Laboratorio 4: Red

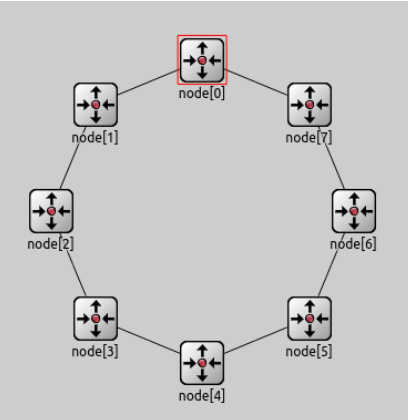
Grupo 22
Integrantes: Lazaro Manuel Cugat, Ezequiel Lauret, Santiago Morales

Abstract

En este laboratorio analizaremos una red con forma de anillo mediante la herramienta de simulación de eventos OMNET + +, con la que estudiaremos el comportamiento de ésta según un criterio de envío de paquetes dado por la cátedra (se envían hacia la “derecha”) y un algoritmo implementado por nosotros para mejorar la eficiencia y desempeño de la red, en el cual básicamente agregamos la opción de que, si conviene, se envíen por la “izquierda”.

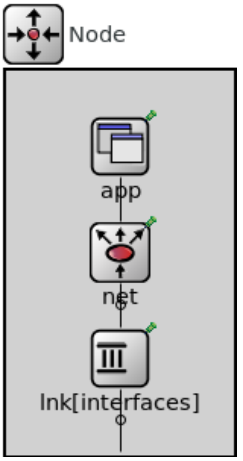
Introducción

El problema de este laboratorio se concentra en la capa de red, más específicamente es un problema de enrutar el tráfico que confluye en este módulo de múltiples entradas y salidas.



Estructura de la red

La estructura de nuestra red es de anillo, compuesta por 8 nodos, cada uno con conexiones hacia sus vecinos.
Cada nodo cuenta con dos capas de enlace (Lnk, una con cada vecino), una capa de red (net) y una capa de aplicación (app). Donde cada paquete generado tiene un nodo de destino que al arribar se envía a la capa de aplicación.



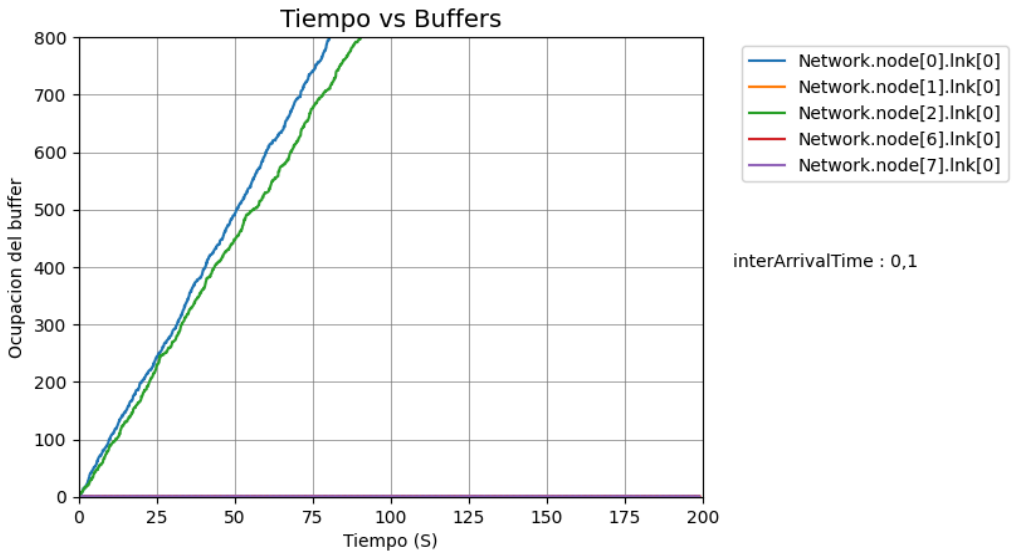
app se encarga de generar los paquetes y servir como destino final de estos, **Lnk[0]** y **Lnk[1]** sirven como canal de comunicación entre el nodo y sus vecinos, 0 es el canal en sentido horario y 1 antihorario. Y **net** es la parte crucial de este lab, se encarga de decidir que que hacer con el paquete cuando llegue al nodo, revisa si el paquete que llegó lo tiene como destino final, y en ese caso lo envía a app. Si no simplemente lo envía al siguiente nodo (primeramente éste lo manda a Lnk[o]).

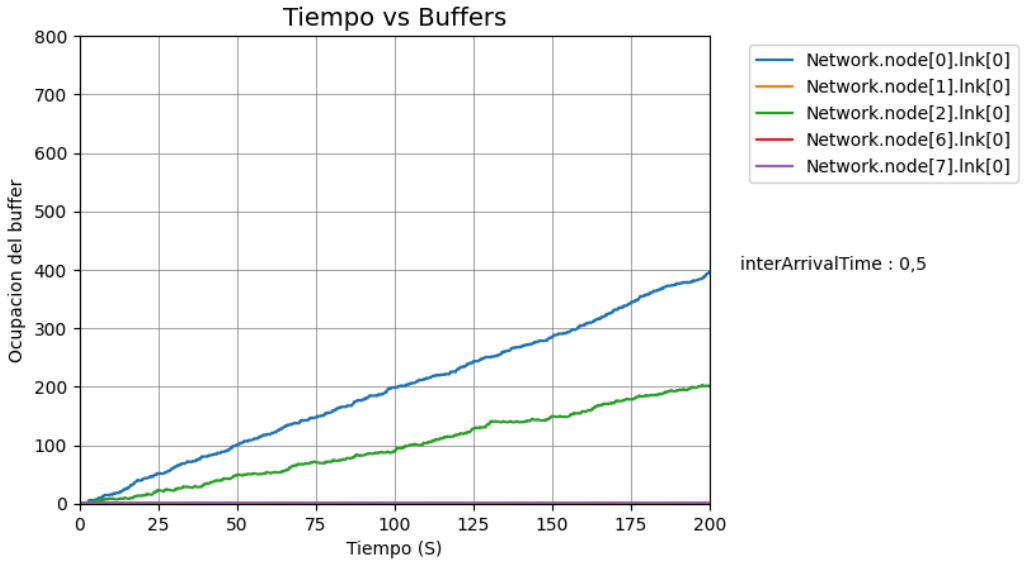
Experimentos

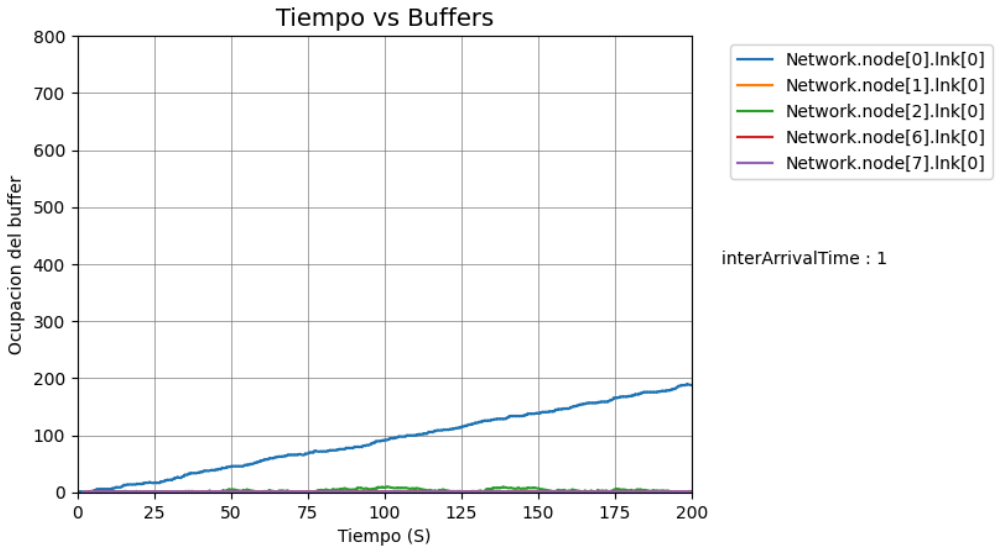
El algoritmo de la cátedra para el enrutamiento de tráfico simplemente envía los paquetes en dirección de sentido horario (a través de Lnk[0]). Al no tener la posibilidad de usar Lnk[1] se produce un retardo en la entrega de los paquetes y además, como todos los paquetes se envían en el mismo sentido, se crea un cuello de botella en los nodos ubicados a la derecha del nodo receptor, ya que todos los paquetes deben pasar por estos.
Para analizar los casos de estudio se utiliza una distribución exponencial y analizamos el comportamiento de la red para los siguientes valores: 0.1, 0.5 y 1.

Caso de estudio 1

- Nodos emisores: 0 y 2
- Nodo receptor: 5

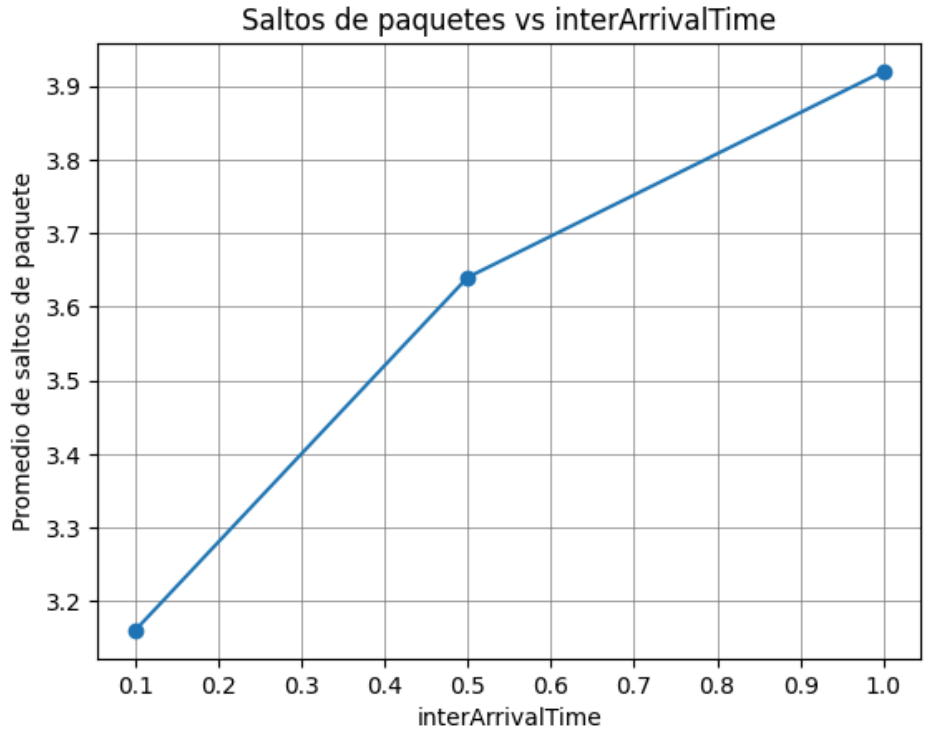






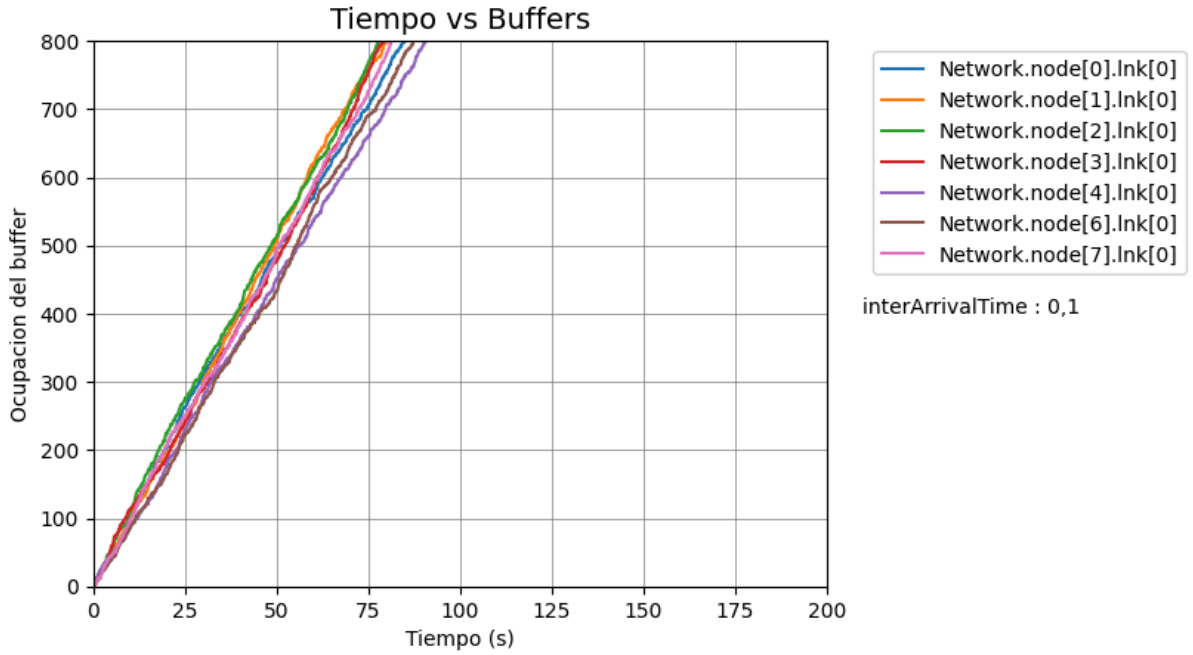
Podemos observar que la ocupación de los búferes disminuye según aumenta el intervalo de envío de paquetes, pues los eventos ocurren cada más tiempo lo que provoca que se envíen menos paquetes.

Además vemos que el buffer del nodo 0 está mucho más ocupado que el del resto, y esto sucede porque todos los paquetes viajan en sentido horario y el nodo 0 actúa como nodo intermedio para los paquetes enviados por el nodo 2, además de gestionar sus propios paquetes.

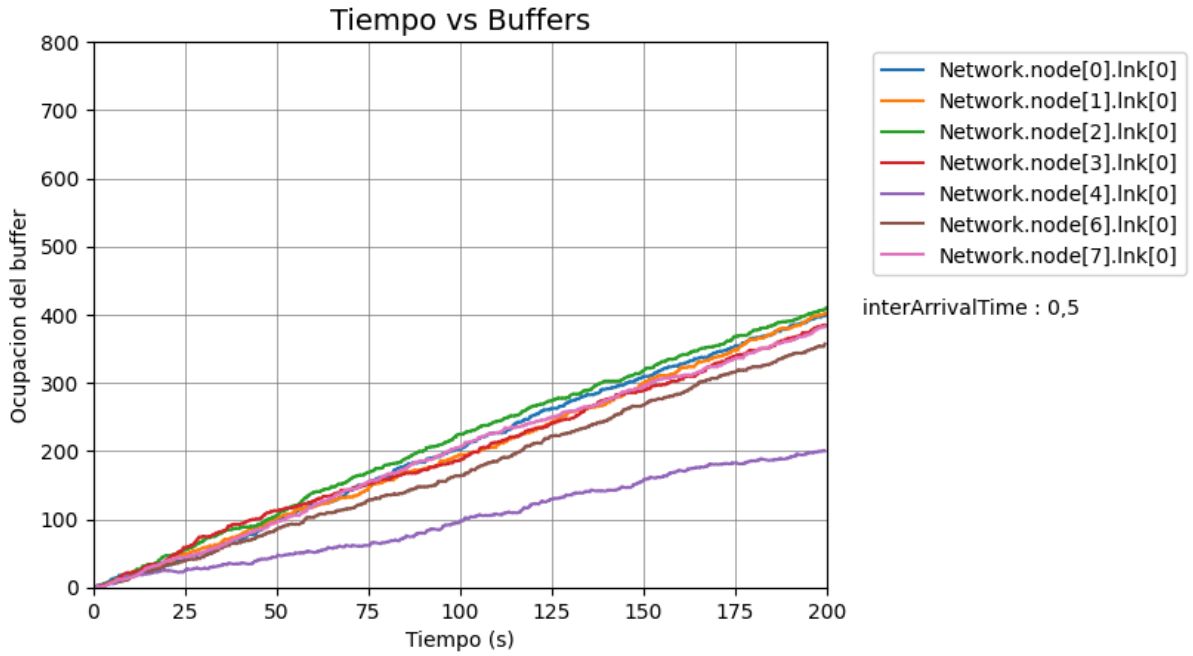


Caso de estudio 2

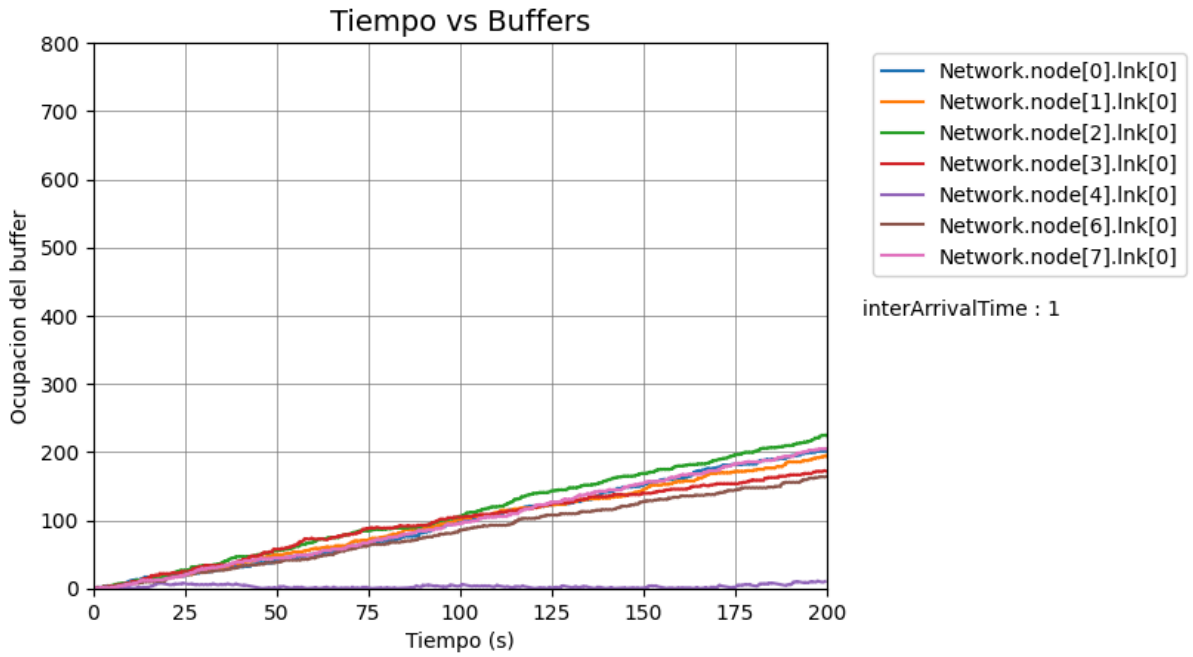
- Nodos emisores: Todos
- Nodo receptor: 5



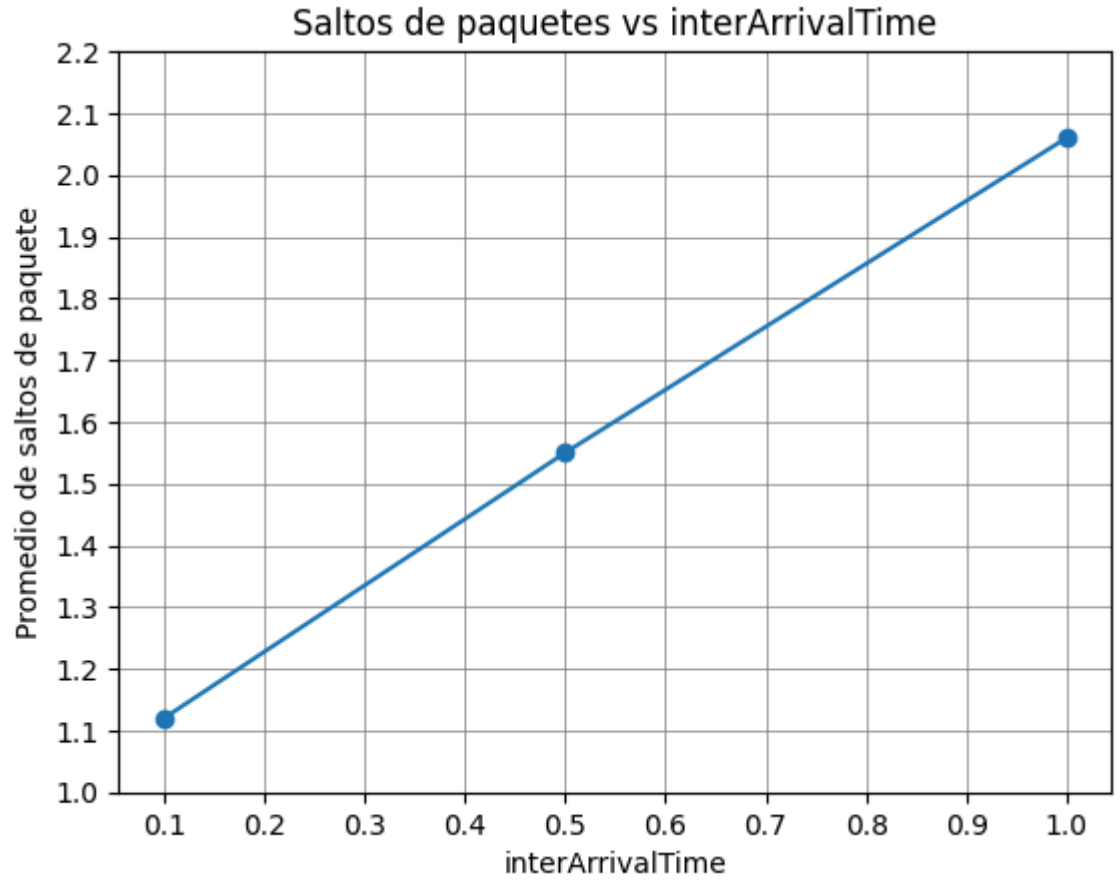
interArrivalTime : 0,1



interArrivalTime : 0,5



En este caso se ve cómo, al ser todos los nodos generadores, todos los buffers tienen tráfico, y notamos que el nodo 4 es el que menos ocupación de buffer tiene pues está a la izquierda del nodo receptor, entonces únicamente genera paquetes a diferencia del resto.



Métodos

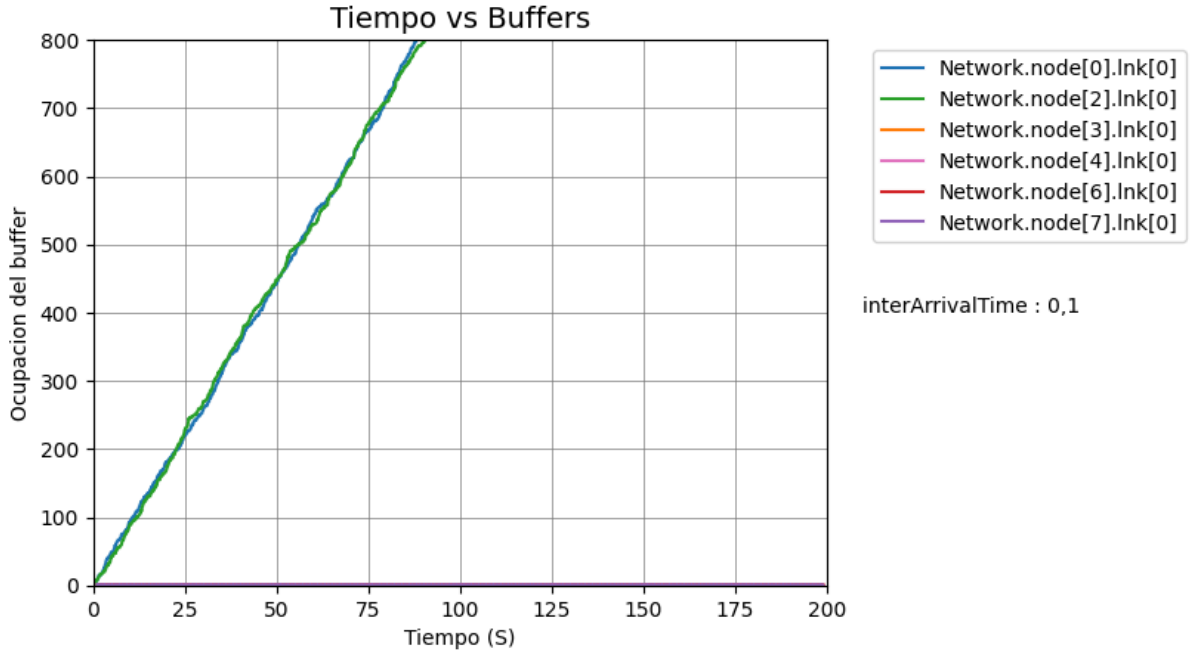
El algoritmo de enrutamiento que implementamos funciona de la siguiente manera:
La capa de red se va a encargar de diferenciar y manejar paquetes de información y paquetes de “investigación”:
Cada nodo emisor va a crear dos paquetes de investigación, uno que va a recorrer el anillo en sentido horario y otro en sentido antihorario. Entonces la idea es que cada paquete de una vuelta completa y guarde cuantos saltos o “hops” le llevó llegar hasta el nodo destino (receptor).
Luego la lógica del módulo de red se puede simplificar en:

- Analizar la información recibida de los paquetes de investigación, decidir qué camino es más corto (cuál tiene menos hops a destino), y guardar esto como el criterio que usará este nodo para el envío de paquetes de información.

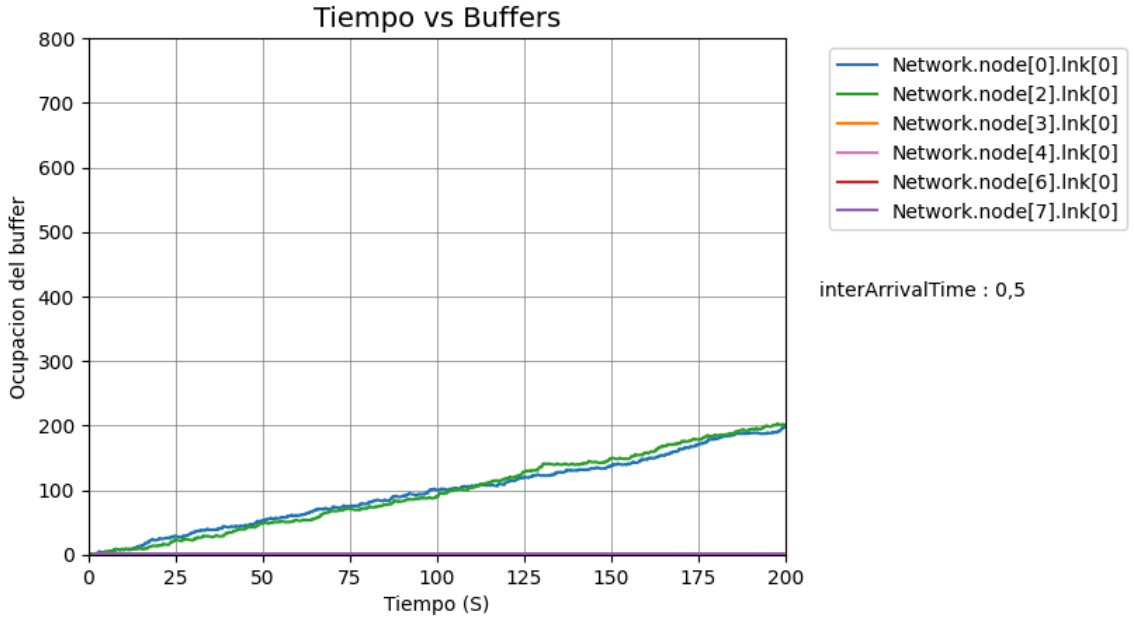
De esta forma si el nodo receptor está a la “izquierda” de un nodo emisor no tiene que dar una vuelta completa al anillo, sino que el nodo va a saber que conviene enviarlo por el otro canal.

Resultado

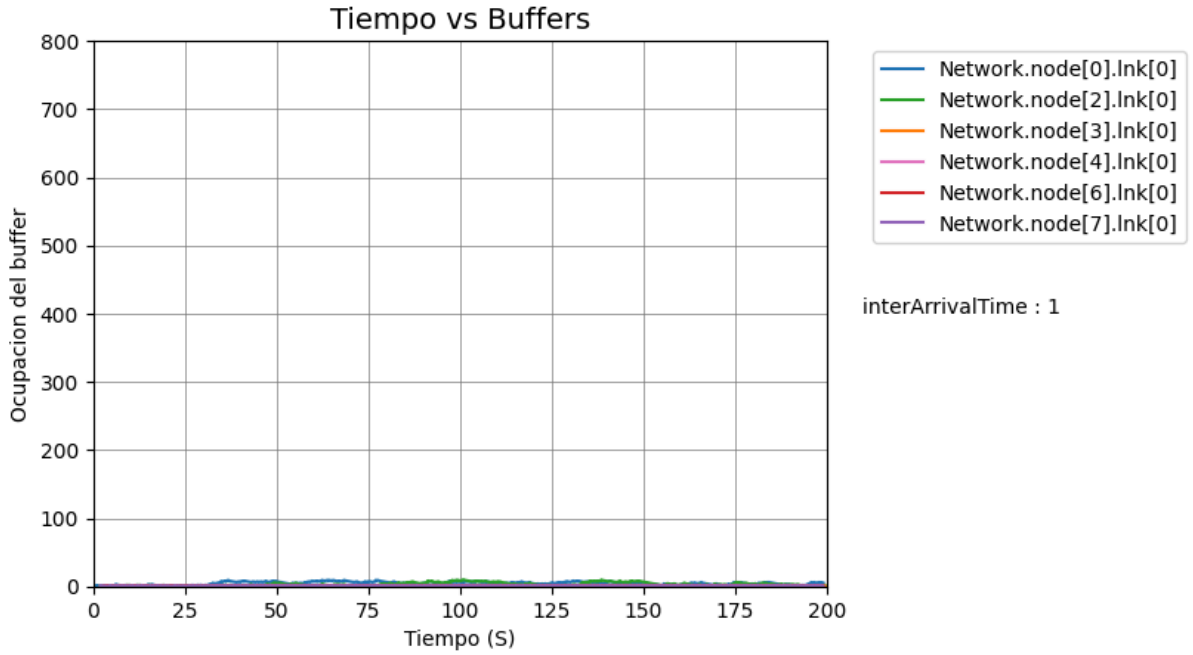
Caso de estudio 1



interArrivalTime : 0,1

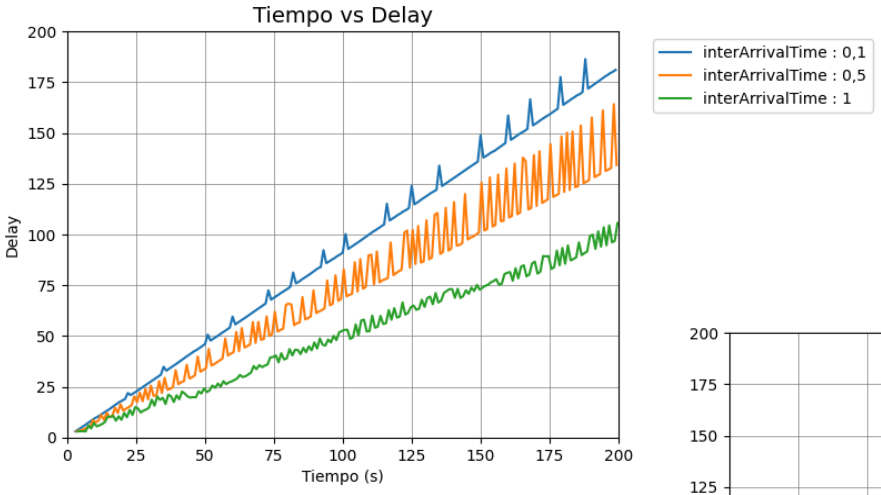


interArrivalTime : 0,5

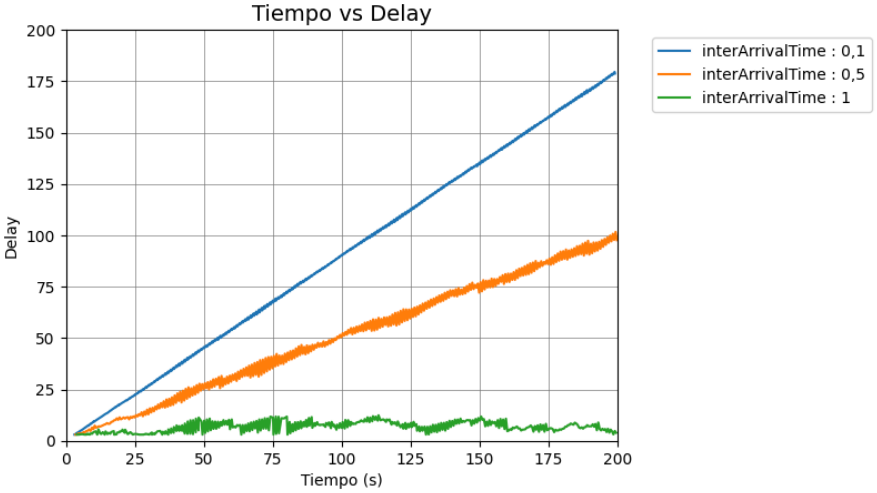


Ahora podemos notar que en todas las variaciones del intervalo, nuestro algoritmo distribuye mucho más equitativamente la carga de tráfico del nodo 0 y 2 , ya que el nodo 2 ahora no envía los paquetes en dirección al nodo 0. Lo que se puede observar en cómo las líneas del gráfico se superponen.

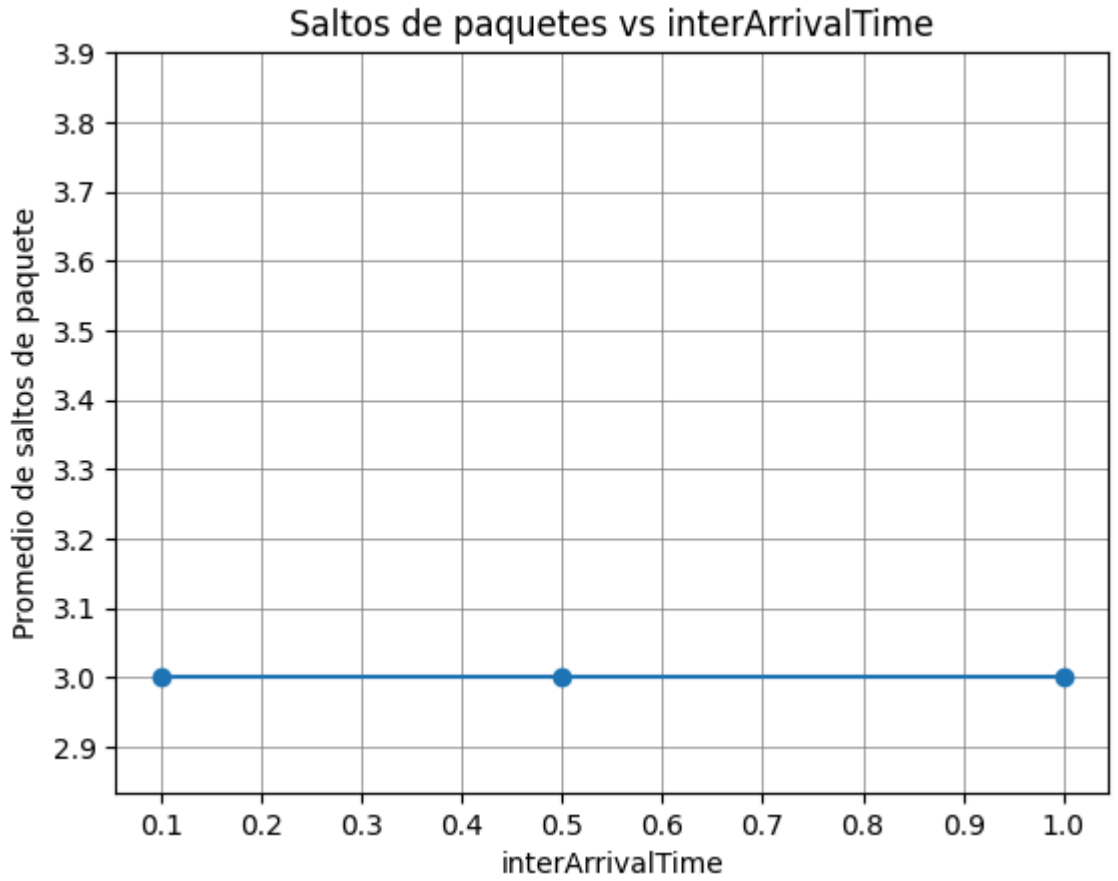
Sin algoritmo:



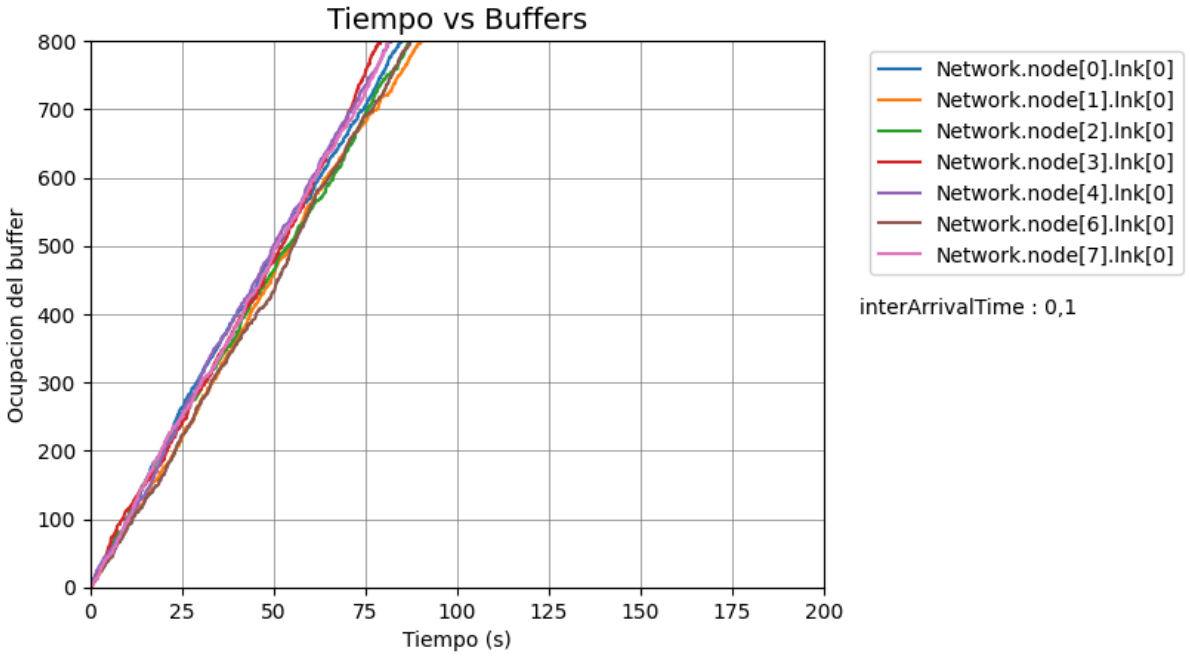
Algoritmo implementado:

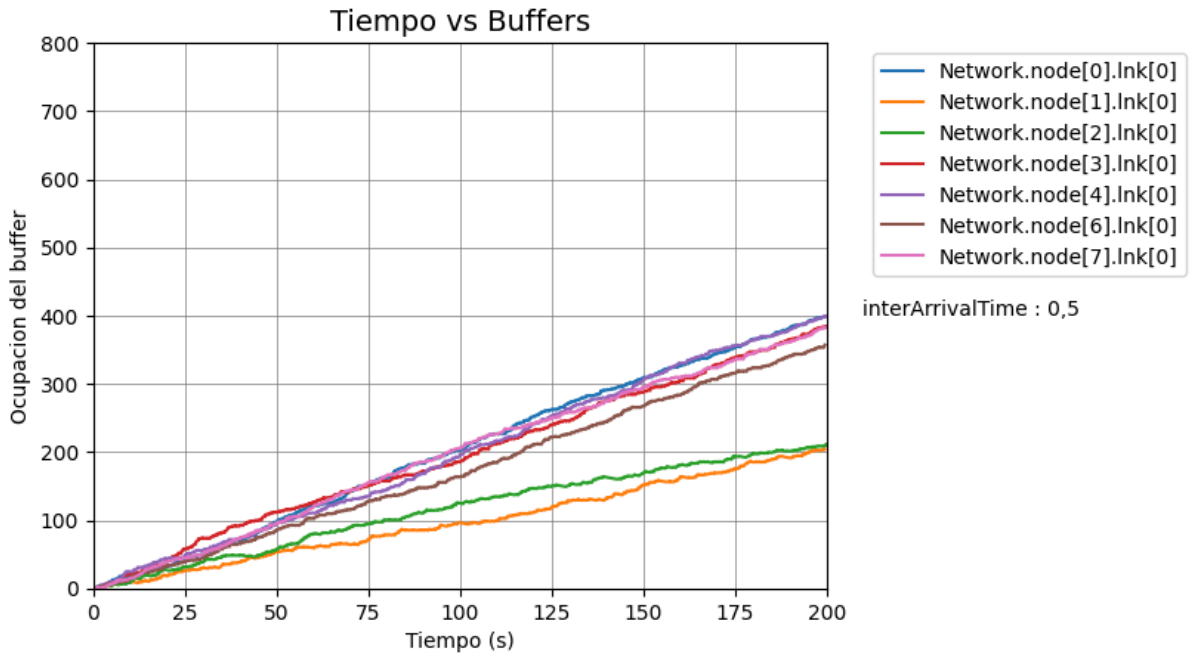


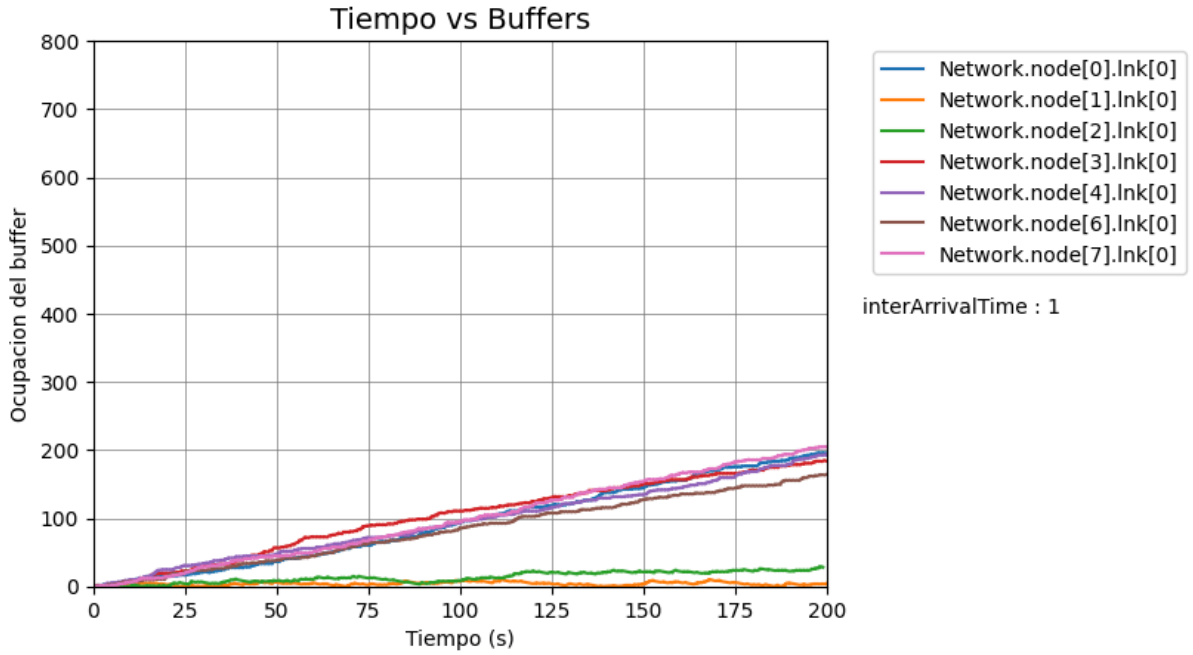
Podemos ver como el delay baja sustancialmente con el algoritmo implementado, lo que sucede gracias al mejor criterio de envío de paquetes.



Caso de estudio 2

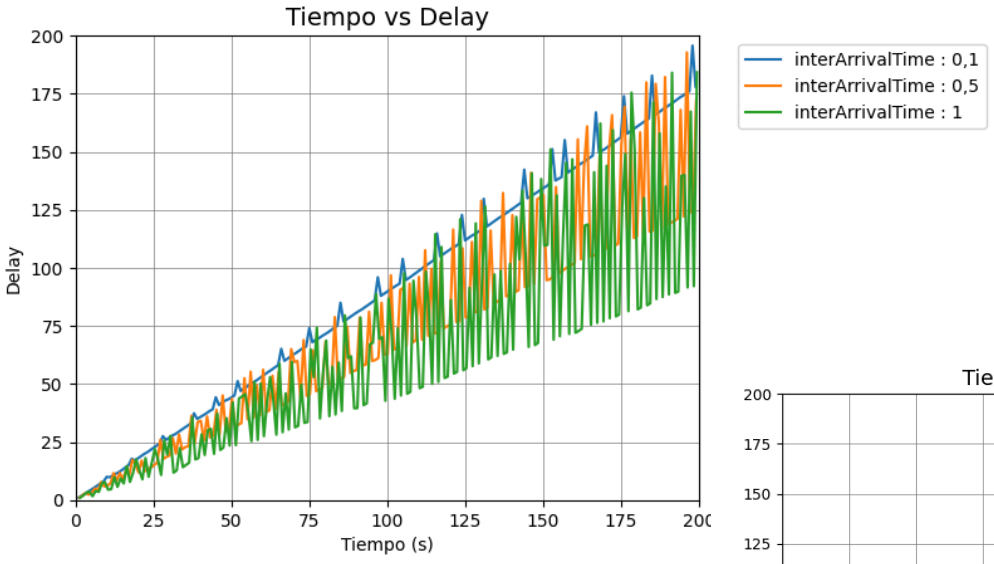




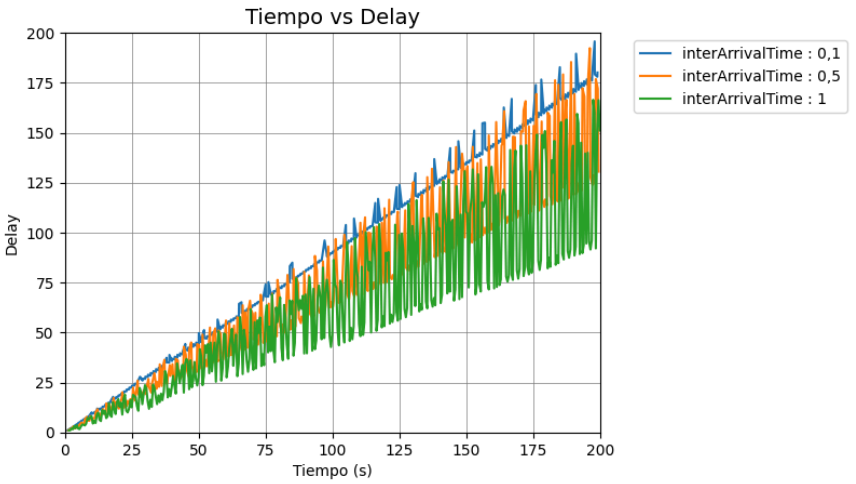


Aquí podemos notar que la ocupación del buffer de los nodos 1 y 2 es más bajo que el resto, lo que ocurre por cómo está implementado nuestro algoritmo, que para el nodo 0 al tomarle la misma cantidad de saltos llegar al nodo 5 por sentido horario y antihorario elige mandarlo en sentido horario. Entonces estos tienen menos carga que el resto.

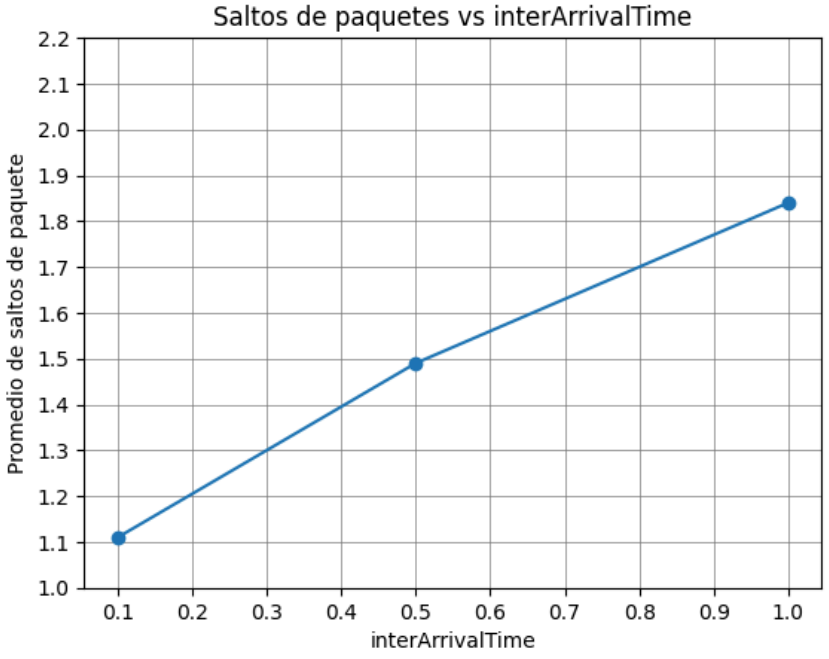
Sin Algoritmo



Algoritmo implementado



Vemos como no hay cambios significativos y mucha relación entre el delay y la ocupación de los buffers, creemos que esto tiene que ver con la gran cantidad de paquetes que se encuentran en la red.



Discusión

En conclusión podemos ver que gracias a la implementación del algoritmo baja el tiempo de entrega de paquetes en ambos casos.

En el primer caso, donde dos nodos envían paquetes a un mismo destino, se observa que el algoritmo modificado consigue reducir tanto el retraso como el número promedio de saltos necesarios para entregar los paquetes, lo que se debe a que el tráfico se distribuye de manera más equitativa entre los nodos.

En el segundo caso, donde todos los nodos envían paquetes al mismo destino, el algoritmo implementado logra una ligera disminución en el retraso promedio del envío de paquetes y en la cantidad promedio de saltos en comparación con el algoritmo original, aunque la mejora no es muy notable.

Cabe destacar que el algoritmo es mejorable, se podría implementar de tal forma que funcione para redes más abstractas y con más interfaces, y no únicamente en forma de anillo. Por ejemplo modificar la lógica y la generación de los paquetes corredores para que se adapten a otras formas de red pero mantengan la tarea de elegir el mejor camino.