



CENTRO DE CIENCIAS BÁSICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
METAHEURÍSTICAS I
7° "A"

**ACTIVIDAD 3.01: ARQUITECTURA CLIENTE – SERVIDOR CON PLANIFICADOR
DE CARGA**

Profesor: Francisco Javier Luna Rosas

Alumnos:

Almeida Ortega Andrea Melissa
Espinoza Sánchez Joel Alejandro
Flores Fernández Óscar Alonso
Gómez Garza Dariana
González Arenas Fernando Francisco
Orocio García Hiram Efraín

Fecha de Entrega: Aguascalientes, Ags., 7 de octubre de 2021

Actividad 3.01: Arquitectura Cliente – Servidor con Planificador de Carga

Antecedentes

La arquitectura cliente/servidor se refiere a un sistema de información distribuido, persigue el objetivo de procesar la información de un modo distribuido. De esta forma, los usuarios finales pueden estar dispersos en un área geográfica más o menos extensa y acceder a un conjunto común de recursos compartidos, el acceso debe ser transparente y multiplataforma (independiente del sistema operativo del software e incluso del hardware).

Otras de las características que debe de tener son:

- Debe utilizar protocolos asimétricos, donde el servidor se limita a escuchar, en espera de que un cliente inicie una solicitud.
- El servidor ofrecerá recursos, tanto lógicos como físicos a una cantidad variable y diversa de clientes.
- El servidor ofrecerá también una serie de servicios, que serán usados por los clientes. Estos servicios estarán encapsulados, para ocultar a los clientes los detalles de su implementación.
- Se facilitará la integridad y el mantenimiento tanto de los datos como de los programas debido a que se encuentran centralizados en el servidor o servidores.
- Los sistemas estarán débilmente acoplados, ya que interactúan mediante el envío de mensajes.
- Se facilitará la escalabilidad, de manera que sea fácil añadir nuevos clientes a la infraestructura (escalabilidad horizontal) o aumentar la potencia del servidor o servidores, aumentando su número o su capacidad de cálculo (escalabilidad vertical)

Los elementos de los que está compuesta la arquitectura cliente/servidor son el servidor, el cliente y el middleware.

El servidor es un proceso que ofrece el recurso o recursos que administra a los clientes que lo solicitan. Según el tipo de servidor implantado, tendremos un tipo de arquitectura cliente/servidor diferente, en algunas ocasiones, un servidor puede actuar, a su vez, como cliente de otro servidor.

Un cliente es un proceso que solicita los servicios de otro, normalmente a petición de un usuario, en entornos cliente/servidor, suele utilizarse el término front-end para referirse a un proceso cliente. Normalmente, un proceso cliente se encarga de interactuar con el usuario, por lo que estará construido con alguna herramienta que permita implementar interfaces gráficas (GUI), además, se encargará de formular las solicitudes al servidor y recibir su respuesta, por lo que deberá encargarse de una parte de la lógica de la aplicación y de realizar algunas validaciones de forma local.

El Middleware es la parte del software del sistema que se encarga del transporte de los mensajes entre el cliente y el servidor, por lo que se ejecuta en ambos lados de la estructura, permite independizar a los clientes y a los servidores que eliminan la necesidad de supeditarse a tecnologías propietarias. Por lo tanto, es el que resuelve la parte del transporte de mensajes y facilita la interconexión de sistemas heterogéneos sin utilizar tecnologías propietarias, permitiendo obtener información desde diferentes orígenes y ofrecerla de manera conjunta. Podemos estructurar el middleware en tres niveles:

- El protocolo de transporte, que será común para otras aplicaciones del sistema.
- El sistema operativo de red
- El protocolo del servicio, que será específico del tipo de sistema cliente/servidor que estemos considerando.

La arquitectura cliente servidor cuenta con 4 niveles los cuales son:

- Un nivel de presentación, que aglutina los elementos relativos al cliente.
- Un nivel de aplicación, compuesto por elementos relacionados con el servidor.

- Un nivel de comunicación, que está formado por los elementos que hacen posible la comunicación entre el cliente y el servidor.
- Un nivel de base de datos, formado por los elementos relacionados con el acceso a los datos.

La topología de red se representa normalmente mediante un dibujo de líneas y objetos que refleja la topología física y lógica general.

Hay dos tipos diferentes de topologías de red:

- La topología de red física es la ubicación de diversos componentes de una red. Los diferentes conectores representan los cables de red físicos y los nodos representan los dispositivos de red físicos (como los switches).
- La topología de red lógica ilustra, en el nivel más alto, cómo fluyen los datos dentro de una red.

Herramientas de detección de la red

Diferentes herramientas pueden crear automáticamente un mapa de topología de red de una red de capa 2 y/o capa 3. Además, la mayoría de las herramientas de supervisión que utilizan SNMP u otros protocolos de supervisión remota pueden proporcionar un mapa de la red.

Detección de una topología de red de capa 2

Para las redes de capa 2, puede usar diferentes protocolos para detectar la topología de la red. Algunos proveedores tienen protocolos registrados o mecanismos de detección de red más esotéricos (como difusiones de red, etc.), pero los protocolos más populares son:

Protocolo de detección de capa de enlace (LLDP)

Es un protocolo de capa de enlace sin proveedor específico utilizado por dispositivos de red para anunciar su identidad, capacidades y vecinos en una red de área local basada en tecnología IEEE 802. Esto le permite detectar y anunciar automáticamente los vecinos del nodo.

Cisco Discovery Protocols (CDP)

Es un protocolo de capa de enlace de datos registrado y desarrollado por Cisco, que utilizan y admiten otros proveedores de red.

Estos protocolos se pueden usar para identificar quién, o qué, está conectado a un puerto de red específico mediante la escucha de los mensajes de LLDP o CDP, o también para anunciar que un dispositivo está conectado a un puerto específico. La mayoría de los switches admiten uno o ambos protocolos.

Otro método más complejo es analizar la tabla de direcciones MAC de cada switch o los paquetes del protocolo de árboles de expansión, para encontrar dónde están conectadas las direcciones MAC. Este método requiere un gran esfuerzo, por lo que suele ser el último recurso.

Detección de una topología de red de capa 3

En una red basada en IP, el Protocolo de mensajes de control de Internet (ICMP) es el protocolo de detección estándar.

Una herramienta común usada para identificar los diferentes saltos de red es traceroute (tracert en Windows), aunque algunas implementaciones podrían usar paquetes UDP en lugar de paquetes ICMP. Con este protocolo, puede encontrar las rutas de un paquete y detectar las redes lógicas y los routers.

En una única red lógica, puede usar un ping de difusión, herramientas específicas de barrido de IP o detección de caché ARP (entre otras herramientas similares) para identificar los diferentes nodos en la misma red. Debido al modo en que estos protocolos interactúan con las barreras de difusión, solo son eficaces dentro de una sola red.

Con el enrutamiento estático, es bastante fácil mostrar la configuración. Cada router puede mostrar las entradas de enrutamiento y los routers más cercanos en esas entradas. Sin embargo, la mayoría de las redes modernas utilizan un protocolo de enrutamiento para intercambiar información.

Con los protocolos de enrutamiento dinámico (como OSPF o BGP), puede consultar los vecinos de IP para identificar los routers que están anunciando o recibiendo las reglas de enrutamiento.

Detección de una topología de red virtual

Puede usar un hipervisor para ayudar a detectar la topología de red compleja. Cada hipervisor añade al menos un switch virtual usado para conectar las redes de VM con las redes físicas.

En función del hipervisor, se utilizan diferentes soluciones para mostrar la topología de red. Por ejemplo, con VMWare vSphere, los switches virtuales pueden admitir CDP y LLDP.

Con VMWare estándar y ESXi, el switch virtual admite solo CDP, en el modo de escucha y/o de anuncio. El switch virtual distribuido por VMware admite CDP y LLDP, pero por desgracia, solo se incluye en las licencias Enterprise Plus (o licencias VSAN o NSX) que normalmente quedan fuera del alcance de empresas como las PYMES.

Detección de una topología de red inalámbrica

En la mayoría de los casos, se utilizan herramientas en las redes Wi-Fi para simplificar la implementación y la configuración. Por ejemplo, con AirWave 8.2.4, Aruba introdujo una característica de topología de red, que es un mapa de capa 2 de la red por cable. Algunas herramientas también pueden proporcionar mapas de localización de puntos de acceso y de cobertura de señal para maximizar la eficacia de la red Wi-Fi.

Varias aplicaciones de smartphone pueden proporcionar los nombres de dispositivo Wi-Fi y SSID, el nivel de señal y los canales usados, pero el uso de estas herramientas no proporciona las mismas funciones o herramientas profesionales usadas para la implementación de redes inalámbricas.

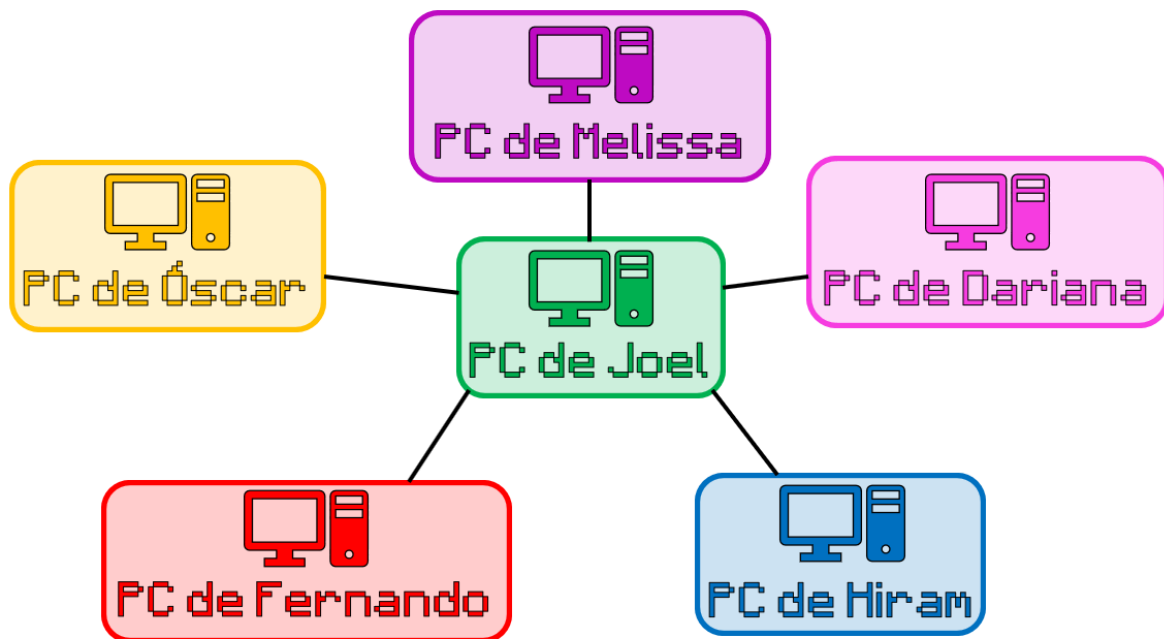
Arquitectura con Planificador de Carga

La arquitectura está planificada con dos tipos base de archivos en código de Python: será un archivo de Python a nivel de servidor y otro a nivel de cliente.

A partir de ahora se denotarán las computadoras como:

- PC de Melissa
- PC de Joel
- PC de Óscar
- PC de Dariana
- PC de Fernando
- PC de Hiram

Con la arquitectura siguiente



La idea de la arquitectura será principalmente un servidor central conectado a cinco clientes a modo de topología tipo estrella. De ser posible el servidor también trataría de actuar como otro cliente más. En el sentido estricto de que para este trabajo, el cliente cumplirá la función de procesar los tweets y realizar el trabajo que regresará

al servidor; para efectos de la actual práctica, se quiere decir, que el servidor también intentará ser un servidor que esté trabajando junto con los demás equipos.

En este sentido, los archivos básicos serán dos: el archivo del servidor y el archivo del cliente.

Por la estandarización de nombres pasada, los equipos cumplen las siguientes funciones:

PC de Melissa	Cliente	
PC de Joel	Cliente*	Servidor
PC de Óscar	Cliente	
PC de Dariana	Cliente	
PC de Fernando	Cliente	
PC de Hiram	Cliente	

* De ser posible que los recursos de su equipo lo dejen tomar a su vez este rol.

Para ello, se desarrollaron dos archivos: el archivo en ejecución del servidor y el archivo del cliente.

El Servidor

El servidor tiene dos fases de código (véase anexo 1), donde la primera esperará la recepción de los otros cinco equipos:

```
### Conexión
import socket

s = socket.socket()
host = socket.gethostname()
port = 8080
s.bind((host,port))
s.listen(1)
print(host)
print("Esperando conexiones...")
conn, addr = s.accept()
print(addr + " se ha conectado")
```

Por este medio, los usuarios se conectarán al servidor, quien esperará su registro en la conexión y posteriormente será notificado el servidor.

Una vez que estén conectados los equipos, el servidor pasará a transmitir archivos de control a los usuarios. Más adelante se explicarán estos archivos de control.

```
### Transfiriendo archivos
filename = "Control.txt"
file = open(filename, 'rb')
file_data = file.read(1024)
conn.send(file_data)
print("Archivo enviado")
```

El Cliente

Por otra parte, el archivo del cliente (véase anexo 2), será quien espere información del servidor, que al ser proporcionada, la ingresará en el programa. Así se podrá conectar en el servidor una vez que el servidor comience a ejecutarse:

```
### Conexión
import socket

s = socket.socket()
print("Ingresa la dirección del host")
host = input()
port = 8080
s.connect((host,port))
print("Conectado")
```

Después, el archivo contiene otro módulo de transferencia de datos. Es por este módulo por el que se enviarán datos del servidor al cliente y viceversa.

```
### Transfiriendo archivos
filename = "Control.txt"
file = open(filename, 'wb')
file_data = s.recv(1024)
file.write(file_data)
file.close()
print("Recibido")
```



Los archivos principales a enviar en futuras entregas del presente proyecto serán los tweets sin analizar del servidor al cliente y éstos mismos analizados del cliente al servidor, sin embargo, otros archivos de control serán enviados entre los equipos que son el núcleo del presente trabajo, pues se tratan de la estructura del planificador de carga.


¿Cómo funciona el planificador de carga?

El planificador de carga en sí será todo documento compartido por medio de archivos de formato .txt entre el servidor y el cliente.



Una prueba de trabajo de ellos es el archivo prueba.txt ubicado en el equipo del servidor junto a su respectivo directorio. En este caso de entrega, el archivo no contiene ninguna información relevante de la planificación de tareas, sin embargo,

pueden mostrarse los dos directorios de servidor y cliente respectivamente a continuación:

 prueba.txt	06/10/2021 06:38 p. m.	Documento de te...	1 KB
 server.py	06/10/2021 11:30 p. m.	Archivo PY	1 KB

 clientJoul.py	06/10/2021 11:30 p. m.	Archivo PY	1 KB
---	------------------------	------------	------

Y al ejecutarse ambos programas y estar ambos equipos en contacto, el cliente obtiene este archivo de texto:

 clientJoul.py	06/10/2021 11:30 p. m.	Archivo PY	1 KB
 prueba.txt	06/10/2021 06:38 p. m.	Documento de te...	1 KB

Sin embargo, el verdadero orden de documentos en cliente y servidor será regularmente el siguiente:

- El servidor:
 - El archivo Python del servidor `server.py`.
 - Un archivo de control de planificación de carga y administración `Control-Servidor.txt`.
- El cliente:
 - El archivo Python del servidor `client.py`.
 - Un archivo de recepción de control de planificación de carga y administración `Recepcion-Nombre.txt`. El nombre del archivo variará según el integrante del equipo. Ejemplos: `Recepcion-Melissa.txt`, `Recepcion-Hiram.txt`.
 - Un archivo de emisión de control de planificación de carga y administración `Emission-Nombre.txt`. El nombre del archivo variará según el integrante del equipo. Ejemplos: `Emission-Dariana.txt`, `Emission-Oscar.txt`.

La función de los archivos de texto plano serán encargarse del plan de carga.

Todo lo narrado a continuación es una simulación que aún no se realiza debido a que no se ha implementado el intercambio de información de tweets pero al realizarse, los números comenzarán a cambiar en los archivos.

Primeramente el archivo de control del servidor (véase anexo 3) tendrá la información general de los clientes. Primero el estado del servidor y de cada cliente. 0 significa que está apagado el servidor o los clientes no están en línea y 1 significa que el servidor está activo o los clientes están conectados a la red.

```
Estado: 0
Cliente Melissa: 0
Cliente Joel: 0
Cliente Oscar: 0
Cliente Dariana: 0
Cliente Fernando: 0
Cliente Hiram: 0
```

Por ejemplo, cuando el servidor esté activo, el primer renglón estará activo y los integrantes conectados en ese momento estarán señalados en el archivo también:

```
Estado: 1
Cliente Melissa: 1
Cliente Joel: 0
Cliente Oscar: 0
Cliente Dariana: 0
Cliente Fernando: 1
Cliente Hiram: 1
```

Posteriormente se encuentra un apartado para cada integrante en el que se reporta el progreso personal de cada integrante y la asignación de cada uno a nivel cíclico dinámico como se muestra a continuación:

```
MELISSA
Procesador (ultimo contacto): 0
Historial del procesador:

Tweets enviados a procesar: 0
Tweets ya procesados: 0
Tiempo: 0
```

Este formato se repite para los seis participantes del proyecto.

En un ejemplo por dar explicación a cada variable tomada en cuenta para cada usuario, se propone el siguiente ejemplo a explicar:

```
MELISSA
Procesador (ultimo contacto): 12
Historial del procesador:
8 4 6 12 29 34 48 37 25
Tweets enviados a procesar: 400
Tweets ya procesados: 1000
Tiempo: 1.2302
```

La imagen anterior estaría proporcionando la información del PC de Melissa que tuvo un 12% de uso del procesador en el último contacto que se tuvo con su equipo, las siguientes líneas serían el historial del procesador en cada contacto con el PC de Melissa, después se habrían indicado que el algoritmo genético habría calculado que lo óptimo es enviarle en el actual periodo de tiempo 400 tweets a procesar, mientras que ya tendría 1,000 tweets reportados al servidor como procesados en un tiempo de 1.2302 segundos.

Se hace hincapié que lo mencionado anteriormente es un caso hipotético y no ha sido posible llevarlo a la realidad dado que todavía no se ha iniciado con la transmisión de tweets del servidor al cliente.

Por parte del cliente, estos datos serían enviados al archivo de recepción del cliente en el que se le indicaría primeramente si su conexión ya se realizó y posteriormente los tweets asignados para que trabaje su equipo, junto con un archivo CSV con los tweets indicados.

```
MELISSA
Conexion: 0
Tweets recibidos para procesar: 0
```

Cuando el cliente termina de trabajar en su procesamiento, escribirá sobre el archivo de emisión que será enviado al servidor con los datos que actualizará sobre el archivo de control, proporcionando los siguientes datos:

```
MELISSA  
Procesador: 0  
Tweets procesados: 0  
Tiempo: 0
```

De modo que la información del plan de carga se mantendría en flujo en sintonía con los archivos CSV enviados a través de los usuarios de esta red y, donde si todo funciona bien, el planificador de carga mantendría una gestión de cada usuario correcta.

Conclusiones

Andrea Melissa Almeida Ortega: Como se pudo observar ya en la práctica de una arquitectura cliente/servidor enfocada en nuestro proyecto, nos damos cuenta que la optimización es algo esencial para el software y no es simplemente priorizar que realice las acciones más rápido, si no de entender lo que está haciendo el programa para poder repartir las distintas tareas y acciones para que se puedan ejecutar de la manera más eficaz posible.

Joel Alejandro Espinoza Sánchez: La implementación y uso de una arquitectura cliente – servidor será muy importante para el proyecto de la materia y más importante con un planificador de carga que evite los fallos a lo largo de su ejecución. Con esta arquitectura se puede conectar un conjunto de computadoras para la comunicación de ellas y obtener ventajas sobre información entre las computadoras.

Óscar Alonso Flores Fernández: Seguir aprendiendo con la elaboración de estas prácticas acerca de los servidores/clientes, sus "complementos" y siendo capaces de poco a poco ir poniendo en práctica nos damos cuenta una vez más de como cuando nos referimos a la optimización no solo nos tenemos que referir a dividir tareas pesadas, es decir, que requieran una cantidad significativa de procesadores trabajando simultáneamente para llevarla a cabo, sino también de incluso las tareas más sencillas poder dividir las para cubrir la mayor área de trabajo posible en el menor tiempo usando las herramientas (en este caso procesadores) para avanzar de una manera eficiente y veloz.

Dariana Gómez Garza: Para realizar esta práctica se tuvo que buscar un poco más de información sobre redes ya que teníamos puro conocimiento teórico, nos dimos cuenta que esto nos serviría para agilizar nuestro trabajo que estamos realizando para el proyecto de esta materia.

También visualizamos aspectos importantes y que no habíamos tomado en cuenta antes y que son fundamentales para la práctica.

Fernando Francisco González Arenas: Usar un servidor permite utilidades en las que podemos comunicar las computadoras fácilmente. En este trabajo podemos observar cómo comunicar nuestras computadoras a partir de un programa en Python y un planificador de carga funcional y tener nuestras computadoras conectadas.

Hiram Efraín Orocio García: Un servidor es útil al momento de hacer uso de recursos, datos, servicios entre otras cosas así que aprender a implementar un así sea en una aplicación básica, en un futuro lo podríamos implementar en problemas mayores donde el compartir estos recursos, serán de gran ayuda.

Referencias bibliográficas

1. Arquitectura cliente/servidor - SomeBooks.es. (2013). Retrieved 7 October 2021, from <http://somebooks.es/arquitectura-clienteservidor/>.
2. Arquitectura cliente-servidor - RedesOrdenadores_Grupoc. (2021). Retrieved 7 October 2021, from <https://sites.google.com/site/redesordenadoresgrupoc/home/arquitectura-cliente-servidor>.
3. Berson, A. (1996). *Client/Server Architecture*. Los Angeles: McGraw Hill.
4. Proper, H. (2017). *Architectural Coordination of Enterprise Transformation*. London: Springer.

Anexos

Anexo 1: Código de server.py en Python:

```
### Conexión
import socket

s = socket.socket()
host = socket.gethostname()
port = 8080
s.bind((host,port))
s.listen(1)
print(host)
print("Esperando conexiones...")
conn, addr = s.accept()
print(addr + " se ha conectado")

### Transfiriendo archivos
filename = "Control.txt"
file = open(filename, 'rb')
file_data = file.read(1024)
conn.send(file_data)
print("Archivo enviado")
```

Anexo 2: Código de client.py en Python:

```
### Conexión
import socket

s = socket.socket()
print("Ingresa la dirección del host")
host = input()
port = 8080
s.connect((host,port))
print("Conectado")

### Transfiriendo archivos
filename = "Control.txt"
file = open(filename, 'wb')
file_data = s.recv(1024)
file.write(file_data)
file.close()
print("Recibido")
```

Anexo 3: Archivo de texto plano Control-Servidor.txt en su estado inicial:

Estado: 0
Cliente Melissa: 0
Cliente Joel: 0
Cliente Oscar: 0
Cliente Dariana: 0
Cliente Fernando: 0
Cliente Hiram: 0

MELISSA
Procesador (ultimo contacto): 0
Historial del procesador:

Tweets enviados a procesar: 0
Tweets ya procesados: 0
Tiempo: 0

JOEL
Procesador (ultimo contacto): 0
Historial del procesador:

Tweets enviados a procesar: 0
Tweets ya procesados: 0
Tiempo: 0

OSCAR
Procesador (ultimo contacto): 0
Historial del procesador:

Tweets enviados a procesar: 0
Tweets ya procesados: 0
Tiempo: 0

DARIANA
Procesador (ultimo contacto): 0
Historial del procesador:

Tweets enviados a procesar: 0
Tweets ya procesados: 0
Tiempo: 0

FERNANDO
Procesador (ultimo contacto): 0
Historial del procesador:

Tweets enviados a procesar: 0
Tweets ya procesados: 0
Tiempo: 0

HIRAM
Procesador (ultimo contacto): 0
Historial del procesador:

Tweets enviados a procesar: 0
Tweets ya procesados: 0
Tiempo: 0

**Anexo 4: Archivo de texto plano Recepcion-Nombre.txt en su estado inicial
(caso de ejemplo: Melissa):**

MELISSA

Conexion: 0

Tweets recibidos para procesar: 0

**Anexo 3: Archivo de texto plano Emision-Nombre.txt en su estado inicial
(caso de ejemplo: Melissa):**

MELISSA

Procesador: 0

Tweets procesados: 0

Tiempo: 0