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

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

Preparación

Comando de uso general

- $ ./client_gbn_bw.py size timeout win 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

Ejecuciones y Resultados

Localhost

- **Un archivo de 1000 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_gbn_bw.py 1000 0.1 wincow files_localhost/big_file.bin outputs_localhost/OUT_1000 127.0.0.1 1818

- Limpiando cache

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	10	0,32	1029	428	29,37	1457	9	0,0008	3.125
1.000	0.1	100	0,45	1029	4662	81,91	5691	99	0,0003	2.222
1.000	0.1	500	0,45	1029	30787	96,76	31816	499	0,0005	2.222
1.000	0.1	900	0,26	1029	10695	91,22	11724	899	0,0024	3.846
1.000	0.3	10	0,48	1029	603	36,94	1632	9	0,0007	2.083
1.000	0.3	100	0,31	1029	1380	57,28	2409	99	0,0028	3.225
1.000	0.3	500	0,34	1029	3599	77,76	4628	499	0,0012	2.941
1.000	0.3	900	0,30	1029	6990	87,16	8019	899	0,0021	3.333
1.000	0.8	100	0,30	1029	1467	58,77	2496	99	0,0011	3.333

Al comparar los timeouts para una misma window = 100 podemos observar que al aumentar el timeout, el bando de ancha aumenta también.

Además, al mantener fijo el timeout e incrementar el tamaño de la window, se observan intervalos intermitentes, la razón principal atribuible es que se esta realizando en localhost y el tiempo real varia en torno a los decimales, y por tanto pequeños retrasos en la red producen estas alteraciones al experimentar.

Anakena

- **Un archivo de 1000 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_gbn_bw.py **1000 timeout window** files_anakena/big_file.bin outputs_anakena/OUT_5000 anakena.dcc.uchile.cl 1818

- Limpiando cache

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	10	6,04	1029	111	9,73	1140	9	0,035	165,56
1.000	0.1	100	1,34	1029	0	0	1029	99	0,142	746,26
1.000	0.1	500	2,48	1029	5259	83,63	6288	499	0,462	403,22
1.000	0.1	900	2,20	1029	5942	85,23	6971	899	0,515	454,54
1.000	0.3	10	5,28	1029	43	4,01	1072	9	0,048	189,39
1.000	0.3	100	1,44	1029	0	0	1029	99	0,137	694,44
1.000	0.3	500	1,73	1029	198	16,13	1227	499	0,332	578,03
1.000	0.3	900	2,30	1029	2204	68,17	3233	899	0,07	434,78
1.000	0.8	100	2,67	1029	0	0	1029	99	0,122	374,53

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 disminuyó 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 aumentó y alcanzó su máximo (para estas mediciones en particular) en una ventana de tamaño 100. Los paquetes retransmitidos también fueron mínimos para una ventana de 100.

Si se graficaran estos datos; eje y ancho de banda/cantidad de paquetes retransmitidos, eje x tamaño de la ventana, observaríamos que se forma un "codo" en la ventana de tamaño 100, lo que nos indicaría que en torno a dicho valor se encuentra el óptimo.

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.

Preguntas

1. Verifique que una ventana de tamaño 1 funcione y demore lo mismo que Stop-and-Wait.

Para Stop and Wait con size 1000 y un archivo de 500KB en 52.67.31.19 con el código del profesor se obtuvo lo siguiente:

Size (bytes)	Tiempo real (s)	Sent packets	Lost packets	Ancho de banda	Tamaño final (KB)
1.000	49,78	514	107	10,04	500

En la tarea 4, utilizando una ventana 1, con timeout = 0.1 (como en la T3), y un mismo archivo de 500KB en 52.67.31.19 a size = 1000 se obtuvo:

Size (bytes)	Timeout	Window	Tiempo real (s)	Sent packets	Retrans packets	% retrans	Total packets	Max window	Rtt est	Ancho de banda (B/s)
1.000	0.1	1	50,01	515	107	20,77	622	0	0,089	9,99

En la tarea 5, utilizando una ventana 1, con timeout = 0.1 (como en la T3), y un mismo archivo de 500KB en 52.67.31.19 a size = 1000 se obtuvo:

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	1	52,08	515	3	0,57	518	0	0,100	9,60

Los resultados fueron bastante similares como para considerar que con una ventana de tamaño 1 demoran lo mismo. Como observación interesante, y concordante con lo propuesto para este protocolo, efectivamente se retransmitieron menos paquetes.

2. ¿Cuál es el tamaño máximo de ventana teórico para Selective Repeat ahora? ¿De verdad falla si usamos un tamaño mayor?

Teóricamente Selective Repeat indica que el tamaño de la ventana cumpla la condición:

$$WINDOW\ SIZE\ \leq \frac{N}{2} = \frac{1000}{2} = 500 \qquad N = [0, 999] = 1000\ valores$$

Por tanto, el valor máximo teórico para una ventana en Selective Repeat es de 500.

El protocolo se probó para ventanas de 500 y funcionó correctamente, sin embargo, para ventanas de tamaño 900 efectivamente dio errores, se pudieron obtener las estadísticas igualmente, pero hubo fallos por el tamaño de ventana que se usaba a causa del fallo por exceder el valor teórico para el protocolo.

Comparación con la T4

Al comparar la T1 con la T3 pudimos observar que el protocolo Stop and Wait suele ser mucho más lento, disminuyendo considerablemente para sizes más grandes debido a la retransmisión de paquetes perdidos. Para el caso de la T4 podemos ver que esto se repite, siendo más lenta que la T1 (donde no se hace nada respecto a retransmitir paquetes), y al compararla con la T3 podemos observar que al aumentar el tamaño de la ventana el ancho de banda supera con creces las mediciones para Stop and Wait.

En particular al comparar el protocolo implementado en la T5 con respecto a la T4, podemos observar que para la T4 mientras que se incrementaba el tamaño de la ventana, el ancho de banda siempre aumentaba, mientras que en la T5 con tamaños de ventana mayores a 100 el ancho de banda disminuye.

Mi hipótesis es que, si bien ahora el tamaño de la ventana del receptor es mayor, cuando se ejecuta el programa con ventanas muy grandes (500 o 900) el receptor de todas formas se satura un poco más y ahí entra en juego el timeout variable implementado, provocando que este aumente y consigo el ancho de banda disminuya.