Tarea 5 Redes: Cliente eco UDP con Selective Repeat adaptativo para medir performance

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

Preparación

Comando de uso general

- $ ./client_sr2_bw.py size timeout win IN OUT host port

Generar un archivo de 500KB o de 5000KB (.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

Ejecuciones y Resultados

Localhost

- **Un archivo de 500 KB**

En base a resultados anteriores, un size de 1.000 suele funcionar bien, por tanto, estableceremos ese size fijo e iremos cambiando los timeouts y windows para la obtención de métricas.

$ time python3 client_sr2_bw.py 1000 0.1 window files_localhost/big_file.bin outputs_localhost/OUT_500 127.0.0.1 1818

- Limpiando cache

Size (bytes)	Timeout	Window	Tiempo real (s)	Sent packets	Retrans packets	% error	Total packets	Max window	Mín Cong	Máx RTT	Ancho de banda (B/s)
1.000	0.1	10	0,15	515	203	28,27	718	9	5	0,0125	3.333
1.000	0.1	100	0,19	515	373	42,00	888	99	50	0,0646	2.631
1.000	0.1	500	0,45	515	504	49,46	1.019	499	1	0,0008	1.111
1.000	0.1	900	0,43	515	422	45,03	937	514	304	0,1645	1.162
1.000	0.3	10	0,28	515	237	31,51	752	9	5	0,0113	1.785
1.000	0.3	100	0,21	515	654	55,94	1.169	99	50	0,0403	2.380
1.000	0.3	500	0,13	515	1.107	68,24	1.622	499	250	0,0832	3.846
1.000	0.3	900	0,35	515	495	49,00	1.010	514	271	0,0973	1.428
1.000	0.8	100	0,18	515	490	48,75	1.005	99	50	0,0345	2.777

Al comparar los timeouts para una misma window = 100 podemos observar que al aumentar el timeout, el bando de ancha pareciera tender a aumentar, aunque por el tamaño pequeño del tiempo real no es algo que se pueda concluir con solo localhost.

Además, al mantener fijo el timeout e incrementar el tamaño de la window, se observan intervalos intermitentes para el timeout de 0.3, mientras que para un tiemeout de 0.1 se observa un claro descenso del ancho de banda. Los intervalos intermitentes en el timeout=0.3 pueden atribuirse a la pequeña diferencia de decimales dado el contexto del tiempo real en localhost.

Anakena

- **Un archivo de 500 KB**

En base a resultados anteriores, un size de 1.000 suele funcionar bien, por tanto, estableceremos ese size fijo e iremos cambiando los timeouts y windows para la obtención de métricas.

$time python3 client_sr2_bw.py **1000 timeout window** files_anakena/big_file.bin outputs_anakena/OUT_500 anakena.dcc.uchile.cl 1818

- Limpiando cache

Size (bytes)	Timeout	Window	Tiempo real (s)	Sent packets	Retrans packets	% error	Total packets	Max window	Mín Cong	Máx RTT	Ancho de banda (B/s)
1.000	0.1	10	24,01	515	8	1,529	523	9	5	0,6645	20,82
1.000	0.1	100	53,54	515	1.503	74,47	2.018	99	1	0,6447	9,28
1.000	0.1	500	45,41	515	8.463	94,26	8.978	499	7	0,6054	11,01
1.000	0.1	900	26,32	515	5.533	91,48	6.048	514	19	0,5672	18,99
1.000	0.3	10	21,98	515	14	2,64	529	9	5	0,6763	22,74
1.000	0.3	100	38,92	515	962	65,13	1477	99	4	0,7085	12,84
1.000	0.3	500	26,98	515	6.513	92,67	7.028	499	5	0,5625	18,53
1.000	0.3	900	13,87	515	2.879	84,82	3.394	514	16	0,5552	36,04
1.000	0.8	100	23,48	515	722	58,36	1.237	99	7	0,5906	21,29

Este experimento resulta más fiable al no ser con localhost y con un servidor alojado en otro dispositivo (hay más latencia). Al observar las mediciones para una window = 100, y aumentar el timeout, podemos comprobar que al ancho de banda aumentó experimentalmente con cada incremento del timeout.

Por otra parte, para un timeout fijo, podemos observar que al ir aumentando la window el ancho de banda disminuyó al pasar de una window de 10 a una de 100, y posteriormente fue en aumentó para las Windows 100 -> 500, y 500->900. En particular es interesante ver que en ambos timeouts (0.1 y 0.3), los paquetes retransmitidos disminuyen al cambiar la window de 500 a 900, pero aumentan al cambiar la window de 100 a 500 (formando un codo en esta zona si se graficaran).

Además, como observación general, siempre se alcanzó una ventana máxima menor en tan solo un valor/dígito respecto a la ventana establecida al ejecutar el comando, excepto para la window = 900.

- **Un archivo de 5000 KB**

$time python3 client_sr2_bw.py **1000 timeout window** files_anakena/big_file.bin outputs_anakena/OUT_5000 anakena.dcc.uchile.cl 1818

Observación: Este experimento lo hice luego de responder las preguntas y notar que, para archivos grandes, el protocolo se comportaba mejor.

Size (bytes)	Timeout	Window	Tiempo real (s)	Sent packets	Retrans packets	% error	Total packets	Max window	Mín Cong	Máx RTT	Ancho de banda (B/s)
1.000	0.1	100	4,67	5137	21	0,40	5158	99	50	0,2191	1.070
1.000	0.3	100	22,58	5137	496	8,80	5633	99	16	0,5971	221,43
1.000	0.8	100	28,26	5137	306	5,62	5443	99	40	0,6016	176,92

Preguntas

1. Compare los resultados medidos con la T1 y la T5. Discuta las diferencias encontradas

Para una comparación más clara, se utilizarán los valores obtenidos en el servidor anakena con un archivo de 500KB.

- T1

Size (bytes)	Tiempo real (s)	Ancho de banda
1.000	1,245	401,60

- T5 (timeout 0.1 y window 100)

Size (bytes)	Timeout	Window	Tiempo real (s)	Sent packets	Retrans packets	% error	Total packets	Max window	Rtt est	Ancho de banda (B/s)
1.000	0.1	100	0,81	515	6	1,15	521	99	0,0763	617,28

Para la tarea 1 el ancho de banda de un paquete de 500 KB con size 1.000 fue de 401,60 B/s, mientras que en la T6 el máximo alcanzado fue de 36,04 B/s, dando un resultado peor. Sin embargo, aspectos importantes que se diferencian entre ambas tareas no son solo el ancho de banda, sino que ahora hay un protocolo implementado para no perder paquetes y por ende datos/información como si ocurría en la T1.

Respecto a la tarea 5, con un timeout de 0.1 y una window de 100 (el máximo para esa tarea), se obtuvo un ancho de banda de 617,28 B/s, el cual es muy superior al obtenido para la T6 (36,04 B/s).

Una de las principales diferencias encontradas entre ambas tareas (5 y 6) fue el como se comportaba el ancho de banda y la cantidad de paquetes retransmitidos. En la T6 los valores máximos de ancho de banda fueron para Windows de 10 y 900, mientras que en la T5 fue para una de 100. Además, en la T5 apenas se retransmitieron paquetes, sin embargo, en la T6 fue muy frecuente que se retransmitieran varios paquetes.

Un punto muy importante que descubrí después es que el protocolo implementado en la T6 se comporta mejor con archivos grandes, para un archivo de **500 KB** en anakena se obtuvo:

Size (bytes)	Timeout	Window	Tiempo real (s)	Sent packets	Retrans packets	% error	Total packets	Max window	Mín Cong	Máx RTT	Ancho de banda (B/s)
1.000	0.1	100	53,54	515	1.503	74,47	2.018	99	1	0,6447	9,28

Mientras que para un archivo de **5000 KB** con el mismo protocolo y en anakena, se obtuvo:

Size (bytes)	Timeout	Window	Tiempo real (s)	Sent packets	Retrans packets	% error	Total packets	Max window	Mín Cong	Máx RTT	Ancho de banda (B/s)
1.000	0.1	100	4,67	5137	21	0,40	5158	99	50	0,2191	1.070

Una diferencia clara es que, aunque el tamaño se aumento en 10 veces, el tiempo real disminuyo también cerca de 10 veces, los paquetes retransmitidos disminuyeron notablemente, así como el % de error, alcanzando un ancho de banda 100 veces superior al comparar ambos archivos con el protocolo de la T6.

De forma específica, tenemos que una diferencia entre la T5 y T6 es que las nuevas implementaciones provocan una ralentización ante archivos pequeños, y por tanto el protocolo es más efectivo con archivos grandes, mientras que para la T5 el protocolo funcionaba bien con archivos pequeños.

2. ¿Hay algún escenario en que esta implementación pueda comportarse peor que la T5?

Tanto el timeout adaptativo como la ventana de congestión adaptativa permiten reducir retransmisiones innecesarias ya que si ocurre una pérdida de reduce a la mitad la ventana de congestión. Esto es algo positivo, sin embargo, añade retraso al tiempo total en que se demora en enviar los paquetes ya que, si el timeout adaptativo se mantiene grande por la inflación, entonces disminuirá el ancho de banda, lo mismo cuando se reduce la ventana de congestión (menos cantidad de paquetes).

En particular, el escenario en que esta implementación puede comportarse peor que la T5 es cuando los archivos son muy pequeños, ya que el peso/tamaño del header con toda la información influirá más y por tanto ralentizará como sucedió al experimentar con un archivo de 500KB y uno de 5000KB en anakena.

Comparación con la T5

Al comparar ambas implementaciones, la T5 mostró ser un buen protocolo (mejor que aquellos de tareas anteriores) para los archivos probados, los cuales fueron generalmente de un tamaño pequeño. Sin embargo, en la T6 se pudo observar que para archivos pequeños se comportaba peor que la tarea 5, pero que en contraste para archivos más grandes el rendimiento del protocolo era notablemente mejor.