UNICEN
Universidad Nacional del Centro
de la Provincia de Buenos Aires

TRABAJO FINAL

Simulador online de red Omega y Baseline

Arquitectura de computadoras y técnicas digitales.

Integrantes: Guerrero, Adrián Pablo.
Oses, Cristian.

INTRODUCCIÓN

El presente trabajo simula el funcionamiento de una red de interconexión bloqueante bidireccional con switches crossbar de 2x2 y conexiones entre etapas compatibles con redes Omega y Baseline. El mismo, permite configurar los siguientes parámetros:

- Número de procesadores.
- Procesadores Activos.
- Periodicidad de los requerimientos.
- Configuración de dirección de memoria.
- Tamaño del buffer de cada switch.

El principal objetivo de este simulador es visualizar el envío de mensajes desde los procesadores hasta la memoria, teniendo en cuenta los distintos conflictos que pudieran llegar a ocurrir en cada unos de los switches.

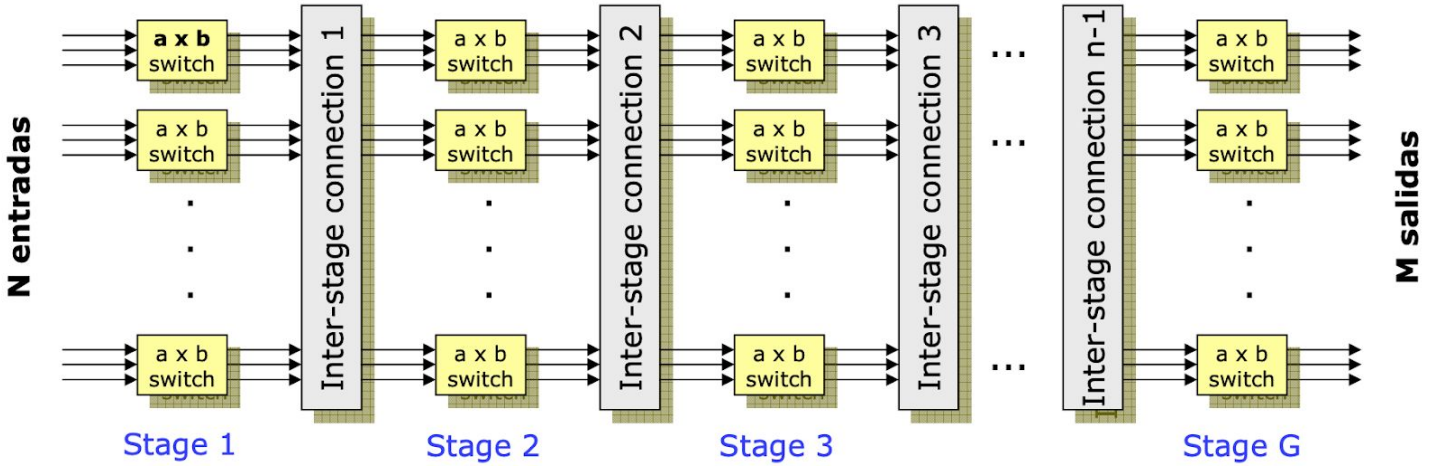
El mismo fue desarrollado como una aplicación web, de manera que cualquier alumno pueda probarlo sin necesidad de instalación y solo accediendo desde el sitio web de la materia.

UNICEN
Universidad Nacional del Centro
de la Provincia de Buenos Aires

TRABAJO FINAL

INTRODUCCIÓN AL PROBLEMA

Una red multietapa se compone de G etapas generalmente iguales. Cada etapa se forma con k switches compuestos por crossbars de **a** x **b** entradas/salidas.
Entre etapas adyacentes se usa una red de interconexión fija donde el número de etapas, la cantidad de crossbars de cada etapa y los valores **a** y **b** determinan la capacidad de encaminamiento de las redes MIN.



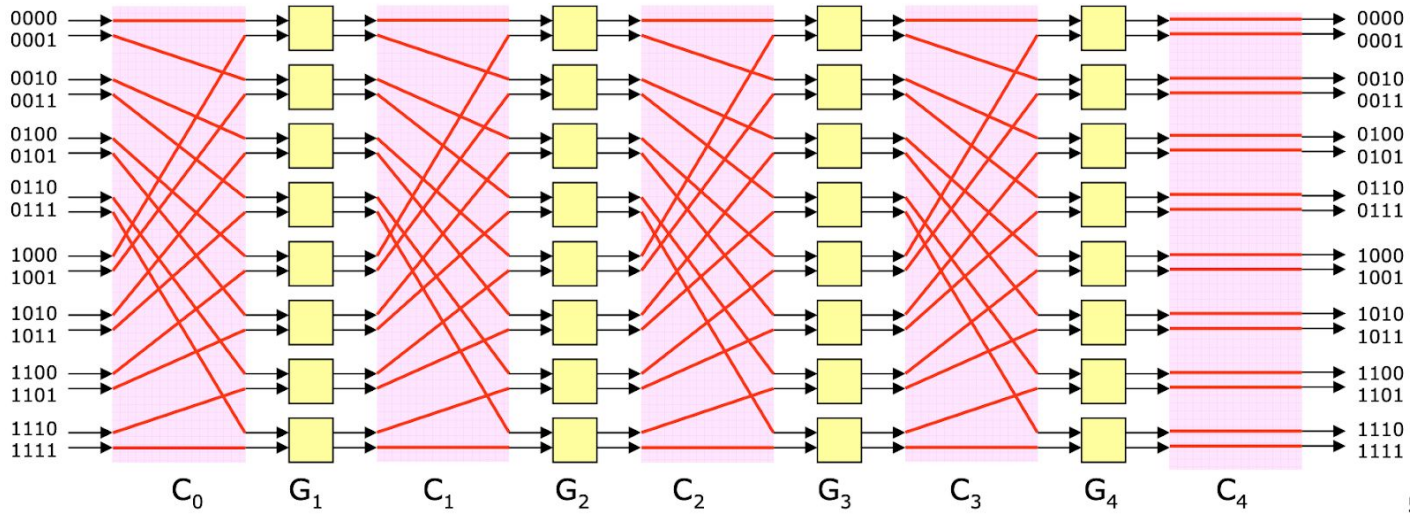
En nuestra aplicación, el número de entradas y salidas de cada switch es el mismo, por lo que el número de entradas N y el número de salidas M, también es el mismo. Por lo que se define como **unicast.**

Las distintas redes multietapa se diferencian en los módulos conmutadores (switches) y en los patrones de las conexiones entre etapas (CEE).

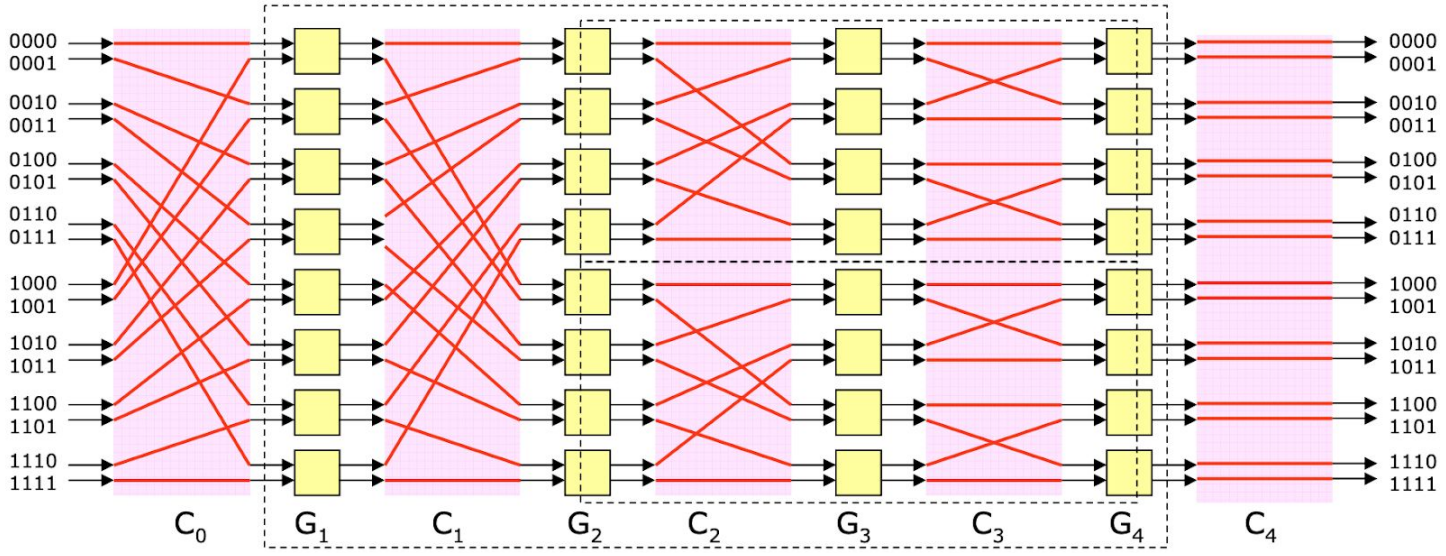
Las más comunes son: Perfect shuffle, Butterfly, Crossbar, Reversal, Baseline, Exchange.

Las redes implementadas son redes **bloqueantes,** es decir que no siempre es posible la conexión entre un puerto de entrada y otro de salida por conflictos entre conexiones existentes.

En una red **Omega** el patrón de conexión Ci está descrito por la (n-i)-ésima permutación perfect-shuffle σ k (1≤i≤n-1). El patrón de conexión Cn está descrito por β 0 (butterfly sub cero = identidad). Posee algoritmo de auto encaminamiento en el que los bits sucesivos de la dirección de destino controlan las sucesivas etapas de la red.



En una red **Baseline** el patrón de conexión Ci está descrito por la (n-i)-ésima permutación en línea base δn-i (1≤i≤n). El patrón de conexión C0 está descrito por σ k (perfect-shuffle). Cada etapa divide el camino en 2 sub caminos, uno hacia la parte alta y otro hacia la parte baja de la subred correspondiente.



Funcionamiento de la red:

En la entrada a la etapa Gi el switch chequea el I-ésimo bit de la dirección de destino si es 0, el switch selecciona la salida superior sino, la salida inferior.

Cada vez que la dirección pasa cada etapa I, si el paquete llegó por el puerto superior, entonces el I-ésimo bit se reemplaza por 0. Si el paquete llegó por el inferior, el I-ésimo bit se reemplaza por 1.

Desarrollo

La aplicación fue desarrollada utilizando *VueJs* (https://vuejs.org/), un framework progresivo que transcribe Javascript. Esta decisión fue tomada por dos cualidades que presenta el framework:

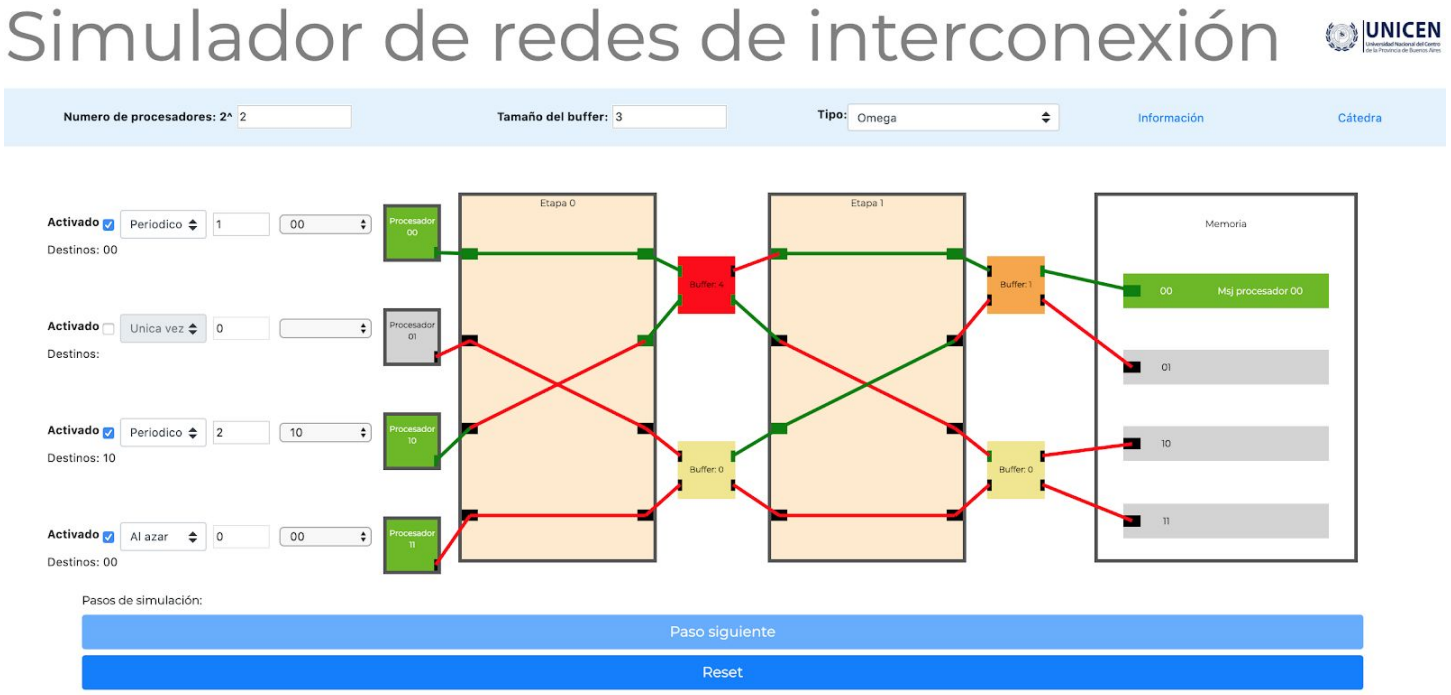
- Reactividad: Vue presenta un sistema de reactividad que utiliza objetos JavaScript simples y una representación optimizada. Cada componente realiza un seguimiento de sus dependencias reactivas durante su renderizado, por lo que el sistema sabe con precisión cuándo volver a renderizar y qué componentes volver a renderizar.
- Componentes: Los componentes Vue extienden elementos HTML básicos para encapsular código reutilizable. En un nivel alto, los componentes son elementos personalizados a los que el compilador de Vue asocia el comportamiento. En Vue, un componente es esencialmente una instancia de Vue con opciones predefinidas. Para nuestro simulador en particular cada crossbar, conexión, procesador, slot de memoria, etapa, puerto, memoria corresponde a un componente Vue.

Como se mencionó anteriormente, en el simulador se podrán graficar completamente redes Baseline u Omega, cada una con la cantidad de procesadores que el usuario determine. Para esto, se deberá seleccionar un número de procesadores, un tipo red, y el tamaño del buffer de cada switch lo cual inmediatamente dibujará el cuerpo de la red. De esta forma, se permitirá seleccionar todos los requerimientos o características propias de cada procesador, de la red propiamente dicha, y del envío de mensajes. Dichas configuraciones serán brindadas de manera selectiva o para completar manualmente, cada una de ellas dependiente de la opción inmediatamente precedente.

Una vez seleccionadas las características de la red y de los envíos de mensajes, se contará con dos botones, “*Paso siguiente*” y “*Reset*”. El primero se encargará de mostrarnos cada paso de la simulación (cabe destacar que cada click en el botón corresponde a cada movimiento o paso del mensaje dentro de la red, desde su

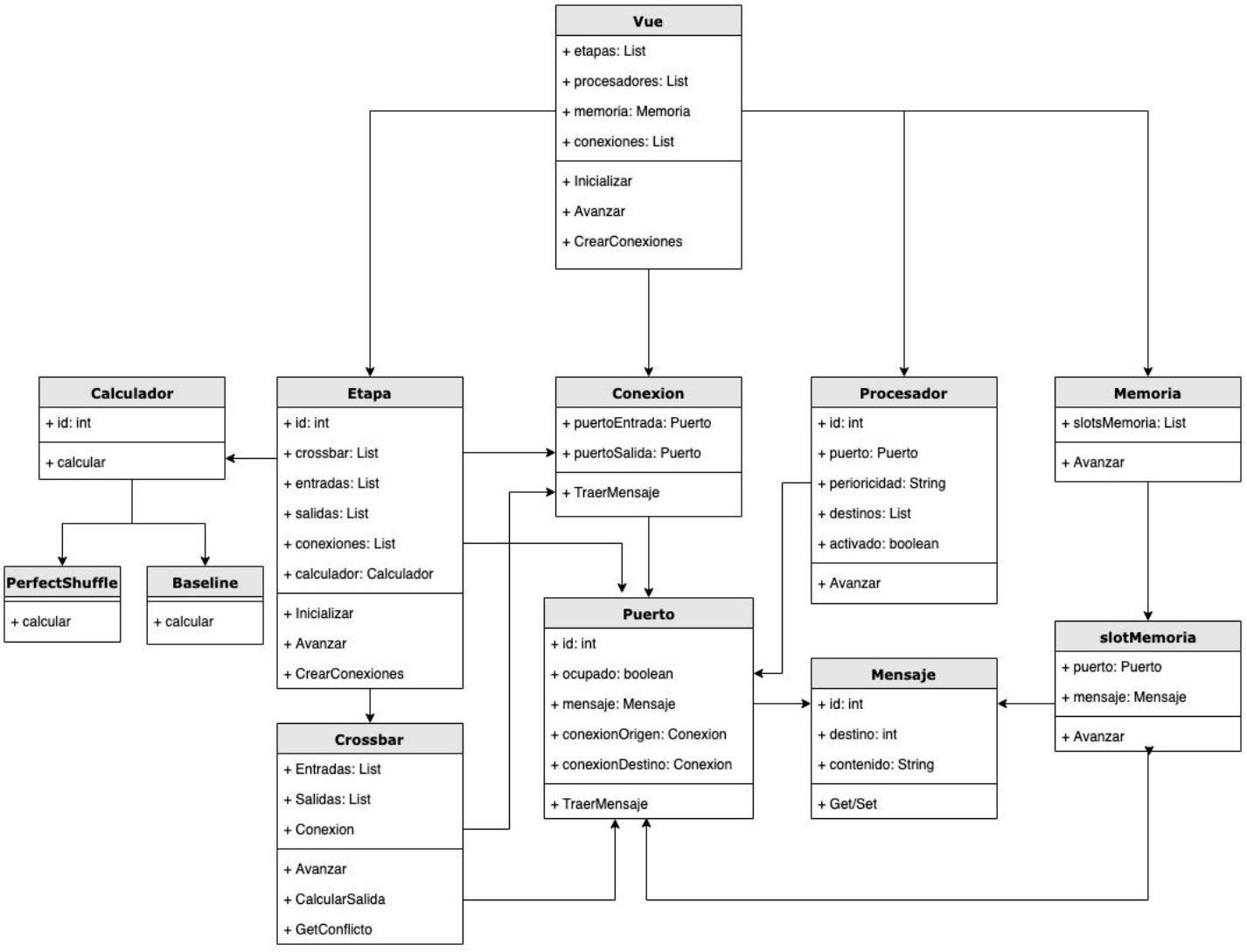
salida del procesador hasta su llegada a memoria, y no a cada ciclo de reloj). El segundo botón, se encarga de resetear completamente el simulador, llevándonos a la pantalla principal la cual nos permitirá realizar una nueva simulación, completamente diferente a la anterior.

Obteniendo la siguiente interfaz:



Estructura del simulador

A continuación, se adjunta un diagrama con las principales clases utilizadas para la implementación de la aplicación:



Para una correcta implementación se decidió utilizar la estructura graficada a fin de encapsular las características y comportamientos de cada objeto.

La clase principal Vue, contiene los distintos elementos para armar la simulación tales como una lista de procesadores, una lista de etapas, una memoria y una lista de conexiones, las cuales representan a las conexiones externas. [1]

Cada uno de los procesadores son iniciados luego de verificar la cantidad ingresada por el usuario, y una vez graficados se permite la configuración individual de cada uno seteando los parámetros correspondientes de cada objeto.

Cada una de las etapas contiene una lista de entradas y salidas. Los crossbar son parte de cada etapa y la cantidad de éstos, dependen de la cantidad de procesadores ingresados por el usuario.

Al necesitar diferentes formas de calcular la conexiones internas [2] en las distintas etapas según el tipo de red, cada una de éstas contiene un calculador que redefine su método calcular a fin de utilizar Perfect Shuffle o Butterfly.

El objeto Memoria solo contiene una lista de SlotsMemoria que reciben y almacenan los mensajes recibidos de los procesadores.

Cada conexión es establecida entre dos puertos, uno de entrada y uno de salida. Asimismo, cada puerto conoce sus conexiones para poder realizar el avance en la simulación.

[1] Las conexiones externas son aquellas que posee el simulador para enviar mensajes y conectar los puertos que se encuentran fuera de cada etapa de interconexión, tanto desde los procesadores hacia la entrada de la primer etapa, desde las salidas de las etapas hasta la entrada de los switch y desde la salida de los switch a las entradas de la próxima etapa o hacia un slot de memoria.

[2] Las conexiones internas son las generadas por la propia etapa en base a lo calculado por su propio calculador, el cual recibe una dirección de entrada y genera una dirección de salida.

Por su parte, cada switch tiene la capacidad de crear un conexión dinámica entre sus entradas y su salida. Dado que la salida es determinada por el i-ésimo bit, cada switch lee el primer bit de la dirección y realiza un desplazamiento, reemplazando el mismo por el valor correspondiente como se explicó con anterioridad.

Calculadores

Al implementar diferentes tipos de tipos de redes, fue necesario incluir en cada etapa un calculador para determinar la salida correspondiente a cada entrada.
Gracias al polimorfismo que es posible utilizar en lenguajes como JavaScript los diferentes calculadores pueden definir métodos con el mismo nombre y de esta forma atender a la solicitud de calcular la salida para las diferentes etapas de distintos tipos de red de forma dinámica.
La asignación de calculadores en cada etapa es determinada por el tipo red como se mencionó anteriormente.
Fue necesario la implementación de dos calculadores:

Perfect Shuffle:

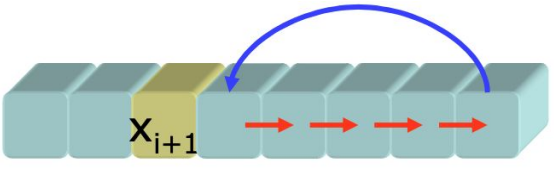
$$\sigma^k(x_{n-1}\ x_{n-2}\ \ldots x_1\ x_0) = x_{n-2}\ \ldots\ x_1\ x_0\ x_{n-1}$$

Perfect shuffle realiza un desplazamiento cíclico hacia la izquierda de los dígitos de X en log2 k posiciones. Con k = 1, 2… 4 y k ≤ N.

Baseline:

$$\delta_i(x_{n-1}\ldots x_{i+1}\,x_i\,x_{i-1}\ldots x_1\,x_0) = x_{n-1}\ldots x_{i+1}\,x_0\,x_i\,x_{i-1}\ldots x_1$$

1 <= i <= N-1



De esta forma cada calculador redefine el método Calcular para retornar la salida correspondiente a la i-ésima etapa.

Decisiones de implementación

Al tratarse de una red bloqueante, debíamos analizar los conflictos que pueden aparecer dentro de la red. Se observó el caso de que dos mensajes pueden arribar al switch de manera simultánea, por lo que deberíamos optar por cierto criterio para poder mostrar por pantalla dicho conflicto y realizar una solución que permitiría continuar con el proceso de simulación. Dicha solución, fue planteada de la siguiente manera:

- Cuando llega un único mensaje al switch, continúa su proceso sin producir ningún tipo de conflicto.
- Si arriban dos mensajes de manera simultánea, ocupando ambas entradas del switch, el mensaje que ocupa el puerto más alto pasa a la salida del switch, y el mensaje que se encontraba en la entrada inferior del switch pasa al buffer alojado dentro del switch.
- Cada vez que saco un mensaje alojado en el buffer, si existe otro mensaje en uno de los puertos del switch, este pasa a almacenarse dentro del buffer para poder liberar una entrada del switch.
- En el caso de querer introducir un elemento en el buffer y esté se encuentra lleno, se genera un conflicto que detiene la simulación.

De esta forma, cada switch tiene tres posibles estados:

- Sin elementos en el buffer (Amarillo).
- Con elementos en el buffer (Naranja).
- Conflicto (Rojo).

En cuanto al proceso de envío de mensajes, el simulador comienza a buscar mensajes desde los procesadores hacia la memoria. Cada uno de los elementos en forma independiente es capaz de verificar cada una de sus conexiones y así de esta forma traer el mensaje. Cada conexión presente en un elemento verifica si tiene un mensaje pendiente en su puerto de entrada, y en tal caso lo posiciona en su puerto de salida. De esta manera, se logra una independencia del mecanismo de envío de mensajes con respecto al tipo de red o lugar de donde se encuentre dicha conexión.

Se decidió parametrizar el tamaño del buffer del switch, no solo para reducir la probabilidad de conflictos que detengan la simulación, sino además para estudiar el impacto en que puede tener en la ejecución tomando en cuenta la cantidad de procesadores y las direcciones de memoria a las cuales pueden comunicarse.

Conclusión

Como estudiantes de la carrera de Ingeniería en Sistemas, creemos que la utilización de este tipo de software conlleva a que los alumnos obtengan un enfoque aún más profundo de una unidad o tema en particular que se dicte en la materia, permitiéndoles realizar una práctica más detallada en el tema, con componentes que se asemejan a los existentes en la vida real.

En nuestro caso particular, la realización de este trabajo nos dio la oportunidad de crear software que corre via web lo cual no habíamos tenido la oportunidad de realizar hasta el momento, realizando un estudio de las tecnologías como Vuejs, Javascript, HTML y CSS. Además, la posibilidad de estudiar en profundidad los distintos tipo de redes de interconexión multietapa, y las características propias que poseen cada una.

Esperamos que este proyecto sea de utilidad para los alumnos de arquitectura de computadoras y sistemas digitales, ayudándolos a una mejor interpretación al igual que a nosotros acerca del funcionamiento de redes de interconexión.