Universidad Politécnica de Cartagena | Campus de Excelencia Internacional

Grado en Ingeniería Telemática
2019-2020

Trabajo Fin de Grado

Simulador de redes de conmutación telefónicas de tres etapas

Eduardo Pérez Soto

Tutor

José María Malgosa Sanahuja

RESUMEN

La presente memoria trata del desarrollo de un simulador de redes de conmutación de tres etapas, el cuál es el objetivo principal del proyecto. El simulador se ha implementado en un proyecto web por lo que, a parte de lo que es el propio simulador, se ha diseñado la web correspondiente para una mayor comodidad en su utilización. La WEB se ha definido a través de HTML, PHP, CSS y JavaScript. El desarrollo del simulador ha conllevado dos partes, la parte gráfica que se ha llevado a cabo mediante SVG (Scalable Vector Graphics) el cuál es un formato de gráficos vectoriales bidimensionales y la parte de generar las conexiones que para ello se ha abordado un simulador programado en C que, dependiendo de los parámetros de entrada, devolverá una secuencia u otra.

Finalmente, el proyecto se ha implementado a través de Linux, mediante Apache para correr la aplicación en local y utilizarse como un recurso de las prácticas de la asignatura de conmutación en la UPCT.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

1. REDES DE TRES ETAPAS

1.1. Introducción

La forma más natural de construir un conmutador espacial es a través de una red *crossbar*. Sin embargo, si el tamaño de la red es muy grade (N >>1), el número de puntos de cruce crece con su cuadrado (N^2). Una alternativa que permite construir redes grandes sin utilizar un número excesivo de puntos de cruce son las redes de tres etapas.

Las redes de conmutación de tres etapas están compuestas por una serie de etapas iniciales, unas etapas intermedias y unas etapas finales (ver figura 1). En los entornos digitales, existen dos tipos de conmutación: espacial y temporal. En este caso, nos centraremos únicamente en la conmutación espacial. Cada una de estas etapas es una matriz de puntos de cruce *crossbar*.

Estas redes de conmutación funcionan bajo el paradigma conocido como Conmutación de Circuitos. En estos sistemas se requiere de la existencia de un camino dedicado entre cada entrada y salida de la red. Este camino es una secuencia de enlaces conectados entre etapas de la red. En cada enlace físico, se dedica un canal lógico para cada conexión. La comunicación en circuitos conmutados conlleva tres fases:

1. Establecimiento del circuito, ya que antes de transmitir cualquier señal, se debe establecer un circuito extremo a extremo.
2. Transferencia de datos de extremo a extremo a través de la red. Los datos podrán ser tanto analógicos como digitales, dependiendo de la naturaleza de la red. Normalmente, la conexión es full dúplex.
3. Desconexión del circuito, una vez ha acabado la fase de transferencia de datos.

Cuando no existe un camino dedicado entre una entrada y salida del sistema, se dice que se ha producido bloqueo.

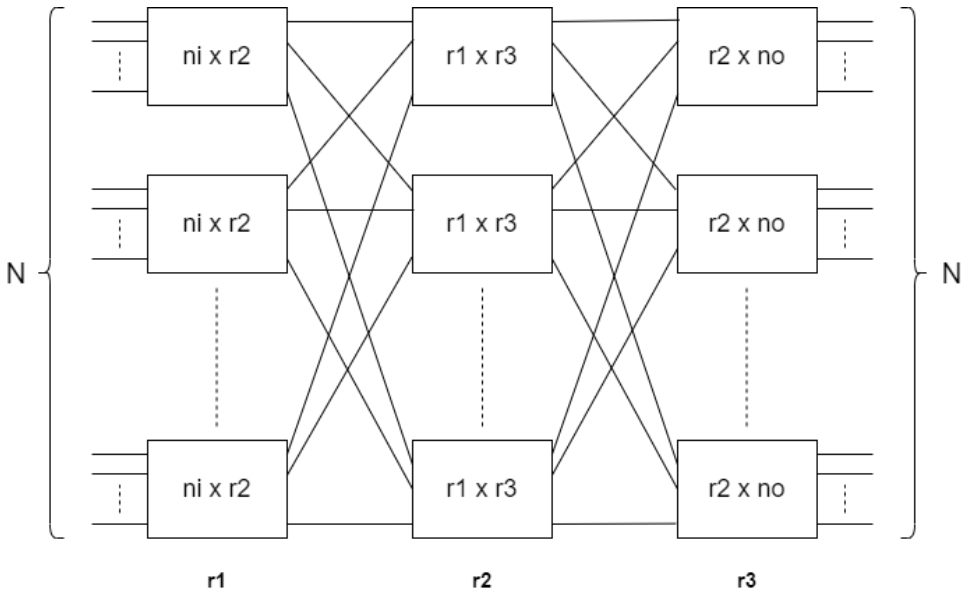


***Figura 1**. Esquema de red de conmutación de tres etapas*

1.2. Bloqueo en redes de tres etapas

En ocasiones, el sistema puede no tener recursos suficientes para establecer una conexión determinada, es decir, es posible que no exista un camino disponible que conecte cada par de entrada y salida de la red. Cuando esto ocurre se dice que se ha producido bloqueo. Existen dos tipos de bloqueo: bloqueo interno y externo.

El bloqueo interno ocurre cuando el conmutador no tiene recursos para conectar un circuito de entrada a una salida.

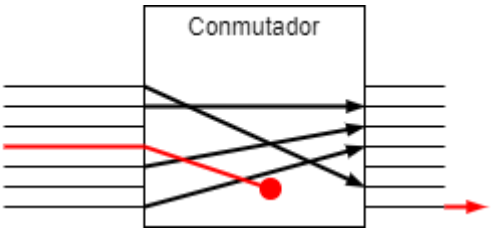


***Figura 2**. Ejemplo de bloqueo interno*

El bloqueo externo se da cuando el conmutador no tiene suficientes recursos de salida para cursar una nueva llamada.

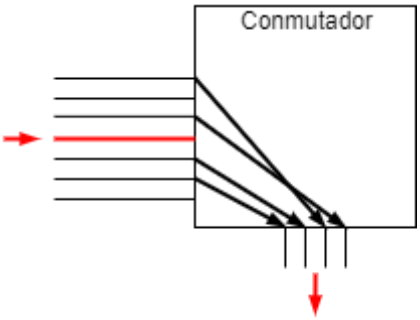


***Figura 3**. Ejemplo de bloqueo externo*

Si existe un camino que conecte cualquier entrada con cualquier salida se dice que el sistema es de accesibilidad total. Las redes de conmutación sin bloqueo y con accesibilidad total son las que se utilizan en la mayoría para datos.

En aquellas situaciones en que la red de conmutación (de tres etapas de cualquier otro tipo) presente bloqueo interno, se utiliza el método de Lee para calcular la probabilidad de bloqueo de la red. Es aproximado, es decir, no es exacto, funciona mejor cuando la carga de la red es baja. Para poder aplicarlo se tienen que cumplir una serie de hipótesis:

- Todas las fuentes generan el mismo tipo de tráfico.
- Los caminos se seleccionan aleatoriamente.
- Las llamadas rechazadas se pierden.
- Las ocupaciones entre etapas sucesivas son independientes.
- La ocupación de los enlaces de una misma etapa también es independiente.

Una vez se cumplen todas estas hipótesis y antes de aplicar el método de Lee, hay que calcular el grafo de red (convirtiendo las matrices en vértices y las conexiones en aristas) y el grafo canal (conjunto de todos los caminos que unen cada par entrada-salida). Un ejemplo sería:

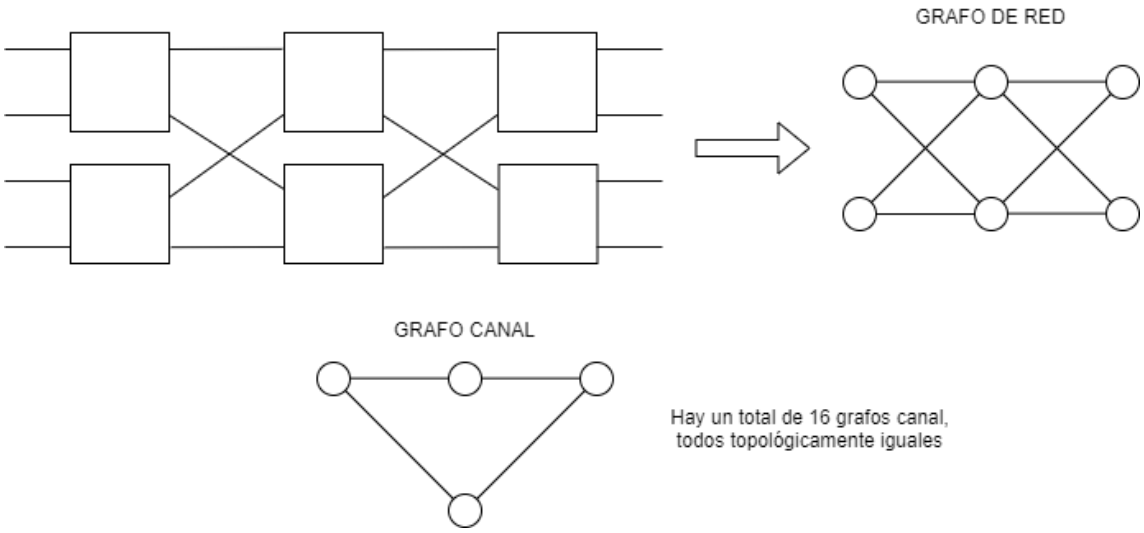


***Figura 4**. Ejemplo de obtención de grafo de red y grafo canal*

Una vez calculados, la probabilidad de bloqueo se calcula sobre los grafos canales de la siguiente forma:

$$PB_{RED} = \sum_i PB_{GC_i} * P_{GC_i} = \sum_i PB_{GC_i} * \frac{Nº\ de\ GC_i}{total\ GC}$$

Donde cada variable corresponde a:

- PB_{RED}: probabilidad de bloqueo de la red.
- PB_{GCi}: probabilidad de bloqueo del grafo canal i.
- P_{GCi}: probabilidad de que sea utilizado el grafo canal i.
- GC: grafo canal.

1.2.1. Condición de Clos

Esta condición se utiliza para eliminar el bloqueo interno de nuestra red. A través de ella se calcula el número de etapas intermedias para asegurar que no haya bloqueo interno. Situándonos en el peor caso, observamos que desde una matriz de entrada se han establecido (n_{in} – 1) conexiones hacia distintas matrices de salida diferentes. Análogamente, para una matriz de salida hay (n_{out} – 1) conexiones establecidas. Por tanto, para este peor caso, se están utilizando (n_{in} – 1) + (n_{out} – 1) matrices de la etapa intermedia. Si se decidiese iniciar otra conexión por esas mismas matrices, para que no

haya posibilidad de bloqueo interno, el número de etapas intermedias debería ser mayor o igual que (n_{in} – 1) + (n_{out} – 1) + 1, que simplificando obtenemos la siguiente ecuación:

$$r_2 \geq n_i + n_o - 1$$

Donde cada variable corresponde a:

- r_2: número de etapas intermedias.
- n_i: número de entradas de cada etapa inicial.
- n_o: número de salidas de cada etapa final.

En redes simétricas, en las que $n_i = n_o = n$, la ecuación se queda de la siguiente manera:

$$r_2 \geq 2n - 1$$

1.3. Redes reordenables

Una red se dice que es reordenable, o reacondicionable, si no cumple la condición de Clos, pero puede efectuar la conexión entre cualquier par entrada-salida, aunque para ello sea necesario cambiar algunas de las conexiones para conservar las comunicaciones existentes. Para ello, cada vez que una llamada entre, el algoritmo de encaminamiento correspondiente busca una etapa intermedia disponible y, si no la encuentra, se reencaminan las llamadas necesarias con el objetivo de establecer la nueva llamada. Por ejemplo:

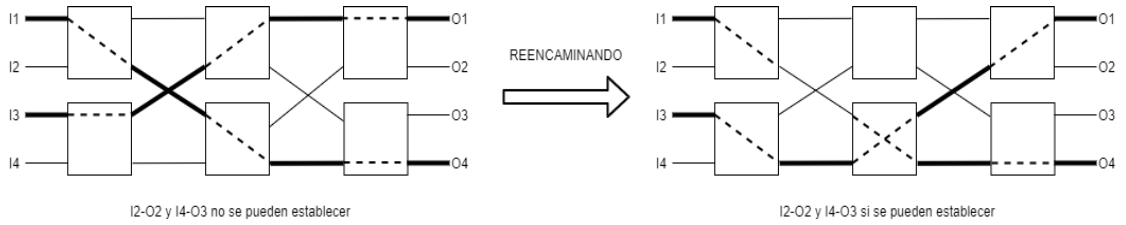


***Figura 5**. Ejemplo de reordenación de una red*

No todas las redes son reordenables, solo lo son aquellas que cumplen el teorema de Slepian-Duguid, explicado posteriormente en el punto 1.3.1. Una vez comprobada si una red es reordenable, según el teorema de Paull, sabemos que existe al menos una combinación cuyo número de reordenaciones es menor o igual a:

$$\min(r1, r3) - 1$$

Por ejemplo, para una red con número de etapas iniciales igual a 4 y un número de etapas finales igual a 5, existirá al menos una combinación cuyo número de reordenaciones sea menor o igual a *min(4,5) – 1* que sería 3.

1.3.1. Teorema de Slepian-Duguid

A través de este teorema podemos conocer si una red de tres etapas es reordenable, comprobando si se cumple la siguiente condición, donde el número de etapas intermedias r_2 debe de ser mayor que el máximo entre n_i y n_o.

$$r_2 \geq \max(n_i, n_o)$$

En caso de que la red sea simétrica ($n_i = n_o = n$), la ecuación se reduce a:

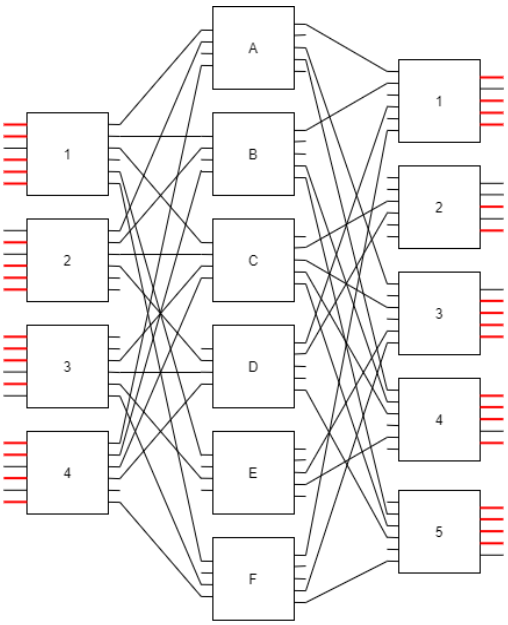
$$r_2 \geq n$$

1.3.2. Algoritmo de reconfiguración

Este algoritmo consiste en realizar las reconfiguraciones necesarias para que la conexión pueda realizarse con éxito. En primer lugar, comprobamos si la red es reordenable, y después, si es necesario reordenarla.

Para la aplicación del algoritmo construimos la matriz de Paull, en la que se representan las conexiones en curso de una red de tres etapas mediante notación matricial:

- Las filas corresponden a los bloques de entrada.
- Las columnas a los bloques de salida.
- Los elementos que se colocan en cada una de las posiciones son los bloques intermedios.



	1	2	3	4	5
1	F		A	B,E	C
2	A,B	D		C	
3		C	E,F		D
4	D		C	A	B,F

Matriz de Paull

Figura 6. Ejemplo de obtención matriz de Paull

Una vez tenemos la matriz, ya podemos empezar a reorganizar. Para ello, tomamos de ejemplo la conexión I2-O1, que corresponde con el bloque de entrada uno y bloque de salida uno.

	1	2	3	4	5
1	F		A	B,E	C
2	A,B				
3					
4	D				

Podemos apreciar como para esta conexión, no hay ninguna etapa intermedia disponible, debido a que una etapa intermedia solo puede aparecer, como máximo, una vez en cada fila/columna de la matriz de Paull. Por tanto, determinamos los bloques disponibles en cada fila y columna.

- Bloques disponibles en la columna: C, E.
- Bloques disponibles en la fila: D.

Posteriormente, determinamos las posibles parejas entre los bloques disponibles, las cuales serían: C-D, E-D, D-C, D-E. Para cada una de ellas, se permuta una con otra iterativamente, hasta que ya no sea posible realizar más permutaciones.

Como ejemplo, realizamos la reordenación de la matriz para el caso de la pareja C-D. El objetivo sería cambiar en la fila la C por la D y realizar las reordenaciones necesarias para que todas las etapas intermedias solo aparezcan, como máximo, una vez en cada fila/columna. Las reordenaciones son las siguientes:

	1	2	3	4	5
1	F		A	B,E	C
2	A,B	D		C	
3		C	E,F		D
4	D		C	A	B,F

Una vez se aplican todas las reordenaciones, la matriz de Paull quedaría de la siguiente forma:

	1	2	3	4	5
1	F		A	B,E	D
2	A,B	C		D	
3		D	E,F		C
4	D		C	A	B,F

Ahora ya podemos realizar la conexión I2-O1, mediante la etapa intermedia C.

	1	2	3	4	5
1	F,C		A	B,E	D
2	A,B	C		D	
3		D	E,F		C
4	D		C	A	B,F

Con esta pareja, se han realizado en total 5 reordenaciones. No obstante, anteriormente hemos visto como el teorema de Paull nos dice que tiene que existir otra combinación con un número de conexiones menor o igual que tres, por tanto, lo óptimo sería buscar una pareja que produzca menos reordenaciones.

1.4. Algoritmos de encaminamiento

En una *crossbar* cuando realizamos una conexión, tenemos la entrada y la salida y la intersección da el punto de cruce. En las redes de tres etapas, para una conexión, tenemos la entrada y la salida, y para poder conectarlas necesitamos seleccionar una etapa intermedia (de entre las disponibles). Este proceso de seleccionar la etapa intermedia se puede hacer de diferentes formas. Hemos estudiado tres configuraciones diferentes:

- Secuencial: Siempre se intenta seleccionar la primera etapa intermedia, si no es posible y esa etapa está ocupada, se intentará con la siguiente y así sucesivamente. Si se llegara a la última etapa y no se pudiera conectar, estaríamos en una situación de bloqueo, ya que no habríamos podido conectar por ninguna. De esta forma, se aprovechará mucho mejor la capacidad de todas las etapas intermedias, y por tanto, es la opción que minimiza el bloqueo interno.

- Aleatorio: La etapa intermedia se elige de forma aleatoria, en caso de que no se haya podido conectar con ninguna de las etapas, se produciría bloqueo.
- Cíclico: En un primer lugar se selecciona la primera etapa intermedia, luego cada conexión irá eligiendo la siguiente etapa para conectarse. Si esa etapa se encuentra ocupada, pasará a la siguiente. Cuando se llega a la última matriz, la próxima será la primera y así sucesivamente. Si no se pudo conectar por ninguna de las etapas, se producirá bloqueo.

2. SIMULADOR DE REDES DE TRES ETAPAS

2.1. Generación de números aleatorios

Es necesario generar números aleatorios en varios procesos del simulador, por ejemplo, en la elección aleatoria de la etapa intermedia, o dada una llamada, seleccionar la salida por la que saldrá dicha llamada. Todos ellos se realizan mediante un algoritmo que consta de dos archivos, *rand.c* donde se encuentra el programa principal, y *rand.h* donde se encuentran declaradas las funciones. El proceso que se lleva a cabo es el siguiente:

- Generar semillas al azar usando el reloj del ordenador. Se realiza mediante la función *int semillas(int p)* la cual, inicializa un total de MAX_SEEDS semillas. El valor de MAX_SEEDS está declarado en "rand.h" a un valor de 2048.
- Generar números entre 0 y 1 al azar. Se generan mediante la función *ranf(int p)*, que genera un número aleatorio entre (0, 1). Puede crear series independientes de números aleatorios si se invoca con semillas distintas (variable p).

Partiendo de la función *ranf*, ya se podría generar la variable aleatoria con cualquiera de las siguientes distribuciones:

- *double expo(double media, int p)*: genera una variable aleatoria exponencial a partir de su media. Puede generar series independientes de números aleatorios si se la invoca con semillas distintas (variable p).
- *int uniforme(int a , int b, int p)*: genera una variable aleatoria uniforme entre [a, b]. Puede generar series independientes de números aleatorios si se la invoca con semillas distintas (variable p).
- *long int geom(double media, int p)*: genera una variable aleatoria geométrica a partir de su media. Puede generar series independientes de números aleatorios si se la invoca con semillas distintas (variable p).
- *double rayleigh (double media, int p)*: genera una variable aleatoria de Rayleigh a partir de su media. Puede generar series independientes de números aleatorios si se la invoca con semillas distintas (variable p).
- *double gauss(double media, double sigma, int p1, int p2)*: genera una variable aleatoria gaussiana a partir de su media y su desviación típica. En este caso se necesitan dos semillas ya que, en este caso, la función de distribución debe generarse empleando dos variables aleatorias de Rayleigh independientes.

Para obtener la variable aleatoria a través de estas distribuciones (a excepción de la de gauss), se utiliza el método de la inversa. El funcionamiento de este método se explica a continuación.

Sea $F_a(x)$ la función de distribución de probabilidad de la variable "a" o función de densidad acumulada (CDF). Sabemos que:

$$F_a(x) = \int_0^x f_a(t)\, dt$$

Siendo $f_a(x)$ la función de densidad de probabilidad (PDF).

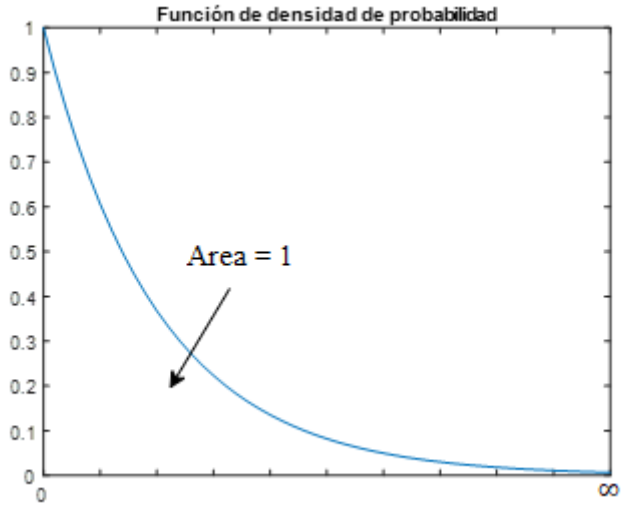


***Figura 7**. Ejemplo de función de densidad de probabilidad de una exponencial*

Poniendo "a" en función de "u", siendo esta uniforme, se tiene que:

$$a = G(u)$$

$$F_a(x) = \Pr\{a < x\} = \Pr\{G(u) < x\} = \Pr\{u < G^{-1}(x)\} = G^{-1}(x)$$

Dado que "u" es uniforme. Por lo tanto,

$$F_a(x) = G^{-1}(x)$$

$$F_a(x) = u$$

$$a = F_a^{-1}(u)$$

con "u" uniformemente distribuida entre 0 y 1, ya que éstos son los valores mínimo y máximo, respectivamente, de la CDF. Será éste un método genérico para la generación de variables aleatorias, aunque no es siempre aplicable ya que la inversa de $F_a(x)$ no se conoce siempre de forma analítica.

La generación de un número uniforme entre 0 y 1 más la posterior proyección sobre x mediante la función de distribución hacen que sean más probables aquellos valores de x para los que la $F_a(x)$ tiene mayor pendiente, es decir, con mayor $f_a(x)$; en definitiva los valores más probables. Este número uniforme entre 0 y 1 lo obtenemos en nuestro proyecto mediante la función *ranf.*

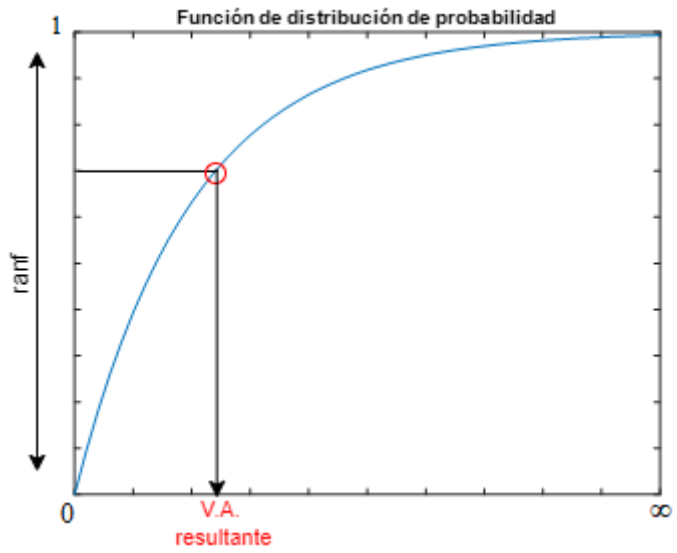


***Figura 8**. Ejemplo de función de distribución de probabilidad de una exponencial*

2.2. Estructura del simulador

El simulador se compone de tres archivos fundamentalmente, desarrollados en C. En primer lugar, está *besa.c*, donde se encuentra el programa principal. En segundo lugar, se encuentra *besa.h*, en el que están declaradas las funciones y parámetros utilizados. Por último, *besa_func.c* es el archivo donde quedan definidas todas las funciones. Profundizando en los archivos:

- *besa.c*: en él se encuentra el programa principal, su funcionamiento está explicado en el diagrama de bloques de la página siguiente, correspondiente a la *Figura 9. Diagrama de bloques del besa*. Cabe destacar que todas las funciones usadas están explicadas en el apartado de *besa_func.c*, donde se encuentran definidas. También, hacer hincapié en que a la hora de imprimir las conexiones, en caso de bloqueo, se indica con un "-1" en la etapa intermedia, y que la numeración, tanto de las entradas, las salidas y las etapas intermedias, comienza en 0.

- *besa.h*: están declaradas las funciones y parámetros que se encuentran compartidos con los otros códigos. Este se incluye mediante la etiqueta *#include besa.h* en los otros códigos.

- *besa_func.c*: se encuentran definidas todas las funciones que se utilizan en el simulador. Entre ellas se hallan *check_opt*, *init_ss*, *atrandom*, *setmstage*, *sequencial*, *cyclical*, *randomly*, *setconnections*, *setavg* y *print_matrix*. *check_opt* es utilizada para comprobar que los parámetros cumplen las especificaciones. *init_ss* inicializa la matriz de datos *ss* que contiene las entradas, salidas y etapas intermedias. *atrandom* se utiliza para obtener las entradas o salidas al azar. *setmstage* encamina las entradas y salidas, eligiendo las etapas intermedias correspondientes, según el algoritmo de encaminamiento seleccionado. *sequencial*, *cyclical* y *randomly* obtienen las etapas intermedias según el algoritmo de encaminamiento seleccionado. *setconnections* se usa para conocer

el número de conexiones que se han establecido. *setavg* obtiene la media de conexiones realizadas y, finalmente *print_matrix*, que imprime la matriz resultante con las conexiones.

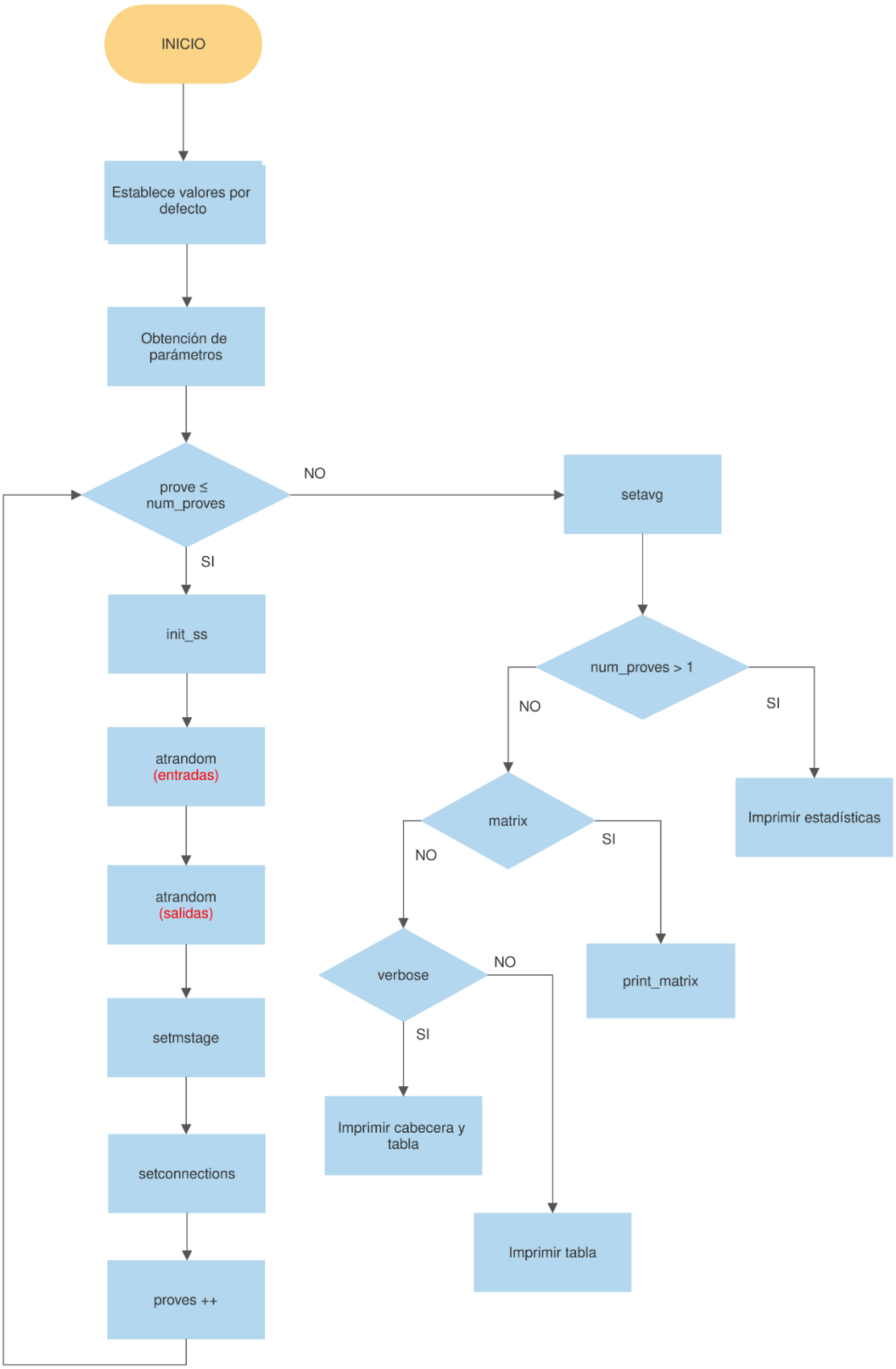


***Figura 9**. Diagrama de bloques del besa*

3. GRÁFICAS 2D EN LA WEB

3.1. Preliminares: JavaScript

JavaScript es un lenguaje de programación interpretado, es decir, que no requiere de compilación, sino que es analizado por otro programa. Es un lenguaje script orientado a objetos con funciones de primera clase, basado en prototipos, dinámico y soporta estilos de programación funcional.

Permite realizar actividades complejas en una página web como mostrar actualizaciones de contenido en el momento, interactuar con mapas, animaciones gráficas 2D/3D etc. Por este motivo, es utilizado para el dibujo de gráficos.

3.2. Cascading Style Sheets (CSS)

CSS es el lenguaje de diseño gráfico utilizado para describir la presentación de documentos HTML o XML, esto incluye varios lenguajes basados en XML como son XHTML o SVG. Este lenguaje describe como debe ser *renderizado* el elemento estructurado. La única forma de diseñar páginas HTML es con CSS. Por lo tanto, se usará CSS independientemente si es para crear gráficos o no.

A la hora de diseñar gráficos, en comparación con SVG (la opción finalmente elegida) podemos destacar los siguientes aspectos. Entre SVG y gráficos de mapa de bits, los gráficos de mapa de bits exportan una imagen y luego la cargan directamente en la página web. El mapa de bits es una imagen estándar. Sin embargo, los mapas de bits no pueden escalar o cambiar de color, como se hace con SVG.

Otro buen beneficio de SVG es que puede personalizar el color con CSS. Por lo tanto, puede hacer que se desvanezca en diferentes colores, o puede tener un gráfico que puede cambiar a cualquier color según sus necesidades. Por tanto, SVG está mucho más orientado a la realización de gráficos y de forma más sencilla.

3.3. Canvas

Es un elemento de HTML que permite dibujar gráficos, aplicarles animaciones y manipular imágenes en la web de forma dinámica, es decir, una vez creado el gráfico, se pueden programar opciones como, por ejemplo, para que el usuario interactúe.

Canvas se define con la etiqueta <canvas> </canvas> creando así un “mapa de bits”, pero cuando hablamos de Canvas nos referimos a toda la API que incluye un conjunto de funciones para dibujar, líneas, rectángulos, círculos, etc.

Canvas, por tanto, es una de las opciones planteadas a usar para el proyecto, para dibujar las redes de tres etapas en la web. En contraposición a la opción elegida, SVG, podemos destacar los siguientes aspectos. Con Canvas la representación visual se crea y modifica por programa, mediante scripting, dependiendo así de JavaScript, mientras que con SVG no es necesario. Canvas no crea objetos vectoriales como SVG, sino mapas de

bits como una imagen fotográfica, por tanto, SVG ofrece gráficos que son mucho más escalables. La API de Canvas no soporta la accesibilidad; deben utilizarse, además del Canvas, otras técnicas basadas en *markup*, mientras que el *markup* de SVG y el modelo de objeto soportan accesibilidad de forma nativa. Mediante Canvas se pueden crear objetos como SVG, pero el contenido del lienzo no es parte del *Document Object Model* (DOM) y, por lo tanto, en comparación con SVG, no es accesible para los lectores de pantalla. DOM es una API que proporciona una representación estructural del documento, permitiendo la modificación de su contenido o su presentación visual. En definitiva, por sencillez, y por las características del grafico que queremos construir, SVG es la mejor opción.

3.4. SVG

Es un lenguaje de marcado para describir gráficos vectoriales escalables, en formato XML, que pueden ser tanto estáticos como animados. Pesa poco y permite una definición mayor a tamaños reducidos.

SVG proporciona elementos para realizar círculos, rectángulos y curvas simples y complejas. Consiste en la etiqueta <svg> </svg>, que es el elemento raíz, y varias formas básicas que crean los diferentes gráficos.

Esta es la opción elegida para el proyecto debido a que, como se ha mencionado anteriormente, es simple, interactivo, escalable, vectorial, soporta accesibilidad de forma nativa, compatibilidad con todos los navegadores, no necesita javascript y es un standard abierto.

3.5. Gráficos 3D en la web: WebGL

WebGL (Web Graphics Library) proporciona gráficos en 3D para un navegador web mediante una API multiplataforma implementada en JavaScript. Está basada en la biblioteca de gráficos OpenGL ES 2.0. Esta nueva tecnología se ejecuta en el elemento Canvas de HTML5, y se integra con todas las interfaces DOM. Este API tiene numerosas ventajas como compatibilidad con distintos navegadores y plataformas, es compatible con todos los elementos del estándar HTML, los gráficos 3D son acelerados por hardware, etc.

Sin embargo, utilizamos SVG en el proyecto debido a la sencillez de los gráficos que se requieren, ya que los gráficos que queremos son en 2D, no es necesario 3D, por lo tanto, SVG es mucho más sencillo, y se adapta perfectamente a nuestras necesidades.

4. APLICACIÓN DESARROLADA

4.1. Diagrama de bloques

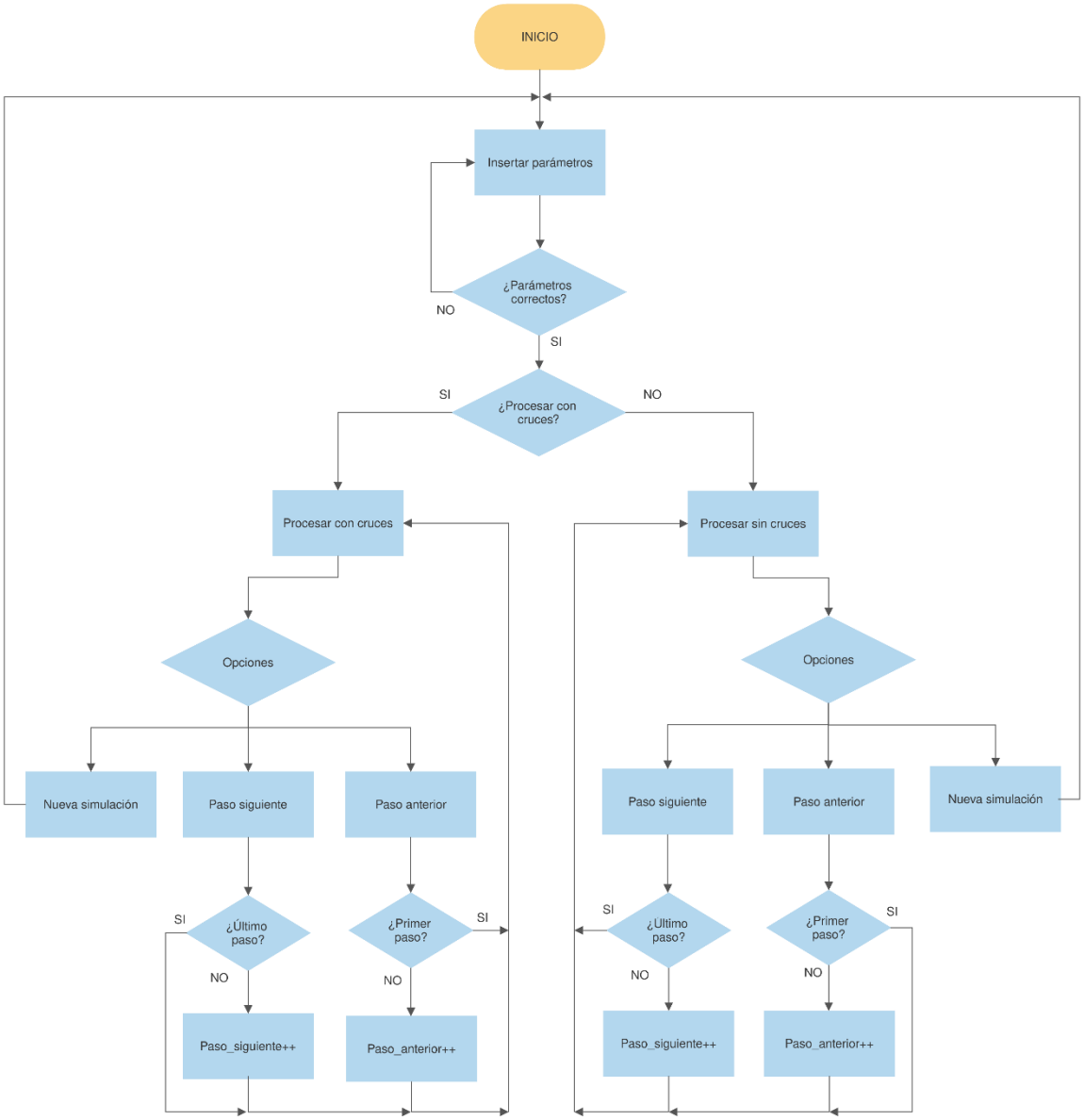


***Figura 10**. Diagrama de bloques de la web*

4.2. Vistas de la web

4.2.1. Índice

La siguiente imagen es la primera página que aparece al entrar a la web. Corresponde al fichero *index.php*, donde se encuentra programada.

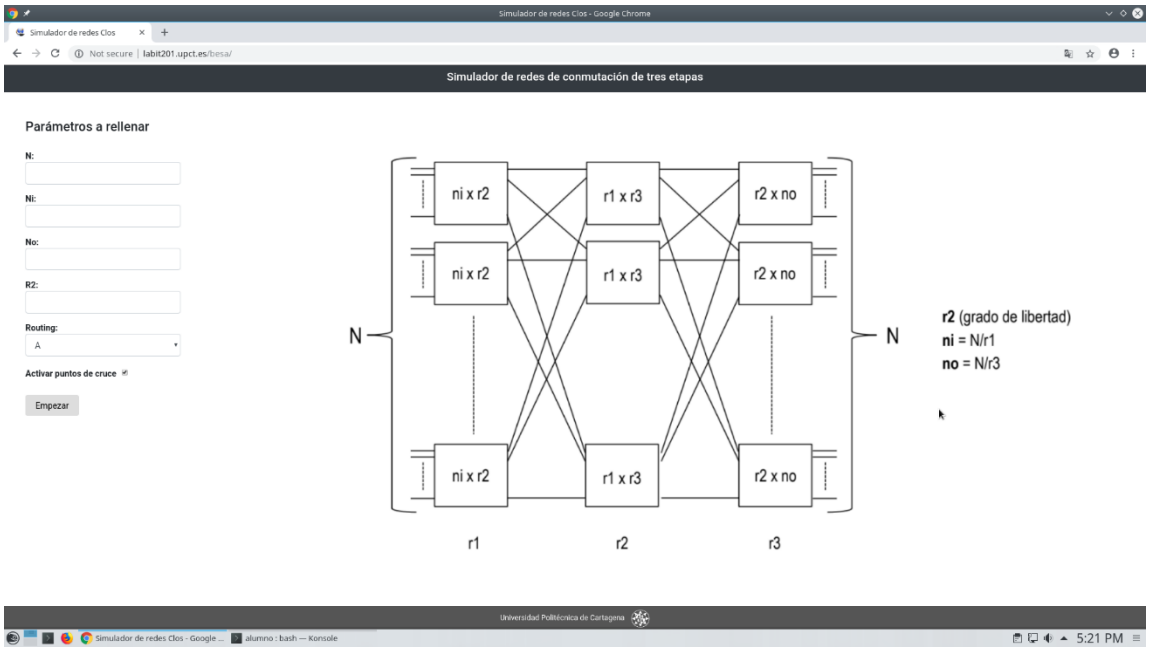


***Figura 11**. Vista de "index.php" en la web*

En ella, se introducen los siguientes parámetros de la simulación:

- N: número de entradas/salidas totales.
- Ni: número de entradas por etapa inicial.
- No: número de salidas por etapa final.
- R2: número de etapas intermedias.

Por último, podemos seleccionar una de las diferentes opciones de enrutamiento, las cuales aparecen como A (secuencial), B (aleatorio) y C (cíclico), y finalmente, si queremos que se muestren los puntos de cruce o no.

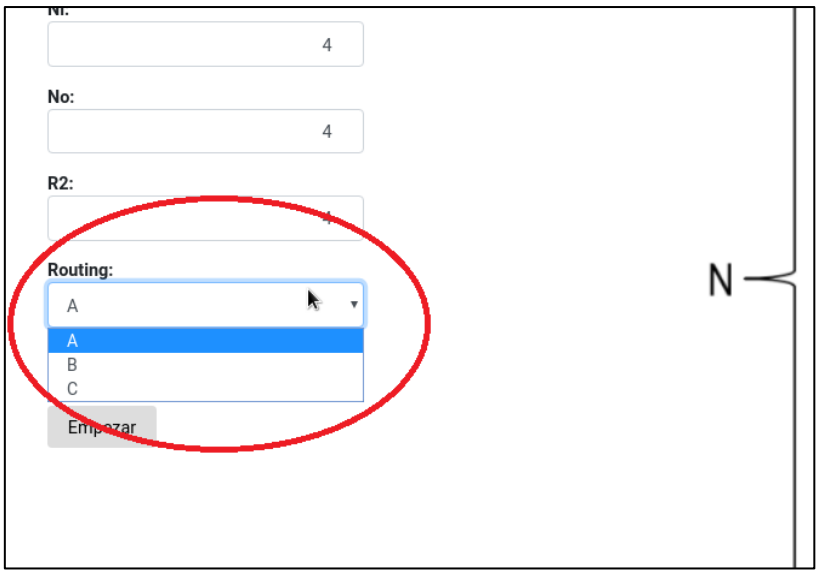


***Figura 12**. Vista de "index.php" en la web con las opciones de enrutamiento*

Los parámetros introducidos deben cumplir una condición, la cual consiste en que la división del número total de entradas/salidas (N) entre el número de entradas de cada etapa inicial (N_i) y entre el número de salidas de cada etapa final, deben de ser ambas un número natural, es decir, un número entero positivo.

$$\frac{N}{N_i} \wedge \frac{N}{N_o} \epsilon \mathbb{N}$$

En caso de que los parámetros introducidos no cumplan esta condición, aparece el siguiente mensaje de error:

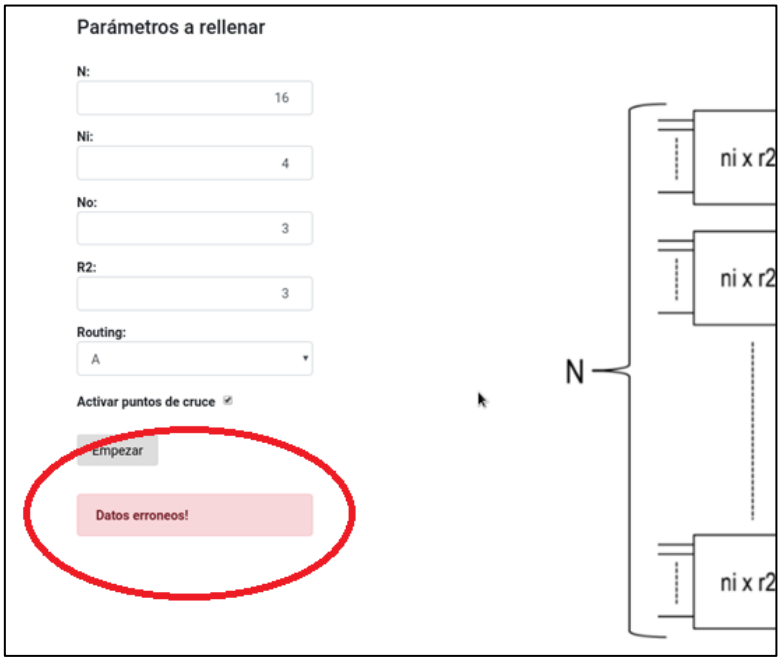


***Figura 13**. Mensaje de error al introducir parámetros que no cumplen los criterios*

También se han introducido restricciones para que no quede ningún parámetro vacío. En el caso de que no se introduzca algún parámetro, aparece el siguiente mensaje:

***Figura 14**. Mensaje de error cuando no se introduce algún parámetro*

4.2.2. Procesar con cruces

Una vez clicamos en *Empezar*, si se había marcado la opción de activar los puntos de cruce, se redireccionará a la siguiente vista, programada en *procesar.php*. En esta vista, se nos generará a la izquierda la red correspondiente y a la derecha, tendremos la tabla con las conexiones que se realizarán. Podremos avanzar y retroceder en la lista de conexiones, mediante los botones de *Paso siguiente* y *Paso anterior*. Podemos apreciar también, como en todas las etapas se pueden ver las conexiones internas, para ver como conmutan cada una de ellas internamente.

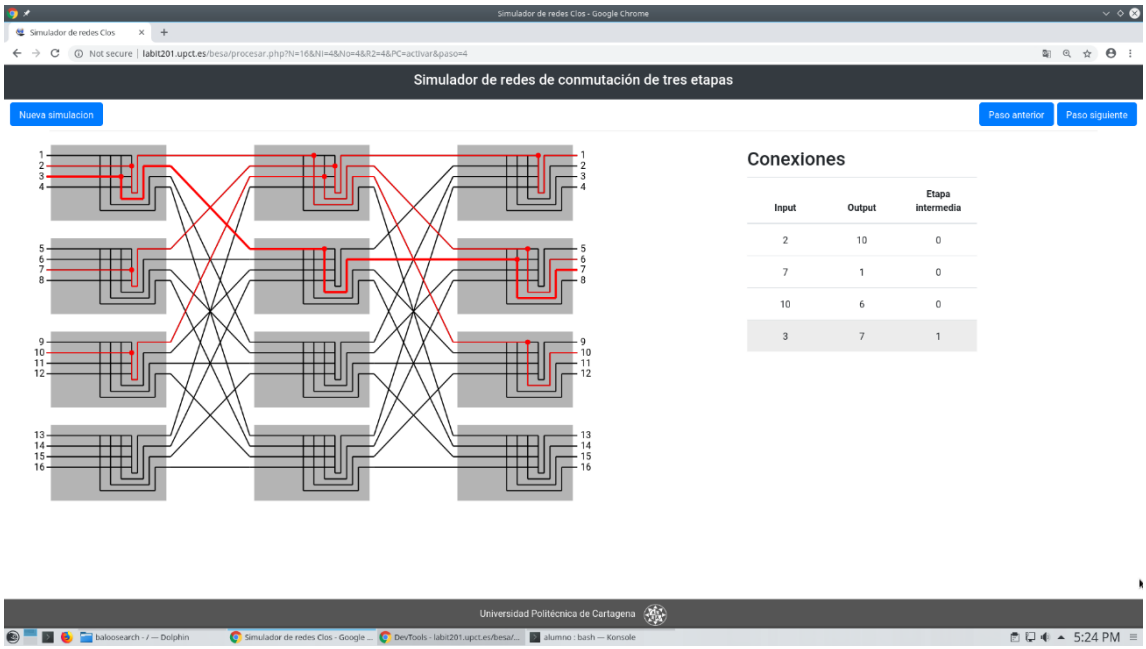


***Figura 15**. Vista de "procesar.php" en la web*

En la imagen anterior, se puede apreciar como la conexión que se está realizando aparece remarcada en la tabla de conexiones mediante un sombreado. En el gráfico, podemos distinguirla ya que el grosor de los cables de conexión es mayor frente al resto. Los puntos de cruce corresponden a los "puntos rojos".

Cuando se produce bloqueo, esa conexión no se dibuja en el gráfico ya que no hay etapa intermedia disponible y, en la tabla de conexiones aparece "Bloqueo" en vez de "-1" en el valor de la etapa intermedia.

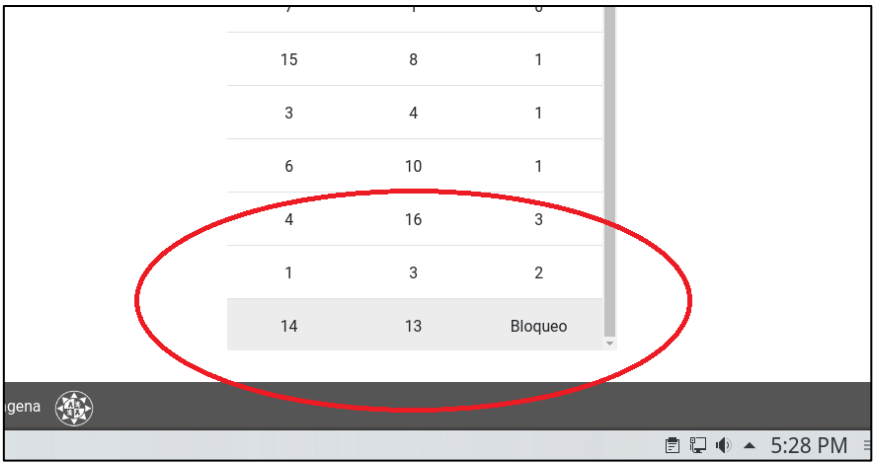


***Figura 16**. Notificación de cuando se produce bloqueo en la web*

Por último, apreciar cómo se redimensionan las etapas cuando generamos una red que sobrepasa los límites de la pantalla.

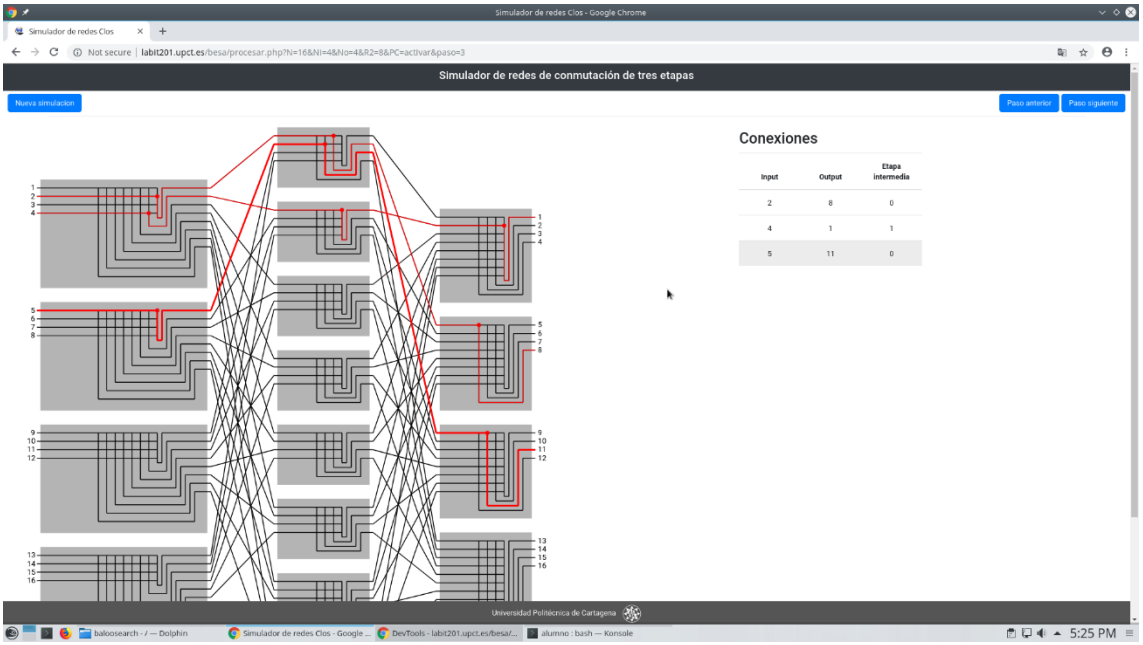


***Figura 17**. Vista de "procesar.php" en la web, donde están las etapas redimensionadas*

4.2.3. Procesar sin cruces

En caso de no haber marcado la opción de activar los puntos de cruce, se nos redireccionará a esta vista, programada en *procesar_sincruces.php*, sin embargo, al contrario de procesar con cruces, en esta vista no podemos apreciar cómo se produce el conexionado dentro de las etapas. Hay que destacar también que, al necesitar menos espacio, con esta opción se reduce el tamaño de las etapas significativamente y, por tanto, esta vista está diseñada para realizar redes más grandes sin ser tan aparatosas, y simplemente para observar cómo se producen las conexiones con cada uno de los

algoritmos implementados. Por último, hay que comentar que los botones funcionan de la misma manera que en *procesar.php*.

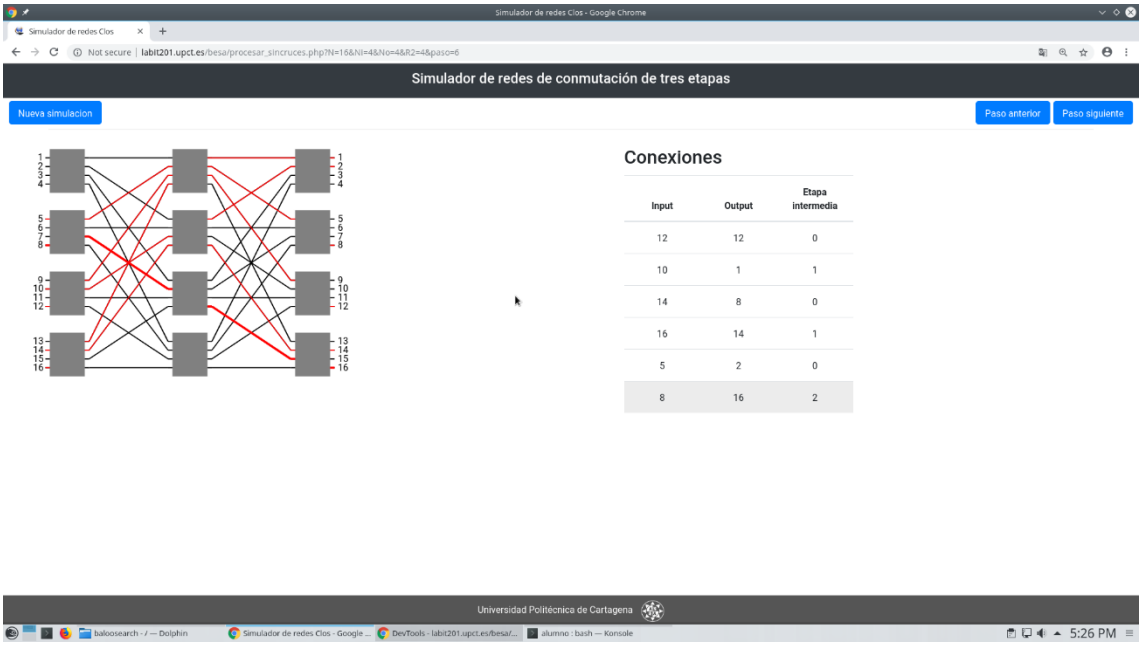


***Figura 18**. Vista de "procesar_sincruces.php" en la web*

4.3. Diseño del gráfico

El gráfico, como se ha comentado anteriormente, se ha realizado con SVG. Todo el gráfico se incluye en un único SVG el cuál se ha diseñado partiendo del origen de coordenadas situado en la esquina superior izquierda, como aparece a continuación:

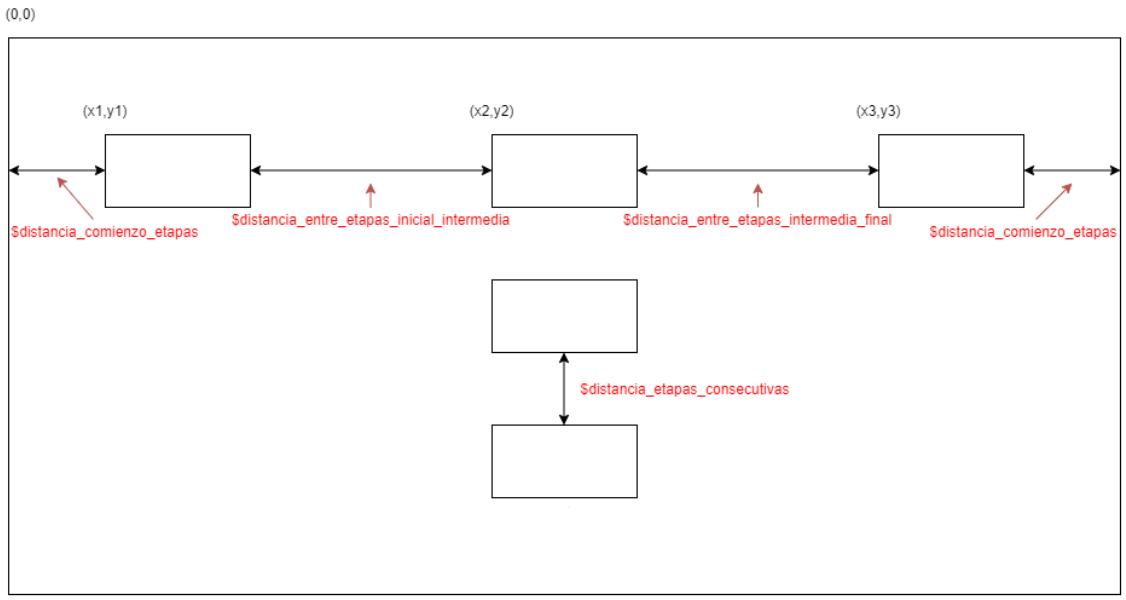


***Figura 19**. Diseño SVG de las distancias entre etapas*

La distancia de las tres etapas, tanto al borde superior como al inferior, no es igual, a no ser que éstas tengan la misma altura. *$distancia_bordes_verticales* es la mínima distancia que puede existir entre las etapas y los bordes superior e inferior. En caso de tener diferentes alturas, la etapa más cercana al borde superior tendrá una distancia de *$distancia_bordes_verticales*, las otras dos se calculan de forma que queden centradas respecto a la mitad del gráfico en función de la etapa con más altura, ya que es la que proporciona la altura del gráfico. Análogamente, esto ocurre con el borde inferior. En la siguiente imagen podemos ver un ejemplo de cómo quedarían ajustadas la distancia de las distintas etapas respecto a los bordes superior e inferior.

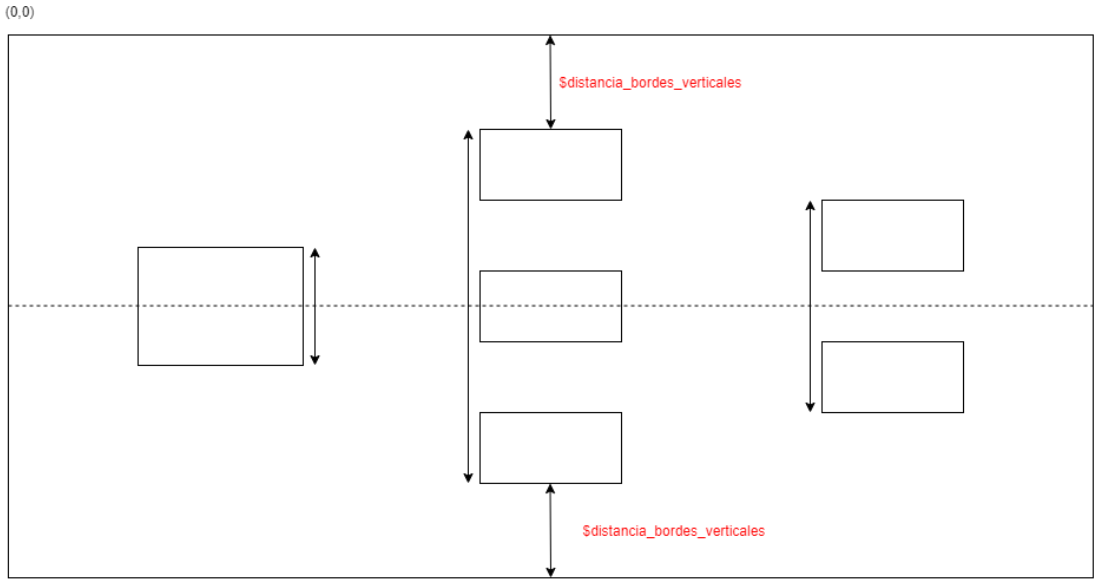


***Figura 20**. Diseño SVG de las distancias entre bordes superior e inferior*

De esta forma, la etapa con mayor altura determina el tamaño del SVG y la distancia de las otras etapas a los bordes superior e inferior, calculándose con estas distancias las coordenadas de las primeras etapas para dibujar el resto.

Así mismo, cada etapa está diseñada internamente mediante las siguientes variables:

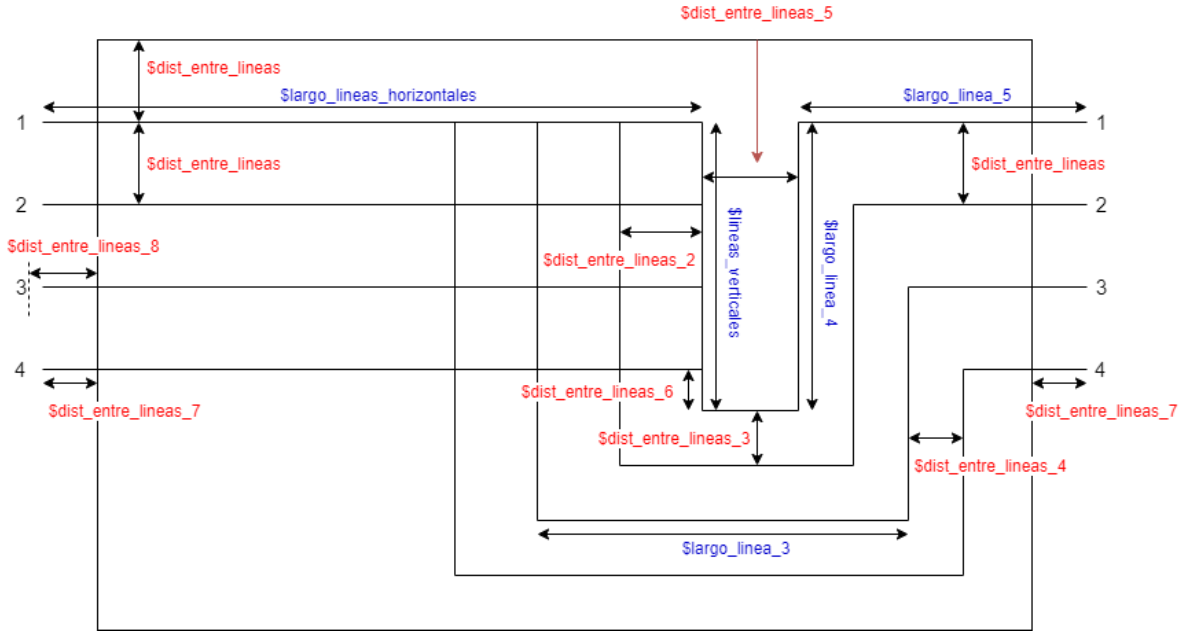


***Figura 21**. Diseño SVG de cada etapa*

Hay que destacar que *$dist_entre_lineas_3* corresponde a las etapas iniciales. Para las etapas intermedias sería *$dist_entre_lineas_3_2* y para las etapas finales, *$dist_entre_lineas_3_3*. También hay que decir que esta distancia varía en función de los parámetros de entrada.

Las dimensiones de las etapas varían en función de si son iniciales, intermedias o finales. De forma que quedan definidas por las siguientes variables:

- Etapas iniciales.
    - Altura: *altura_rect*
    - Ancho: *ancho_rectangulo*
- Etapas intermedias.
    - Altura: *altura_rect_medio*
    - Ancho: *ancho_rectangulo_medio*
- Etapas finales.
    - Altura: *altura_rect_final*
    - Ancho: *ancho_rectangulo_final*

En el caso de procesar sin cruces, debido a que no se dibuja el interior de la etapa, ya no se requieren todas las variables, pero las que se usan siguen teniendo el mismo nombre, con la diferencia de que los valores son distintos en algunos casos.

4.4. Realización de la web

La base de la web se ha realizado en HTML, que es un lenguaje que se utiliza para el desarrollo de páginas de Internet. Es la pieza más básica para la construcción de la web y se usa para definir el sentido y estructura del contenido. A parte de HTML, también se ha usado JavaScript (lo mínimo imprescindible), PHP, CSS y SVG.

4.4.1. JavaScript

En este proyecto, se ha utilizado para una mejor experiencia del cliente a la hora de hacer uso de la web, ya que éste permite mejoras en la interfaz de usuario y páginas web dinámicas.

A través de este lenguaje, hemos podido implementar una página dinámica. Se ha utilizado a la hora de adaptar la longitud de la tabla que contiene las conexiones al alto de la página, dependiendo del tamaño del grafico SVG.

4.4.2. PHP

PHP es un lenguaje de código abierto muy popular, especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. Fue desarrollado específicamente para la programación del lado del servidor, y tiene muchas características integradas que no están disponibles en otros lenguajes de script. No se necesita usar configuraciones tediosas o características extrañas para que PHP funcione en un entorno web. Lo mejor de utilizar PHP es su extrema simplicidad, pero a su vez ofrece muchas características avanzadas para los programadores profesionales.

El PHP se incrusta mediante las siguientes etiquetas dentro del HTML:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Ejemplo</title>
    </head>
    <body>

        <?php
            echo "¡Hola, soy un script de PHP!";
        ?>

    </body>
</html>
```

***Figura 22**. Ejemplo para introducir PHP en HTML*

En la web se ha utilizado principalmente para obtener parámetros de los formularios, realizar los cálculos de todas las variables necesarias para la obtención del gráfico y

redireccionar a diferentes rutas. También nos ha permitido realizar bucles, comprobaciones de variables, imprimir código HTML y SVG usando valores de las variables, etc. En las siguientes imágenes podemos ver algunos de los ejemplos mencionados:

```
$y1_lineas = $y1 + $dist_entre_lineas;
$y1_numeros = $y1_lineas + 5;
$largo_lineas_horizontales = $x1 + $ancho_rectangulo*0.70;
$lineas_verticales = $y1 + $Ni*$dist_entre_lineas + $dist_entre_lineas_6;
$largo_linea3 = $largo_lineas_horizontales + $dist_entre_lineas_5;
$largo_linea4 = $y1_lineas;
$largo_linea5 = $x1 + $ancho_rectangulo + $dist_entre_lineas_7;
echo"<rect x='".$x1."' y='".$y1."' width='".$ancho_rectangulo."' height='".$altura
```

***Figura 23**. Ejemplo de uso de PHP para la realización de cálculos de las distancias*

```
for ( $s = 1; $s<=$No ; $s++){
    $numeros = $s + $No*($i-1);
    echo"<line x1='".$largo_lineas_horizontales."' y1='".$y3_lineas."' x2='".$
    echo"<line x1='".$largo_lineas_horizontales."' y1='".$lineas_verticales."'
    echo"<line x1='".$largo_linea3."' y1='".$lineas_verticales."' x2='".$largo_
    echo"<line x1='".$largo_linea3."' y1='".$largo_linea4."' x2='".$largo_line
    echo"<text x='".$x3_numeros."' y='".$y3_numeros."' >".$numeros."</text>";
```

***Figura 24**. Ejemplo de uso de PHP para imprimir código SVG*

Además, se ha utilizado para ejecutar el programa del simulador en Linux para obtener la lista de conexiones en función de los parámetros dados. Para ello mediante la función *system()* se ejecuta el comando con los parámetros introducidos, como observamos a continuación:

```
//Ejecutamos besa y leemos el resultado
if(isset($select)){
    system("besa -n ".$N." --ni ".$Ni." --no ".$No." --k ".$R2." --routing ".$select."
    > /tmp/resultado_pasos_simulacion.txt");
}

$archivo = file_get_contents("/tmp/resultado_pasos_simulacion.txt");
$conexiones = explode( "\n", $archivo);
```

Figura 25. Ejecución del simulador mediante PHP

Una vez obtenemos el archivo resultante, observamos la siguiente forma:

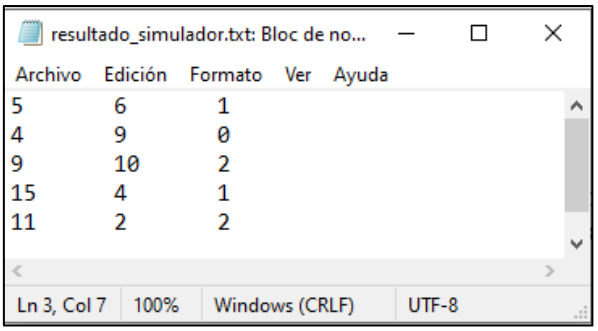


***Figura 26**. Ejemplo de archivo devuelto por el programa del simulador*

Mediante la función *explode()* de PHP conseguimos procesar el archivo fila por fila y columna por columna, teniendo así cada conexión por separado, y pudiendo obtener de cada conexión el valor que queramos.

4.4.3. CSS

CSS se ha utilizado en este proyecto para describir la presentación de documentos HTML y SVG. Este lenguaje trata de aplicar reglas a los elementos mediante selectores como el id del elemento, tipo de elemento, la clase, etc. Las reglas son de la forma "propiedad:valor" todas ellas separados por ";".

Las hojas de estilo se pueden aplicar de diferentes formas en la web:

- Forma interna: mediante un elemento <style> dentro del html.

```
<head>
...
<style type="text/css">
    elementoAfectadoPorElEstilo {
        propiedad:valor;
        propiedad2:valor;
    }
</style>
</head>
```

Figura 27. Ejemplo de aplicación de CSS de forma interna

- Forma externa: creamos un archivo .css donde se incluirán todos los estilos y luego se agregará a la web de la siguiente forma en el HTML

```
<head>
...
<link rel="stylesheet" type="text/css" href="rutaDelArchivo.css">
</head>
```

Figura 28. Ejemplo de aplicación de CSS de forma externa

- Forma en línea: se aplica individualmente a cada elemento, mediante el atributo style, de la siguiente forma:

```
<div style="margin-bottom:15px">
```

Figura 29. Ejemplo de aplicación de CSS de forma en línea

La forma en línea es la que se ha utilizado en la realización del proyecto en todos los elementos HTML como en los elementos SVG para darle el estilo a la web.

4.4.4. SVG

Como se ha mencionado en apartados anteriores, SVG se ha utilizado para la realización del gráfico de la red. Para ello se ha declarado una única etiqueta <svg></svg> para todo el gráfico. En ella es donde se introducen todas las figuras, y para su declaración es necesario especificar las dimensiones mediante las etiquetas *width* y *height.* Entre las figuras utilizadas se encuentran las siguientes:

- Líneas: se declaran mediante la etiqueta <line />. Para definir una línea hay que introducir las coordenadas del punto inicial y final (x1 y1 x2 y2), en pixeles o porcentaje. Si es en píxeles no hay que añadir *px* al valor. Además, mediante *stroke*, hay que asignarle un color para que la muestre. Se pueden añadir más propiedades como el grosor de línea, el tipo de línea o la forma de los extremos, pero solo se ha utilizado el grosor de la línea que se especifica mediante *Stroke-width*.

- Rectángulos: se declaran mediante la etiqueta <rect />. Para definirlo hay que introducir las coordenadas de donde se situará (x, y) y como en la etiqueta del svg, es necesario especificar las dimensiones mediante las etiquetas *width* y *height*. Se pueden utilizar otras propiedades, en nuestro caso se utiliza *fill* para rellenarlo.

- Círculos: se declaran mediante la etiqueta <circle />. Es necesario introducir el centro del círculo y su radio (cx, cy, r). Se pueden personalizar con los mismos parámetros *stroke* empleados para la línea. Para el relleno se utiliza la propiedad *fill* y para cambiar su opacidad *fill-opacity*, con valores entre 0 y 1, aunque esta última no se ha utilizado en el proyecto.

5. CONCLUSIONES

En conclusión, comentar que se han completado todos los objetivos del proyecto y se han abordado correctamente todos los pasos y procedimientos para la obtención del simulador. En versiones futuras, cabe la posibilidad de una mejora en la optimización de la Web y la adición de nuevas funcionalidades, para proporcionar a sus usuarios mayores conocimientos. Por ejemplo, respecto a la optimización se podría implementar el uso de VUE en el "front-end" ya que trae numerosos beneficios como que no se recargue la página cada vez que se presiona un botón, y respecto a nuevas funcionalidades, se podría añadir una que muestre en la web la reordenación de conexiones en caso de que no se pueda realizar con éxito una conexión, pero sí se pudiera reordenando.

La obtención de números aleatorios para el simulador ha llevado al estudio de diversos métodos, en concreto, el método de la inversa, que fue elegido debido a su relativa sencillez y porque es el más utilizado en la obtención de variables aleatorias para experimentos de simulación. Otro estudio, ha sido el lenguaje de programación del simulador. Se optó por usar lenguaje C debido a la flexibilidad, fácil aprendizaje y gran optimización. El desarrollo del código tuvo cierta complejidad por falta de familiarización con el lenguaje, pero se desarrolló satisfactoriamente todo lo necesario para el proyecto.

Respecto a la web, hay que comentar que el diseño ha sido muy laborioso, pero, sobre todo, el diseño del gráfico. Aunque debido a la facilidad de usar SVG en la realización de los gráficos, ha hecho que fuese más ameno, ya que es un lenguaje con una curva de aprendizaje sencilla. Se debatió bastante su elección, pero debido a que también no utilizaba ningún recurso externo fue la opción elegida, ya a que se pretendía que la web fuese lo más sencilla posible y con el menor número de vulnerabilidades respecto a la seguridad. PHP también ha sido muy importante en este proyecto, ya que todos los cálculos se han llevado a cabo a través de él. Se valoró la posibilidad de haber realizado la web con Frameworks y diversas tecnologías, pero como se ha comentado, debido al objetivo del proyecto, no eran estrictamente necesarios y, únicamente con PHP, se podían cumplir nuestras necesidades.

6. BIBLIOGRAFÍA

- Apuntes de la asignatura de Conmutación del segundo curso del Grado de Ingeniería telemática de la Universidad Politécnica de Cartagena. Sistemas de conmutación de circuitos.

- Telecommunications switching, traffic and networks. J. E. Flood, Editorial Pearson Education. ISBN 0-13-033309-3

- Apuntes de la asignatura de Arquitecturas Avanzadas del tercer curso del Grado de Ingeniería informática de la Universidad de Valladolid. Redes de interconexión.

- Campos, Oscar. «Introducción al elemento canvas de HTML5». Genbeta, 20 de junio de 2011. https://www.genbeta.com/desarrollo/introduccion-al-elemento-canvas-de-html5.

- «Canvas: gráficos y animaciones en HTML 5». Accedido 7 de marzo de 2020. https://www.aulaclic.es/articulos/canvas_1.html.

- «Cómo elegir Canvas o SVG para tus sitios Web». Accedido 7 de marzo de 2020. https://desarrolloweb.com/articulos/elegir-canvas-svg.html.

- Documentación web de MDN. «CSS». Accedido 7 de marzo de 2020. https://developer.mozilla.org/es/docs/Web/CSS.

- «Desarrollo de Aplicaciones WEB. Páginas web dinámicas. Rafael Barzanallana. Universidad de Murcia». Accedido 7 de marzo de 2020. https://www.um.es/docencia/barzana/DAWEB/2017-18/daweb-tema-13-paginas-web-dinamicas.html.

- «Gráficos vectoriales escalables». En Wikipedia, la enciclopedia libre, 13 de febrero de 2020. https://es.wikipedia.org/w/index.php?title=Gr%C3%A1ficos_vectoriales_escalables&oldid=123521747.

- Documentación web de MDN. «HTML». Accedido 7 de marzo de 2020. https://developer.mozilla.org/es/docs/Web/HTML.

- Documentación web de MDN. «JavaScript». Accedido 7 de marzo de 2020. https://developer.mozilla.org/es/docs/Web/JavaScript.

- Martínez, por Juan Manuel. «SVG en HTML: Crear dibujos vectoriales en la web». Covalenciawebs (blog), 24 de septiembre de 2015. https://covalenciawebs.com/crear-dibujos-vectoriales-directamente-en-la-web-svg-en-html/.

- CódigoFacilito. «Qué es HTML». Accedido 7 de marzo de 2020. https://codigofacilito.com/articulos/que-es-html.

- ¿Que es WebGL? «¿Que es WebGL?» Accedido 7 de marzo de 2020. http://ayudasprogramacionweb.blogspot.com/2012/08/que-es-webgl.html.

- «Scalable Vector Graphics (SVG) 1.1 (Second Edition)», s. f., 826. https://www.w3.org/TR/SVG11/REC-SVG11-20110816.pdf

- «SVG and Canvas and WebGL, Oh My». Accedido 7 de marzo de 2020. http://dataquarium.io/svg-canvas-webgl/.

- Documentación web de MDN. «Tutorial de SVG». Accedido 7 de marzo de 2020. https://developer.mozilla.org/es/docs/Web/SVG/Tutorial.

- CSS-Tricks. «Using SVG», 5 de marzo de 2013. https://css-tricks.com/using-svg/.

- «WebGL». En Wikipedia, la enciclopedia libre, 7 de marzo de 2020. https://es.wikipedia.org/w/index.php?title=WebGL&oldid=124073047.

- Documentación web de MDN. «What is JavaScript?» Accedido 7 de marzo de 2020. https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Qu%C3%A9_es_JavaScript.

- CSS-Tricks. «When to Use SVG vs. When to Use Canvas», 12 de noviembre de 2019. https://css-tricks.com/when-to-use-svg-vs-when-to-use-canvas/.

7. ANEXO

7.1. Código del simulador: besa.c

```
/* besa: Bloqueo En Sentido Amplio*/

#include <stdio.h>
#include <string.h>
#include <getopt.h>
#include <stdlib.h>
#include <math.h>
#include "rand.h"
#include "besa.h"

char *prgname;
char usage[]="%s --help shows the options list\n";


int main(int argc, char **argv)
{
    int numproves=1, matrix=0, verbose=0, trampa=0;
    struct switch_param sp;
    struct switch_state ss;
    struct switch_avg   sa;

    if (semillas(MAX_SEEDS) != 0) exit (-1);

    prgname = strrchr(argv[0], '/');
    if (prgname == (char *)NULL) prgname = argv[0];
    else prgname++;

    int i;
    for (i=0; i<N_MAX; i++) sa.connections[i] = 0;
    sa.con_avg = 0;
    sp.N  = 16;
    sp.r1 = sp.r2 = sp.r3 = 4;
    sp.ni = sp.no = 4;
    sp.ralg = 0;

    while(1){
        int c;
        static struct option long_options[] =
        {
          {"help", no_argument, NULL, 'h'},
          {"n", required_argument, NULL, 'n'},
          {"ni", required_argument, NULL, 'i'},
          {"no", required_argument, NULL, 'o'},
          {"k", required_argument, NULL, 'k'},
          {"routing", required_argument, NULL, 'r'},
          {"proves", required_argument, NULL, 'p'},
          {"matrix", no_argument, NULL, 'm'},
          {"verbose", no_argument, NULL, 'v'},
          {"trampa", no_argument, NULL, 't'},
          {"ib", no_argument, NULL, 'b'},
          {0,   0,                  0,     0}
       };
       c = getopt_long(argc,argv,"hi:o:k:n:p:r:tmvb", long_options, (int *)0);

       if (c == EOF) break;

       switch(c){
          case 'h': printf("%s [options], available options are:\n", prgname);
                    fputs("-h  Help\n", stdout);
                    fputs("-n switch size (default 16)\n", stdout);
                    fputs("--ni input size (default 4)\n", stdout);
```

```
      fputs("--no output siz (default 4)e\n", stdout);
      fputs("-k number of middle-stage switches (default 4)\n", stdout);
      fputs("-r routing algorithm (default a; sequential)\n", stdout);
      fputs("-p number of proves (default 1)\n", stdout);
      fputs("-m matrix representation\n", stdout);
      exit(0);
  case 'n': sp.N = atoi(optarg);
            break;
  case 'i': sp.ni = atoi(optarg);
            break;
  case 'o': sp.no = atoi(optarg);
            break;
  case 'k': sp.r2  = atoi(optarg);
            break;
  case 'p': numproves = atoi(optarg);
            break;
  case 'r': if (*optarg == 'a') sp.ralg = 0;
            else if (*optarg == 'b') sp.ralg = 1;
            else if (*optarg == 'c') sp.ralg = 2;
            else sp.ralg = atoi(optarg);
            break;
  case 'm': matrix = 1;
            break;
  case 'v': verbose = 1;
            break;
  case 't': trampa = 1;
            break;
  case 'b': trampa = 1;
            break;
  default : fprintf(stderr, usage, prgname);
            exit(1);
   }
}
if (check_opt(sp) != 0) exit(-1);

sp.r1 = sp.N/sp.ni;
sp.r3 = sp.N/sp.no;

int prove;
for (prove = 1; prove <= numproves; prove++){
    init_ss(&ss);
    atrandom(sp, ss.in, 0);
    atrandom(sp, ss.out, sp.N);
    setmstage(sp, &ss);
    setconnections(sp, ss, &sa);
}
setavg(sp, &sa);

if (!matrix && numproves == 1){
   if (verbose) {
       printf("Input\tOutput\tMiddle stage\n");
       for (i=0; i<sp.N; i++){
        if (ss.mstage[i] == BLCK) {
           printf("%3d\t%4d\t    B\n", ss.in[i], ss.out[i]);
           break;
        }
        else printf("%3d\t%4d\t%5d\n", ss.in[i], ss.out[i], ss.mstage[i]);
       }
   }
   else {
       for (i=0; i<sp.N; i++){
          printf("%d\t%d\t%d\n", ss.in[i], ss.out[i], ss.mstage[i]);
          if (ss.mstage[i] == BLCK) break;
       }
   }
}

if (matrix && numproves == 1){
```

```
        int m[sp.N*sp.N];
        int row, col;

        for (i=0; i<sp.N*sp.N; i++) m[i] = EMPTY;

        for (i=0; i<sp.N; i++){
            row = ss.in[i];
            col = ss.out[i];
            m[row*sp.N+col] = ss.mstage[i];
            if (ss.mstage[i] == BLCK) break;
        }
        print_matrix(m, sp);
    }

    if (numproves > 1) {
        int num_c=0, num_b=0;
        printf("\nDistribution of connection successfully made\n");
        for (i=1; i<=sp.N; i++) {
            printf("%d\t%6d (%3.2lf%%)\n", i, sa.connections[i],
100*(double)sa.connections[i]/(double)numproves);
            num_c += i*sa.connections[i];
            if (i != sp.N) num_b += sa.connections[i];
            }
        printf("\n");
        printf("Avarage number of successful connections: %.2lf\n\n",
sa.con_avg/(double)numproves);
        if (trampa) printf("\nIB = %0.6lf\n", (double)num_b/(double)(num_c +
num_b));
    }
    return(0);
}
```

7.2. Código del simulador: besa.h

```
#ifndef _BESA_H_
#define _BESA_H_ 1

#define N_MAX 64
#define BLCK -1
#define EMPTY -2

enum ralg {SEQUENTIAL, CYCLICAL, RANDOM};

struct switch_param
{
  int N;
  int r1, r2, r3;
  int ni, no;
  int ralg;
};


struct switch_state
{
  int in[N_MAX];
  int out[N_MAX];
  int mstage[N_MAX];
};


struct switch_avg
{
   int connections[N_MAX+1];
   double con_avg;
};


int check_opt(struct switch_param sp);
void init_ss(struct switch_state *ss);
void atrandom(struct switch_param sp, int *matrix, int seed);
void setmstage(struct switch_param sp, struct switch_state *ss);
int sequencial(struct switch_param sp, int *fms);
int cyclical(struct switch_param sp, int *fms);
int randomly(struct switch_param sp, int *fms);
void setconnections(struct switch_param sp, struct switch_state ss, struct
switch_avg *sa);
void setavg(struct switch_param sp, struct switch_avg *sa);
void print_matrix(int *m, struct switch_param sp);

#endif
```

7.3. Código del simulador: besa_func.c

```
#include "besa.h"
#include "rand.h"
#include <stdio.h>

int check_opt(struct switch_param sp)
{
   if (sp.N > N_MAX) {printf("switch size must be lower or equal to %d\n",
N_MAX); return(-1);}
   if (sp.N % sp.ni != 0) {printf("switch size must be divisible by %d
(ni)\n", sp.ni); return(-1);}
   if (sp.N % sp.no != 0) {printf("switch size must be divisible by %d
(no)\n", sp.no); return(-1);}
   if (sp.ralg > 2 && sp.ralg < 0 ) {printf("routing algorithm should be 0, 1
or 2\n"); return(-1);}

   return 0;
}

void init_ss(struct switch_state *ss)
{
   int i;
   for (i=0; i<N_MAX; i++) ss->in[i] = ss->out[i] = ss->mstage[i] = EMPTY;
}


void atrandom(struct switch_param sp, int *matrix, int seed)
{
   int i, j, max, value[N_MAX], index;

   for (i=0; i<sp.N; i++) value[i] = i;
   for (i=0; i<sp.N; i++){
       max = sp.N-(i+1);
       index = uniforme(0, max, seed+i);
       matrix[i] = value[index];
       if (index != max) for (j=index; j<max; j++) value[j] = value[j+1];
   }
}


void setmstage(struct switch_param sp, struct switch_state *ss)
{
   int i, j, k, available[N_MAX];
   int (*ptr_ralg)();

   for (i=0; i<sp.N; i++){
    for (k=0; k<sp.r2; k++) available[k] = k;
    for (j=0; j<i; j++){
      if (ss->in[j]/sp.ni == ss->in[i]/sp.ni)   available[ss->mstage[j]]=BLCK;
      if (ss->out[j]/sp.no == ss->out[i]/sp.no) available[ss->mstage[j]]=BLCK;
    }
    switch (sp.ralg){
           case SEQUENTIAL: ptr_ralg = sequencial; break;
           case CYCLICAL  : ptr_ralg = cyclical; break;
           case RANDOM    : ptr_ralg = randomly; break;
    }

    ss->mstage[i] = (*ptr_ralg)(sp, available);
    if (ss->mstage[i] == BLCK) return;
  }
}
```

```
int sequencial(struct switch_param sp, int *fms)
{
   int i;

   for (i=0; i<sp.r2; i++) if(fms[i] != BLCK) return fms[i];

   return BLCK;
}


int cyclical(struct switch_param sp, int *fms)
{
   int i, last, value;
   static int cyclic=-1;

   value = BLCK;

   for (i=0; i<sp.r2; i++) {
       cyclic = (cyclic + 1) % sp.r2;
       if (i == 0) last = cyclic;
       if(fms[cyclic] != BLCK) {
          value = fms[cyclic];
          cyclic = last;
          break;
       }
   }
   return value;
}


int randomly(struct switch_param sp, int *fms)
{
   int i, numfree=0, rmd, cont=1;

   for (i=0; i<sp.r2; i++) if(fms[i] != BLCK) numfree++;
   rmd = uniforme(1, numfree, 2*sp.N);
   for (i=0; i<sp.r2; i++){
       if(fms[i] != BLCK){
          if(cont == rmd) return fms[i];
          cont++;
       }
  }

  return BLCK;
}


void setconnections(struct switch_param sp, struct switch_state ss, struct
switch_avg *sa)
{
   int i;

   for (i=0; i<sp.N; i++){
       if (ss.mstage[i] == BLCK) {sa->connections[i]++; return;}
   }
   sa->connections[sp.N]++;

}

void setavg(struct switch_param sp, struct switch_avg *sa)
{
   int i;

   for (i=1; i<=sp.N; i++) sa->con_avg += sa->connections[i]*i;
}
```

```
void print_matrix(int *m, struct switch_param sp)
{
   int i, j;

   printf("%2c",' ');
   for(i=0; i< sp.N; i++) printf(" %3d ", i);
   printf("\n");
   printf("%2c",' ');
   for(i=0; i< sp.N; i++) printf("|----");
   printf("|\n");

   for(i=0; i< sp.N; i++) {
      printf("%2d", i);
      for (j=0; j<sp.N; j++){
          if (m[i*sp.N+j] == EMPTY) printf("|%4c", ' ');
          else {
             if(m[i*sp.N+j] != BLCK) printf("|%3d ", m[i*sp.N+j]);
             else printf("|  B ");
          }
      }
   printf("|\n");
   printf("%2c",' ');
   for(j=0; j< sp.N; j++) printf("|----");
   printf("|\n");
   }
}
```

7.4. Código de la web: index.php

```
<!DOCTYPE html>
<html style="min-height: 100%; position: relative">
  <head>
    <meta charset = "utf-8">
    <meta name="escala" content="width=device-width, initial-scale=1.0">
    <!--Definimos la escala de la web -->
    <title>Simulador de redes Clos</title>
    <!--El título de nuestra web -->
    <!-- Recursos externos -->
    <link rel="icon" type="image/x-icon" href="imagenes/ICOFX.png" />
    <link rel="stylesheet" href="css/bootstrap.css">
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link rel="stylesheet" href="css/bootstrap-grid.css">
    <link rel="stylesheet" href="css/bootstrap-grid.min.css">
    <link rel="stylesheet" href="css/bootstrap-reboot.css">
    <link rel="stylesheet" href="css/bootstrap-reboot.min.css">
  </head>
  <body class="overflow-auto" style="min-height:100%; width:100%;
overflow:auto;">
    <div style="position:fixed; width:100%; z-index:1000; background-
color:white; zoom:79%">
      <nav class="navbar navbar-expand-md navbar-dark bg-dark d-block" style="
width:100%; max-height:70px; position:fixed">
        <h4 style="text-align: center; color: white; width: 100%; height:100%;
white-space: nowrap; overflow: hidden; text-overflow: ellipsis;"> Simulador de
redes de conmutación de tres etapas</h4>
      </nav>
    </div>
    <div>
    <div style="display:flex; zoom:90%" >
      <div class="row w-100">
        <div class="col-xs-4 col-sm-4 col-md-4 col-lg-3 col-xl-2 d-inline"
style="z-index:500">
          <div style="float:left; margin-top:60px; min-width:100%; margin-
bottom: 60px;">
            <h4 style="margin-left:40px; margin-top:40px"> Parámetros a
rellenar </h4>
            <form action="procesar.php?" method="GET" style="margin-left:40px;
margin-top:30px;width: 290px;">
              <?php
                if(isset($_GET['N'], $_GET['Ni'], $_GET['No'], $_GET['R2'])){
                $N = $_GET['N'];
                $Ni = $_GET['Ni'];
                $No = $_GET['No'];
                $R2 = $_GET['R2'];
                echo '<b style="float:left">N:</b><input name="N" class="form-
control" type="number" min="1" style="margin-bottom:15px; text-align:right;
float:left" value="'.$N.'" required><br>';
                echo '<b style="float:left">Ni:</b><input name="Ni"
class="form-control" type="number" min="1" style="margin-bottom:15px; text-
align:right; float:left" value="'.$Ni.'" required><br>';
                echo '<b style="float:left">No:</b><input name="No"
class="form-control" type="number" min="1" style="margin-bottom:15px; text-
align:right; float:left" value="'.$No.'" required><br>';
                echo '<b style="float:left">R2:</b><input name="R2"
class="form-control" type="number" min="1" style="margin-bottom:15px; text-
align:right; float:left " value="'.$R2.'" required><br>';
                }else{
                echo '<b style="float:left">N:</b><input name="N" class="form-
control" type="number" min="1" style="margin-bottom:15px; text-align:right;
float:left" required><br>';
                echo '<b style="float:left">Ni:</b><input name="Ni"
class="form-control" type="number" min="1" style="margin-bottom:15px; text-
align:right; float:left" required><br>';
```

```
                echo '<b style="float:left">No:</b><input name="No"
class="form-control" type="number" min="1" style="margin-bottom:15px; text-
align:right; float:left" required><br>';
                echo '<b style="float:left">R2:</b><input name="R2"
class="form-control" type="number" min="1" style="margin-bottom:15px; text-
align:right; float:left " required><br>';
                }
                ?>
              <b style="float:left">Routing:</b>
              <select name="select" class="form-control" style="margin-
bottom:20px; text-align:right; float:left " required>
                <option value="0"> A </option>
                <option value="1"> B </option>
                <option value="2"> C </option>
              </select>
              <br>
              <b style="float:left">Activar puntos de cruce</b><input
name="PC" type="checkbox" class="d-block" style="margin-bottom:35px; margin-
right: 30px; text-align:right; margin-left:10px;margin-top:5px; float:left"
value="activar" checked>
              </br>
              <input value="Empezar" class="btn d-block" type="submit"
style="width:100px; float:none;margin-right: 250px;">
              <input name="paso" class="form-control" type="hidden" value="0">
              <?php
                if (!empty($_GET['R'])) {
                    echo '<div class="alert alert-danger" style="margin-
top:35px">';
                    echo '<strong>Datos erroneos!</strong>';
                    echo '</div>';
                }
                ?>
            </form>
          </div>
        </div>
        <div class="col-xl-1 col-lg-1 col-md-2 col-sm-0"></div>
        <div class="col-sm-8 col-md-6 col-lg-8 col-xl-9 d-table"
style="margin-top: 10%;">
          <img src="./imagenes/red.png" style="margin-top:80px;float:left;
margin-bottom:100px; margin-left: 4%; margin-right: 4%; width: 92%; min-
width:50%;"/>
        </div>
      </div>
    </div>
    <footer class="container-fluid text-center" style="bottom:0px ;
background-color: #555;width:100%; color: white; padding: 7px;position:fixed;
max-height:50px; z-index:999 ; zoom:79%">
      <p class="d-inline">Universidad Politécnica de Cartagena</p>
      <img class="d-inline" width="40" height="40" style="margin-left:10px"
src="imagenes/upct_icon.png"/>
    </footer>
  </body>
</html>
```

7.5. Código de la web: procesar.php

```
<!DOCTYPE html>
<html style="min-height: 100%; position: relative; min-
width:100%;display:flex">
  <head>
    <meta charset = "utf-8">
    <!--Definimos la fuente -->
    <meta name="escala"  content="width=device-width, initial-scale=1.0">
    <!--Definimos la escala de la web -->
    <title>Simulador de redes Clos</title>
    <!--El título de nuestra web -->
    <!-- Recursos externos -->
    <link rel="icon" type="image/x-icon" href="imagenes/ICOFX.png" />
    <link rel="stylesheet" href="css/bootstrap.css">
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link rel="stylesheet" href="css/bootstrap-grid.css">
    <link rel="stylesheet" href="css/bootstrap-grid.min.css">
    <link rel="stylesheet" href="css/bootstrap-reboot.css">
    <link rel="stylesheet" href="css/bootstrap-reboot.min.css">
    <!-- Script -->
    <script>
      function adjustTable(){
          var table = document.getElementById('table-scroll');
          var svg = document.getElementById('svg');
          var h = document.documentElement.clientHeight;
          table.scrollTop = '9999';

          if (h < svg.height.animVal.value){
              var alt = Math.trunc(svg.height.animVal.value) - 200;
              table.style.maxHeight = alt+"px";
          }
      }
    </script>
  </head>
  <body onload="adjustTable()" class="overflow-auto" style="min-
height:100%;width:100%; zoom:79%">
    <!-- Parte superior Web y obtención de parámetros-->
    <div style="position:fixed; width:100%; z-index:4; background-
color:white;">
      <!-- Navbar-->
      <div style="margin-bottom:15px">
        <nav class="navbar navbar-expand-md navbar-dark bg-dark d-block"
style=" width:100%;max-height:70px; position:fixed; white-space: nowrap;">
          <h4 style="text-align: center; color: white; width:100%;
height:100%; white-space: nowrap; overflow: hidden; text-overflow: ellipsis;">
Simulador de redes de conmutación de tres etapas</h4>
        </nav>
      </div>
      <!-- Parámetros y botones -->
      <div style="margin-top:64px;">
        <?php
          //Obtención de parámetros
          $N = $_GET['N'];
          $Ni = $_GET['Ni'];
          $No = $_GET['No'];
          $R2 = $_GET['R2'];
          $pasos = $_GET['paso'];
          $select = $_GET['select'];

          //Botón nueva simulación
          echo "<a role='button' class='btn btn-primary btn-md'
href='index.php?N=".$N."&Ni=".$Ni."&No=".$No."&R2=".$R2."' style=' margin-
bottom:0px; bottom:0px; margin-left:10px'>Nueva simulacion</a>";

          //Comprueba si los parametros cumplen las especificaciones
```

```
          $resultado_division = $N / $Ni;
          $resultado_division_2 = $N / $No;

          if(!is_int($resultado_division) || !is_int($resultado_division_2)){
              header("Location: index.php?R=1&N=$N&Ni=$Ni&No=$No&R2=$R2");
          }else{
              if (empty($_GET['PC'])) {
                  header("Location:
procesar_sincruces.php?N=$N&Ni=$Ni&No=$No&R2=$R2&select=$select&paso=0");
              }
          }

          //Ejecutamos besa y leemos el resultado
          if(isset($select)){
              system("besa -n ".$N." --ni ".$Ni." --no ".$No." --k ".$R2." --
routing ".$select." > /tmp/resultado_pasos_simulacion.txt");
          }

          $archivo = file_get_contents("/tmp/resultado_pasos_simulacion.txt");
          $conexiones = explode( "\n", $archivo);

          //Regulamos los pasos
          $paso_siguiente = $pasos + 1;

          if($pasos != 0 ){
          $paso_anterior = $pasos - 1;
          }else{
          $paso_anterior = $pasos;
          }

          if($pasos != (sizeof($conexiones)-1)){
          $paso_siguiente = $pasos + 1;
          }else{
          $paso_siguiente = $pasos;
          }

          //Botones paso anterior y siguiente
          echo "<a role='button' class='btn btn-primary btn-md'
href='procesar.php?N=".$N."&Ni=".$Ni."&No=".$No."&R2=".$R2."&PC=activar&paso="
.$paso_siguiente."' style='margin-bottom:0px; margin-right:10px; margin-
left:5px; float:right' >Paso siguiente</a>";
          echo "<a role='button' class='btn btn-primary btn-md'
href='procesar.php?N=".$N."&Ni=".$Ni."&No=".$No."&R2=".$R2."&PC=activar&paso="
.$paso_anterior."' style='margin-bottom:0px;float:right' >Paso anterior</a>";
          ?>
      </div>
      <hr style="width: 92%; margin-left: 4% !important; margin-right: 4%
!important; margin-top: 8px; margin-bottom: 0px;">
    </div>
    <!-- Calculo de variables y parametros utilizados en la simulación -->
    <?php
      $etapas_iniciales = $N / $Ni;
      $etapas_finales = $N / $No;

      $distancia_etapas_consecutivas = 30;
      $distancia_bordes_verticales = 25;
      $distancia_comienzo_etapas = 80;
      $distancia_entre_etapas_inicial_intermedia = 150;
      $distancia_entre_etapas_intermedia_final = 150;
      $dist_entre_lineas = 18;
      $dist_entre_lineas_2 = 18;

      if ( $R2 >= 7 || $Ni<3 ){
       $dist_entre_lineas_3 = 18;
      }else{
       $dist_entre_lineas_3 = 10;
      }
```

```
      if ( $etapas_finales >=7 || $etapas_iniciales<3 ){
       $dist_entre_lineas_3_2 = 18;
      }else{
       $dist_entre_lineas_3_2 = 10;
      }

      if ( $No >=7 || $R2<3){
       $dist_entre_lineas_3_3 = 18;
      }else{
       $dist_entre_lineas_3_3 = 10;
      }

      $dist_entre_lineas_4 = 10;
      $dist_entre_lineas_5 = 10;
      $dist_entre_lineas_6 = 10;
      $dist_entre_lineas_7 = 7.5;
      $dist_entre_lineas_8 = 21;

      $altura_rect = $dist_entre_lineas*($Ni + 1)+ ($dist_entre_lineas_3 -
0.18)*$R2;
      $ancho_rectangulo = $R2*40+37;

      $altura_rect_medio = $dist_entre_lineas*($etapas_iniciales + 1)+
($dist_entre_lineas_3_2 - 0.18)*$etapas_finales;
      $ancho_rectangulo_medio = $etapas_finales*40+37;

      $altura_rect_final = $dist_entre_lineas*($R2 + 1)+
($dist_entre_lineas_3_3 - 0.18)*$No;
      $ancho_rectangulo_final = $No*40+37;


      $x1 = $distancia_comienzo_etapas;
      $x2 = $x1 + $ancho_rectangulo +
$distancia_entre_etapas_inicial_intermedia;
      $x3 = $x2 + $ancho_rectangulo_medio +
$distancia_entre_etapas_intermedia_final;

      $x1_lineas = $x1 - $dist_entre_lineas_7;
      $x2_lineas = $x2 - $dist_entre_lineas_7;
      $x3_lineas = $x3 - $dist_entre_lineas_7;

      $x1_numeros = $x1 - $dist_entre_lineas_7 - $dist_entre_lineas_8;

      $x3_numeros = $x3 + $ancho_rectangulo_final + $dist_entre_lineas_8 - 8;

      $ancho_svg = $x3 + $ancho_rectangulo_final + $distancia_comienzo_etapas;
      $altura_svg;
      ?>
    <!-- Contenido de la Web -->
    <div style="display:flex; margin-bottom:60px; width:100%; margin-
top:111px; z-index:1">
      <!-- SVG -->
      <div  style=" min-height:100%; float:left">
        <?php
          //DECLARAMOS EL SVG
          $altura_etapas_iniciales = $distancia_bordes_verticales*2 +
$altura_rect*$etapas_iniciales +
$distancia_etapas_consecutivas*($etapas_iniciales - 1);
          $altura_etapas_intermedias = $distancia_bordes_verticales*2 +
$altura_rect_medio*$R2 + $distancia_etapas_consecutivas*($R2 - 1);
          $altura_etapas_finales = $distancia_bordes_verticales*2 +
$altura_rect_final*$etapas_finales +
$distancia_etapas_consecutivas*($etapas_finales - 1);

          if($altura_etapas_iniciales >= $altura_etapas_intermedias ) {
          if($altura_etapas_iniciales > $altura_etapas_finales ){
          $y1 = $distancia_bordes_verticales;
          $y1_para_lineas = $y1;
```

```
          $y2 = ($altura_etapas_iniciales/2)-(($altura_etapas_intermedias -
($distancia_bordes_verticales*2))/2);
          $y2_para_lineas = $y2;
          $y3 = ($altura_etapas_iniciales/2)-(($altura_etapas_finales -
($distancia_bordes_verticales*2))/2);
          $y3_para_lineas = $y3;
          $altura_svg = $altura_etapas_iniciales;
          echo "<svg id='svg' x='0' y='0' width='".$ancho_svg."'
height='".$altura_svg."'>";
          }else{
          $y3 = $distancia_bordes_verticales;
          $y3_para_lineas = $y3;
          $y2 = ($altura_etapas_finales/2)-(($altura_etapas_intermedias -
($distancia_bordes_verticales*2))/2);
          $y2_para_lineas = $y2;
          $y1 = ($altura_etapas_finales/2)-(($altura_etapas_iniciales -
($distancia_bordes_verticales*2))/2);
          $y1_para_lineas = $y1;
          $altura_svg = $altura_etapas_finales;
          echo "<svg id='svg' x='0' y='0' width='".$ancho_svg."'
height='".$altura_svg."'>";
          }
          }else{
          if($altura_etapas_intermedias > $altura_etapas_finales ){
          $y2 = $distancia_bordes_verticales;
          $y2_para_lineas = $y2;
          $y1 = ($altura_etapas_intermedias/2)-(($altura_etapas_iniciales -
($distancia_bordes_verticales*2))/2);
          $y1_para_lineas = $y1;
          $y3 = ($altura_etapas_intermedias/2)-(($altura_etapas_finales -
($distancia_bordes_verticales*2))/2);
          $y3_para_lineas = $y3;
          $altura_svg = $altura_etapas_intermedias;
          echo "<svg id='svg' x='0' y='0' width='".$ancho_svg."'
height='".$altura_svg."'>";
          }else{
          $y3 = $distancia_bordes_verticales;
          $y3_para_lineas = $y3;
          $y1 = ($altura_etapas_finales/2)-(($altura_etapas_iniciales -
($distancia_bordes_verticales*2))/2);
          $y1_para_lineas = $y1;
          $y2 = ($altura_etapas_finales/2)-(($altura_etapas_intermedias -
($distancia_bordes_verticales*2))/2);
          $y2_para_lineas = $y2;
          $altura_svg = $altura_etapas_finales;
          echo "<svg id='svg' x='0' y='0' width='".$ancho_svg."'
height='".$altura_svg."'>";
          }
          }

          // GENERACION RECTANGULO ETAPAS INICIALES
          for ( $i = 1; $i<= $etapas_iniciales; $i++){
          $y1_lineas = $y1 + $dist_entre_lineas;
          $y1_numeros = $y1_lineas + 5;
          $largo_lineas_horizontales = $x1 + $ancho_rectangulo*0.70;
          $lineas_verticales = $y1 + $Ni*$dist_entre_lineas +
$dist_entre_lineas_6;
          $largo_linea3 = $largo_lineas_horizontales + $dist_entre_lineas_5;
          $largo_linea4 = $y1_lineas;
          $largo_linea5 = $x1 + $ancho_rectangulo + $dist_entre_lineas_7;
          echo"<rect x='".$x1."' y='".$y1."' width='".$ancho_rectangulo."'
height='".$altura_rect."' fill='rgb(180, 180, 180)'/>";
          for ( $q = 1; $q<=$Ni; $q++){
          $numeros = $q + $Ni*($i-1);
          if ( $numeros <= 9 ){
          $x1_numeros = $x1 - $dist_entre_lineas_7 - $dist_entre_lineas_8 + 8;
          }else{
          $x1_numeros = $x1 - $dist_entre_lineas_7 - $dist_entre_lineas_8;
```

```
          }
          echo"<line x1='".$x1_lineas."' y1='".$y1_lineas."'
x2='".$largo_lineas_horizontales."' y2='".$y1_lineas."' style='stroke:#000;
stroke-width:0.5mm' />";
          echo"<text x='".$x1_numeros."' y='".$y1_numeros."'
>".$numeros."</text>";
          $y1_lineas = $y1_lineas + $dist_entre_lineas;
          $y1_numeros = $y1_numeros + $dist_entre_lineas;
          }

          $y1_lineas = $y1 + $dist_entre_lineas;

          for ( $s = 1; $s<=$R2 ; $s++){

          echo"<line x1='".$largo_lineas_horizontales."' y1='".$y1_lineas."'
x2='".$largo_lineas_horizontales."' y2='".$lineas_verticales."'
style='stroke:#000; stroke-width:0.5mm' />";
          echo"<line x1='".$largo_lineas_horizontales."'
y1='".$lineas_verticales."' x2='".$largo_linea3."' y2='".$lineas_verticales."'
style='stroke:#000; stroke-width:0.5mm' />";
          echo"<line x1='".$largo_linea3."' y1='".$lineas_verticales."'
x2='".$largo_linea3."' y2='".$largo_linea4."' style='stroke:#000; stroke-
width:0.5mm' />";
          echo"<line x1='".$largo_linea3."' y1='".$largo_linea4."'
x2='".$largo_linea5."' y2='".$largo_linea4."' style='stroke:#000; stroke-
width:0.5mm' />";
          $lineas_verticales = $lineas_verticales + $dist_entre_lineas_3;
          $largo_lineas_horizontales = $largo_lineas_horizontales -
$dist_entre_lineas_2;
          $largo_linea3 = $largo_linea3 + $dist_entre_lineas_4;
          $largo_linea4 = $largo_linea4 + $dist_entre_lineas;
          }
          $y1 = $y1 + $altura_rect + $distancia_etapas_consecutivas;
          }

          // GENERACION RECTANGULO ETAPAS INTERMEDIAS
          for ( $i = 1; $i<= $R2; $i++){
          $y2_lineas = $y2 + $dist_entre_lineas;
          $largo_lineas_horizontales = $x2 + $ancho_rectangulo_medio*0.70;
          $lineas_verticales = $y2 + $etapas_iniciales*$dist_entre_lineas +
$dist_entre_lineas_6;
          $largo_linea3 = $largo_lineas_horizontales + $dist_entre_lineas_5;
          $largo_linea4 = $y2_lineas;
          $largo_linea5 = $x2 + $ancho_rectangulo_medio +
$dist_entre_lineas_7;
          echo"<rect x='".$x2."' y='".$y2."'
width='".$ancho_rectangulo_medio."' height='".$altura_rect_medio."'
fill='rgb(180, 180, 180)'/>";
          for ( $q = 1; $q<=$etapas_iniciales; $q++){
          echo"<line x1='".$x2_lineas."' y1='".$y2_lineas."'
x2='".$largo_lineas_horizontales."' y2='".$y2_lineas."' style='stroke:#000;
stroke-width:0.5mm' />";
          $y2_lineas = $y2_lineas + $dist_entre_lineas;

          }

          $y2_lineas = $y2 + $dist_entre_lineas;

          for ( $s = 1; $s<=$etapas_finales ; $s++){
          echo"<line x1='".$largo_lineas_horizontales."' y1='".$y2_lineas."'
x2='".$largo_lineas_horizontales."' y2='".$lineas_verticales."'
style='stroke:#000; stroke-width:0.5mm' />";
          echo"<line x1='".$largo_lineas_horizontales."'
y1='".$lineas_verticales."' x2='".$largo_linea3."' y2='".$lineas_verticales."'
style='stroke:#000; stroke-width:0.5mm' />";
          echo"<line x1='".$largo_linea3."' y1='".$lineas_verticales."'
x2='".$largo_linea3."' y2='".$largo_linea4."' style='stroke:#000; stroke-
width:0.5mm' />";
```

```
          echo"<line x1='".$largo_linea3."' y1='".$largo_linea4."'
x2='".$largo_linea5."' y2='".$largo_linea4."' style='stroke:#000; stroke-
width:0.5mm' />";
          $lineas_verticales = $lineas_verticales + $dist_entre_lineas_3_2;
          $largo_lineas_horizontales = $largo_lineas_horizontales -
$dist_entre_lineas_2;
          $largo_linea3 = $largo_linea3 + $dist_entre_lineas_4;
          $largo_linea4 = $largo_linea4 + $dist_entre_lineas;
          }
          $y2 = $y2 + $altura_rect_medio + $distancia_etapas_consecutivas;
          }

          // GENERACION RECTANGULO ETAPAS FINALES
          for ( $i = 1; $i<= $etapas_finales; $i++){
          $y3_lineas = $y3 + $dist_entre_lineas;
          $y3_numeros = $y3_lineas + 5;
          $largo_lineas_horizontales = $x3 + $ancho_rectangulo_final*0.70;
          $lineas_verticales = $y3 + $R2*$dist_entre_lineas +
$dist_entre_lineas_6;
          $largo_linea3 = $largo_lineas_horizontales + $dist_entre_lineas_5;
          $largo_linea4 = $y3_lineas;
          $largo_linea5 = $x3 + $ancho_rectangulo_final +
$dist_entre_lineas_7;
          echo"<rect x='".$x3."' y='".$y3."'
width='".$ancho_rectangulo_final."' height='".$altura_rect_final."'
fill='rgb(180, 180, 180)'/>";
          for ( $q = 1; $q<=$R2; $q++){
          echo"<line x1='".$x3_lineas."' y1='".$y3_lineas."'
x2='".$largo_lineas_horizontales."' y2='".$y3_lineas."' style='stroke:#000;
stroke-width:0.5mm' />";
          $y3_lineas = $y3_lineas + $dist_entre_lineas;

          }

          $y3_lineas = $y3 + $dist_entre_lineas;

          for ( $s = 1; $s<=$No ; $s++){
          $numeros = $s + $No*($i-1);
          echo"<line x1='".$largo_lineas_horizontales."' y1='".$y3_lineas."'
x2='".$largo_lineas_horizontales."' y2='".$lineas_verticales."'
style='stroke:#000; stroke-width:0.5mm' />";
          echo"<line x1='".$largo_lineas_horizontales."'
y1='".$lineas_verticales."' x2='".$largo_linea3."' y2='".$lineas_verticales."'
style='stroke:#000; stroke-width:0.5mm' />";
          echo"<line x1='".$largo_linea3."' y1='".$lineas_verticales."'
x2='".$largo_linea3."' y2='".$largo_linea4."' style='stroke:#000; stroke-
width:0.5mm' />";
          echo"<line x1='".$largo_linea3."' y1='".$largo_linea4."'
x2='".$largo_linea5."' y2='".$largo_linea4."' style='stroke:#000; stroke-
width:0.5mm' />";
          echo"<text x='".$x3_numeros."' y='".$y3_numeros."'
>".$numeros."</text>";
          $lineas_verticales = $lineas_verticales + $dist_entre_lineas_3_3;
          $largo_lineas_horizontales = $largo_lineas_horizontales -
$dist_entre_lineas_2;
          $largo_linea3 = $largo_linea3 + $dist_entre_lineas_4;
          $largo_linea4 = $largo_linea4 + $dist_entre_lineas;
          $y3_numeros = $y3_numeros + $dist_entre_lineas;
          }
          $y3 = $y3 + $altura_rect_final + $distancia_etapas_consecutivas;
          }

          //Dibujar lineas entre etapas iniciales e intermedias

          for( $i = 1; $i<=$etapas_iniciales; $i++){
          $x1_lineas_intermedias = $x1 + $ancho_rectangulo +
$dist_entre_lineas_7;
```

```
          $y1_lineas_intermedias = $y1_para_lineas + ($altura_rect +
$distancia_etapas_consecutivas)*($i - 1) + $dist_entre_lineas ;
          $x2_lineas_intermedias = $x2 - $dist_entre_lineas_7;
          $y2_lineas_intermedias = $y2_para_lineas + $dist_entre_lineas*$i;

          for( $j = 1; $j<=$R2 ; $j++){
          echo"<line x1='".$x1_lineas_intermedias."'
y1='".$y1_lineas_intermedias."' x2='".$x2_lineas_intermedias."'
y2='".$y2_lineas_intermedias."' style='stroke:#000; stroke-width:0.5mm' />";
          $y1_lineas_intermedias = $y1_lineas_intermedias +
$dist_entre_lineas;
          $y2_lineas_intermedias = $y2_lineas_intermedias + $altura_rect_medio
+ $distancia_etapas_consecutivas;
          }
          }

          //Dibujar lineas entre etapas intermedias y finales

          for( $i = 1; $i<=$R2; $i++){
          $x1_lineas_intermedias = $x2 + $ancho_rectangulo_medio +
$dist_entre_lineas_7;
          $y1_lineas_intermedias = $y2_para_lineas + ($altura_rect_medio +
$distancia_etapas_consecutivas)*($i - 1) + $dist_entre_lineas ;
          $x2_lineas_intermedias = $x3 - $dist_entre_lineas_7;
          $y2_lineas_intermedias = $y3_para_lineas + $dist_entre_lineas*$i;

          for( $j = 1; $j<=$etapas_finales; $j++){
          echo"<line x1='".$x1_lineas_intermedias."'
y1='".$y1_lineas_intermedias."' x2='".$x2_lineas_intermedias."'
y2='".$y2_lineas_intermedias."' style='stroke:#000; stroke-width:0.5mm' />";
          $y1_lineas_intermedias = $y1_lineas_intermedias +
$dist_entre_lineas;
          $y2_lineas_intermedias = $y2_lineas_intermedias + $altura_rect_final
+ $distancia_etapas_consecutivas;
          }
          }

          //LINEAS DE CADA PASO

          for( $i = 0; $i<($pasos); $i++ ){
          $data = explode ( "\t", $conexiones[$i]);
          $input = round($data[0],0)+1;
          $comprobacion = $input - 0.2;
          $output = round($data[1],0)+1;
          $comprobacion_2 = $output - 0.2;
          $etapa_intermedia = round($data[2],0);
          if( $etapa_intermedia != -1){
          if ( $i != $pasos - 1 ){
          //Dibujar la etapa inicial

          $etapa_inicial_dibujar = intval($comprobacion/$Ni);

          $entrada = $input - $Ni*($etapa_inicial_dibujar);
          $y1_dibujar = $y1_para_lineas + ($altura_rect +
$distancia_etapas_consecutivas)*$etapa_inicial_dibujar +
$dist_entre_lineas*$entrada;
          $x2_dibujar = $x1 + $ancho_rectangulo*0.70 -
$etapa_intermedia*$dist_entre_lineas_2;
          $y2_dibujar = $y1_para_lineas+($altura_rect +
$distancia_etapas_consecutivas)*$etapa_inicial_dibujar +
$Ni*$dist_entre_lineas + $dist_entre_lineas_6
+$dist_entre_lineas_3*$etapa_intermedia;
          $x3_dibujar = $x2_dibujar + $dist_entre_lineas_5 +
$dist_entre_lineas_4*($etapa_intermedia)
+$dist_entre_lineas_2*$etapa_intermedia;
          $y3_dibujar = $y1_para_lineas + ($altura_rect +
$distancia_etapas_consecutivas)*$etapa_inicial_dibujar +
$dist_entre_lineas*($etapa_intermedia+1);
```

```
          $x4_dibujar = $x1 + $ancho_rectangulo + $dist_entre_lineas_7;

          echo"<line x1='".$x1_lineas."' y1='".$y1_dibujar."'
x2='".$x2_dibujar."' y2='".$y1_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";
          echo"<circle cx='".$x2_dibujar."' cy='".$y1_dibujar."' r='4'
style='stroke:rgb(255,0,0); fill:red;'/>";
          echo"<line x1='".$x2_dibujar."' y1='".$y1_dibujar."'
x2='".$x2_dibujar."' y2='".$y2_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";
          echo"<line x1='".$x2_dibujar."' y1='".$y2_dibujar."'
x2='".$x3_dibujar."' y2='".$y2_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";
          echo"<line x1='".$x3_dibujar."' y1='".$y2_dibujar."'
x2='".$x3_dibujar."' y2='".$y3_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";
          echo"<line x1='".$x3_dibujar."' y1='".$y3_dibujar."'
x2='".$x4_dibujar."' y2='".$y3_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";


          //Dibujar lineas entre etapa inicial e intermedia

          $x5_dibujar = $x1 + $ancho_rectangulo + $dist_entre_lineas_7;
          $x6_dibujar = $x1 + $ancho_rectangulo +
$distancia_entre_etapas_inicial_intermedia - $dist_entre_lineas_7;
          $y4_dibujar = $y2_para_lineas +
$etapa_intermedia*($altura_rect_medio + $distancia_etapas_consecutivas) +
$dist_entre_lineas*($etapa_inicial_dibujar+1);

          echo"<line x1='".$x5_dibujar."' y1='".$y3_dibujar."'
x2='".$x6_dibujar."' y2='".$y4_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";


          //Dibujar la etapa intermedia

          $etapa_final_dibujar = intval($comprobacion_2/$No);
          $x7_dibujar = $x2 + $ancho_rectangulo_medio*0.70 -
$etapa_final_dibujar*$dist_entre_lineas_2;
          $y5_dibujar = $y2_para_lineas + ($altura_rect_medio +
$distancia_etapas_consecutivas)*$etapa_intermedia +
$etapas_iniciales*$dist_entre_lineas + $dist_entre_lineas_6 +
$dist_entre_lineas_3_2*$etapa_final_dibujar;
          $x8_dibujar = $x7_dibujar + $dist_entre_lineas_5 +
$dist_entre_lineas_4*($etapa_final_dibujar)
+$dist_entre_lineas_2*$etapa_final_dibujar;
          $y6_dibujar = $y2_para_lineas + ($altura_rect_medio +
$distancia_etapas_consecutivas)*$etapa_intermedia +
$dist_entre_lineas*($etapa_final_dibujar+1);
          $x9_dibujar = $x2 + $ancho_rectangulo_medio + $dist_entre_lineas_7;

          echo"<line x1='".$x6_dibujar."' y1='".$y4_dibujar."'
x2='".$x7_dibujar."' y2='".$y4_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";
          echo"<circle cx='".$x7_dibujar."' cy='".$y4_dibujar."' r='4'
style='stroke:rgb(255,0,0); fill:red;'/>";
          echo"<line x1='".$x7_dibujar."' y1='".$y4_dibujar."'
x2='".$x7_dibujar."' y2='".$y5_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";
          echo"<line x1='".$x7_dibujar."' y1='".$y5_dibujar."'
x2='".$x8_dibujar."' y2='".$y5_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";
          echo"<line x1='".$x8_dibujar."' y1='".$y5_dibujar."'
x2='".$x8_dibujar."' y2='".$y6_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";
```

```
          echo"<line x1='".$x8_dibujar."' y1='".$y6_dibujar."'
x2='".$x9_dibujar."' y2='".$y6_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";


          //Dibujar lineas entre etapa intermedia y final

          $x10_dibujar = $x2 + $ancho_rectangulo_medio + $dist_entre_lineas_7;
          $x11_dibujar = $x2 + $ancho_rectangulo_medio +
$distancia_entre_etapas_intermedia_final - $dist_entre_lineas_7;
          $y7_dibujar = $y3_para_lineas +
$etapa_final_dibujar*($altura_rect_final + $distancia_etapas_consecutivas) +
$dist_entre_lineas*($etapa_intermedia+1);

          echo"<line x1='".$x10_dibujar."' y1='".$y6_dibujar."'
x2='".$x11_dibujar."' y2='".$y7_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";


          //Dibujar la etapa final
          $salida = $output - $No*($etapa_final_dibujar);
          $x12_dibujar = $x3 + $ancho_rectangulo_final*0.70 - ($salida-
1)*$dist_entre_lineas_2;
          $y8_dibujar = $y3_para_lineas + ($altura_rect_final +
$distancia_etapas_consecutivas)*$etapa_final_dibujar + $R2*$dist_entre_lineas
+ $dist_entre_lineas_6 + $dist_entre_lineas_3_3*($salida-1);
          $x13_dibujar = $x12_dibujar + $dist_entre_lineas_5 +
$dist_entre_lineas_4*($salida-1) +$dist_entre_lineas_2*($salida-1);
          $y9_dibujar = $y3_para_lineas + ($altura_rect_final +
$distancia_etapas_consecutivas)*$etapa_final_dibujar +
$dist_entre_lineas*($salida);
          $x14_dibujar = $x3 + $ancho_rectangulo_final + $dist_entre_lineas_7;

          echo"<line x1='".$x11_dibujar."' y1='".$y7_dibujar."'
x2='".$x12_dibujar."' y2='".$y7_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";
          echo"<circle cx='".$x12_dibujar."' cy='".$y7_dibujar."' r='4'
style='stroke:rgb(255,0,0); fill:red;'/>";
          echo"<line x1='".$x12_dibujar."' y1='".$y7_dibujar."'
x2='".$x12_dibujar."' y2='".$y8_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";
          echo"<line x1='".$x12_dibujar."' y1='".$y8_dibujar."'
x2='".$x13_dibujar."' y2='".$y8_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";
          echo"<line x1='".$x13_dibujar."' y1='".$y8_dibujar."'
x2='".$x13_dibujar."' y2='".$y9_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";
          echo"<line x1='".$x13_dibujar."' y1='".$y9_dibujar."'
x2='".$x14_dibujar."' y2='".$y9_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";
          }else{
          //Dibujar la etapa inicial

          $etapa_inicial_dibujar = intval($comprobacion/$Ni);

          $entrada = $input - $Ni*($etapa_inicial_dibujar);
          $y1_dibujar = $y1_para_lineas + ($altura_rect +
$distancia_etapas_consecutivas)*$etapa_inicial_dibujar +
$dist_entre_lineas*$entrada;
          $x2_dibujar = $x1 + $ancho_rectangulo*0.70 -
$etapa_intermedia*$dist_entre_lineas_2;
          $y2_dibujar = $y1_para_lineas+($altura_rect +
$distancia_etapas_consecutivas)*$etapa_inicial_dibujar +
$Ni*$dist_entre_lineas + $dist_entre_lineas_6
+$dist_entre_lineas_3*$etapa_intermedia;
          $x3_dibujar = $x2_dibujar + $dist_entre_lineas_5 +
$dist_entre_lineas_4*($etapa_intermedia)
+$dist_entre_lineas_2*$etapa_intermedia;
```

```
          $y3_dibujar = $y1_para_lineas + ($altura_rect +
$distancia_etapas_consecutivas)*$etapa_inicial_dibujar +
$dist_entre_lineas*($etapa_intermedia+1);
          $x4_dibujar = $x1 + $ancho_rectangulo + $dist_entre_lineas_7;

          echo"<line x1='".$x1_lineas."' y1='".$y1_dibujar."'
x2='".$x2_dibujar."' y2='".$y1_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";
          echo"<circle cx='".$x2_dibujar."' cy='".$y1_dibujar."' r='4'
style='stroke:rgb(255,0,0); fill:red;'/>";
          echo"<line x1='".$x2_dibujar."' y1='".$y1_dibujar."'
x2='".$x2_dibujar."' y2='".$y2_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";
          echo"<line x1='".$x2_dibujar."' y1='".$y2_dibujar."'
x2='".$x3_dibujar."' y2='".$y2_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";
          echo"<line x1='".$x3_dibujar."' y1='".$y2_dibujar."'
x2='".$x3_dibujar."' y2='".$y3_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";
          echo"<line x1='".$x3_dibujar."' y1='".$y3_dibujar."'
x2='".$x4_dibujar."' y2='".$y3_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";


          //Dibujar lineas entre etapa inicial e intermedia

          $x5_dibujar = $x1 + $ancho_rectangulo + $dist_entre_lineas_7;
          $x6_dibujar = $x1 + $ancho_rectangulo +
$distancia_entre_etapas_inicial_intermedia - $dist_entre_lineas_7;
          $y4_dibujar = $y2_para_lineas +
$etapa_intermedia*($altura_rect_medio + $distancia_etapas_consecutivas) +
$dist_entre_lineas*($etapa_inicial_dibujar+1);

          echo"<line x1='".$x5_dibujar."' y1='".$y3_dibujar."'
x2='".$x6_dibujar."' y2='".$y4_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";


          //Dibujar la etapa intermedia

          $etapa_final_dibujar = intval($comprobacion_2/$No);
          $x7_dibujar = $x2 + $ancho_rectangulo_medio*0.70 -
$etapa_final_dibujar*$dist_entre_lineas_2;
          $y5_dibujar = $y2_para_lineas + ($altura_rect_medio +
$distancia_etapas_consecutivas)*$etapa_intermedia +
$etapas_iniciales*$dist_entre_lineas + $dist_entre_lineas_6 +
$dist_entre_lineas_3_2*$etapa_final_dibujar;
          $x8_dibujar = $x7_dibujar + $dist_entre_lineas_5 +
$dist_entre_lineas_4*($etapa_final_dibujar)
+$dist_entre_lineas_2*$etapa_final_dibujar;
          $y6_dibujar = $y2_para_lineas + ($altura_rect_medio +
$distancia_etapas_consecutivas)*$etapa_intermedia +
$dist_entre_lineas*($etapa_final_dibujar+1);
          $x9_dibujar = $x2 + $ancho_rectangulo_medio + $dist_entre_lineas_7;

          echo"<line x1='".$x6_dibujar."' y1='".$y4_dibujar."'
x2='".$x7_dibujar."' y2='".$y4_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";
          echo"<circle cx='".$x7_dibujar."' cy='".$y4_dibujar."' r='4'
style='stroke:rgb(255,0,0); fill:red;'/>";
          echo"<line x1='".$x7_dibujar."' y1='".$y4_dibujar."'
x2='".$x7_dibujar."' y2='".$y5_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";
          echo"<line x1='".$x7_dibujar."' y1='".$y5_dibujar."'
x2='".$x8_dibujar."' y2='".$y5_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";
```

```
          echo"<line x1='".$x8_dibujar."' y1='".$y5_dibujar."'
x2='".$x8_dibujar."' y2='".$y6_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";
          echo"<line x1='".$x8_dibujar."' y1='".$y6_dibujar."'
x2='".$x9_dibujar."' y2='".$y6_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";


          //Dibujar lineas entre etapa intermedia y final

          $x10_dibujar = $x2 + $ancho_rectangulo_medio + $dist_entre_lineas_7;
          $x11_dibujar = $x2 + $ancho_rectangulo_medio +
$distancia_entre_etapas_intermedia_final - $dist_entre_lineas_7;
          $y7_dibujar = $y3_para_lineas +
$etapa_final_dibujar*($altura_rect_final + $distancia_etapas_consecutivas) +
$dist_entre_lineas*($etapa_intermedia+1);

          echo"<line x1='".$x10_dibujar."' y1='".$y6_dibujar."'
x2='".$x11_dibujar."' y2='".$y7_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";


          //Dibujar la etapa final
          $salida = $output - $No*($etapa_final_dibujar);
          $x12_dibujar = $x3 + $ancho_rectangulo_final*0.70 - ($salida-
1)*$dist_entre_lineas_2;
          $y8_dibujar = $y3_para_lineas + ($altura_rect_final +
$distancia_etapas_consecutivas)*$etapa_final_dibujar + $R2*$dist_entre_lineas
+ $dist_entre_lineas_6 + $dist_entre_lineas_3_3*($salida-1);
          $x13_dibujar = $x12_dibujar + $dist_entre_lineas_5 +
$dist_entre_lineas_4*($salida-1) +$dist_entre_lineas_2*($salida-1);
          $y9_dibujar = $y3_para_lineas + ($altura_rect_final +
$distancia_etapas_consecutivas)*$etapa_final_dibujar +
$dist_entre_lineas*($salida);
          $x14_dibujar = $x3 + $ancho_rectangulo_final + $dist_entre_lineas_7;

          echo"<line x1='".$x11_dibujar."' y1='".$y7_dibujar."'
x2='".$x12_dibujar."' y2='".$y7_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";
          echo"<circle cx='".$x12_dibujar."' cy='".$y7_dibujar."' r='4'
style='stroke:rgb(255,0,0); fill:red;'/>";
          echo"<line x1='".$x12_dibujar."' y1='".$y7_dibujar."'
x2='".$x12_dibujar."' y2='".$y8_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";
          echo"<line x1='".$x12_dibujar."' y1='".$y8_dibujar."'
x2='".$x13_dibujar."' y2='".$y8_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";
          echo"<line x1='".$x13_dibujar."' y1='".$y8_dibujar."'
x2='".$x13_dibujar."' y2='".$y9_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";
          echo"<line x1='".$x13_dibujar."' y1='".$y9_dibujar."'
x2='".$x14_dibujar."' y2='".$y9_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";
          }
          }
          }
          echo "</svg>";
          ?>
      </div>
      <!-- Tabla conexiones -->
      <div style="float:left; margin-top:30px; margin-left:3% !important;
display:flex; width:91%; justify-content:center; min-width:513px; margin-
right: 6% !important;">
        <h2 style="width:400px;">Conexiones</h2>
        <table class="table" style="position:absolute; width:400px; margin-
top:50px; text-align:center">
          <thead class="d-block">
            <tr>
```

```
              <th style='width:130px' >Input</th>
              <th style='width:130px'>Output</th>
              <th style='width:130px'>Etapa intermedia</th>
            </tr>
          </thead>
          <tbody class="d-block" id="table-scroll" style="max-height:
calc(100vh - 150px);overflow-y: auto;">
            <?php
              for( $i = 0; $i<$pasos; $i++ ){
                    $data = explode ( "\t", $conexiones[$i]);
                    $input = $data[0] + 1;
                    $output = $data[1] + 1;
                    $etapa_intermedia = $data[2];
                    $etapa_intermedia_dibujar = $data[2];

                    if( $etapa_intermedia == -1){
                          $etapa_intermedia_dibujar = "Bloqueo";
                    }
                    if( $i == ($pasos-1)){
                          echo "<tr bgcolor='#ECECEC'>";
                    }else{
                          echo "<tr>";
                    }
                    echo "<td style='width:130px'>".$input."</td>";
                    echo "<td style='width:130px'>".$output."</td>";
                    echo "<td
style='width:130px'>".$etapa_intermedia_dibujar."</td>";
                    echo "</tr>";

              }
              ?>
          </tbody>
        </table>
      </div>
    </div>
    <!-- Parte inferior de la Web -->
    <footer class="container-fluid text-center" style="bottom:0px ;
position:fixed; background-color: #555; width:100%; color: white; padding:
7px; max-height:50px">
      <p class="d-inline">Universidad Politécnica de Cartagena</p>
      <img class="d-inline" width="40" height="40" style="margin-left:10px"
src="imagenes/upct_icon.png"/>
    </footer>
  </body>
</html>
```

7.6. Código de la web: procesar_sincruces.php

```
<!DOCTYPE html>
<html style="min-height: 100%; position: relative; min-
width:100%;display:flex">
  <head>
    <meta charset = "utf-8">
    <!--Definimos la fuente -->
    <meta name="escala"  content="width=device-width, initial-scale=1.0">
    <!--Definimos la escala de la web -->
    <title>Simulador de redes Clos</title>
    <!--El título de nuestra web -->
    <!-- Recursos externos -->
    <link rel="icon" type="image/x-icon" href="imagenes/ICOFX.png" />
    <link rel="stylesheet" href="css/bootstrap.css">
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link rel="stylesheet" href="css/bootstrap-grid.css">
    <link rel="stylesheet" href="css/bootstrap-grid.min.css">
    <link rel="stylesheet" href="css/bootstrap-reboot.css">
    <link rel="stylesheet" href="css/bootstrap-reboot.min.css">
    <!-- Script -->
    <script>
      function adjustTable(){
          var table = document.getElementById('table-scroll');
          var svg = document.getElementById('svg');
          var h = document.documentElement.clientHeight;
          table.scrollTop = '9999';

          if (h < svg.height.animVal.value){
              var alt = Math.trunc(svg.height.animVal.value) - 200;
              table.style.maxHeight = alt+"px";
          }
      }
    </script>
  </head>
  <body onload="adjustTable()" class="overflow-auto" style="min-
height:100%;width:100%; zoom:79%">
    <!-- Parte superior Web y obtención de parámetros -->
    <div style="position:fixed; width:100%; z-index:4; background-
color:white;">
      <!-- Navbar-->
      <div style="margin-bottom:15px">
        <nav class="navbar navbar-expand-md navbar-dark bg-dark d-block"
style=" width:100%;max-height:70px; position:fixed; white-space: nowrap;">
          <h4 style="text-align: center; color: white; width:100%;
height:100%; white-space: nowrap; overflow: hidden; text-overflow: ellipsis;">
Simulador de redes de conmutación de tres etapas</h4>
        </nav>
      </div>
      <!-- Parámetros y botones -->
      <div style="margin-top:64px;">
        <?php
          //Obtención de parámetros
          $N = $_GET['N'];
          $Ni = $_GET['Ni'];
          $No = $_GET['No'];
          $R2 = $_GET['R2'];
          $pasos = $_GET['paso'];
          $select = $_GET['select'];

          //Botón nueva simulación
          echo "<a role='button' class='btn btn-primary btn-md'
href='index.php?N=".$N."&Ni=".$Ni."&No=".$No."&R2=".$R2."' style=' margin-
bottom:0px; bottom:0px; margin-left:10px'>Nueva simulacion</a>";

          //Ejecutamos besa y leemos el resultado
```

```
          if(isset($select)){
              system("besa -n ".$N." --ni ".$Ni." --no ".$No." --k ".$R2." --
routing ".$select." > /tmp/resultado_pasos_simulacion.txt");
          }

          $archivo = file_get_contents("/tmp/resultado_pasos_simulacion.txt");
          $conexiones = explode( "\n", $archivo);

          //Regulamos los pasos
          $paso_siguiente = $pasos + 1;

          if($pasos != 0 ){
          $paso_anterior = $pasos - 1;
          }else{
          $paso_anterior = $pasos;
          }

          if($pasos != (sizeof($conexiones)-1)){
          $paso_siguiente = $pasos + 1;
          }else{
          $paso_siguiente = $pasos;
          }

          //Botones paso anterior y siguiente
          echo "<a role='button' class='btn btn-primary btn-md'
href='procesar_sincruces.php?N=".$N."&Ni=".$Ni."&No=".$No."&R2=".$R2."&paso=".
$paso_siguiente."' style='margin-bottom:0px; margin-right:10px; margin-
left:5px; float:right' >Paso siguiente</a>";
          echo "<a role='button' class='btn btn-primary btn-md'
href='procesar_sincruces.php?N=".$N."&Ni=".$Ni."&No=".$No."&R2=".$R2."&paso=".
$paso_anterior."' style='margin-bottom:0px;float:right' >Paso anterior</a>";
          ?>
      </div>
      <hr style="width: 92%; margin-left: 4% !important; margin-right: 4%
!important; margin-top: 8px; margin-bottom: 0px;">
    </div>
    <!-- Calculo de variables y parametros utilizados en la simulación -->
    <?php
      $etapas_iniciales = $N / $Ni;
      $etapas_finales = $N / $No;

      $distancia_etapas_consecutivas = 30;
      $distancia_bordes_verticales = 35;
      $distancia_comienzo_etapas = 80;
      $distancia_entre_etapas_inicial_intermedia = 150;
      $distancia_entre_etapas_intermedia_final = 150;
      $dist_entre_lineas = 15;
      $dist_entre_lineas_7 = 7.5;
      $dist_entre_lineas_8 = 21;

      $porcentaje_ancho_rect = 0.8;

      if ( $Ni > $R2){
       $altura_rect = $dist_entre_lineas*($Ni + 1);
       $ancho_rectangulo = $altura_rect*$porcentaje_ancho_rect;
      }else{
       $altura_rect = $dist_entre_lineas*($R2 + 1);
       $ancho_rectangulo = $altura_rect*$porcentaje_ancho_rect;
      }

      if ( $etapas_iniciales > $etapas_finales){
       $altura_rect_medio = $dist_entre_lineas*($etapas_iniciales + 1);
       $ancho_rectangulo_medio = $altura_rect_medio*$porcentaje_ancho_rect;
      }else{
       $altura_rect_medio = $dist_entre_lineas*($etapas_finales+ 1);
       $ancho_rectangulo_medio = $altura_rect_medio*$porcentaje_ancho_rect;
      }
```

```
      if ( $R2 > $No){
       $altura_rect_final = $dist_entre_lineas*($R2 + 1);
       $ancho_rectangulo_final = $altura_rect_final*$porcentaje_ancho_rect;
      }else{
       $altura_rect_final = $dist_entre_lineas*($No + 1);
       $ancho_rectangulo_final = $altura_rect_final*$porcentaje_ancho_rect;
      }

      $x1 = $distancia_comienzo_etapas;
      $x2 = $x1 + $ancho_rectangulo +
$distancia_entre_etapas_inicial_intermedia;
      $x3 = $x2 + $ancho_rectangulo_medio +
$distancia_entre_etapas_intermedia_final;

      $x1_lineas = $x1 - $dist_entre_lineas_7;
      $x2_lineas = $x2 - $dist_entre_lineas_7;
      $x3_lineas = $x3 - $dist_entre_lineas_7;

      $x1_numeros = $x1 - $dist_entre_lineas_7 - $dist_entre_lineas_8;

      $x3_numeros = $x3 + $ancho_rectangulo_final + $dist_entre_lineas_8 - 8;

      $ancho_svg = $x3 + $ancho_rectangulo_final + $distancia_comienzo_etapas;
      $altura_svg;
      ?>
    <!-- Contenido de la Web -->
    <div style="display:flex; margin-bottom:60px;width:100%; margin-top:111px;
z-index:1">
      <!-- SVG -->
      <div style=" min-height:100%; float:left">
        <?php
          //DECLARAMOS EL SVG
          $altura_etapas_iniciales = $distancia_bordes_verticales*2 +
$altura_rect*$etapas_iniciales +
$distancia_etapas_consecutivas*($etapas_iniciales - 1);
          $altura_etapas_intermedias = $distancia_bordes_verticales*2 +
$altura_rect_medio*$R2 + $distancia_etapas_consecutivas*($R2 - 1);
          $altura_etapas_finales = $distancia_bordes_verticales*2 +
$altura_rect_final*$etapas_finales +
$distancia_etapas_consecutivas*($etapas_finales - 1);

          if($altura_etapas_iniciales >= $altura_etapas_intermedias ) {
          if($altura_etapas_iniciales > $altura_etapas_finales ){
          $y1 = $distancia_bordes_verticales;
          $y1_para_lineas = $y1;
          $y2 = ($altura_etapas_iniciales/2)-(($altura_etapas_intermedias -
($distancia_bordes_verticales*2))/2);
          $y2_para_lineas = $y2;
          $y3 = ($altura_etapas_iniciales/2)-(($altura_etapas_finales -
($distancia_bordes_verticales*2))/2);
          $y3_para_lineas = $y3;
          $altura_svg = $altura_etapas_iniciales;
          echo "<svg id='svg' x='0' y='0' width='".$ancho_svg."'
height='".$altura_svg."'>";
          }else{
          $y3 = $distancia_bordes_verticales;
          $y3_para_lineas = $y3;
          $y2 = ($altura_etapas_finales/2)-(($altura_etapas_intermedias -
($distancia_bordes_verticales*2))/2);
          $y2_para_lineas = $y2;
          $y1 = ($altura_etapas_finales/2)-(($altura_etapas_iniciales -
($distancia_bordes_verticales*2))/2);
          $y1_para_lineas = $y1;
          $altura_svg = $altura_etapas_finales;
          echo "<svg id='svg' x='0' y='0' width='".$ancho_svg."'
height='".$altura_svg."'>";
          }
          }else{
```

```
          if($altura_etapas_intermedias > $altura_etapas_finales ){
          $y2 = $distancia_bordes_verticales;
          $y2_para_lineas = $y2;
          $y1 = ($altura_etapas_intermedias/2)-(($altura_etapas_iniciales -
($distancia_bordes_verticales*2))/2);
          $y1_para_lineas = $y1;
          $y3 = ($altura_etapas_intermedias/2)-(($altura_etapas_finales -
($distancia_bordes_verticales*2))/2);
          $y3_para_lineas = $y3;
          $altura_svg = $altura_etapas_intermedias;
          echo "<svg id='svg' x='0' y='0' width='".$ancho_svg."'
height='".$altura_svg."'>";
          }else{
          $y3 = $distancia_bordes_verticales;
          $y3_para_lineas = $y3;
          $y1 = ($altura_etapas_finales/2)-(($altura_etapas_iniciales -
($distancia_bordes_verticales*2))/2);
          $y1_para_lineas = $y1;
          $y2 = ($altura_etapas_finales/2)-(($altura_etapas_intermedias -
($distancia_bordes_verticales*2))/2);
          $y2_para_lineas = $y2;
          $altura_svg = $altura_etapas_finales;
          echo "<svg id='svg' x='0' y='0' width='".$ancho_svg."'
height='".$altura_svg."'>";
          }
          }
          //GENERACION RECTANGULO ETAPAS INICIALES
          for ( $i = 1; $i<= $etapas_iniciales; $i++){
          $y1_lineas = $y1 + $dist_entre_lineas;
          $y1_numeros = $y1_lineas + 5;
          $x1_lineas_2_start = $x1 + $ancho_rectangulo;
          $x1_lineas_2_end = $x1 + $ancho_rectangulo+ $dist_entre_lineas_7;

          echo"<rect x='".$x1."' y='".$y1."' width='".$ancho_rectangulo."'
height='".$altura_rect."' fill='grey'/>";
          for ( $q = 1; $q<=$Ni; $q++){
          $numeros = $q + $Ni*($i-1);
          if ( $numeros <= 9 ){
          $x1_numeros = $x1 - $dist_entre_lineas_7 - $dist_entre_lineas_8 + 8;
          }else{
          $x1_numeros = $x1 - $dist_entre_lineas_7 - $dist_entre_lineas_8;
          }
          echo"<line x1='".$x1_lineas."' y1='".$y1_lineas."' x2='".$x1."'
y2='".$y1_lineas."' style='stroke:#000; stroke-width:0.5mm' />";
          echo"<text x='".$x1_numeros."' y='".$y1_numeros."'
>".$numeros."</text>";
          $y1_lineas = $y1_lineas + $dist_entre_lineas;
          $y1_numeros = $y1_numeros + $dist_entre_lineas;
          }

          $y1_lineas = $y1 + $dist_entre_lineas;

          for ( $s = 1; $s<=$R2 ; $s++){

          echo"<line x1='".$x1_lineas_2_start."' y1='".$y1_lineas."'
x2='".$x1_lineas_2_end."' y2='".$y1_lineas."' style='stroke:#000; stroke-
width:0.5mm' />";
          $y1_lineas = $y1_lineas + $dist_entre_lineas;
          }
          $y1 = $y1 + $altura_rect + $distancia_etapas_consecutivas;
          }

          //GENERACION RECTANGULO ETAPAS INTERMEDIAS
          for ( $i = 1; $i<= $R2; $i++){
          $y2_lineas = $y2 + $dist_entre_lineas;
          $x2_lineas_2_start = $x2 + $ancho_rectangulo_medio;
          $x2_lineas_2_end = $x2 + $ancho_rectangulo_medio+
$dist_entre_lineas_7;
```

```
          echo"<rect x='".$x2."' y='".$y2."'
width='".$ancho_rectangulo_medio."' height='".$altura_rect_medio."'
fill='grey'/>";
          for ( $q = 1; $q<=$etapas_iniciales; $q++){
          echo"<line x1='".$x2_lineas."' y1='".$y2_lineas."' x2='".$x2."'
y2='".$y2_lineas."' style='stroke:#000; stroke-width:0.5mm' />";
          $y2_lineas = $y2_lineas + $dist_entre_lineas;
          }

          $y2_lineas = $y2 + $dist_entre_lineas;

          for ( $s = 1; $s<=$etapas_finales ; $s++){

          echo"<line x1='".$x2_lineas_2_start."' y1='".$y2_lineas."'
x2='".$x2_lineas_2_end."' y2='".$y2_lineas."' style='stroke:#000; stroke-
width:0.5mm' />";
          $y2_lineas = $y2_lineas + $dist_entre_lineas;
          }
          $y2 = $y2 + $altura_rect_medio + $distancia_etapas_consecutivas;
          }

          //GENERACION RECTANGULO ETAPAS INTERMEDIAS
          for ( $i = 1; $i<= $etapas_finales; $i++){
          $y3_lineas = $y3 + $dist_entre_lineas;
          $y3_numeros = $y3_lineas + 5;
          $x3_lineas_2_start = $x3 + $ancho_rectangulo_final;
          $x3_lineas_2_end = $x3 + $ancho_rectangulo_final+
$dist_entre_lineas_7;

          echo"<rect x='".$x3."' y='".$y3."'
width='".$ancho_rectangulo_final."' height='".$altura_rect_final."'
fill='grey'/>";
          for ( $q = 1; $q<=$R2; $q++){
          echo"<line x1='".$x3_lineas."' y1='".$y3_lineas."' x2='".$x3."'
y2='".$y3_lineas."' style='stroke:#000; stroke-width:0.5mm' />";
          $y3_lineas = $y3_lineas + $dist_entre_lineas;
          }

          $y3_lineas = $y3 + $dist_entre_lineas;

          for ( $s = 1; $s<=$No ; $s++){
          $numeros = $s + $No*($i-1);
          echo"<line x1='".$x3_lineas_2_start."' y1='".$y3_lineas."'
x2='".$x3_lineas_2_end."' y2='".$y3_lineas."' style='stroke:#000; stroke-
width:0.5mm' />";
          echo"<text x='".$x3_numeros."' y='".$y3_numeros."'
>".$numeros."</text>";
          $y3_lineas = $y3_lineas + $dist_entre_lineas;
          $y3_numeros = $y3_numeros + $dist_entre_lineas;
          }
          $y3 = $y3 + $altura_rect_final + $distancia_etapas_consecutivas;
          }

          //Dibujar lineas entre etapas iniciales e intermedias

          for( $i = 1; $i<=$etapas_iniciales; $i++){
          $x1_lineas_intermedias = $x1 + $ancho_rectangulo +
$dist_entre_lineas_7;
          $y1_lineas_intermedias = $y1_para_lineas + ($altura_rect +
$distancia_etapas_consecutivas)*($i - 1) + $dist_entre_lineas ;
          $x2_lineas_intermedias = $x2 - $dist_entre_lineas_7;
          $y2_lineas_intermedias = $y2_para_lineas + $dist_entre_lineas*$i;

          for( $j = 1; $j<=$R2 ; $j++){
          echo"<line x1='".$x1_lineas_intermedias."'
y1='".$y1_lineas_intermedias."' x2='".$x2_lineas_intermedias."'
y2='".$y2_lineas_intermedias."' style='stroke:#000; stroke-width:0.5mm' />";
```

```
          $y1_lineas_intermedias = $y1_lineas_intermedias +
$dist_entre_lineas;
          $y2_lineas_intermedias = $y2_lineas_intermedias + $altura_rect_medio
+ $distancia_etapas_consecutivas;
          }
          }

          //Dibujar lineas entre etapas intermedias y finales

          for( $i = 1; $i<=$R2; $i++){
          $x1_lineas_intermedias = $x2 + $ancho_rectangulo_medio +
$dist_entre_lineas_7;
          $y1_lineas_intermedias = $y2_para_lineas + ($altura_rect_medio +
$distancia_etapas_consecutivas)*($i - 1) + $dist_entre_lineas ;
          $x2_lineas_intermedias = $x3 - $dist_entre_lineas_7;
          $y2_lineas_intermedias = $y3_para_lineas + $dist_entre_lineas*$i;

          for( $j = 1; $j<=$etapas_finales; $j++){
          echo"<line x1='".$x1_lineas_intermedias."'
y1='".$y1_lineas_intermedias."' x2='".$x2_lineas_intermedias."'
y2='".$y2_lineas_intermedias."' style='stroke:#000; stroke-width:0.5mm' />";
          $y1_lineas_intermedias = $y1_lineas_intermedias +
$dist_entre_lineas;
          $y2_lineas_intermedias = $y2_lineas_intermedias + $altura_rect_final
+ $distancia_etapas_consecutivas;
          }
          }

          for( $i = 0; $i<($pasos); $i++ ){
          $data = explode ( "\t", $conexiones[$i]);
          $input = $data[0];
          $output = $data[1];
          $etapa_intermedia = $data[2];

          }

          //LINEAS DE CADA PASO

          for( $i = 0; $i<($pasos); $i++ ){
          $data = explode ( "\t", $conexiones[$i]);
          $input = round($data[0],0) + 1;
          $comprobacion = $input - 0.2;
          $output = round($data[1],0) + 1;
          $comprobacion_2 = $output - 0.2;
          $etapa_intermedia = round($data[2],0);

          if( $etapa_intermedia != -1){

          if($i != $pasos -1){
          //Dibujar la etapa inicial

          $etapa_inicial_dibujar = intval($comprobacion/$Ni);

          $entrada = $input - $Ni*($etapa_inicial_dibujar);
          $y1_dibujar = $y1_para_lineas + ($altura_rect +
$distancia_etapas_consecutivas)*$etapa_inicial_dibujar +
$dist_entre_lineas*$entrada;
          $x3_dibujar = $x1 + $ancho_rectangulo;
          $y3_dibujar = $y1_para_lineas + ($altura_rect +
$distancia_etapas_consecutivas)*$etapa_inicial_dibujar +
$dist_entre_lineas*($etapa_intermedia+1);
          $x4_dibujar = $x1 + $ancho_rectangulo + $dist_entre_lineas_7;

          echo"<line x1='".$x1_lineas."' y1='".$y1_dibujar."' x2='".$x1."'
y2='".$y1_dibujar."' style='stroke:rgb(255,0,0); stroke-width:0.5mm' />";
          echo"<line x1='".$x3_dibujar."' y1='".$y3_dibujar."'
x2='".$x4_dibujar."' y2='".$y3_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";
```

```
          //Dibujar lineas entre etapa inicial e intermedia

          $x5_dibujar = $x1 + $ancho_rectangulo + $dist_entre_lineas_7;
          $x6_dibujar = $x1 + $ancho_rectangulo +
$distancia_entre_etapas_inicial_intermedia - $dist_entre_lineas_7;
          $y4_dibujar = $y2_para_lineas +
$etapa_intermedia*($altura_rect_medio + $distancia_etapas_consecutivas) +
$dist_entre_lineas*($etapa_inicial_dibujar+1);

          echo"<line x1='".$x5_dibujar."' y1='".$y3_dibujar."'
x2='".$x6_dibujar."' y2='".$y4_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";


          //Dibujar la etapa intermedia

          $etapa_final_dibujar = intval($comprobacion_2/$No);

          $x8_dibujar = $x2 + $ancho_rectangulo_medio;
          $y6_dibujar = $y2_para_lineas + ($altura_rect_medio +
$distancia_etapas_consecutivas)*$etapa_intermedia +
$dist_entre_lineas*($etapa_final_dibujar+1);
          $x9_dibujar = $x2 + $ancho_rectangulo_medio + $dist_entre_lineas_7;

          echo"<line x1='".$x6_dibujar."' y1='".$y4_dibujar."' x2='".$x2."'
y2='".$y4_dibujar."' style='stroke:rgb(255,0,0); stroke-width:0.5mm' />";
          echo"<line x1='".$x8_dibujar."' y1='".$y6_dibujar."'
x2='".$x9_dibujar."' y2='".$y6_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";


          //Dibujar lineas entre etapa intermedia y final

          $x10_dibujar = $x2 + $ancho_rectangulo_medio + $dist_entre_lineas_7;
          $x11_dibujar = $x2 + $ancho_rectangulo_medio +
$distancia_entre_etapas_intermedia_final - $dist_entre_lineas_7;
          $y7_dibujar = $y3_para_lineas +
$etapa_final_dibujar*($altura_rect_final + $distancia_etapas_consecutivas) +
$dist_entre_lineas*($etapa_intermedia+1);

          echo"<line x1='".$x10_dibujar."' y1='".$y6_dibujar."'
x2='".$x11_dibujar."' y2='".$y7_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";


          //Dibujar la etapa final

          $salida = $output - $No*($etapa_final_dibujar);

          $x13_dibujar = $x3 + $ancho_rectangulo_final;
          $y9_dibujar = $y3_para_lineas + ($altura_rect_final +
$distancia_etapas_consecutivas)*$etapa_final_dibujar +
$dist_entre_lineas*($salida);
          $x14_dibujar = $x3 + $ancho_rectangulo_final + $dist_entre_lineas_7;

          echo"<line x1='".$x11_dibujar."' y1='".$y7_dibujar."' x2='".$x3."'
y2='".$y7_dibujar."' style='stroke:rgb(255,0,0); stroke-width:0.5mm' />";
          echo"<line x1='".$x13_dibujar."' y1='".$y9_dibujar."'
x2='".$x14_dibujar."' y2='".$y9_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:0.5mm' />";
          }else{
          //Dibujar la etapa inicial

          $etapa_inicial_dibujar = intval($comprobacion/$Ni);

          $entrada = $input - $Ni*($etapa_inicial_dibujar);
```

```
          $y1_dibujar = $y1_para_lineas + ($altura_rect +
$distancia_etapas_consecutivas)*$etapa_inicial_dibujar +
$dist_entre_lineas*$entrada;
          $x3_dibujar = $x1 + $ancho_rectangulo;
          $y3_dibujar = $y1_para_lineas + ($altura_rect +
$distancia_etapas_consecutivas)*$etapa_inicial_dibujar +
$dist_entre_lineas*($etapa_intermedia+1);
          $x4_dibujar = $x1 + $ancho_rectangulo + $dist_entre_lineas_7;

          echo"<line x1='".$x1_lineas."' y1='".$y1_dibujar."' x2='".$x1."'
y2='".$y1_dibujar."' style='stroke:rgb(255,0,0); stroke-width:1mm' />";
          echo"<line x1='".$x3_dibujar."' y1='".$y3_dibujar."'
x2='".$x4_dibujar."' y2='".$y3_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";


          //Dibujar lineas entre etapa inicial e intermedia

          $x5_dibujar = $x1 + $ancho_rectangulo + $dist_entre_lineas_7;
          $x6_dibujar = $x1 + $ancho_rectangulo +
$distancia_entre_etapas_inicial_intermedia - $dist_entre_lineas_7;
          $y4_dibujar = $y2_para_lineas +
$etapa_intermedia*($altura_rect_medio + $distancia_etapas_consecutivas) +
$dist_entre_lineas*($etapa_inicial_dibujar+1);

          echo"<line x1='".$x5_dibujar."' y1='".$y3_dibujar."'
x2='".$x6_dibujar."' y2='".$y4_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";


          //Dibujar la etapa intermedia

          $etapa_final_dibujar = intval($comprobacion_2/$No);

          $x8_dibujar = $x2 + $ancho_rectangulo_medio;
          $y6_dibujar = $y2_para_lineas + ($altura_rect_medio +
$distancia_etapas_consecutivas)*$etapa_intermedia +
$dist_entre_lineas*($etapa_final_dibujar+1);
          $x9_dibujar = $x2 + $ancho_rectangulo_medio + $dist_entre_lineas_7;

          echo"<line x1='".$x6_dibujar."' y1='".$y4_dibujar."' x2='".$x2."'
y2='".$y4_dibujar."' style='stroke:rgb(255,0,0); stroke-width:1mm' />";
          echo"<line x1='".$x8_dibujar."' y1='".$y6_dibujar."'
x2='".$x9_dibujar."' y2='".$y6_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";


          //Dibujar lineas entre etapa intermedia y final

          $x10_dibujar = $x2 + $ancho_rectangulo_medio + $dist_entre_lineas_7;
          $x11_dibujar = $x2 + $ancho_rectangulo_medio +
$distancia_entre_etapas_intermedia_final - $dist_entre_lineas_7;
          $y7_dibujar = $y3_para_lineas +
$etapa_final_dibujar*($altura_rect_final + $distancia_etapas_consecutivas) +
$dist_entre_lineas*($etapa_intermedia+1);

          echo"<line x1='".$x10_dibujar."' y1='".$y6_dibujar."'
x2='".$x11_dibujar."' y2='".$y7_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";

          //Dibujar la etapa final

          $salida = $output - $No*($etapa_final_dibujar);

          $x13_dibujar = $x3 + $ancho_rectangulo_final;
          $y9_dibujar = $y3_para_lineas + ($altura_rect_final +
$distancia_etapas_consecutivas)*$etapa_final_dibujar +
$dist_entre_lineas*($salida);
```

```
          $x14_dibujar = $x3 + $ancho_rectangulo_final + $dist_entre_lineas_7;

          echo"<line x1='".$x11_dibujar."' y1='".$y7_dibujar."' x2='".$x3."'
y2='".$y7_dibujar."' style='stroke:rgb(255,0,0); stroke-width:1mm' />";
          echo"<line x1='".$x13_dibujar."' y1='".$y9_dibujar."'
x2='".$x14_dibujar."' y2='".$y9_dibujar."' style='stroke:rgb(255,0,0); stroke-
width:1mm' />";
          }
          }
          }
          echo "</svg>";
          ?>
      </div>
      <!-- Tabla conexiones -->
      <div style="float:left; margin-top:30px; margin-left:3% !important;
display:flex; width:91%; justify-content:center; min-width:513px; margin-
right: 6% !important;">
        <h2 style="width:400px;">Conexiones</h2>
        <table class="table" style="position:absolute; width:400px; margin-
top:50px; text-align:center">
          <thead class="d-block">
            <tr>
              <th style='width:130px'>Input</th>
              <th style='width:130px'>Output</th>
              <th style='width:130px'>Etapa intermedia</th>
            </tr>
          </thead>
          <tbody id="table-scroll" class="d-block" style="max-height:
calc(100vh - 150px);overflow-y: auto;">
            <?php
              for( $i = 0; $i<$pasos; $i++ ){
                    $data = explode ( "\t", $conexiones[$i]);
                    $input = $data[0] + 1;
                    $output = $data[1] + 1;
                    $etapa_intermedia = $data[2];
                    $etapa_intermedia_dibujar = $data[2];

                    if( $etapa_intermedia == -1){
                          $etapa_intermedia_dibujar = "Bloqueo";
                    }
                    if( $i == ($pasos-1)){
                                        echo "<tr bgcolor='#ECECEC'>";
                                }else{
                                        echo "<tr>";
                    }
                    echo "<td style='width:130px'>".$input."</td>";
                    echo "<td style='width:130px'>".$output."</td>";
                    echo "<td
style='width:130px'>".$etapa_intermedia_dibujar."</td>";
                    echo "</tr>";
              }
              ?>
          </tbody>
        </table>
      </div>
    </div>
    <!-- Parte inferior de la Web -->
    <footer class="container-fluid text-center" style="bottom:0px ;
position:fixed; background-color: #555; width:100%; color: white; padding:
7px; max-height:50px">
      <p class="d-inline">Universidad Politécnica de Cartagena</p>
      <img class="d-inline" width="40" height="40" style="margin-left:10px"
src="imagenes/upct_icon.png"/>
    </footer>
  </body>
</html>
```