Simulación de tráfico de internet

AyED TP - 3

Desarrollar un programa que simule el tráfico de datos, al ”estilo” del funcionamiento de Internet.
Existen n máquinas que cumplen la funci´on de routers que se encargan de rutear los datos desde una máquina de origen hacia la máquina de destino.
Existen otras k máquinas, las terminales, que son las emisoras - receptoras de páginas. Cada una de estas máquinas está conectada a un único router que es el encargado de enviar / recibir las paginas hacía / desde el destino final.
Cada router está conectado a 1 o más routers para transmitir los paquetes. Cada router sabe cuales son las máquinas terminales que tiene conectadas y cuales son los routers vecinos que tiene, es decir a que otros routers está conectado directamente. Además, cada router tiene una tabla que le indica a qué router enviar los datos con un determinado destino. Cada router tiene una conexión directa con sus vecinos de un determinado ancho de banda. Cuando un router recibe una página para enviar de una de sus terminales, este la divide en paquetes de igual tamaño y va enviando por la ruta elegida de a un paquete por vez. Es decir que una página pedida por otra terminal se divide y se envía de a segmentos. A su vez, cuando un router va recibiendo de otro router paquetes con un determinado destino, debe enviarlo al router correspondiente en la ruta, o bien, si el destino final es una m´aquina terminal conectada directamente, debe ir almacenando los paquetes recibidos hasta que est´en todos los que correspondan a la p´agina enviada, rearma la pagina y recien alli´ı se la envía a la máquina destino.
Las direcciones de las máquinas, son tipo IP, pero simplificadas. Tienen dos partes de 1 byte cada una: la primera indica el router y la segunda la máquina terminal conectada al router. Es decir que pueden haber 256 routers con 256 máquinas cada uno.
¿Cómo hace cada router para computar la tabla de destinos que posee? Si la dirección del paquete corresponde a la de un router vecino, hay una conexión directa, por lo que no hay más trámite. Para routers que no son vecinos pueden haber varias rutas alternativas, debiendo el router elegir aquella que tiene la menor carga de tráfico. Una vez determinada la mejor ruta, todos los paquetes enviados a un determinado destino, se envían al router vecino que conforma el camino elegido.
En resumen, cada router tiene las siguientes funciones:

- Recibir una página en una máquina cliente, dividirla en los paquetes que corresponda, y enviarla a la cola de tráfico de la ruta que corresponda.
- Recibir paquetes de los routers vecinos y redireccionarlos hacia el router vecino que corresponda si la dirección del paquete no es la propia del router, o bien si la dirección del paquete es la del router en cuestión, debe esperar a recibir todos los paquetes que corresponden a la página enviada y una vez sucedido esto, enviar la página a la máquina de destino

Cada router tiene una cola de envíos para cada router vecino, en donde van encolando los paquetes que tienen que enviarse por ese canal y que luego envía por cada turno, todos los que su ancho de banda le permita. En la cola no se deben colocar todos los paquetes de una página consecutivos: deben ser intercalados con los paquetes que provengan de otra máquina, para que se vayan enviado parcialmente de todas las máquinas al mismo tiempo.

Esto evita que un envío muy pesado atore al servidor y los otros paquetes demoren mucho en ser enviados.
Existe un administrador del sistema que recopila las rutas óptimas de todos los routers periódicamente. Para ello cada router le envía el tama˜no de la cola de espera de envíos de paquetes hacia cada router vecino, y con ello el administrador determina la ruta ´óptima pasando por los routers que tengan menor tráfico pendiente en relación al ancho de banda de la conexión que tenga con ese router. Hay que tener en cuenta que cada router envía k paquetes por vez en un canal, según el ancho de banda que tenga el canal.
Para determinar el óptimo, lo que importa es la cantidad de ciclos que un nuevo paquete debe esperar hasta ser enviado. Además se pierde un ciclo al entrar a un router y volver a salir.
O sea que, si un router tiene la cola vacía, el envío de un paquete a ese router no tiene un costo de cero, si no de uno, porque el paquete debe esperar hasta el próximo turno para ser reenviado.
Una vez que el administrador determina los caminos óptimos, se les informa a cada router. Esos caminos son utilizados a partir de ese momento hasta que vuelven a computarse.
Puede suceder que los paquetes pendientes de enviar de una página, utilicen un camino distinto de los enviados previamente, porque se cambió el camino a utilizar por uno con menos tráfico.
El caso es el así: suponga una página que se dividió en 50 paquetes. Se enviaron 20. Se recupera el camino óptimo y se cambia de ruta. Los 30 paquetes restantes van por otra ruta, que al ser tomada como ´óptima, pueden llegar a destino antes que los primeros 20. Tener esto en cuenta cuando el router debe rearmar la página.
Ud deberá simular este proceso. El mismo consta de un ciclo en que cada uno de los routers haga las tareas de recepción y reenvío o almacenamiento de paquetes que tienen que hacer, de a uno por vez. Cada 2 ciclos, tomará el control el administrador para computar los caminos óptimos y volver a al cómputo de ciclos.
Deberá utilizar números aleatorios para simular la generación de páginas a ser enviadas, el destino y el tamaño de cada página. La cantidad de routers, la cantidad de terminales por router, las conexiones directas de los routers y el ancho de banda entre los routers y entre cada terminal y el router asociado deberá ser configurable y definido en un archivo que parametrice el sistema.

Matrices

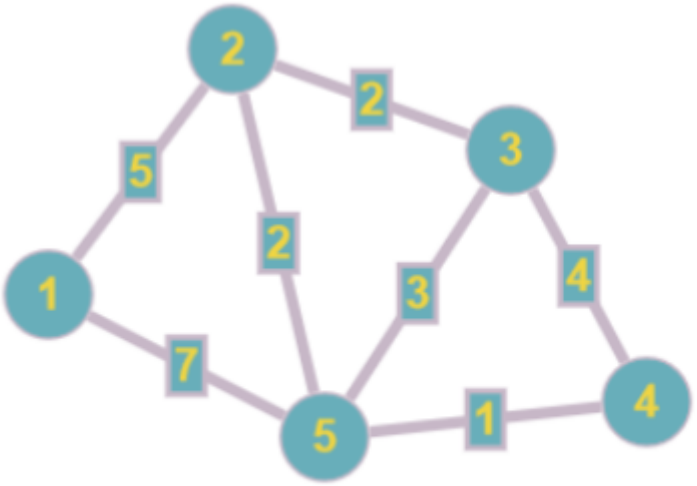
Adjacency-Bandwidth routers matrix.

Como paso inicial se tiene un archivo de configuración “configuration.txt” donde se declaran las matrices de adyacencia, de ancho de banda y la lista de terminales disponibles. Estos parámetros son leídos y luego cargados dentro del programa en los objetos declarados previamente.
Una matriz de adyacencia declara las interconexiones entre routers, teniendo un “1” si hay conexión o un “0” si no la hay. A partir de esta matriz se pueden deducir los nodos en el diagrama de grafos.
En la matriz de ancho de banda se declaran las capacidades máximas de transferencia por el camino que hay entre nodos.
También al ser una simulación de tráfico, los datos que viajan a través de los routers una vez llegados al router final estos son enviados a los terminales finales que se conectan a

estos routers, estos en la práctica se podrán deducir como PC's a las cuales llegan o s alen los datos a enviar.

1	1	0	0	1
1	1	1	0	1
0	1	1	1	1
0	0	1	1	1
1	1	1	1	1

0	5	0	0	7
5	0	2	0	2
0	2	0	4	3
0	0	4	0	1
7	2	3	1	0



A partir de estas dos matrices es que se deduce el mapa sobre el que vamos a trabajar utilizando dijkstra para obtener el camino más corto en el envío de datos.

Datos a enviar.

Los datos que se generan en los terminales y luego se envían hacia otra terminal a través del camino de routers están dentro de un paquete de datos, este paquete contiene una parte del dato completo y los paquetes se envían de manera intercalada para no saturar los canales de comunicación. Para la simulación se envían cadenas de 16 bits generados de manera aleatoria y de tamaño aleatorio, siempre declarando un rango máximo de tamaño en los archivos de configuración.

DATA.

Los datos se trabajan en una lista de objetos bitset de tamaño 16. Los primeros 32 bits contienen información útil para saber a donde enviarse. Para el envío se extrae esta información y luego se extrae una sola de estas cadenas de datos generadas para empaquetarse y enviarse por la red, donde el terminal final rearma estos datos para tener su totalidad de la información.

```
DATA<48:> = <FinalTerminal FinalRouter 0 InitTerminal 16 .. 16>
      FinalTerminal es el terminal a donde tiene que llegar.
      FinalRouter es el router a donde tiene que llegar.
      8 bits sin uso.
      InitTerminal es el terminal inicial.
      Cadena de datos a enviar.
```

PACKETS.

Los paquetes se arman de manera que contengan sólo una parte del dato obtenido en DATA. Estos paquetes son los que se envían dentro de la red y se dividen en diferentes **bitset<64>**. Estos aparte de contener los datos contienen información sobre el viaje que van a realizar.

```
PACKET<64> = <PKTINFO PARTPOS TOTPARTS NEXTROUT INITTRML FNLTRNL FNLRT>
	PacketInfo es la información a enviar, es decir una parte del dato.
	PartPosition es el número de parte del dato.
	TotalParts es la cantidad total de partes del dato.
	NextRouter es el camino que tomará el paquete.
	InitTerminal es el terminal inicial.
	FinalTerminal es el terminal a donde tiene que llegar.
	FinalRouter es el router a donde tiene que llegar
```

Administrador

La clase “Administrator” cumple la función de administrar y controlar la simulación de una red de routers y terminales, es responsable de establecer la configuración de la red, generar datos aleatorios, enrutar paquetes y supervisar el flujo de datos a través de la red.

Esta simulación funciona por “steps” es decir que con cada iteración se puede ver el tráfico de datos a través de la red y estos steps son controlados por el método simulationStep().

```
void Administrator::simulationStep(void) {
	NodeList<Router *> *node_router = routers_list->getHead();
	Router *router;
	while(node_router != NULL) {
		router = node_router->getData();
		router->sendPacketsToRouters();
		router->sendDatasToTerminals();
		router->recieveDatasFromTerminals();
		router->checkBuildAndSortDatas();
		node_router = node_router->getNext();
	}
	this->calculateTimeouts();
	this->setPathToPackets();
	cout<<endl;
	this->showPacketsInputs();
	node_router = routers_list->getHead();
	while(node_router != NULL) {
		router = node_router->getData();
		router->movePacketsToOutputsQueues();
		router->sortPacketsQueues();
		node_router = node_router->getNext();
	}
	cout<<endl;
	this->showDatasOutputsQueues();
	cout<<endl;
```

```
        this->showPacketsOutputsQueues();
        cout<<endl;
        this->showTimeoutMatrix();
        cout<<endl;
        this->showParcialDatas();
}
```

Como primer paso se envían los paquetes armados hacia los routers origen, luego si es que existen, se envían los datos generados hacia los terminales destino, y se envían los datos que ya están en camino hacia el router receptor. Si algún dato en cuestión ya llegó completo a su router destino se rearma y se envían a la terminal.

Para la elección del camino se calculan los timeouts de la matriz de grafos, es decir el peso que tiene esta conexión entre routers, dividiendo la cantidad de paquetes a enviar sobre el ancho de banda máximo soportado, así actualizando routers_timeout_matrix. Que luego es utilizada con el algoritmo de Dijkstra para encontrar el camino más corto entre un router inicial y un router destino. Una vez encontrado este camino más corto, se modifica el bitset del “nextRouter” del paquete para así luego ser enviado por ese camino.

Como paso final se muestran en consola los nuevos paquetes ingresados, la cola de paquetes de salida, la cola de datas de salida hacia los terminales destino, la matriz de timeouts que representa el tráfico “en vivo” de los datos y los datos parciales que ya llegaron a su router destino.

Una vez que todos los datos llegaron a su destino se rearman y se envían al terminal final para así finalizar la ejecución del programa. Como también el programa puede finalizar antes si es que se supera un máximo de steps sin que lleguen todos los datos.

Armado y desarmado de datos.

Como se explicó anteriormente los datos se dividen en “nodos” de 16 bits los cuales cada uno se envía envuelto en un paquete para no enviar datos muy largos en una sola parte, estos paquetes contienen la información de router inicial, final, cantidad de partes y número de partes, para así cuando llegan al router final se pueden filtrar y mediante el método checkBuildAndSortDatas() se pueden rearmar correctamente verificando si esta el total de partes y si es el paquete correcto que pertenece al paquete a armar.

Ejemplo de ejecución del programa con step inicial y primer step.

```
-ADJACENCY ROUTERS MATRIX-

| 1 1 0 0 1 |
| 1 1 1 0 1 |
| 0 1 1 1 1 |
| 0 0 1 1 1 |
| 1 1 1 1 1 |

-BANDWIDTH CONNECTIONS MATRIX-

| 0 5 0 0 7 |
| 5 0 2 0 2 |
| 0 2 0 4 3 |
| 0 0 4 0 1 |
| 7 2 3 1 0 |

-CONNECTIONS-

router IP: 00000000
connected terminals IP's: 00000000 00000001
connected routers IP's: 00000001 00000100

router IP: 00000001
connected terminals IP's: 00000010
connected routers IP's: 00000000 00000010 00000100

router IP: 00000010
connected terminals IP's: 00000011
connected routers IP's: 00000001 00000011 00000100

router IP: 00000011
connected terminals IP's: 00000100
connected routers IP's: 00000010 00000100

router IP: 00000100
connected terminals IP's: 00000101
connected routers IP's: 00000000 00000001 00000010 00000011

-GENERATED DATAS-

router IP: 00000000
terminal IP: 00000000
- 0000001100000010 0000000000000000 0011111000111100 0110110000010111
0000100011111011
terminal IP: 00000001
- 0000001000000001 0000000000000001 0100110001000110 0111000000100100

router IP: 00000001
```

```
terminal IP: 00000010
- 0000000000000000 0000000000000010 0111101100011111 0001101110101101

router IP: 00000010
terminal IP: 00000011
- 0000010100000100 0000000000000011 0010101001010001 0100000010010000

router IP: 00000011
terminal IP: 00000100
- 0000001000000001 0000000000000100 0010011001101100

router IP: 00000100
terminal IP: 00000101
- 0000001100000010 0000000000000101 0110110000010001

total generated datas: 6

Press any key to continue . . .
```

```
================================================================

simulation step: 0

-PACKETS INPUTS-

router IP: 00000000
0011111000111100 00000000 00000011 00000001 00000000 00000011 00000010
0110110000010111 00000001 00000011 00000001 00000000 00000011 00000010
0000100011111011 00000010 00000011 00000001 00000000 00000011 00000010
0100110001000110 00000000 00000010 00000001 00000001 00000010 00000001
0111000000100100 00000001 00000010 00000001 00000001 00000010 00000001

router IP: 00000001
0111101100011111 00000000 00000010 00000000 00000010 00000000 00000000
0001101110101101 00000001 00000010 00000000 00000010 00000000 00000000

router IP: 00000010
0010101001010001 00000000 00000010 00000100 00000011 00000101 00000100
0100000010010000 00000001 00000010 00000100 00000011 00000101 00000100

router IP: 00000011
0010011001101100 00000000 00000001 00000010 00000100 00000010 00000001

router IP: 00000100
0110110000010001 00000000 00000001 00000010 00000101 00000011 00000010

-DATAS OUTPUTS QUEUES-
```

```
router IP: 00000000
(terminal 00000000) EMPTY
(terminal 00000000) EMPTY

router IP: 00000001
(terminal 00000001) EMPTY

router IP: 00000010
(terminal 00000010) EMPTY

router IP: 00000011
(terminal 00000011) EMPTY

router IP: 00000100
(terminal 00000100) EMPTY

-PACKETS OUTPUTS QUEUES-

router IP: 00000000
(router 00000001) 0011111000111100 00000000 00000011 00000001 00000000 00000011
00000010
                  0100110001000110 00000000 00000010 00000001 00000001 00000010
00000001
                  0000100011111011 00000010 00000011 00000001 00000000 00000011
00000010
                  0111000000100100 00000001 00000010 00000001 00000001 00000010
00000001
                  0110110000010111 00000001 00000011 00000001 00000000 00000011
00000010
(router 00000100) EMPTY

router IP: 00000001
(router 00000000) 0111101100011111 00000000 00000010 00000000 00000010 00000000
00000000
                  0001101110101101 00000001 00000010 00000000 00000010 00000000
00000000
(router 00000010) EMPTY
(router 00000100) EMPTY

router IP: 00000010
(router 00000001) EMPTY
(router 00000011) EMPTY
(router 00000100) 0010101001010001 00000000 00000010 00000100 00000011 00000101
00000100
                  0100000010010000 00000001 00000010 00000100 00000011 00000101
00000100

router IP: 00000011
(router 00000010) 0010011001101100 00000000 00000001 00000010 00000100 00000010
00000001
```

```
(router 00000100) EMPTY

router IP: 00000100
(router 00000000) EMPTY
(router 00000001) EMPTY
(router 00000010) 0110110000010001 00000000 00000001 00000010 00000101 00000011
00000010
(router 00000011) EMPTY

-ROUTERS TIMEOUT MATRIX-

| 9000    0 9000 9000    0 |
|    0 9000   0 9000    0 |
| 9000    0 9000    0    0 |
| 9000 9000   0 9000    0 |
|    0    0    0    0 9000 |

-PARCIAL DATAS-

router IP: 00000000
EMPTY

router IP: 00000001
EMPTY

router IP: 00000010
EMPTY

router IP: 00000011
EMPTY

router IP: 00000100
EMPTY

Press any key to continue . . .
```