Tarea 2

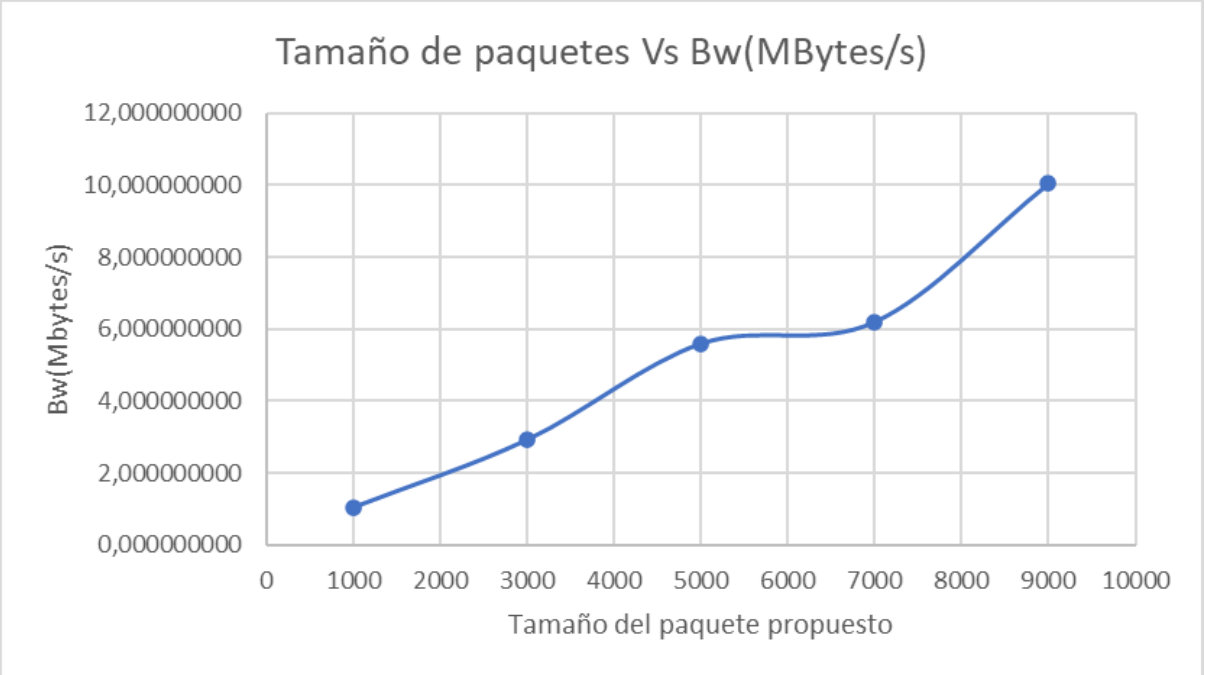
Nombre: Oscar Eduardo Salazar Figueroa
Rut: 20.889.666-0

Pregunta 1:Genere algunos experimentos con diversos tamaños de paquete, timeout y pérdidas y haga una recomendación de valores a utilizar para las distintas pérdidas. Grafique sus resultados.

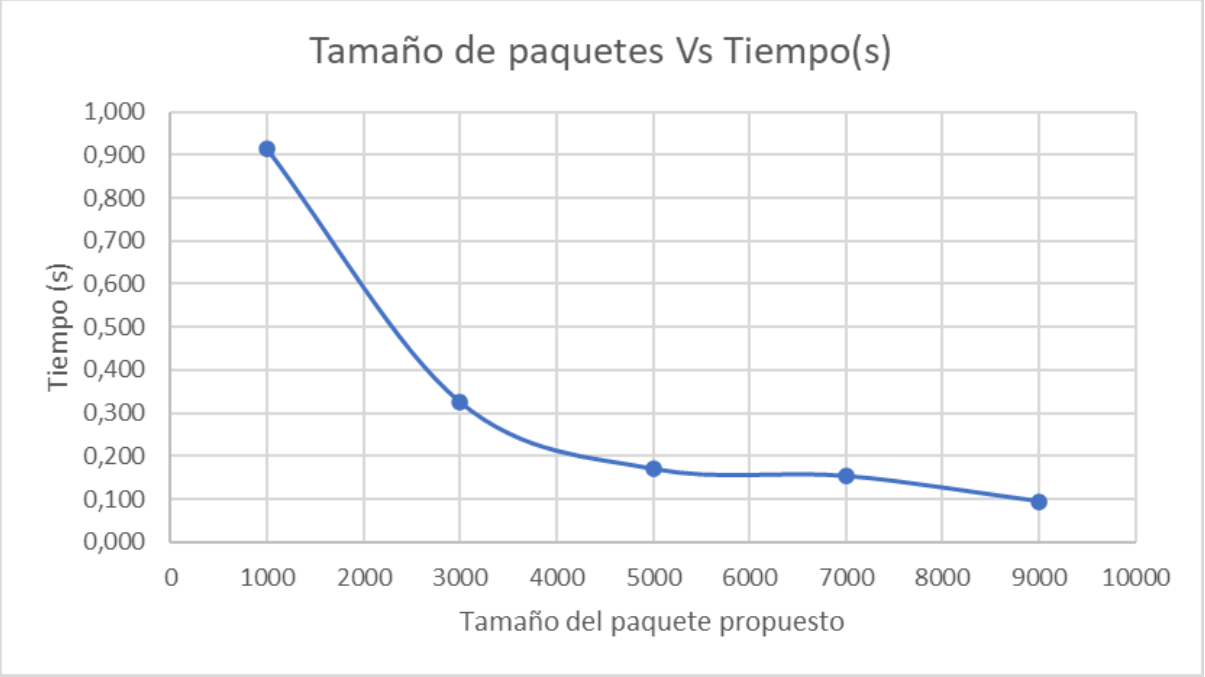
Se realizaron 15 tests para el experimento. Todos los tests de realizaron con un tamaño total de 1000000 de bytes. En primer lugar se varió el tamaño del paquete de 1000 a 9000 bytes manteniendo el Timeout en 25ms y el Loss rate en 15%. A continuación se muestran los datos generados junto a los gráficos de comparación entre tamaño de paquete vs Tiempo(s)/Bw(MBytes/s)/Errores.

Notar que hacemos diferencia en errores de descarte y errores de desorden, pero en los gráficos se usa los errores totales

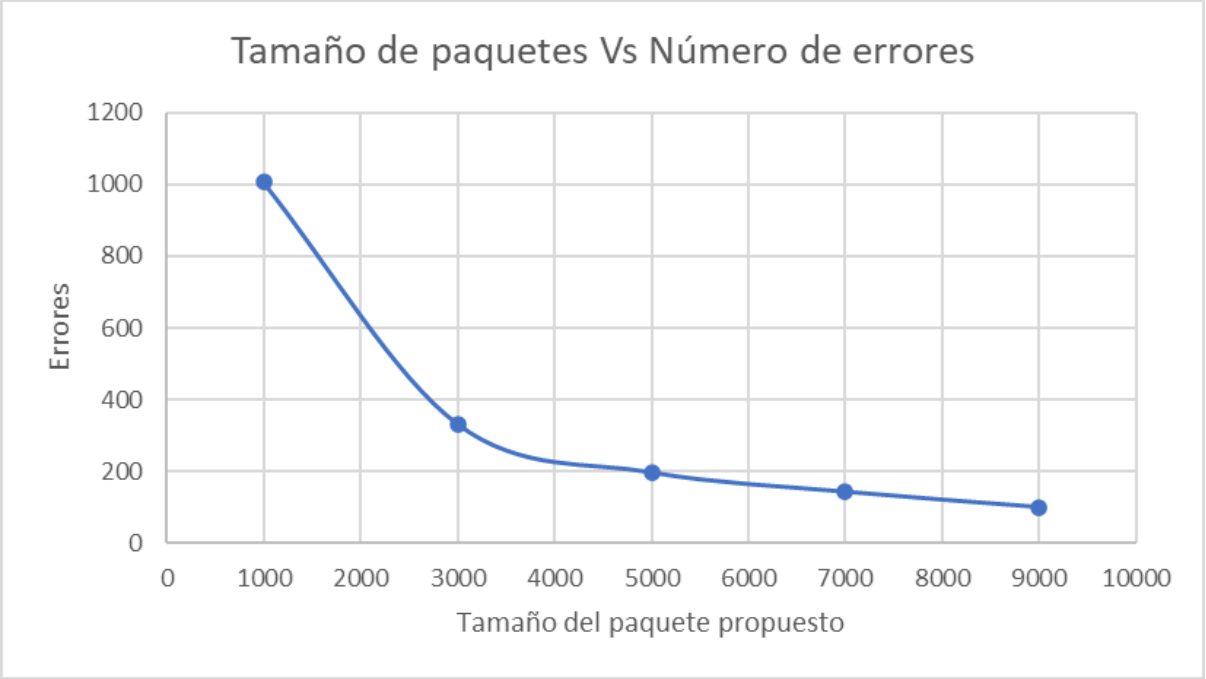
Tamaño de paquete	Time(s)	Bw(MByets/s)	errores totales	Errores desorden	Errores descarte
1000	0,915	1,042494690	1006	907	99
3000	0,326	2,926231175	331	299	32
5000	0,170	5,599300087	196	180	16
7000	0,154	6,203906910	142	129	13
9000	0,095	10,049165542	99	96	3



Se puede ver que mientras mayor es el tamaño del paquete propuesto, mayor es el ancho de banda (Bw). Esto concuerda con lo que sabemos dado a que a mayor tamaño de paquete menor será la cantidad de paquetes totales para completar la cantidad de tamaño total del archivo pedido al servidor. Notar que de 5000 a 7000 la diferencia no fue mucha, pero se puede deber a una coincidencia y la tendencia dice que al aumentar el tamaño del paquete, aumenta también el ancho de banda.



Se puede ver que a mayor tamaño de paquetes menor será el tiempo total. Esto concuerda con lo que sabemos y con el gráfico anterior debido a que estamos recibiendo la misma cantidad de bytes pero en paquetes más grandes lo que deriva en menor cantidad de paquetes totales y menor tiempo de ejecución. Se puede apreciar que desde el tamaño de paquete de 5000 bytes empieza a disminuir menos, esto debido a que empieza a ser limitante el hardware o el propio medio a la hora de mandar y recibir paquetes.

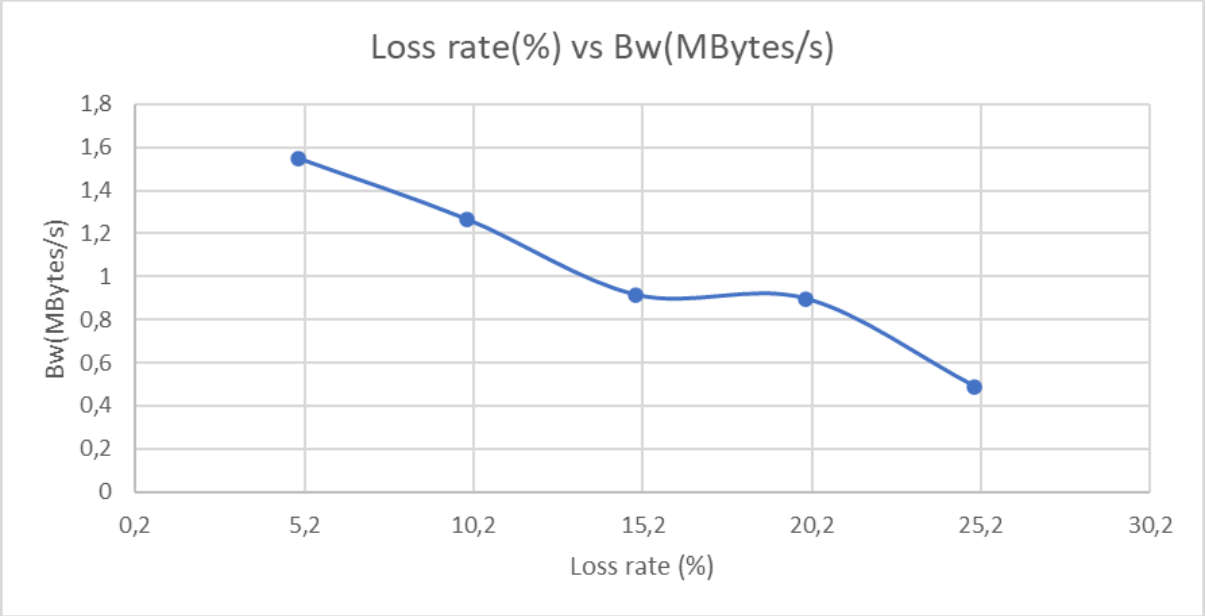


Se puede ver una disminución de la cantidad de errores con respecto al aumento del tamaño de los paquetes. Esto se debe a que al aumentar el tamaño de paquetes disminuye la cantidad total de paquetes a recibir, por lo que hay menos paquetes que tienen probabilidad de perderse lo que deriva en una menor cantidad de posibles errores.

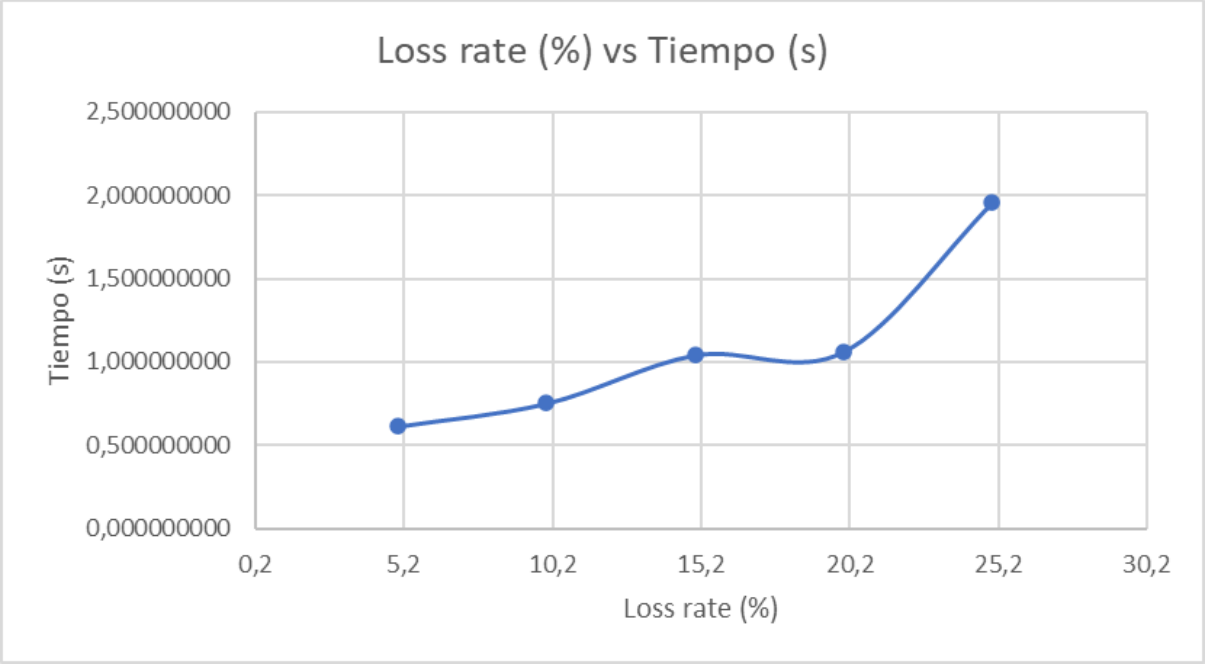
Por lo anterior podemos decir que es mejor tener un tamaño de paquete mayor para así aumentar el ancho de banda y disminuir tanto el tiempo de ejecución como la cantidad de errores. Dicho lo anterior, sabemos que para paquetes mucho más grandes es más probable de que se pierdan o que se corrompan en el camino en una red convencional, por lo que esto tiene una limitante que se debe medir en cada caso.

En segundo lugar se hicieron experimentos variando el Loss rate y fijando el tamaño del paquete con el Timeout. Se varió el Loss rate desde 5% a 25% y se fijó el tamaño de paquete en 9000 bytes y el Timeout en 25ms. A continuación se mostrarán los resultados junto con sus gráficos.

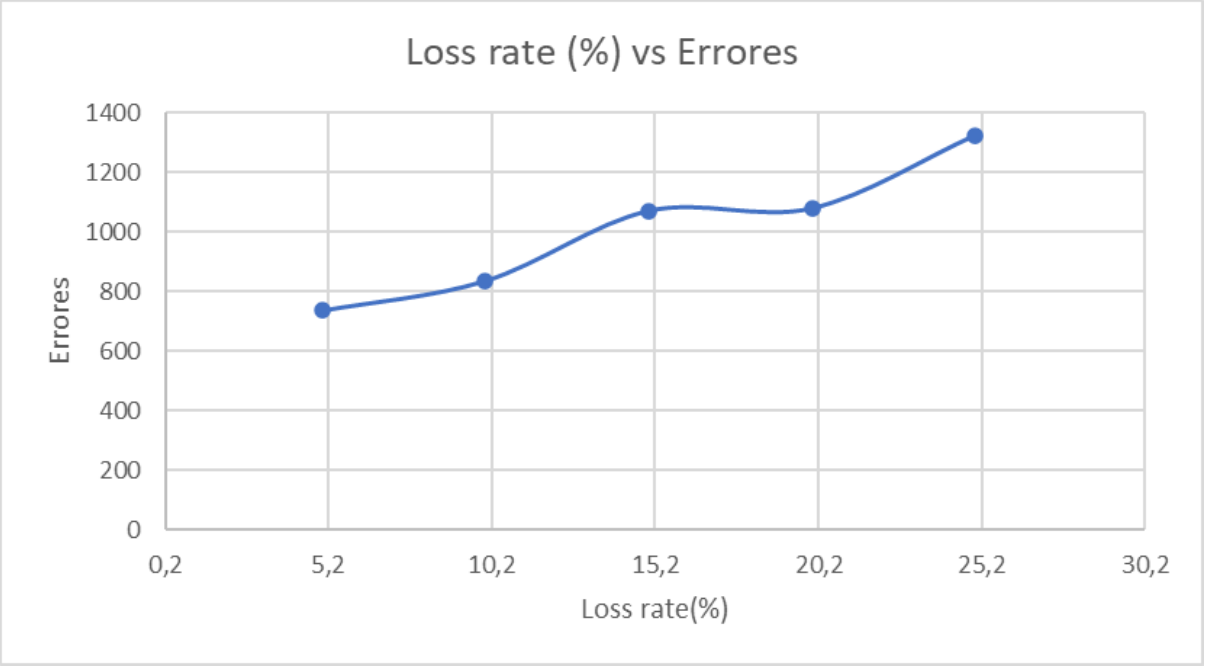
Loss Rate	Time(s)	Bw(MByets/s)	Errores totales	Errores desorden	Errores descarte
5	0,615470648	1,549504139	736	693	43
10	0,753979921	1,264853731	836	768	68
15	1,043170929	0,914207145	1071	941	130
20	1,063254595	0,896938815	1080	949	131
25	1,954926252	0,487831352	1325	1124	201



Podemos ver que a mayor Loss rate menor es el ancho de banda. Esto coincide con lo que sabemos ya que a mayor probabilidad de error mayor es la cantidad de errores lo que ralentiza el tráfico de datos lo que se traduce en un menor ancho de banda.



Podemos ver que a mayor Loss Rate mayor es el tiempo de ejecución total. Esto se debe a la misma razón explicada en el gráfico anterior ya que al haber más errores el programa debe volver a pedir un paquete lo que se traduce en un tiempo mayor de ejecución.

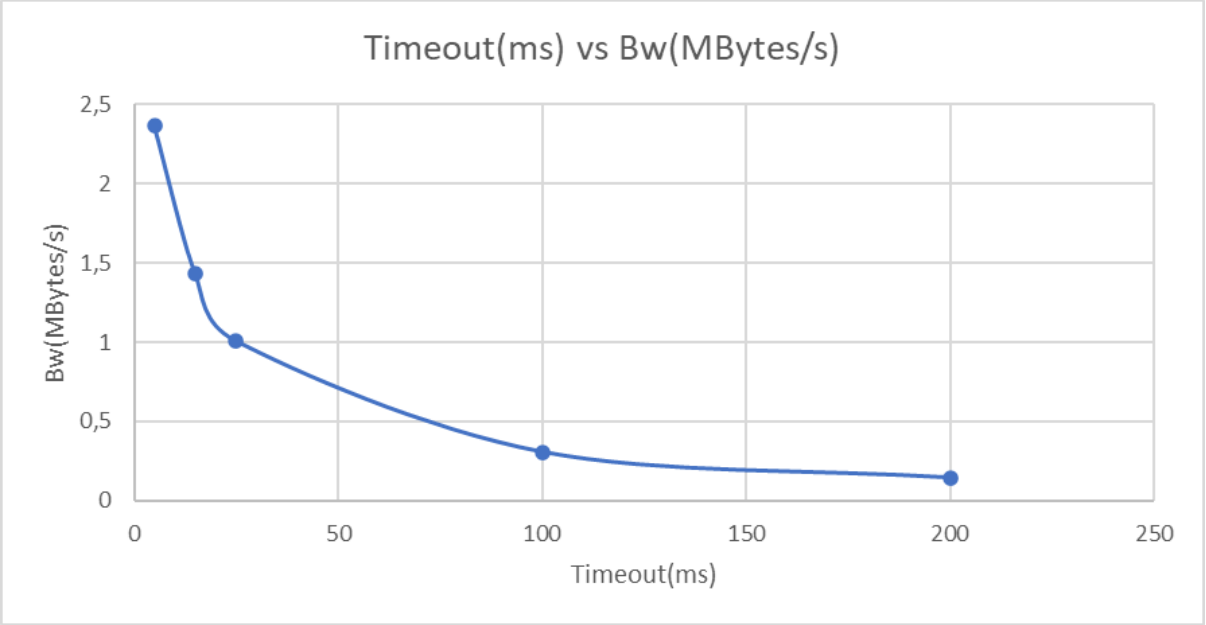


Se puede ver a mayor Loss rate mayor cantidad de errores. Esto se tiene por la definición de Loss Rate que se traduce en una mayor probabilidad de que un error ocurra, lo que se traduce en mayor cantidad de errores.

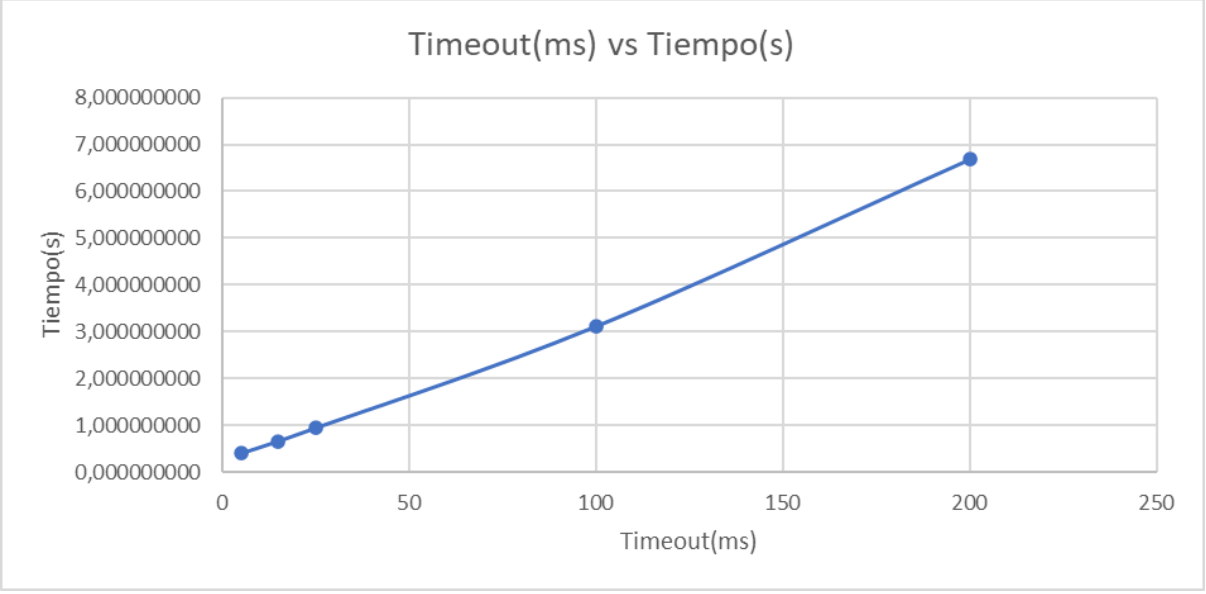
Podemos inferir de los datos anteriores que la queremos disminuir el Loss rate lo más posible, para así aumentar el ancho de banda y minimizar los errores. Esto en la práctica no depende enteramente del cliente si no de la calidad de la conexión y otros factores que no podemos controlar como en este experimento, pero es importante ver que el objetivo final sería el minimizar la probabilidad de errores total.

En último lugar se hicieron experimentos variando el Timeout y fijando el tamaño del paquete con el Loss Rate. Se varió el Timeout desde 5ms a 200 ms y se fijó el tamaño de paquete en 9000bytes y el Loss Rate en 15%. A continuación se mostrarán los resultados junto con sus gráficos

Timeout(ms)	Time(s)	Bw(MByets/s)	Errores totales	Errores desorden	Errores descarte
5	0,403040886	2,363835985	2803	1056	1747
15	0,665458202	1,433109267	1061	939	122
25	0,947478533	1,006539234	1032	932	100
100	3,115257740	0,306130149	1026	924	102
200	6,678687572	0,142793671	1022	922	100

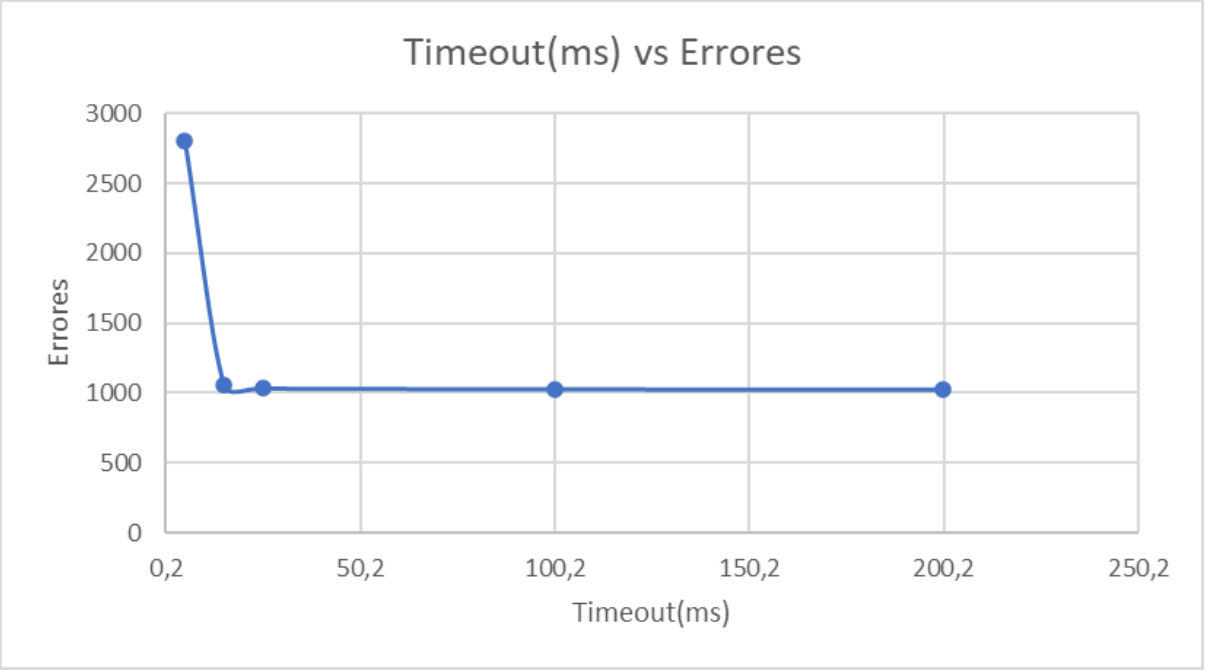


Podemos ver que al aumentar el Timeout disminuye el ancho de banda. Sin embargo, se tuvo varios problemas para timeouts pequeños, en concreto para timeout de 5ms ya que el servidor muchas veces no era capaz de mandar la información al ritmo que requería le timeout. El comportamiento se debe en particular ya que al aumentar el timeout estamos esperando más tiempo a que el servidor responda lo que se traduce en una ralentización del programa y por tanto una diminución del ancho de banda.



Podemos ver que el tiempo de ejecución aumenta con respecto al aumento del Timeout. Lo anterior tiene relación con la explicación del gráfico visto ya que al

aumentar el tiempo de espera a la respuesta del servidor se aumenta el tiempo total de ejecución del programa.



Podemos ver que la cantidad de errores es alta solamente para Timeouts pequeños mientras que para el segundo timeout en adelante se tiene prácticamente la misma cantidad de errores. Esto se debe a que el Timeout de 5ms hace que el servidor en muchas ocasiones no alcance enviar el paquete y se tome como error hasta que pueda enviar en el tiempo pedido el paquete o bien encontrarse con el paquete anterior que había pensado que no había llegado por timeout y tomarlo como válido, pero aumentando de igual manera el número de errores. Si aumentamos un poco el timeout el servidor alcanza en la mayoría de los casos a mandar el paquete y ahora se contabiliza en su mayoría la pérdida de paquetes.

Por lo anterior se debe de tener cuidado con el timeout, en general se busca el mínimo timeout pero hay un límite para enviar paquetes del servidor el cual hace que no necesariamente el menor timeout es el mejor. Se debe de intentar minimizar el timeout sin llegar al punto crítico donde el servidor no alcanza a mandar el paquete.

En conclusión se debe tener un tamaño de paquete considerable (que el servidor pueda soportar), un Loss rate lo más bajo posible y un Timeout lo suficientemente bajo pero que no impida que el servidor alcance a mandar los paquetes.

Podemos hacer una comparación con la tarea 1 y es que, si bien tenemos los mismos comportamientos en los gráficos, en su mayoría (si no en todos los casos)

se tiene un mayor ancho de banda, gracias al algoritmo que implementamos en esta ocasión. Esto debido a que stop&wait va recibiendo y leyendo 1 sólo paquete a la vez, mientras que selective repeat tiene la capacidad de leer y guardar varios paquetes a la ves (según el tamaño de la ventana) y escribir varios paquetes a la vez (dependiendo de cuantos paquetes se tenga en la ventana seguidos del paquete 0) por lo que este algoritmo mejora enormemente la eficiencia y el ancho de banda.

Pregunta 2. Explique porqué el cliente de la tarea 1 igual funciona, a pesar que el servidor envía implementando selective-repeat y con una ventana de tamaño 50.

Funciona debido a que el cliente de la tarea 1 recibe y escribe sólo los paquetes que espera en orden, lo demás lo descarta. El servidor recibirá un ACK acumulativo (empieza por A) cada vez que se equivoque, luego en algún momento el servidor le mandará el correcto, el cliente avanza una posición y esperará el siguiente paquete. Si no hay desorden ni pérdida de paquetes el cliente recibirá siempre el paquete que está esperando y se irá moviendo conforme vayan llegando, por lo que los demás espacios de la ventana nunca se ocuparán. Si, por el contrario, hay desorden o errores, el cliente sólo escribirá el que está esperando y los demás serán descartados, notificando al servidor cual fue el último que se recibió correctamente. Podrá ser menos eficiente pero en algún punto terminará por recibir todos los paquetes hasta el final.

Pregunta 3: Calcule si el tamaño 50 de ventana está bien para la conexión que Ud tiene. Debería ser más? menos? por qué?

Tenemos que el tamaño de la ventana debe ser proporcional a la cantidad de paquetes que nos llegan por segundo y al RTT. Usando ping al servidor de anakena nos entrega un RTT de aproximadamente 7ms. Teniendo un aproximado de 2458 paquetes por segundo tenemos que el BDP es aproximadamente 17.21, lo que redondeando sería una ventana de 17.

Pregunta 4: La ventana de recepción del cliente es igual a del enviador. ¿ Serviría de algo que la ventana del cliente fuera más grande?

Sorpresivamente si sirve, ya que en el caso de que los ACK de, por ejemplo, los primeros 10 elementos de la ventana se hayan perdido en la conexión, la ventana de recepción se habrá movido pero la del servidor no. Lo anterior implica a que si la ventana del servidor es más grande, puede seguir enviando paquetes más grandes

mientras arregla el problema con los ACK anteriores, ya que estos paquetes pueden ser recibidos por el cliente pues ya se había movido. Esto ayuda a que la conexión siga su curso mientras se arreglan errores anteriores.