatagramPacket (byte buf[], int longitud, InetAddress dir, int puerto)**. Se agrega información sobre la dirección y el puerto destino.

DATAGRAMPACKET

Métodos de instancia

- **InetAddress getAddress().** Obtiene la dirección IP del destino. Se utiliza habitualmente para envíos.
- **int getPort**. Devuelve el número de puerto remoto hacia el cual va dirigido el DatagramPacket.
- **byte[] getData()**. Devuelve un arreglo de bytes con los datos contenidos en el datagrama. Se utiliza para recuperar los datos del datagrama después de haberlo recibido.
- **int getLength ()**. Devuelve la longitud de los datos válidos contenidos en arreglo de bytes, que se obtendrían con el método *getData()*. Habitualmente no coincide con la longitud del arreglo de bytes.

DATAGRAMSOCKET

- Esta clase se utiliza tanto para enviar y recibir DatagramPackets.
- Se utiliza un único DatagramSocket para enviar paquetes a diferentes destinos y recibir paquetes de diferentes fuentes. **No hay forma de controlar los equipos de los cuales se van a recibir los paquetes**; simplemente se reciben todos los enviados hacia al puerto y la dirección.
- Al crear un DatagramSocket se puede asignar un puerto o dejar que el sistema operativo asigne uno por defecto.
- Generalmente, en el servidor se tomará un puerto particular para operar, mientras que los clientes permitirán la asignación aleatoria del puerto.

DATAGRAMSOCKET

Constructores

- **DatagramSocket() throws SocketExeption**
  - Se elige un número de puerto aleatorio. Utilizado en el **lado del cliente.**
- **DatagramSocket(int puerto) throws SocketException**
  - Se especifica el puerto y es **utilizado en el servidor.**
- **DatagramSocket(int puerto, InetAddress local) throws SocketExeption**
  - Se especifica el puerto y la dirección. Es útil cuando el servidor tiene más de una interfaz y se desea especificar a cual de ellas la aplicación va a establecer la comunicación UDP.

DATAGRAMSOCKET

Métodos se instancia

Esta clase proporciona los métodos para enviar y recibir DatagramPackets, y métodos para cerrar el Socket, obtener información acerca de la dirección local y fijar un tiempo limite de recepción.

- **void send(DatagramPacket paquete) throws IOException**
  - Envía el "paquete" hacia la red.
- **void receive(DatagramPacket paquete) throws IOException**
  - Recibe un único paquete UDP, dentro del DatagramPacket especificado en el parámetro "paquete". Se espera que el paquete determine la dirección IP y el puerto de su origen, su contenido y la longitud de los datos. Este método espera hasta recibir correctamente un paquete o hasta que se alance el timeout.
- **InetAddress getLocalAddress ()**

DATAGRAMSOCKET

Métodos se instancia

Esta clase proporciona los métodos para enviar y recibir DatagramPackets, y métodos para cerrar el Socket, obtener información acerca de la dirección local y fijar un tiempo limite de recepción.

- **void close()**
- **InetAddress getInetAddress()**
  - Este método devuelve la InetAddress a la cual está conectado el DatagramSocket.
- **int getPort()**

DATAGRAMSOCKET

- Conocer o cambiar el tiempo de espera para el bloque de alguna operación (*timeout*) y conocer o cmabiar el tamaño de los buffers de recepción y el envío del DatagramSocket.
  - **void setSoTimeout(int timeout) throws SocketException**
  - **int getSoTimeout() throws SocketException**
  - **void setSendBufferSize(int size) throws SocketException**
  - **int getSendBufferSize() throws SocketException**
  - **void setReceiveBufferSize(int size) throws SocketException**
  - **int getReceiveBufferSize()**

- Se le ha suministrado un ejemplo en el GitHub para una prueba de cliente / servidor con comunicación no orientada a la conexión.
- Revise el material faltante y las páginas (142-167) del libro Reese, R. M. (2015). Learning Network Programming with Java. Packt Publishing Ltd.
- El objetivo es implementar el ejemplo de UDP Streaming para el procesamiento de audio.

MINDEFENSA

UNIVERSIDAD ICESI | A OTRO NIVEL

PRÓXIMA CLASES - COMPETENCIAS

- Describir la aplicación de subredes (**Jueves**)
- Describir NAT (**Jueves**)
- Describir los protocolos ICMP, ARP y DHCP (**Jueves**)
- Describir TCP (**Jueves**)
- Aplicar la clase Multicastsocket en Java. (**Viernes – No hay clase**)
- Aplicar la API de Java la resolución de problemas con UDP Streaming. (**Viernes – No hay clase**)

LECTURAS

Material utilizado	1. Arboleda, L. (2012). Programación en Red con Java. 2. Harold, E. (2004). Java network programming. " O'Reilly Media, Inc.". 3. Tanenbaum, A. S. (2003). Redes de computadoras. Pearson educación. 4. Reese, R. M. (2015). Learning Network Programming with Java. Packt Publishing Ltd.
Actividades DESPUÉS clase – jueves (11 y 18)	A1. Leer del libro 3, la sección 5.6.1 hasta la sección 5.6.3, y 5.6.4 (**Ya estaba asignado**) A1. A1. Leer del libro 3 el contenido desde la sección 6.5.1 hasta la 6.5.7
Actividades DESPUÉS clase – viernes (19 septiembre)	A3. Leer la sección 9 del libro 1 A2. Leer la sección 8 del libro 1

REFERENCIAS

1. https://www.google.com.co/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwju5o62x-vdAhXjtlkKHaouDcAQjRx6BAgBEAU&url=http%3A%2F%2Feltallerdelbit.com%2Fdireccionamiento-ip%2F&psig=AOvVaw3E_T5IpV-ANtL0eEQbHtkg&ust=1538700259199893