Sara Soriano - 240007
Rubén Vera - 241456
Eneko Treviño - 241679

DOCUMENTACIÓN PRÁCTICA 6

Después de ejecutar todos los comandos que se nos indicaba en la práctica, hemos ejecutado el comando “grep” para poder saber el ancho de banda usado en las descargas de cada uno de los modos de ejecución (origin, random, optimize) y a partir de ahí poder sacar las siguientes conclusiones:

- En el caso de origin, el recorrido desde todos los nodos hasta el “origin” para poder bajarse el archivo es el mayor de todos. Por lo cual, el ancho de banda va a ser menor ya que hay menor velocidad a causa de la distancia entre los dos nodos. La *request* del archivo tendrá que pasar por más nodos de la topología y eso conlleva una penalización temporal ya que, por ejemplo, cuando llegue al router este tendrá que mirar su router table para llegar a la red del “origin” y así sucesivamente hasta llegar a ella. De la manera en que está hecho encontraremos el archivo que buscamos seguro ya que nos estamos yendo al servidor que contiene todos los archivos, pero sacrificando velocidad por lo comentado anteriormente.

```
mininet@mininet:~/Software/cdn/results$ grep " MB/s)" *origin* | cut -d '(' -f 2
 | cut -d ' ' -f 1 | sed "s/,/./g" | awk -vx=0 '{x += (($1==int($1))?$1/1000:$1)
} END {print x/NR}'
1
```

- Sin embargo, en el caso de random, el ancho de banda será diferente en cada ejecución por el hecho de que todos los nodos irán a un *pop* diferente. Y, por lo tanto, en alguna ejecución irán a uno más cercano y en otra a uno más lejano. Random puede ser mejor que el origin o incluso peor, en nuestro caso ha sido mejor. De todas formas, no es la manera óptima ya que no está yendo al más cercano, sino a uno aleatorio. De esta forma, además, ni nos aseguramos de que el *pop* al que vayamos contenga el archivo que buscamos ni garantizamos máxima velocidad por el motivo de la distancia hasta él. El “grep” devuelve la media de ancho de banda de todos los paquetes, pero si mirásemos el ancho de banda de un solo cliente, encontraríamos lo mencionado de que en algunos casos sería incluso menor que el ancho de banda del mismo host ejecutado en modo origin.

```
mininet@mininet:~/Software/cdn/results$ grep " MB/s)" *random* | cut -d '(' -f 2
 | cut -d ' ' -f 1 | sed "s/,/./g" | awk -vx=0 '{x += (($1==int($1))?$1/1000:$1)
} END {print x/NR}'
5,125
```

Sara Soriano - 240007
Rubén Vera - 241456
Eneko Treviño - 241679

- Finalmente, en el caso optimize, el ancho de banda obtenido será mayor respecto a las ejecuciones anteriores, debido a que en cada ejecución las ejecuciones irán al *pop* más cercano. Y, aunque no nos aseguremos de que el *pop* contenga el archivo y haya de ser pedido al "origin", una vez descargado y obtenido en el *pop*, sabemos que en la siguiente ejecución pidiendo el mismo archivo nos aseguramos que la velocidad será máxima. Por lo tanto, el ancho de banda también.

```
mininet@mininet:~/Software/cdn/results$ grep " MB/s)" *optimize* | cut -d '(' -f
 2 | cut -d ' ' -f 1 | sed "s/,/./g" | awk -vx=0 '{x += (($1==int($1))?$1/1000:$
1)} END {print x/NR}'
8,04167
```

Para conseguir el modo optimize hemos modificado el archivo "cdn_selector.py" como se pide en el enunciado. La modificación ha sido añadir una matriz con todas las IP's de los *hosts* adjudicando cada fila a una LAN. Además, hemos usado una función para conseguir la IP del nodo que está haciendo la *request*.

Por cómo está hecha la topología y colocados los *pops* simplemente hemos de buscar la fila de topology_matrix en la que se encuentra esa IP, dicha fila será el número de *pop* al que se le hará *redirect*. Esto es así ya que previamente la IP de los *pop*s estaban guardadas en una array (pop_ips) en la cual se distinguen las diferentes LAN's y, por lo tanto, la fila equivale al número de *pop* adecuado.

```
21      elif option == 'optimize':
22          pop_ips = ['10.10.15.2','10.10.25.2','10.10.35.2','10.10.45.2']
23          client_ip = request.remote_addr
24          topology_matrix = [['10.1.1.11', '10.1.1.12','10.1.1.13','10.1.1.14','10.1.1.15','10.1.1.16'],['10.1.2.11',
    '10.1.2.12','10.1.2.13','10.1.2.14','10.1.2.15','10.1.2.16'], ['10.1.3.11', '10.1.3.12','10.1.3.13','10.1.3.14','10.1.3.15','10.1.3.16'],
    ['10.1.4.11', '10.1.4.12','10.1.4.13','10.1.4.14','10.1.4.15','10.1.4.16']]
25          '''
26          TODO
27          Create your own code to improve download times for each user,
28          depending on client IP, calculate index in pop_ips array to give a redirect url to client.
29          '''
30          for i in range(len(topology_matrix)):
31              for j in range(len(topology_matrix[i])):
32                  if(topology_matrix[i][j] == client_ip):
33                      index = i
34          redirect_url = 'http://%s:5200/repository/%s'%(pop_ips[index],filename)
```