UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN
INFRAESTRUCTURA DE COMUNICACIONES
Profesores: Yezid Donoso / Jesse Padilla
ydonoso@uniandes.edu.co / pa.jesse10@uniandes.edu.co
Asist. De laboratorio: Jonatan Legro Pastrana
j.legro@uniandes.edu.co

LABORATORIO 4 – IMPLEMENTACIÓN DE SERVIDORES TCP Y UDP

1. OBJETIVO (S)

Este laboratorio tiene por objetivo la implementación de servidores TCP y UDP con sus respectivos clientes 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 servidores y clientes TCP para diferentes servicios.
- Implementar servidores y clientes UDP para diferentes servicios.
- Implementar y evaluar pruebas de carga y desempeño para la comunicación. Diseñar, implementar y evaluar diferentes tipos de pruebas de desempeño y seguridad a cada uno de los servidores implementados.

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 TCP, 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. En la Figura 1 se presenta la arquitectura esperada.

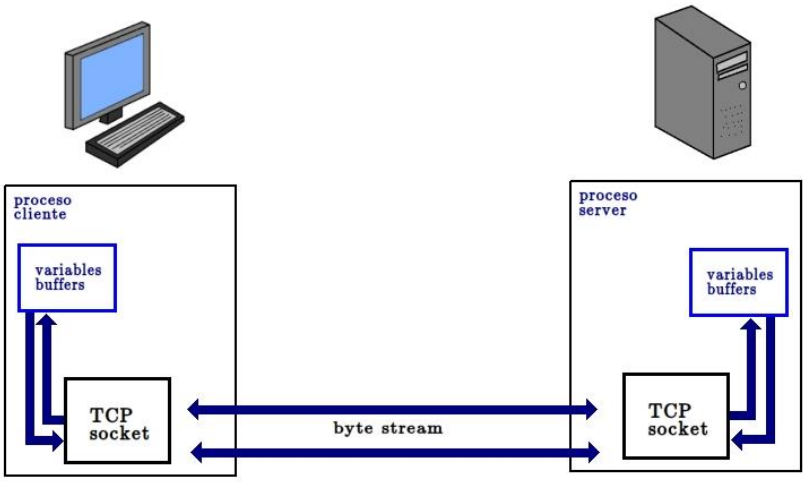

Figura 1. Diagrama de comunicación por sockets TCP (http://wiki.inf.utfsm.cl/index.php?title=Socket_programming_with_TCP)

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.

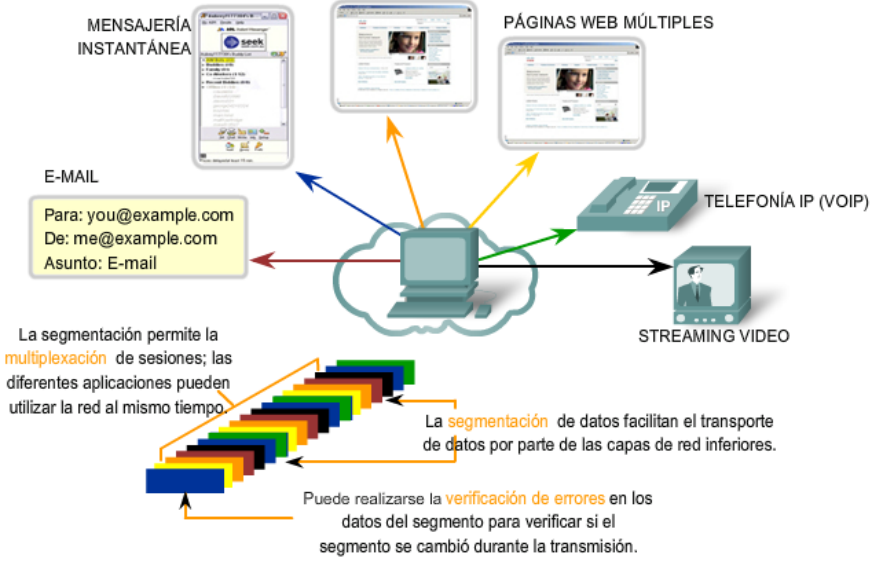

Figura 2. Servicios de la capa de transporte

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.

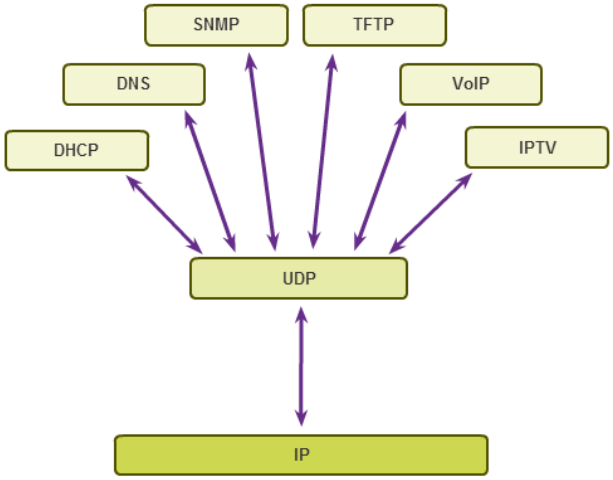

Figura 3. Aplicaciones que utilizan UDP

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.

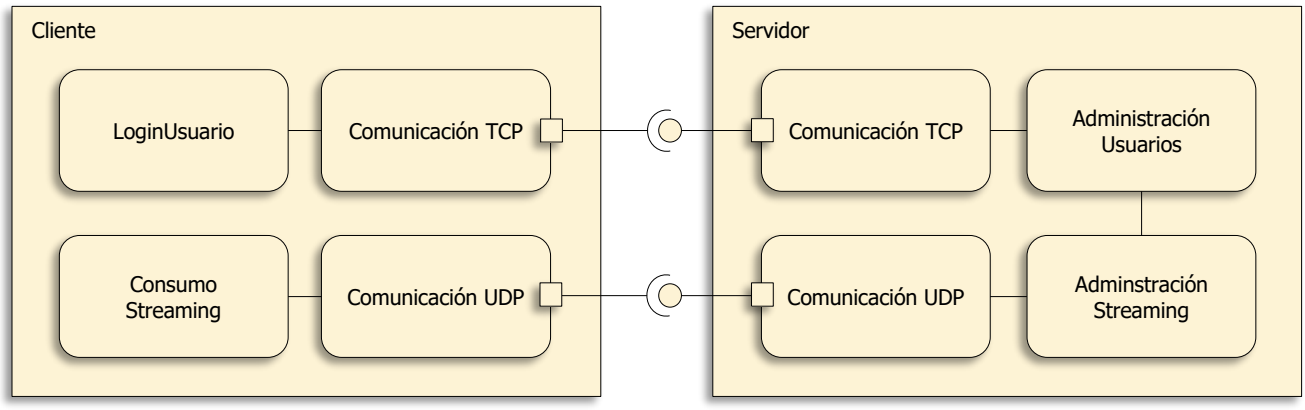

Figura 4. Diagrama de componentes

Las primeras 4 partes de la práctica se enfocarán en la implementación de los servidores y de los clientes, seguidamente está una parte que se enfocará en mediciones de desempeño sobre los servidores implementados con el fin de evaluar las características ofrecidas por el protocolo de comunicación. Las 2 partes siguientes de la práctica tienen un nuevo requerimiento de servidor en la nube y su respectivo cliente. Finalmente, se realizará un análisis de las aplicaciones del mercado.

4. PROCEDIMIENTO

Para el presente laboratorio debe desarrollar los siguientes requerimientos:

4.1. Implementación de un servidor TCP para autenticación de usuarios

Implementar un servidor utilizando sockets que realice la autenticación de usuarios desde la capa de aplicación. El servidor recibe desde un cliente un flujo de datos con la información del usuario (nombre de usuario y contraseña), realiza el proceso de autenticación, y, después informa al cliente sobre el desenlace de la solicitud.

El servidor debe tener las siguientes características:

- El servidor debe ejecutarse sobre una máquina con sistema operativo Linux, de preferencia Ubuntu 16 o Ubuntu 18.
- El servidor debe garantizar la persistencia de información.
- El servidor debe atender múltiples clientes en paralelo.
- El servidor debe estar en capacidad de soportar un mínimo de 150 clientes concurrentes.
- El servidor debe implementar mecanismos de medición que permitan determinar el tiempo promedio de atención de los clientes, número de clientes en cola, número de clientes que están siendo atendidos y tiempo promedio en cola.
- El servidor debe estar implementado de tal forma que no sea posible realizar un ataque de denegación de servicio.
- La elección del puerto por el cual escucha el servidor es libre.
- El servidor puede ser implementado utilizando Python o Java. Para el primer lenguaje se recomienda el uso del módulo *socket*, para Java se recomienda el uso de las clases *ServerSocket* del paquete *java.net*.

4.2. Implementación de un cliente TCP para autenticación de usuarios

Diseñar e implementar un cliente que utilice los servicios implementados en la numeral 4.1. Aunque la elección del tipo de interfaz de usuario es libre, se recomienda implementarla haciendo uso de un framework web que maneje tecnologías HTML y JavaScript. Debe validarse la correcta autenticación del usuario en pantalla y en los logs del servidor.

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

Añadir al servidor implementado en el numeral 4.1 una 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:

- Una vez finaliza la autenticación del usuario, se debe permitir el almacenamiento de videos por parte del usuario, en el formato de preferencia del equipo. Adicionalmente, los videos deben ser anexados a una lista de reproducción/emisión (además de ser enviado al cliente).
- El servidor debe realizar el streaming utilizando el protocolo UDP. Cada video recibido 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.

- Para un servidor implementado en Python se recomienda el uso del módulo *socket*, para Java se sugiere el uso de la clase *DatagramSocket* del paquete *java.net*.

4.4. 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. Aunque la elección del tipo de interfaz de usuario es libre, se recomienda extender la interfaz web del numeral 4.2, teniendo en cuenta la necesidad de ejecutar acciones básicas de un reproductor como "Conectar Streaming a Canal" (representado a través de un par dirección-puerto), "Play" y "Stop".

Las primeras cuatro secciones del laboratorio se deben evidenciar durante la sustentación de la práctica, además de las secciones 4.6 y 4.7. Se tendrá en cuenta buenas prácticas de programación al momento de efectuar la revisión.

4.5. Pruebas de carga y desempeño

- Implementar pruebas de carga y desempeño sobre el servidor TCP, con el fin de determinar el tiempo promedio de atención de los clientes, porcentaje de error frente a peticiones de usuario y throughput.
- Hacer la misma implementación, pero esta vez sobre el servidor UDP. Tener en cuenta los tres elementos de medición determinados.

Para los dos contextos de prueba donde solo varía el servidor objetivo, se recomienda hacer uso de la herramienta de pruebas de carga Apache JMeter, particularmente de los resultados manejados en el Aggregate Report. A continuación, se definen los requerimientos para realizar el análisis de resultados:

- Efectuar pruebas sobre escenarios de concurrencia. Por ejemplo, 25, 60, 120 y 150 usuarios. La selección de estos escenarios dependerá directamente del comportamiento de su desarrollo, por lo que será necesario hacer pruebas exploratorias que le permitan determinar el número de solicitudes a evaluar[1]. Manejar un mismo período de ramp-up[2] para los 3 escenarios seleccionados, y efectuar 2 iteraciones para cada uno de estos escenarios.
- Sustentar la selección de escenarios de concurrencia y de ramp-up.
- Generar los elementos gráficos necesarios para poder evaluar los resultados obtenidos.
- Concluir sobre el desempeño y concurrencia manejados por cada tipo de servidor, y establecer un comparativo entre los mismos. Así mismo, analizar una posible degradación de servicio con la variación en el número de usuarios concurrentes.

[1] No exceder los 300 usuarios concurrentes, ya que para este escenario sería necesario crear un clúster de JMeter.
[2] Para definir adecuadamente el ramp-up, consultar la siguiente fuente: http://jmeter.apache.org/usermanual/test_plan.html.

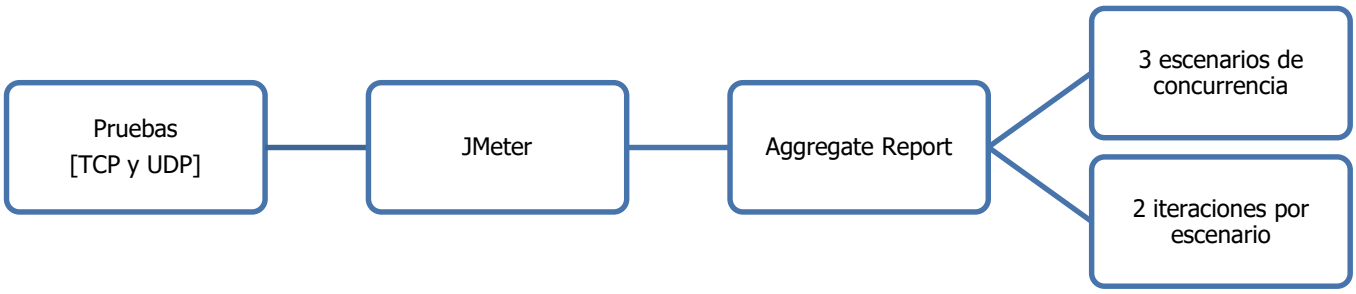

Figura 5. Vista general plan de pruebas

1. **Para esta parte de la práctica, se deberá hacer una descripción del diseño e implementación de cada conjunto de pruebas. Así mismo, se solicita la muestra de resultados, análisis y conclusiones para el grupo de pruebas. Apoyarse en las gráficas necesarias para sustentar lo anterior.**
2. **Consulte 3 herramientas adicionales para realizar pruebas de carga y explique la operación de cada una de ellas. (Sólo explicar cómo funciona, no deben realizar pruebas adicionales de carga).**

4.6. Implementación de un servidor TCP para emisión de archivos de video

Generar un servidor con funcionalidad de emisión (streaming) utilizando el protocolo TCP.

- Este servidor debe ser implementado con Java o Python; deben garantizar que se cumplan con las características del protocolo.
- Establecer las preferencias de rendimiento en términos de tiempo de conexión, baja latencia y alto ancho de banda.
- El servidor debe estar diseñado para no ser afectado contra ataques de denegación de servicios.
- Aclarar los detalles que tuvieron en cuenta a la hora de diseñar el servidor para cumplir con las características del protocolo.
- El servidor debe ser ejecutado en una máquina virtual Linux, que se encuentre ubicada en la nube (AWS o Azure o Google).

4.7. Implementación de un cliente TCP para visualización de videos

Diseñar e implementar un cliente que utilice los servicios de streaming implementados en el numeral 4.6. La elección del tipo de interfaz de usuario es libre, pero debe tener en cuenta la necesidad de ejecutar acciones básicas de un reproductor como "Conectar Streaming a Canal" (representado a través de un par dirección-puerto), "Play" y "Stop".

4.8. Evaluación y análisis de aplicaciones del mercado

Para la última parte del laboratorio deberá dar respuesta a los siguientes puntos:

3. **Analizar las aplicaciones Waze, Netflix, y Skype. Guarde una captura de wireshark mientras usa Netflix y otra captura de wireshark mientras realiza una videollamada en Skype, analice y explique las diferencias entre ambas aplicaciones con base en los protocolos que pueda visualizar en los paquetes capturados.**
4. **Realizar un análisis de las medidas de desempeño realizadas al servidor TCP (Numeral 4.1). ¿Es el comportamiento de estas variables acorde con los modelos teóricos encontrados la literatura de la materia?**
5. **Consulte y explique detalladamente qué funciones tiene el comando "netstat", el cual puede usarse sobre Windows y Linux.**

6. ENTREGABLES

- Informe digital 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.
- Datos generados en la sección 4.5.
- Carpeta con el desarrollo efectuado para los clientes y servidores (el código debe comentarse).
- Enlace a un video, en donde expliquen sus aplicaciones y muestren el funcionamiento.

Nota: Deben hacer una sola entrega en una carpeta comprimida con todos archivos. Deben realizar la entrega por Sicua Plus.

7. 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
22/02/2019	**Jonatan Legro Pastrana** j.legro@uniandes.edu.co	Combinación de laboratorios TCP y UDP. Tomando de base guías de R. Cáliz, L. Ochoa, J. Padilla.