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 1 – IMPLEMENTACIÓN DE SERVIDOR Y CLIENTE TCP

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 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. [1, 2, 3]
- Programación con sockets en lenguaje Python y PHP [5, 6, 7]
- Ejecución de pruebas de carga y desempeño con Apache JMeter [9]

3. INFORMACIÓN BÁSICA

En esta práctica se desarrollará un servidor que implemente servicios TCP, para comunicarse respectivamente con un cliente. 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.

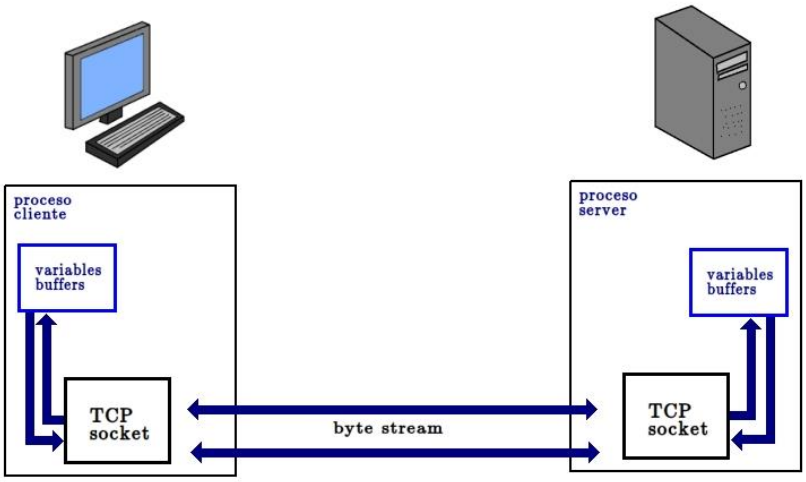

Figura 1. Diagrama de comunicación por sockets TCP (http://wiki.inf.utfsm.cl/index.php?title=Socket_programming_with_TCP)

Adicionalmente, se espera que se implementen pruebas de carga y desempeño sobre la arquitectura desarrollada, con el fin de evaluar las características ofrecidas por el protocolo de comunicación. Esto permitirá evaluar otras aplicaciones del mercado, analizar su desempeño y sugerir los protocolos de comunicación empleados en las mismas.

Las primeras dos partes del laboratorio se centran en la implementación del cliente y del servidor, junto con los mecanismos necesarios para garantizar una comunicación sobre TCP. La tercera parte se enfoca en el diseño e implementación de pruebas de carga y desempeño sobre el servidor. La práctica finaliza con una evaluación del protocolo de comunicación implementado, y análisis de aplicaciones del mercado.

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.

La implementación del servidor puede ser llevada a cabo en una máquina virtual, proporcionada en el laboratorio, o puede hacer su implementación en una máquina virtual de un proveedor en la nube. Por otra parte, el cliente debe ser implementado en el equipo de algún integrante del equipo.

4. CONFIGURACIÓN

Deberá tener en cuenta esta información si usted desea implementar su servidor TCP en una máquina virtual Ubuntu Server 18.04 proporcionada en este laboratorio.
Primero deberá tener instalado en su equipo el software VMWare Workstation Player 15 para windows o VMware Fusion para MAC.
Ahora descargue la máquina virtual desde el enlace que le fue proporcionado en la actividad de SicuaPlus. El archivo descargado se encuentra comprimido en **.rar**. Descomprima el archivo y entre en la carpeta nombrada UbuntuServer18.04 donde encontrara los archivos de la máquina virtual.

Intente abrir el archivo con extensión **.vmx**, esto abrirá la maquina virtual en el software Vmware Workstation Player.

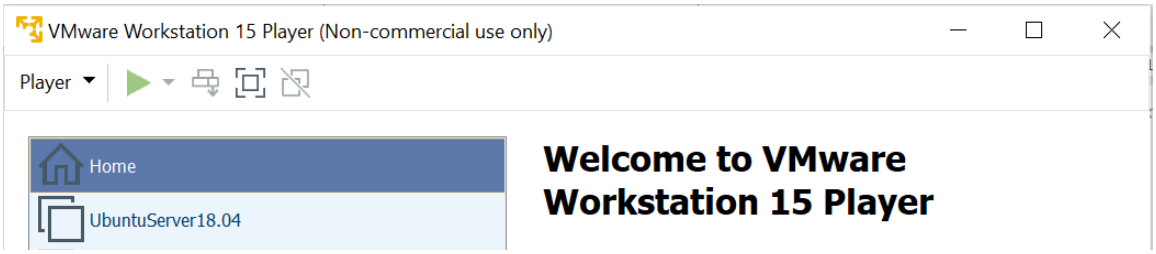

Figura 2. Ventana VMWare Workstation Player

La máquina virtual se encuentra configurada en modo Bridge, lo que significa que se conectara a su red como si fuera un equipo más. Al iniciar la maquina use las credenciales de acceso **usuario: infracom y contraseña: infracom.** Al iniciar podrá observar la dirección IP que obtiene su maquina virtual, para el presente caso se observa que la maquina ha recibido la dirección 192.168.0.15.

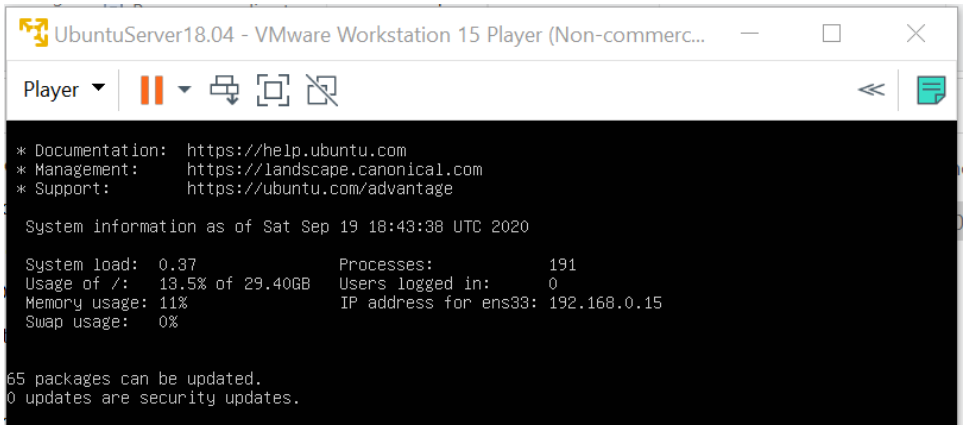

Figura 3. Funcionamiento Máquina virtual Ubuntu18.04 Server.

Ahora puede usar la ventana de Vmware Workstation Player para trabajar en la maquina o usar una conexión por SSH al puerto 22 usando un software de terminal como Putty o la consola de su sistema operativo.

4. PROCEDIMIENTO

Para el presente laboratorio deberán desarrollar los siguientes requerimientos para un cliente y un servidor que se encuentran en dos máquinas diferentes en la misma red. Los requerimientos son los siguientes:

4.1. Requerimientos de cliente TCP

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

1. Correr en el equipo local del estudiante.

2. Conectarse al servidor TCP 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 TCP.

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 valor suministrado por el servidor.

6. Enviar notificación de recepción del archivo al servidor.

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. (Revisar sección de recomendaciones).

4.2. Requerimientos de servidor TCP

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 puede correr de las siguientes maneras:
   a. Máquina virtual UbuntuServer 18.04, configurada en modo Bridge. Esta maquina puede correr en el mismo equipo en el que funcione el cliente o en otro equipo que se encuentre conectado a la misma red.
   b. Servidor en la nube usando alguno de los siguientes proveedores: AWS, GoogleCloud o Azure.

1. Recibir conexiones TCP. La aplicación debe soportar 25 conexiones en simultáneo.

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

3. 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. El servidor deberá enviar también un valor hash calculado para ese archivo. Este valor será usado para validar la integridad del archivo enviado.

4. 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.

5. Definir el tamaño del buffer apropiado para su diseño. Realice diferentes pruebas para obtener el mejor desempeño en términos de tiempo de transmisión.

6. 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.

7. Disponer un repositorio de los archivos enviados y logs para cada una de las pruebas. (Revisar sección de recomendaciones).

4.3. Recomendaciones para la construcción de los logs

1. Construya un archivo por prueba.
2. Incluya el fecha y hora de la prueba en el archivo.
3. Incluya el nombre del archivo enviado y su tamaño.
4. Identifique cada cliente al que se realiza la transferencia de archivos.
5. Identifique si la entrega del archivo fue exitosa o no.
6. 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.
7. Incluya el número de paquetes enviados, numero de paquetes recibidos y el número de paquetes transmitidos, al igual que el valor total en bytes transmitidos y recibidos (estos valores están dados por el tamaño del archivo, los paquetes retransmitidos y los encabezados). Apóyese de comandos como iptraf, ngrep, ethstatus, entre otras para obtener estas estadísticas.

4.4. Pruebas de carga y desempeño sobre la arquitectura

Se espera que para este laboratorio el grupo efectúe pruebas de carga y desempeño sobre el servidor TCP implementado en la segunda parte de la práctica. Para ello, haga un grupo de pruebas haciendo uso de JMeter.

Grupo de pruebas en JMeter:

- Se recomienda hacer uso del Aggregate Report de Apache JMeter.
- Se deben efectuar pruebas de carga y desempeño para el servicio TCP de manera independiente. Para ambos casos se debe determinar el tiempo promedio de atención de los clientes, porcentaje de error frente a peticiones de usuario y throughput.
- Efectuar pruebas sobre escenarios de concurrencia. Por ejemplo, 10, 20, 50 y 100 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 3 escenarios seleccionados, y efectuar 2 iteraciones para cada uno de estos escenarios.

[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.

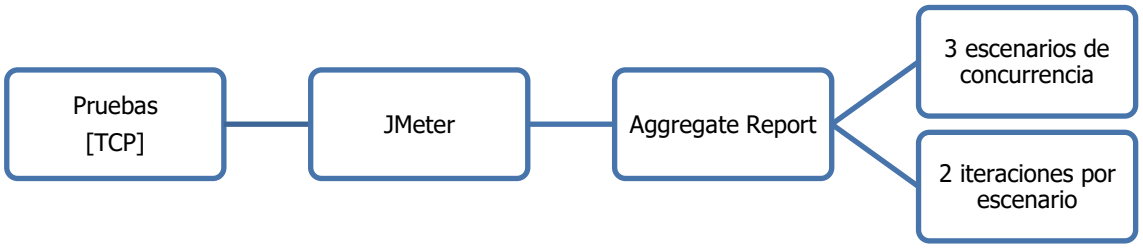

Figura 4. Vista general plan de pruebas

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 2 herramientas adicionales para realizar pruebas de carga y explique la operación de cada una de ellas.

6. 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.
- Incluir en el informe tablas, Gráficas y Análisis de Resultados.
- Incluir en el informe un enlace a un repositorio en la nube de las aplicaciones de Servidor y cliente efectuadas: Github o Gitlab.
- Incluir en el informe un Link a un video en donde se muestre el funcionamiento de transferencia de archivos de la aplicación.

Nota: Deben hacer una sola entrega en una carpeta comprimida con todos los 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

[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.

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

HISTORIAL DE REVISIONES

FECHA	AUTOR	OBSERVACIONES
11/09/2020	**Arnold Andres Lara** a.larav@uniandes.edu.co	Actualización del laboratorio.
28/02/2020	**Arnold Andres Lara** a.larav@uniandes.edu.co	Ajustes de redacción y actualización del laboratorio
24/07/2019	**Jonatan Legro Pastrana** j.legro@uniandes.edu.co	Actualización de documento.