Universidad de Costa Rica

Sede Rodrigo Facio
Facultad de Ingeniería
Escuela de Ciencias de la Computación e Informática

Redes de computadoras

Reporte

Tarea programada 1

Adrián Lara

Danilo Acosta Villalobos

B30044

Erick Guillén Torelli

B30044

Montes de Oca, 2016

1. Resumen

A continuación se presenta un trabajo de investigación acerca del protocolo de capa de enlace *Selective Repeat*, dicho trabajo fue enfocado a la comprensión de la implementación básica del algoritmo. Se implementó una simulación del mismo en Java y se realizaron experimentos con la implementación, con tal de estudiar su funcionamiento y el impacto de las variables a las cuales está sujeto.

2. Introducción

El mundo está conectado a través de Internet, es decir, a través de redes. La información viaja de un lado a otro en pocos segundos y el usuario no se preocupa por cómo hacer que la información sepa para dónde va, qué hacer si se pierde algo en el camino o cómo encriptarla por seguridad. De todo lo anterior se encargan los algoritmos que poseen las computadoras.

Todos los "cómo" quedan delegados a las computadoras y los programadores. Estos se encargan de direccionar la información, verificar que llegue a su destino, comprimirla para reducir tiempos de envío, encriptarla para evitar robos de datos y otras tareas necesarias. Sin embargo, el usuario no percibe todo lo que sucede y, gracias a esto, el uso de las redes se ve simplificado y usuarios sin ningún conocimiento técnico las pueden utilizar.

3. Marco Teórico

Muchas de las redes de computadoras se componen de un cliente, un servidor y uno o varios intermediarios. El cliente es el encargado de codificar la información, enviarla al intermediario y verificar que haya sido recibido por el servidor. Las tareas del intermediario radican en transmitir los datos entre el servidor y el cliente. Por el lado del servidor, este debe recibir los datos, ordenarlos de manera adecuada y enviar una

respuesta al cliente para informarle que los recibió correctamente.

La codificación realizada por el cliente debe proveer suficiente información para saber la dirección del destinatario, el orden de los datos y la información del emisor, entre otros. Además el intermediario y el cliente deben ser capaces de interpretar la codificación de los datos que reciben.

El mecanismo de envío de información debe ser capaz de retransmitir los datos que no hayan llegado a su destino y puede ser implementado de muchas formas, una de esas formas es llamada *selective repeat*. El algoritmo de *selective repeat* utiliza una ventana de un tamaño predefinido. Los datos que se encuentren en la ventana son enviados y se espera la confirmación del primer dato enviado antes mover la ventana. Además, se tiene un tiempo mientras se espera la confirmación de los datos enviados y se revisa la ventana continuamente para que, en caso de que no se haya recibido la confirmación en cierto tiempo, se reenvién los datos.

Por otra parte, la implementación de los paquetes difiere en cada algoritmo, pero en el caso de *selective repeat* se utiliza un contador de un tamaño mínimo del doble del tamaño de la ventana del algoritmo. Dicho contador permite saber el orden de los paquetes y acomodarlos de acuerdo al mensaje original.

4. Descripción del programa implementado

El programa consta de cinco clases:

Servidor Esta clase se encarga principalmente de recibir los paquetes provenientes del Cliente a través del Intermediario, enviar los mensajes de respuesta para cada paquete recibido (ACK) y, finalmente, ordenar los paquetes y guardar lo recibido en un archivo nuevo.

Cliente La clase cliente es la que emite los mensajes. Esta clase lee el archivo que será enviado, envía la información en paquetes y se encarga de manejar el protocolo de reenvío de paquetes extraviados (*Selective Repeat*) y mover la ventana conforme

se obtienen las confirmaciones del Servidor.

Intermediario Dicha clase realiza la tarea de recibir mensajes desde el Cliente (mensajes de datos) y desde el Servidor (ACK), y retransmitirlos al destino correspondiente. Adicionalmente en este punto se simula la pérdida de paquetes en el canal con la probabilidad P, determinada por el usuario.

Paquete Encapsula la estructura básica de un paquete, en este caso utlizaremos paquetes de la forma (*#sec:character*) donde "secçorresponde al número del paquete y çharacter.ªl contenido del mismo, cada paquete con datos solo va a transmitir un caracter la vez. Adicionalmente esta clase define los métodos necesarios para manipularlos.

Util Encapsula los métodos que transmiten o esperan paquetes en la red dentro de las clases principales del programa.

El flujo básico del programa consiste en iniciar la ejecucción de las clases Servidor, Intermediario y Cliente, en ese orden. De esta forma el Cliente comienza leyendo el archivo y enviándolo en grupos de paquetes del tamaño de la ventana definida por el usuario.

Durante esta transmisión el Intermediario recibe los paquetes que, con una probabilidad $1 - P$, serán renviados al Servidor, el cual envía un mensaje de respuesta al Cliente (a través del Intermediario) indicando que el paquete X fue recibido con éxito.

Si algún paquete se pierde, el protocolo utilizado reenviará el paquete perdido después de que se cumpla un *timeout*. Este *timeout* indica que se ha esperado el mensaje de confirmación por mucho tiempo y que se debe reenviar el paquete.

Una vez que todos los paquetes han sido recibidos en el Servidor y se obtienen todos los mensajes de confirmación en el Cliente, este envía un paquete con una bandera con el valor -1 lo cual indica que el archivo se ha completado y se puede guardar el mensaje recibido por el servidor en disco.

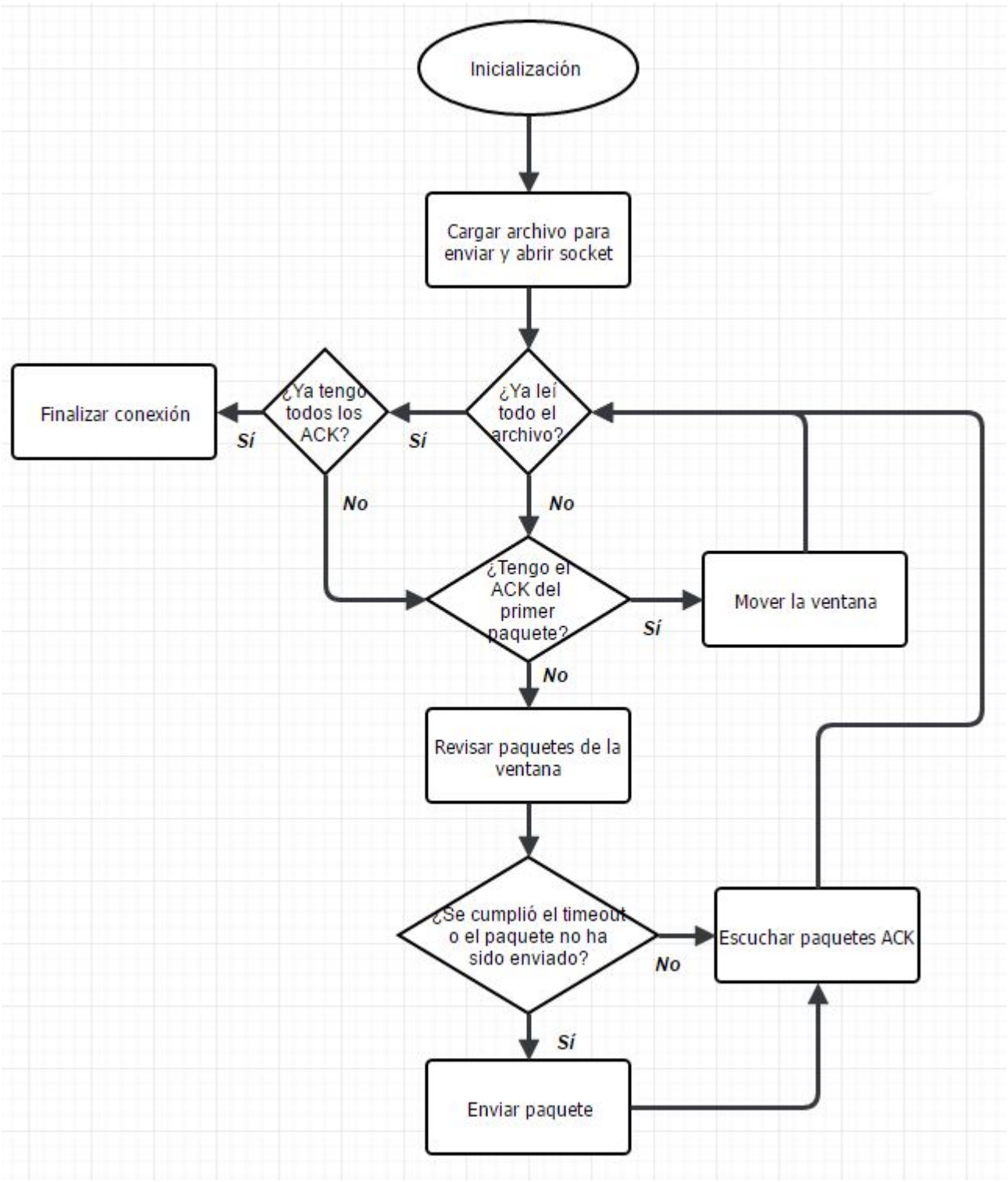


Figura 1: Flujo de trabajo del cliente de la red.

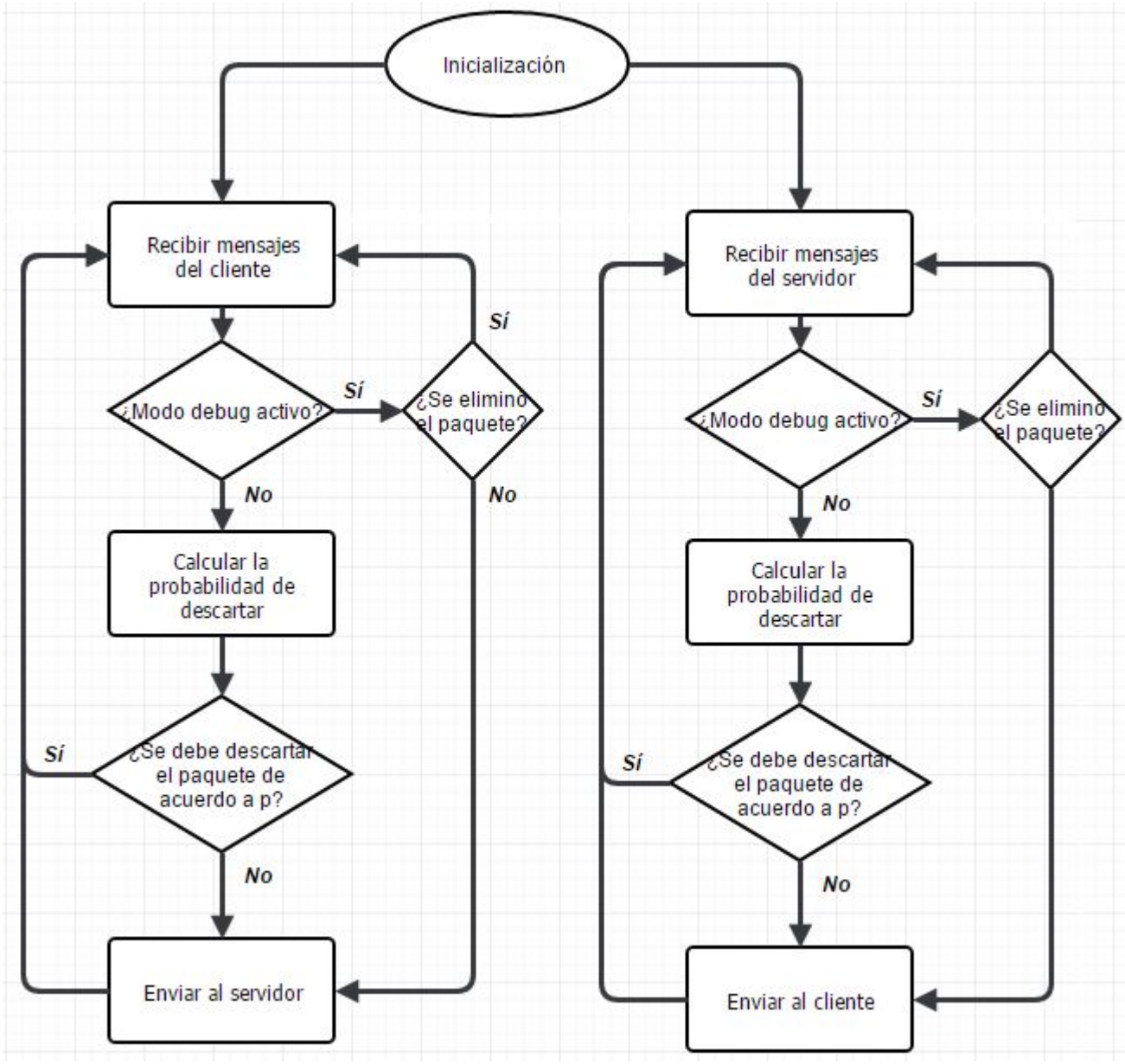


Figura 2: Flujo de trabajo del intermediario de la red.

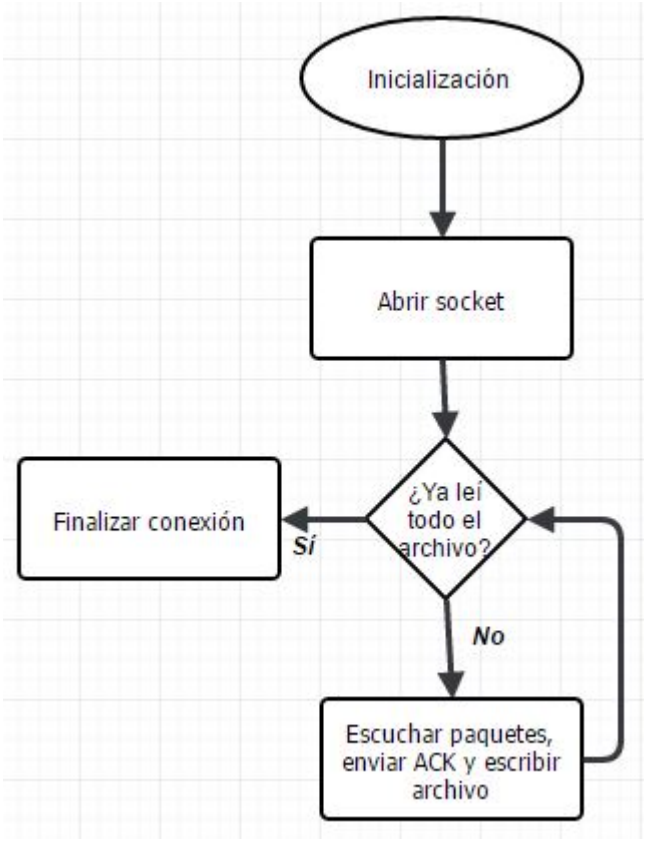


Figura 3: Flujo de trabajo del servidor de la red.

5. Descripción de los experimentos

Para un análisis del algoritmo y el impacto de las variables, se van a ejecutar tres tipos de pruebas que miden los distintos impactos:

1. Impacto de la probabilidad de pérdida de paquetes sobre el tiempo de envío: Se graficarán los tiempos tomando $p = 10, 20, 30, 40, 50, 60, 70, 80, 90$, un *timeout* de 1000ms y un tamaño de ventana de 50 sobre un archivo de tamaño de 100 caracteres, 1050 caracteres y 2000 caracteres.

2. Impacto de la probabilidad de pérdida de paquetes sobre el porcentaje de paquetes enviados y reenviados: Se graficarán los porcentajes de paquetes enviados y reenviados más de 1 vez tomando $p = 10, 20, 30, 40, 50, 60, 70, 80, 90$, un *timeout* de 2000ms y un tamaño de ventana de 50 sobre sobre un archivo de tamaño de 100 caracteres, 1050 caracteres y 2000 caracteres.

3. Impacto de la distintos tamaños de ventana sobre el tiempo de envío: Se graficarán los tiempos tomando $p = 10$ y un *timeout* de 2000ms sobre un archivo de tamaño de 100 caracteres y una ventana con tamaño 10, 25 y 50; 1050 caracteres y una ventana con tamaño 105, 260 y 1025, y 2000 caracteres con una ventana de tamaño 200, 500 y 1000.

4. Impacto de la distintos tiempos de *timeout* sobre la cantidad de paquetes enviados y reenviados: Se graficarán los tiempos tomando $p = 10$ sobre un archivo de tamaño de 100 caracteres y una ventana con tamaño 10, 25 y 50; 1050 caracteres y una ventana con tamaño 105, 260 y 5025, y 2000 caracteres con una ventana de tamaño 200, 500 y 1000.

6. Resultados

Los experimentos planeados no fueron completados con los archivos de tamaño 1050 y 2000 caracteres debido a que como podemos ver con el archivo de 100 el orden crece exponencialmente conforme aumenta el porcentaje de perdida y los tiempos se volvían intratables.

Sin embargo, se logró analizar el impacto de la probabilidad de pérdida de paquetes sobre el tiempo de retraso en la figura 4, el impacto de la probabilidad de pérdida de paquetes sobre la cantidad de mensajes reenviados en la figura 5 y los retrasos al variar la ventana en la figura 6.

Figura 4: Resultados de los retrasos en un archivo de 100 caracteres y distintas probabilidades de pérdidas de paquetes.

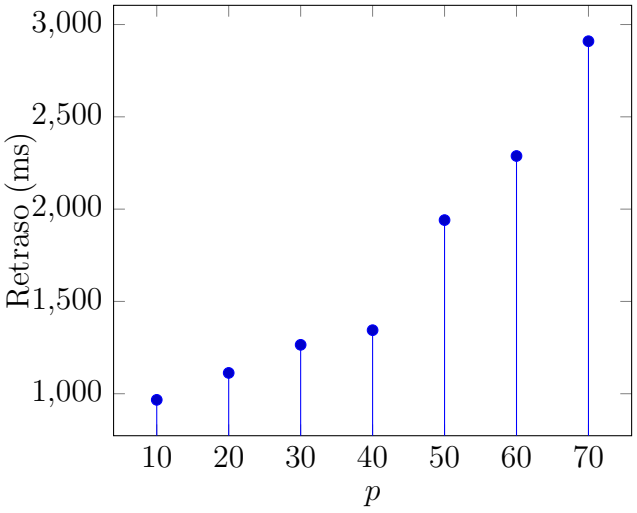


Figura 5: Resultados de los paquetes enviados y reenviados en un archivo de 100 caracteres y distintas probabilidades de pérdidas de paquetes.

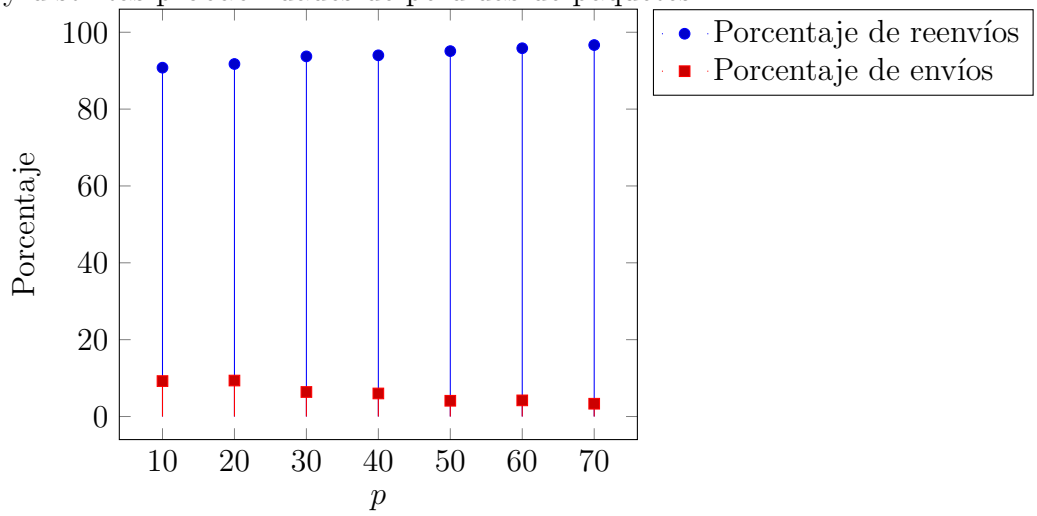
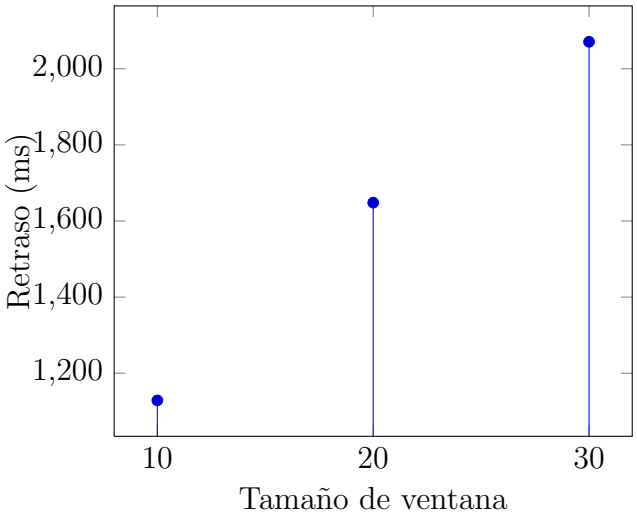


Figura 6: Resultados de los retrasos en un archivo de 100 caracteres y distintos tamaños de ventana.



7. Conclusiones

El protocolo de *Selective Repeat* es un protocolo complejo pero efectivo. Como se puede observar en los experimentos, conforme el porcentaje de pérdida de paquetes aumenta, el orden de duración crece de forma exponencial. De acuerdo a la anterior, podemos deducir que si se desea utilizar este protocolo, se debe asegurar que el porcentaje de paquetes perdidos no sea alto y que sea menor al 10(%). Todo esto se puede

observar de acuerdo a la figura 4.

Además, de acuerdo a la figura 5 podemos deducir que el porcentaje de pérdida afecta la cantidad de paquetes reenviados y si el *timeout* es muy grande, se puede impactar enormemente el tiempo de ejecución.

Y con respecto a la figura 6, podemos observar que un tamaño de ventana correspondiente al 10 % del archivo nos permite una menor duración en retrasos, ya que la ventana se puede revisar de forma constante y se puede mover al recibir 1 paquete de 10, contrario a esperar 1 paquete de 30.