Tarea 2 Redes: Cliente eco UDP para medir performance

Nombre: Patricio Espinoza A. **Sección**: 1

Preparación

Comando de uso general

- $ ./client_bw.py size IN OUT host port

Generar un archivo de 1 GB, uno de 500 KB, y 100 archivos de 10.000 KB y de 5KB (.bin)

- $ python3 files_generator.py

Ejecución del servidor local

- $ python3 server_echo_udp3.py 1818

Limpiar cache

- $ sudo sync && echo 3 | sudo tee /proc/sys/vm/drop_caches

Observaciones:
[1] En los casos de timeout o message too long se registró el tiempo real obtenido, aunque este no debería de usarse para calcular un ancho de banda pues la ejecución falló.

[2] En los casos de ejecuciones en paralelo (localhost y anakena), ocurrió que algunas recibían el EOF y otras arrojaban timeout. Se guardo como tamaño total la suma de todos los tamaños de los archivos generados. Para sizes muy grandes fue notoria la pérdida de paquetes.

Ejecuciones y Resultados

Localhost

- **Un archivo de 1000000 KB (1GB) → 1000000000 bytes**

$ time python3 client_bw.py ***size*** files_localhost/big_file.bin outputs_localhost/OUT_***size*** 127.0.0.1 1818

- Limpiando cache

Size (bytes)	Tiempo real (s)	Ancho de banda (B/s)	Tamaño final (KB)
100	350,57	0,28	99.618
500	14,68	34,05	246.600
1.000	9,65	103,62	195.631
10.000	4,130	2.421,3	169.307
100.000	3,21	Msg too long	0
1.000.000	3,10	Msg too long	0
10.000.000	3,04	Msg too long	0

- **100 archivos de 10.000 KB**

  $ time for i in {0..99}; do
  python3 client_bw.py ***size*** files_localhost/file_$i.bin outputs_localhost/OUT_$i 127.0.0.1 1818 &
  done
  wait

- Limpiando el cache

Size (bytes)	Tiempo real (s)	Ancho de banda	Tamaño total recibido de los archivos (KB)
10	1,472	6,79	209 KB
100	1,026	97,46	2.579 KB
500	0,166	3,01	1.205 KB
1.000	0,655	1526	9.851 KB
10.000	1,069	9354	16.264 KB
100.000	0,483	Msg too long	5KB[1] + 0KB[99]
1.000.000	0,763	Msg too long	5KB[1] + 0KB[99]
10.000.000	1,059	Msg too long	5KB[1] + 0KB[99]

Observaciones:
[1] En los casos de timeout o message too long se registró el tiempo real obtenido, aunque este no debería de usarse para calcular un ancho de banda pues la ejecución falló.

[2] En los casos de ejecuciones en paralelo (localhost y anakena), ocurrió que algunas recibían el EOF y otras arrojaban timeout. Se guardo como tamaño total la suma de todos los tamaños de los archivos generados. Para sizes muy grandes fue notoria la pérdida de paquetes.

Anakena

sudo sync && echo 3 | sudo tee /proc/sys/vm/drop_caches

- **Un archivo de 500 KB**

$ time python3 client_bw.py ***size*** files_anakena/big_file.bin outputs_anakena/OUT_***size*** anakena.dcc.uchile.cl 1818

- Limpiando cache

Size (bytes)	Tiempo real (s)	Ancho de banda	Tamaño final (KB)
10	4,84	2,06	54
100	4,45	timeout	249
500	0,982	509,1	80
1.000	0,376	2659,57	280
10.000	3,13	timeout	0
100.000	3,16	Msg too long	0
1.000.000	3,14	Msg too long	0
10.000.000	3,15	Msg too long	0

- **100 archivos de 5 KB**

```
$ time for i in {0..99}; do
  python3 client_bw.py size files_anakena/file_$i.bin outputs_anakena/OUT_$i
anakena.dcc.uchile.cl 1818 &
done
wait
```

- Limpiando el cache

Size (bytes)	Tiempo real (s)	Ancho de banda	Tamaño total recibido de los archivos (KB)
10	1,57	6,36	39,2
100	1,52	65,78	146
500	1,10	454,54	496
1.000	1,049	953	478
10.000	0,797	12547	10
100.000	0,327	0305,81	5
1.000.000	0,401	2493	5
10.000.000	0,826	0	0

Observaciones:
[1] En los casos de timeout o message too long se registró el tiempo real obtenido, aunque este no debería de usarse para calcular un ancho de banda pues la ejecución falló.

[2] En los casos de ejecuciones en paralelo (localhost y anakena), ocurrió que algunas recibían el EOF y otras arrojaban timeout. Se guardo como tamaño total la suma de todos los tamaños de los archivos generados. Para sizes muy grandes fue notoria la pérdida de paquetes.

Preguntas

1. Si el archivo de salida es más pequeño que el de entrada, ¿es correcto medir el ancho de banda disponible como tamaño recibido/tiempo?

   Si es lo correcto, este caso nos permite observar cuantos datos del archivo original que enviamos efectivamente llegaron al receptor. Dependiendo de la red puede suceder que haya perdida de paquetes, y por tanto es correcto hacer esta medición para determinar la calidad de la red en base al ancho de banda.

2. Si el archivo de salida es más pequeño que el de entrada, ¿es correcto medir el ancho de banda disponible como tamaño enviado/tiempo?

   No es correcto ya que no nos entrega una información útil para determinar la calidad de la red. Al medirlo de esta forma, pensaríamos equivocadamente que no hay diferencias entre UDP y TCP ya que en ambas solo tomamos el tamaño enviado, pero en UDP se perdieron paquetes lo que afecta el tamaño enviado.

3. Una medición que termina por timeout, ¿podría usarse de alguna forma para medir el ancho de banda disponible?

   No se podría usar ya que el timeout implica que hubo alguna falla en la ejecución, por tanto, se obtendría un ancho de banda basado en métricas incorrectas. Esto puede deberse a que no llega el paquete EOF, o que el receptor no esta recibiendo más. Para lo que es útil es que, en caso de que sea algo constante, y falle con ***sizes*** que no debería, podría dar indicios de fallas a nivel del código/estructura de la red (cliente/servidor), pero no para ancho de banda.

Comparación

Dentro de las diferencias principales entre la tarea 1 con TCP, y la tarea 2 con UDP, es que esta última resulta ser más rápida con tiempos reales generalmente menores. Otra diferencia notable que recalca lo visto en clases, es la pérdida de paquetes y la estabilidad entre las redes, en donde TCP se demora más, pero es más fiable, mientras que en UDP ocurre mucho la pérdida de paquetes. También en esta última se pudo observar el límite al enviar chunks, en donde para sizes muy grandes dejó de responder y falló.

Observaciones:
[1] En los casos de timeout o message too long se registró el tiempo real obtenido, aunque este no debería de usarse para calcular un ancho de banda pues la ejecución falló.

[2] En los casos de ejecuciones en paralelo (localhost y anakena), ocurrió que algunas recibían el EOF y otras arrojaban timeout. Se guardo como tamaño total la suma de todos los tamaños de los archivos generados. Para sizes muy grandes fue notoria la pérdida de paquetes.