



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

Práctica 6: Balanceador de carga

Unidad de aprendizaje: Aplicaciones para comunicaciones en red

Grupo: 3CM7

Alumnos:

- Ontiveros Salazar Alan Enrique
- Sánchez Valencia Sergio Gabriel

Profesor:

Moreno Cervantes Axel Ernesto

3 de diciembre de 2018

Práctica 6: Balanceador de carga

3CM7

3 de diciembre de 2018

1. Marco teórico

1.1. Balanceador de carga

Es una técnica para distribuir o compartir cargas de trabajo a través de múltiples recursos de cómputo, tales como computadoras, clústers, enlaces de red, unidades de proceso o discos duros. Su objetivo es optimizar el uso de recursos, maximizar el rendimiento y minimizar el tiempo de respuesta, evitando la sobrecarga de un único recurso.

El uso de múltiples componentes con balance de carga en lugar de uno solo puede incrementar la confiabilidad y la disponibilidad, a costa de una mayor redundancia. El balance de carga se lleva a cabo con software o hardware dedicado.

Internamente, está basado en algún algoritmo que divide de la manera más equitativa posible la carga de trabajo, para evitar los famosos cuellos de botella.

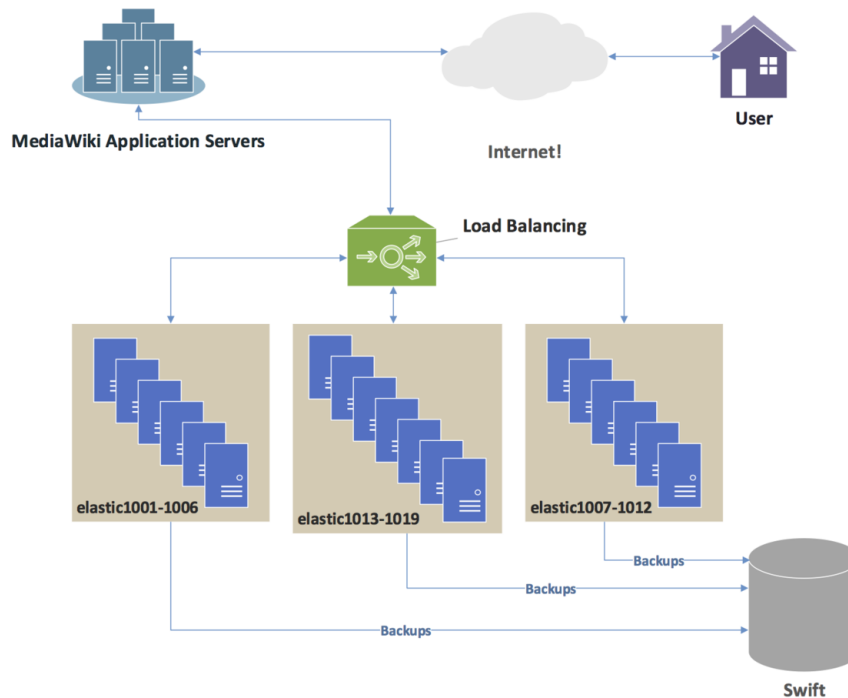


Figura 1: Diagrama básico de un balanceador de carga

1.2. Round-robin

Es un algoritmo de planificación en computación. Consiste en asignarle a cada proceso unidades de tiempo iguales (quantums) en un orden circular, manejando todos los procesos sin prioridad. Es un algoritmo simple y

fácil de implementar.

Distribuye todo de manera equitativa y en un orden racional, normalmente comenzando por el primer elemento de la lista hasta llegar al último y empezando de nuevo desde el primer elemento.

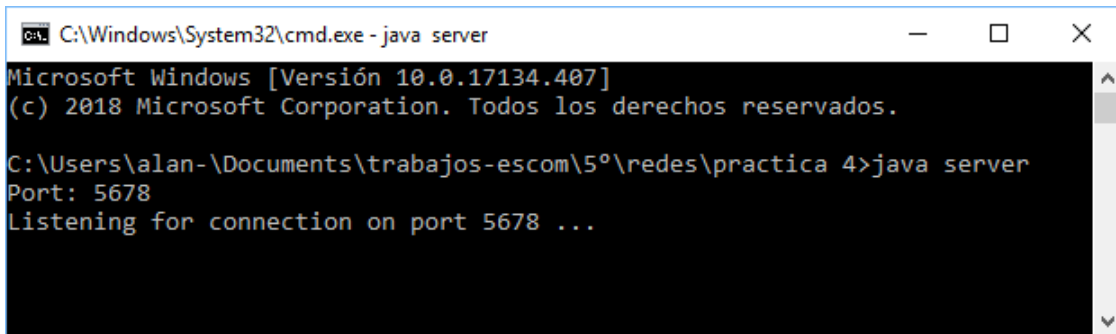
2. Desarrollo

En esta práctica usaremos el servidor HTTP que hicimos en la práctica 4, solo que esta vez lo correremos simultáneamente en distintos puertos (del 5678 al 5681) para tener cuatro disponibles. Haremos un nuevo servidor en el puerto 1234 que actuará de forma similar a un proxy: recibirá todas las solicitudes HTTP que le lleguen y las repartirá a un puerto de los disponibles, luego esperará su respuesta mediante un socket bloqueante y la regresará al cliente que la solicitó mediante el socket no bloqueante.

Internamente llevaremos una variable que nos dirá la posición del servidor actual, de tal forma que se estará incrementando cada vez que llegue un nuevo cliente, y regresando a 0 si ya usamos todos los servidores (esto se logra de manera muy sencilla con el operador módulo), algo parecido a una cola circular. Así logramos que la repartición de la carga sea lo más igual posible entre todos los servidores disponibles.

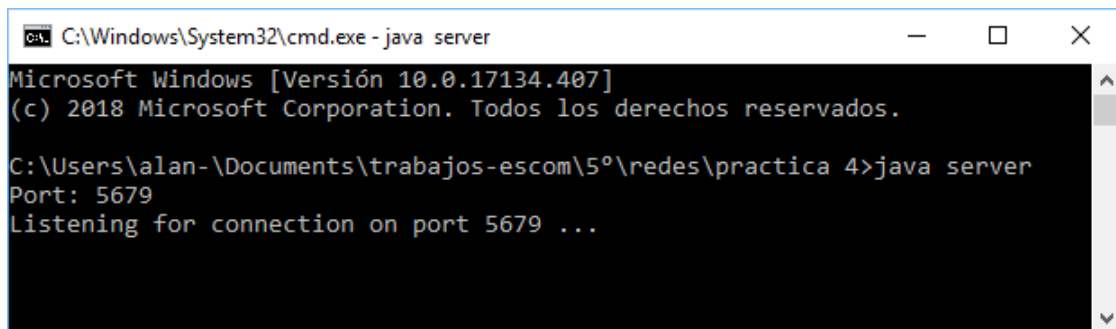
3. Pruebas

Corremos los cuatro servidores:



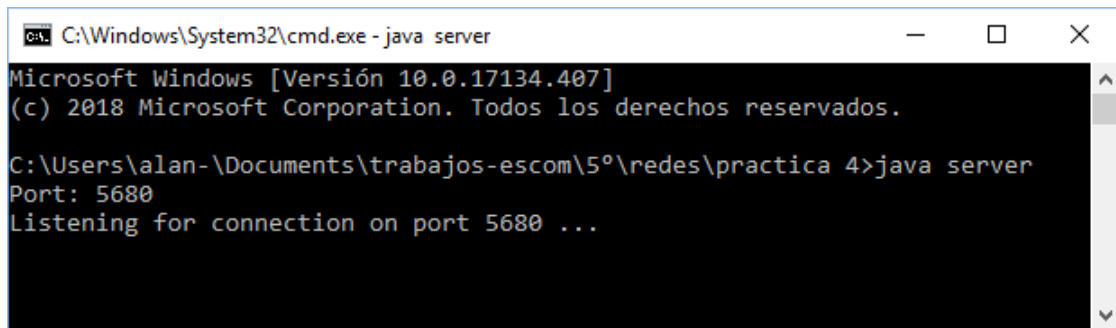
```
C:\Windows\System32\cmd.exe - java server
Microsoft Windows [Versión 10.0.17134.407]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\alan-\Documents\trabajos-escom\5º\redes\practica 4>java server
Port: 5678
Listening for connection on port 5678 ...
```



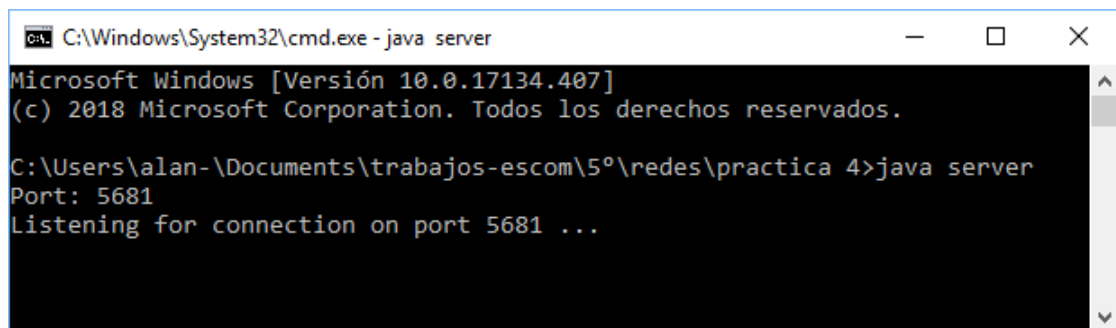
```
C:\Windows\System32\cmd.exe - java server
Microsoft Windows [Versión 10.0.17134.407]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\alan-\Documents\trabajos-escom\5º\redes\practica 4>java server
Port: 5679
Listening for connection on port 5679 ...
```



```
C:\Windows\System32\cmd.exe - java server
Microsoft Windows [Versión 10.0.17134.407]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

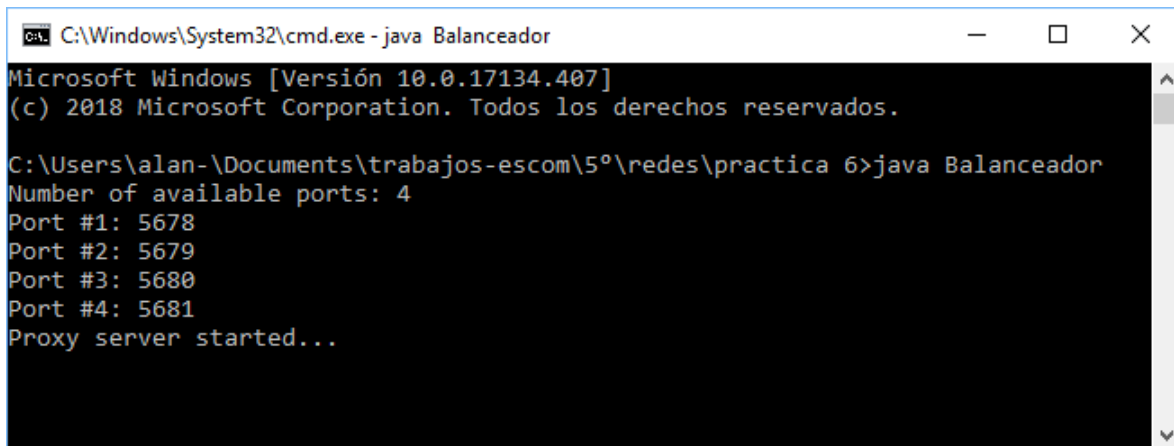
C:\Users\alan-\Documents\trabajos-escom\5º\redes\practica 4>java server
Port: 5680
Listening for connection on port 5680 ...
```



```
C:\Windows\System32\cmd.exe - java server
Microsoft Windows [Versión 10.0.17134.407]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\alan-\Documents\trabajos-escom\5º\redes\practica 4>java server
Port: 5681
Listening for connection on port 5681 ...
```

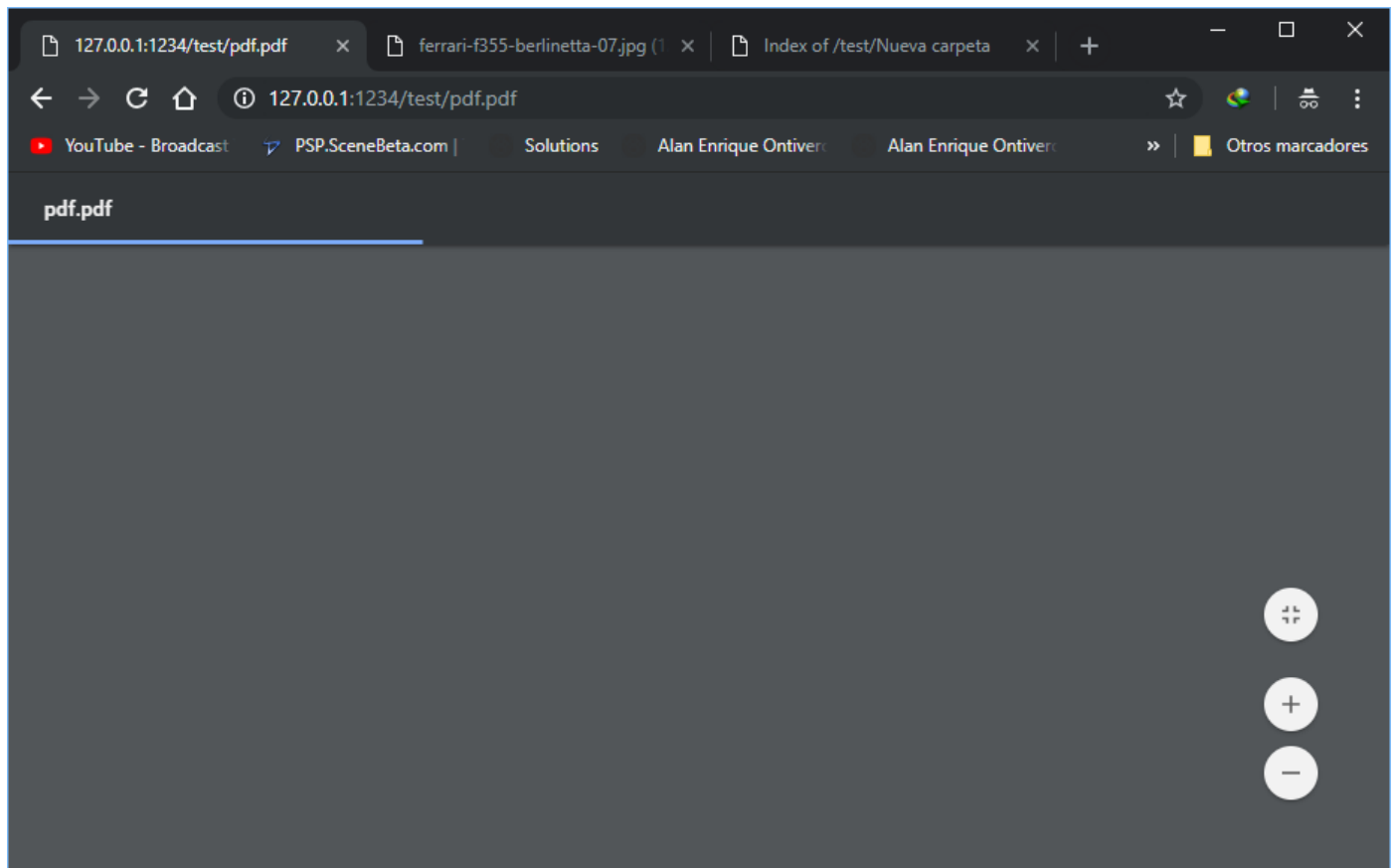
Corremos el balanceador de carga:



```
C:\Windows\System32\cmd.exe - java Balanceador
Microsoft Windows [Versión 10.0.17134.407]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\alan-\Documents\trabajos-escom\5º\redes\practica 6>java Balanceador
Number of available ports: 4
Port #1: 5678
Port #2: 5679
Port #3: 5680
Port #4: 5681
Proxy server started...
```

Realizamos algunas solicitudes desde el navegador:



Y vemos cómo éstas se repartieron entre los cuatro servidores:

```
C:\Windows\System32\cmd.exe - java server
(c) 2018 Microsoft Corporation. Todos los derechos reservados.
C:\Users\alan-\Documents\trabajos-escom\5º\redes\practica 4>java server
Port: 5678
Listening for connection on port 5678 ...
New client connected from /127.0.0.1:14298
  Got a request from /127.0.0.1:14298 of type GET with url /
New client connected from /127.0.0.1:14303
  Got a request from /127.0.0.1:14303 of type GET with url /test
New client connected from /127.0.0.1:14312
  Got a request from /127.0.0.1:14312 of type GET with url /test/pdf.pdf
New client connected from /127.0.0.1:14332
New client connected from /127.0.0.1:14336
```

```
C:\Windows\System32\cmd.exe - java server
Listening for connection on port 5679 ...
New client connected from /127.0.0.1:14299
Got a request from /127.0.0.1:14299 of type GET with url /favicon.ico
New client connected from /127.0.0.1:14304
Got a request from /127.0.0.1:14304 of type GET with url /test/carpeta
New client connected from /127.0.0.1:14323
New client connected from /127.0.0.1:14333
New client connected from /127.0.0.1:14337
```

```
C:\Windows\System32\cmd.exe - java server
New client connected from /127.0.0.1:14301
Got a request from /127.0.0.1:14301 of type GET with url /test
New client connected from /127.0.0.1:14305
Got a request from /127.0.0.1:14305 of type GET with url /test/Nueva%20car
peta
New client connected from /127.0.0.1:14328
New client connected from /127.0.0.1:14334
New client connected from /127.0.0.1:14338
```

```
C:\Windows\System32\cmd.exe - java server
New client connected from /127.0.0.1:14302
Got a request from /127.0.0.1:14302 of type GET with url /test
New client connected from /127.0.0.1:14306
Got a request from /127.0.0.1:14306 of type GET with url /test/carpeta/fe
rrari-f355-berlinetta-07.jpg
New client connected from /127.0.0.1:14329
New client connected from /127.0.0.1:14335
New client connected from /127.0.0.1:14339
```

4. Conclusiones

Al usar sockets no bloqueantes evitamos la creación de un hilo por cada cliente que llegue; en lugar de eso Java implementa la técnica de los selectores, que nos dicen los eventos con los que nosotros podemos interactuar (lectura, escritura, nueva conexión, etc.), y justo en el momento que los necesitamos. De esta manera solo necesitamos crear un hilo, optimizando memoria y tiempo de procesamiento.