UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN
INFRAESTRUCTURA DE COMUNICACIONES
Profesores: Yezid Donoso / Carlos Lozano
[ydonoso/calozanog] @uniandes.edu.co
Asist. De laboratorio: Arnold Andres Lara
a.larav@uniandes.edu.co

LABORATORIO #3: ACTIVIDAD 2 – IMPLEMENTACIÓN DE SERVIDOR Y CLIENTE UDP

1. OBJETIVO (S)

Este laboratorio tiene por objetivo la implementación de una arquitectura cliente/servidor para hacer transferencia de información usando el protocolo UDP en un lenguaje de programación seleccionado por el grupo de trabajo. Adicionalmente, se busca que el estudiante pueda implementar mecanismos para evaluar el desempeño de los servidores y analizarlos mediante diferentes pruebas.

Al finalizar la práctica, el estudiante estará en capacidad de:

- Implementar aplicaciones servidor y clientes, para enviar/recibir archivos soportada en sockets TCP.
- Implementar aplicaciones servidor y clientes, para enviar/recibir archivos soportada en sockets UDP.
- Evaluar pruebas de carga y desempeño para la comunicación. Diseñar, implementar y evaluar diferentes tipos de pruebas de desempeño a cada uno de los servidores implementados.
- Completar el desarrollo y el análisis pertinente para uno de los requerimientos del proyecto.

2. LECTURAS PREVIAS

Los temas a tratar en esta práctica son los siguientes:

- Generalidades de los protocolos TCP y UDP. [1]
- Implementación de servidores TCP y UDP en Java. [1, 2, 3]
- Programación con sockets en lenguaje Python y PHP [5, 6, 7]
- Conversión y manipulación de archivos multimedia en Python y Java. [4, 8]
- Ejecución de pruebas de carga y desempeño con Apache JMeter [9]

3. INFORMACIÓN BÁSICA

En unas secciones de esta práctica se desarrollarán servidores que implementen servicios UDP, para comunicarse respectivamente con clientes. De tal forma, se supone el diseño e implementación de una arquitectura cliente-servidor, donde la comunicación se establezca a través de sockets.

Para motivar el uso de UDP en la práctica de laboratorio, suponga que está interesado en diseñar un protocolo de transporte sin overhead. ¿Cómo podría desarrollar este protocolo? Primero, considere usar un protocolo de transporte vacío. En el emisor, puede tomar los mensajes de la aplicación y pasarlos directamente a la capa de internet; en el receptor, puede tomar los mensajes que llegan de la capa de red y pasarlos directamente a la aplicación. Sin embargo, a nivel de transporte se requiere proporcionar como mínimo un servicio de multiplexación / demultiplexación para pasar datos entre la capa de red y la aplicación correcta.

UDP hace tan poco como un protocolo de transporte puede. Aparte de la función de multiplexación / demultiplexación, realiza comprobación de errores. De hecho, si un desarrollador selecciona UDP en lugar de TCP, entonces la aplicación está hablando casi directamente con la capa de internet.

UDP toma los datos de la capa de aplicación, agrega los campos de número de puerto de origen y destino para el servicio de multiplexación / demultiplexación, agrega otros dos campos de menor importancia y pasa el "segmento o datragrama" resultante a la capa de red. La capa de red encapsula el datagrama en un paquete IP y luego realiza un intento de mejor esfuerzo para entregar el datagrama al receptor.

Figura 1. Servicios de la capa de transporte

UDP se utiliza comúnmente en aplicaciones multimedia, como la telefonía IP, streaming de audio y video, dado que todas estas aplicaciones pueden tolerar pequeñas pérdidas de información, por lo que la transferencia de datos confiable no es absolutamente crítica para el éxito de la aplicación. Finalmente, debido a que TCP no se puede emplear con multidifusión, las aplicaciones de multidifusión se ejecutan a través de UDP.

En esta práctica se desarrollarán servidores que implementen protocolos a nivel de capa de transporte (TCP y UDP), así como los clientes que consumirán dichos servicios.

La práctica plantea el desarrollo de una aplicación en arquitectura cliente-servidor que debe ser implementada utilizando sockets. Deben definirse protocolos a nivel de aplicación para la funcionalidad requerida, y adicionalmente, se debe implementar algún mecanismo para poder analizar métricas de escalabilidad y desempeño, ya que sobre el servidor se realizarán pruebas para determinar sus parámetros de desempeño ante un número creciente de clientes.

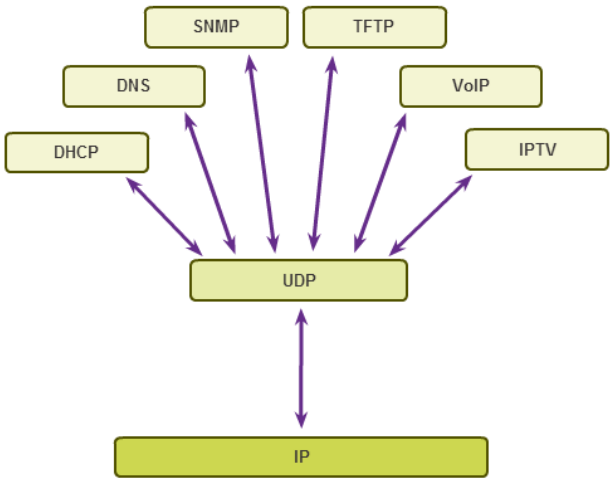

Figura 2. Aplicaciones que utilizan UDP

Las primeras 2 partes de la práctica se enfocarán en la implementación de servidor y de cliente para transferencia de archivos, seguidamente las dos partes siguientes se enfocarán en la implementación de servidor y de cliente para streaming.

4. PROCEDIMIENTO

Para el presente laboratorio debe desarrollar los siguientes requerimientos:

4.1. Requerimientos cliente UDP (transferencia de archivos)

El grupo deberá desarrollar y desplegar un cliente en el lenguaje de programación de su preferencia (Java o Python). Este deberá cumplir con los siguientes requerimientos:

1. Correr en el equipo local del estudiante.

2. Conectarse al servidor UDP y mostrar que se ha realizado dicha conexión. Mostrar el estado de la conexión.

3. Enviar notificación de preparado para recibir datos de parte del servidor.

4. Recibir un archivo del servidor por medio de una comunicación a través de sockets UDP.

5. Verificar la integridad del archivo con respeto a la información entregada por el servidor.

   Nota: La aplicación cliente debe calcular el hash del archivo (utilice el mismo algoritmo utilizado por el servidor) y compararlo con el indicado por el servidor.

6. Enviar notificación de recepción del archivo al servidor, si este está correcto o no (resultado de la comparación del hash).

7. La aplicación debe permitir medir el tiempo de transferencia de un archivo en segundos. Este tiempo debe calcularse desde el momento en que se envía el primer paquete con datos del archivo en el servidor hasta el momento en el que se recibe el último paquete del archivo en el cliente. Al final de cada transferencia la aplicación debe reportar si el archivo está completo y

correcto y el tiempo total de transferencia. Para esto, genere un log para cada intercambio de datos entre cliente y servidor.

8. Disponer un repositorio de los archivos recibidos y logs.

4.2. Requerimientos servidor UDP (transferencia de archivos)

El equipo deberá implementar un servidor en el lenguaje de programación de su preferencia (Java o Python). Este deberá cumplir con los siguientes requerimientos:

1. El servidor debe correr en una máquina virtual UbuntuServer18.04 configurada con su tarjeta de red en modo Bridge, proporcionada en esta actividad de laboratorio. Esta máquina puede ser ejecutada en el mismo equipo en el que funcione el cliente o en otro equipo que se encuentre conectado a la misma red.

2. Recibir conexiones UDP. La aplicación debe soportar 25 conexiones en simultáneo.

3. Definir el tamaño de los mensajes en que se van a fragmentar los archivos. Nota: Recuerde que los fragmentos UDP pueden ser hasta de 64 KB.

4. Tener dos archivos disponibles para su envío a los clientes: un archivo de tamaño 100 MB, y otro de 250 MB. Se sugiera que uno de estos archivos sea multimedia.

5. La aplicación debe permitir seleccionar qué archivo desea enviarse a los clientes conectados y a cuántos clientes en simultáneo. A todos se les envía el mismo archivo durante una transmisión.

6. Se debe entregar el valor hash calculado para cada archivo. Este dato se usará para que los clientes validen la integridad del mismo.

7. Realizar la transferencia de archivos a los clientes definidos en la prueba. El envío debe realizarse solo cuando el número de clientes indicados estén conectados y su estado sea listo para recibir.

8. La aplicación debe permitir medir el tiempo de transferencia de un archivo en segundos. Este tiempo debe calcularse desde el momento en que se envía el primer paquete con datos del archivo en el servidor hasta el momento en el que se recibe el último paquete del archivo en el cliente. Al final de cada transferencia la aplicación debe reportar si el archivo está completo y correcto y el tiempo total de transferencia. Para esto, genere un log para cada intercambio de datos entre cliente y servidor.

9. Disponer un repositorio de los archivos recibidos y logs para cada una de las pruebas. (Revisar sección de recomendaciones de la guía anterior).

4.3. Recomendaciones para la construcción de los logs

1. Construya un archivo por prueba.
2. Nombre el archivo con la fecha y hora de la prueba.
3. Incluya el fecha y hora de la prueba en el archivo.
4. Incluya el nombre del archivo enviado y su tamaño.

5. Identifique cada cliente al que se realiza la transferencia de archivos.
6. Registrar el número de fragmentos a Enviado o Recibidos.
7. Identifique si la entrega del archivo fue exitosa o no.
8. Tome los tiempos de transferencia a cada uno de los clientes, calcule este tiempo desde el momento que se envía el primer paquete archivo y hasta que se recibe la confirmación de recepción del último paquete del archivo.

4.4. Implementación de un servidor UDP para emisión de archivos de video

Implementar la funcionalidad de emisión (streaming) de archivos de video en formato .mp4, de forma similar a un canal de televisión IP.

El servidor debe ser extendido con las siguientes características:

1. El servidor debe correr en una máquina virtual UbuntuServer18.04 configurada con su tarjeta de red en modo Bridge, proporcionada en esta actividad de laboratorio. Esta máquina puede ser ejecutada en el mismo equipo en el que funcione el cliente o en otro equipo que se encuentre conectado a la misma red.

2. Tener 3 archivos de video que serán emitidos a clientes por medio de canales.

3. El servidor debe realizar el streaming utilizando el protocolo UDP. Cada video debe ser emitido a una dirección multicast diferente, e incluso es posible contar con puertos diferentes (de elección libre), de forma que cada par dirección-puerto represente un "canal" con un video diferente.

4.5. Implementación de un cliente UDP visualización de videos

Diseñar e implementar un cliente que utilice los servicios de streaming implementados en el numeral 4.3. La elección del tipo de interfaz de usuario es libre, teniendo en cuenta la necesidad de cumplir con las siguientes características:

1. Correr en el equipo local del estudiante.

2. Visualizar los canales disponibles (cada uno representado por un par dirección-puerto)

3. Permitir conexión con algún canal disponible.

4. Visualizar el video emitido en el canal escogido, permitiendo la acción de "Stop" que detendrá la visualización de video.

5. Volver a permitir al usuario la elección de algún canal.

4.6. Análisis de Resultados

1. Realice pruebas de desempeño para la transferencia de archivos con UDP. Establezca comparativos y análisis de parámetros en las pruebas de las aplicaciones de TCP y UDP.
2. ¿Son congruentes los resultados obtenidos en las dos actividades de este laboratorio con la teoría presentada en la clase teórica?
3. ¿Qué características tiene un servicio de streaming de video que usa el protocolo TCP?

4. ¿Es posible desarrollar aplicaciones UDP que garanticen la entrega confiable de archivos? Que consideraciones deben tenerse en cuenta para garantizar un servicio de entrega confiable utilizando dicho protocolo. Justifique su respuesta.

5. ENTREGABLES

- Informe digital en formato **.pdf** que contenga el proceso de solución de los requerimientos efectuados en cada una de las secciones de la práctica. Si estos no se evidencian en el mismo, se asumirá que no fueron desarrollados. Tablas, gráficas y logs. Analizando las mismas. Adicional incluir en el informe:
  - Enlace de repositorio remoto de las aplicaciones de Servidor y cliente de transferencia de archivos UDP efectuadas: Github o Gitlab.
  - Enlace de repositorio remoto de las aplicaciones de Servidor y cliente de streaming de video efectuadas: Github o Gitlab.
  - Enlace a un video corto, en donde expliquen sus aplicaciones y muestren el funcionamiento.
- Sustentación de la funcionalidad por algún integrante del grupo elegido de forma aleatoria.

La calificación de este laboratorio será distribuida de la siguiente manera:

Entregable	Valor
Actividad 4.1	15%
Actividad 4.2	15%
Actividad 4.3	5%
Actividad 4.4	15%
Actividad 4.5	15%
Actividad 4.6	20%
Sustentación	10%
Calidad informe	5%
Total	100%

6. REFERENCIAS

[1] Kurose, James. Ross, Keith. Computer Networking: A Top-Down Approach. 6th edition. Addison-Wesley. Capítulos 2 y 3.

[2] The Java Tutorials. Trail: Custom Networking.
http://docs.oracle.com/javase/tutorial/networking/index.html

[3] Java Secure Sockets Extension (JSSE).
http://docs.oracle.com/javase/7/docs/technotes/guides/security/jsse/JSSERefGuide.html

[4] Manipulación de audio y sonido en lenguaje Java. Tutorial.
http://docs.oracle.com/javase/tutorial/sound/TOC.html

[5] Documentación oficial Python: Socket – Low-level Networking Interface.
https://docs.python.org/2/library/socket.html

[6] Extensión Socket. http://php.net/manual/es/book.sockets.php

[7] Rhodes, Brandon. Goerzen, John. Foundations of Python Network Programming. 2nd edition. 2010.

[8] Librería PyMedia para grabación, reproducción y streaming multimedia en Python. http://pymedia.org/tut/

[9] Página oficial de Apache JMeter. http://jmeter.apache.org/

HISTORIAL DE REVISIONES

FECHA	AUTOR	OBSERVACIONES
24/09/2020	**Arnold Andres Lara** a.larav@uniandes.edu.co	Ajustes de redacción y actualización del laboratorio
10/03/2020	**Arnold Andres Lara** a.larav@uniandes.edu.co	Ajustes de redacción y actualización del laboratorio
25/07/2019	**Jonatan Legro Pastrana** j.legro@uniandes.edu.co	Actualización de documento.