

Tarea 3 - Redes de Computadores

Juan Carlos Arriagada 2973565-4
Celeste Bertin 201092008-9

20 de Junio del 2014

1. Introducción

El presente informe tiene por objetivo mostrar al lector cual es el camino que sigue la información para llegar a su destino, explicando brevemente porque son así las rutas. Además, se espera familiarizar al lector con el algoritmo de Vector-Distancia para calcular la ruta más corta para el envío de paquetes y explicar cómo es que este algoritmo permite darle robustez a las redes en caso de falla de algún enlace físico.

2. Desarrollo

El presente informe se divide en 3 items, los cuales son "comunicación intercontinental", "tabla de costo de routers" y "tabla de costo de routers en caso de modificación de conexiones", los cuales se detallan a continuación:

2.1. Comunicación Intercontinental

Desde la creación de la telegrafía, la humanidad ha tratado de comunicarse a largas distancias de forma rápida y efectiva. Un gran desafío fue la comunicación entre continentes, para lo cual se tendió un cable de cobre entre Europa y América del Norte, logrando reducir el tiempo de envío de mensajes a unos pocos minutos, en comparación a los 10 días que tomaba enviar un mensaje por barco [4]. Hoy en día, la información viaja de forma rápida y efectiva entre continentes a través de cables submarinos de fibra óptica, los cuales transmiten las señales digitales utilizadas para telecomunicaciones internacionales, entre los cuales está el internet. Estos cables están instalados entre continentes de tal forma que si se daña alguno, existe redundancia de las conexiones tal que no se pierda comunicación en caso de contratiempos, como por ejemplo, un cable cortado. Los cables usados actualmente están compuestos de varias capas, para protección de los cables de fibra óptica los cuales son usados para el envío de información. Para transmitir información a través de tan largas distancia, se le agregan al cable repetidores cada 60 kilómetros, los cuales amplifican la señal [2]. En caso de dañarse el cable, una embarcación de reparación va al punto de

ruptura, sube ambos extremos a la superficie y los conecta de nuevo con una extensión nueva de cable. El cable luego de ser reparado es más largo que el cable original, por ende se deja el cable en forma de "U" en el fondo del mar [6].

Chile está conectado al resto del mundo a través de varios cables submarinos de fibra óptica [5], los cuales son:

- Panamericano (PAN-AM)
- South America-1 (SAM-1)
- South American Crossing (SAC)/Latin American Nautilus (LAN)

Los cuales se pueden ver en la siguiente figura:

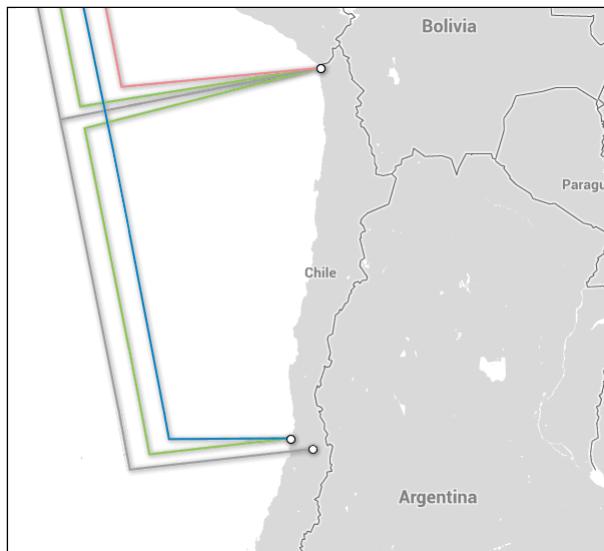


Figura 1: Cables submarinos que conectan a Chile a otros continentes

A continuación se utiliza Open Visual Trace Route 5.1.1 para investigar la ruta tomada por los paquetes al acceder a los siguientes sitios web desde Santiago de Chile:

Moodle

Se ingresó <http://moodle.inf.utfsm.cl/> a Open Visual Trace Route, el cual mostró las rutas tomadas para llevar al server final. Se puede ver que el server de Moodle queda en Valparaíso. Cabe notar que si se accede a Moodle desde el internet de la Universidad misma, se accede directamente, ya que Moodle está en los servidores de la universidad.

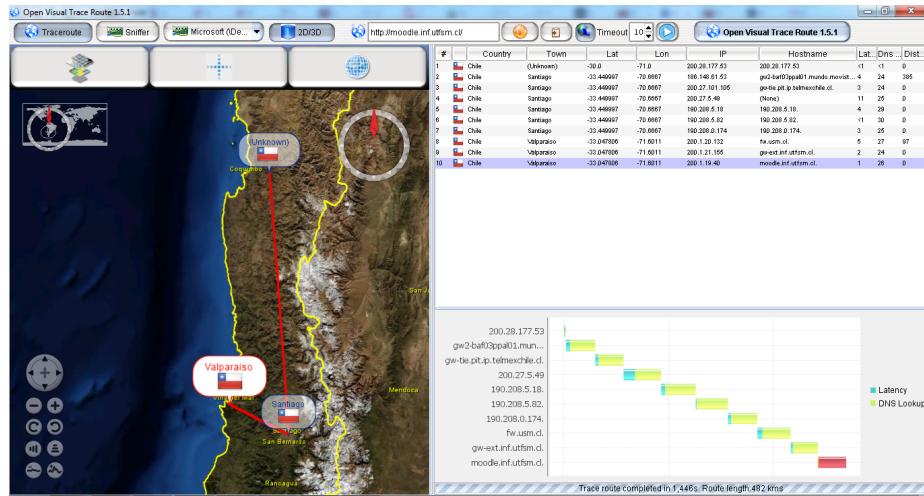


Figura 2: Trace Route de moodle.inf.utfsm.cl desde Santiago

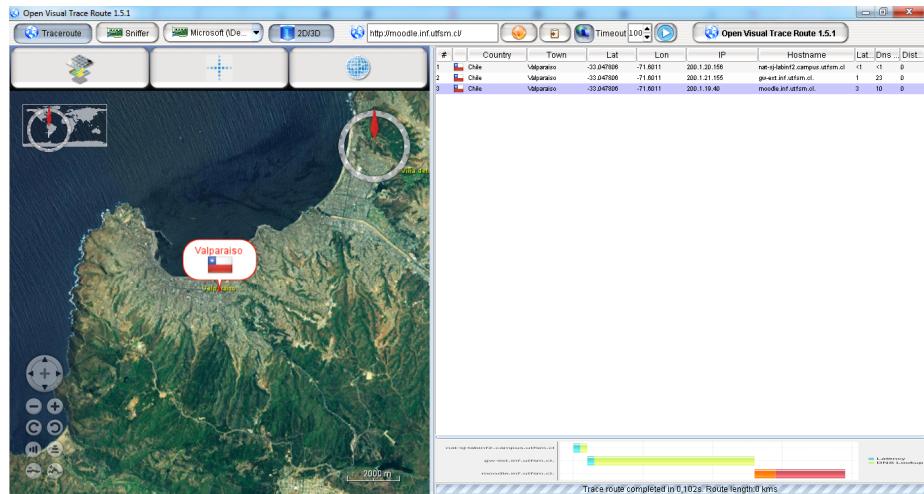


Figura 3: Trace Route de moodle.inf.utfsm.cl dentro de la red de la Universidad Técnica Federico Santa María

Google.cl

Se ingresó <http://google.cl/> a Open Visual Trace Route, el cual mostró las rutas tomadas para llevar al server final. Se puede ver que el server de Google.cl está ubicado en Mountain View, Estados Unidos, lo cual concuerda con la sede principal de Google[1].

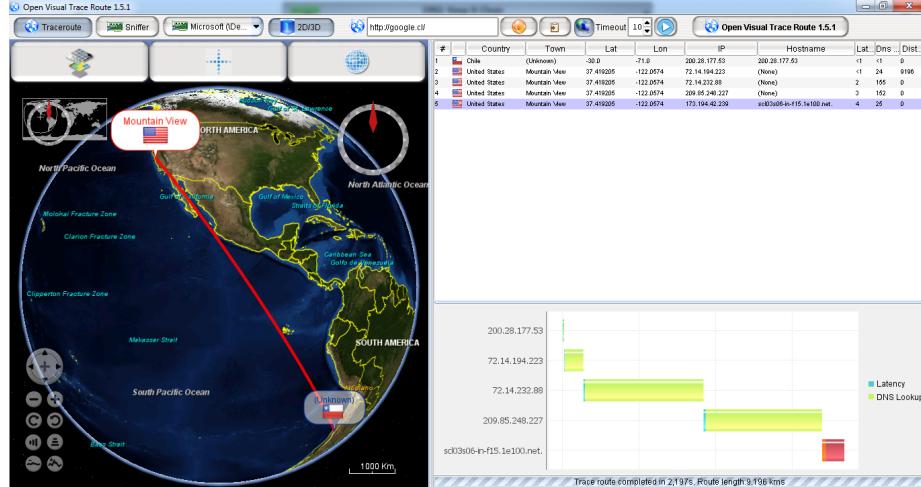


Figura 4: Trace Route de google.cl

Cime

Se ingresó <http://cime.cl/> a Open Visual Trace Route, el cual mostró las rutas tomadas para llevar al server final. Se puede ver que el server de Cime está en Nueva York, Estados Unidos.

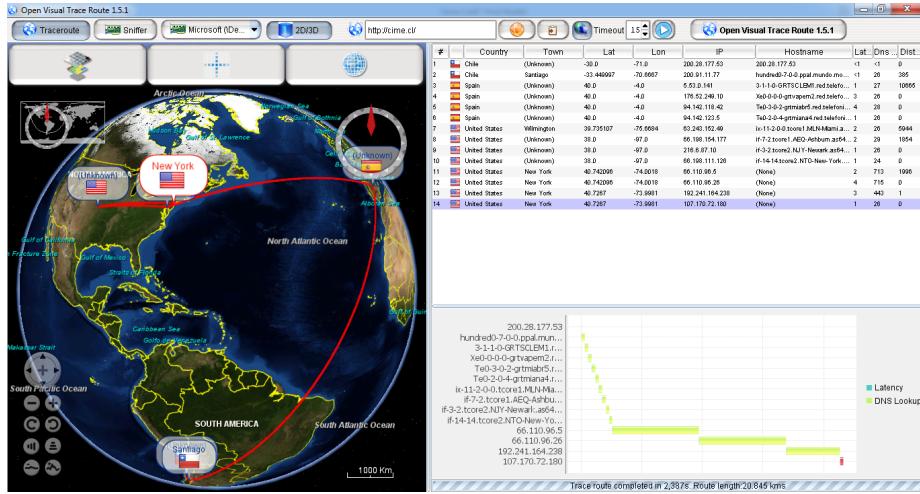


Figura 5: Trace Route de cime.cl

Wikipedia

Se ingresó <http://wikipedia.com/> a Open Visual Trace Route, el cual mostró las rutas tomadas para llevar al server final. Se puede ver que el server de Wikipedia está en San Francisco, Estados Unidos. Esto concuerda con la localización de los servidores de Wikimedia[3].

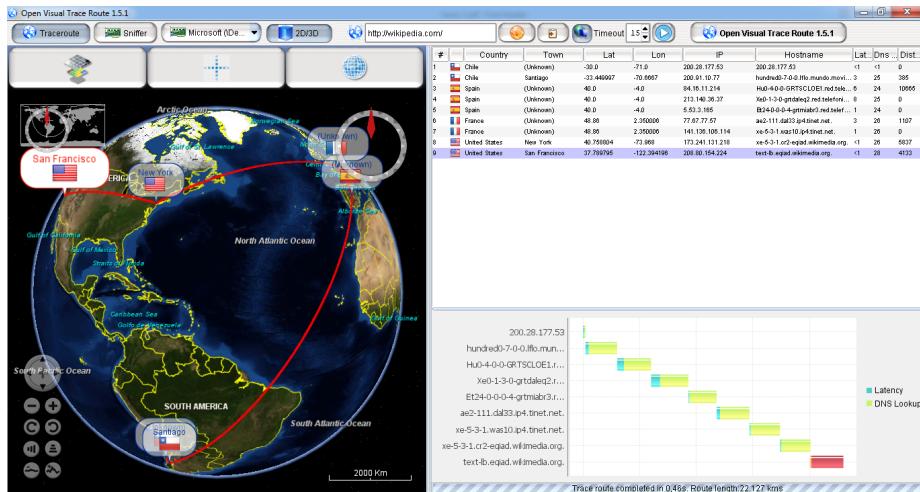


Figura 6: Trace Route de wikipedia.com

2.2. Tabla de costos de routers

Se dispone una red con nueve routers, donde el PC desea acceder a los archivos que se encuentran en el servidor, tal como se ve en la siguiente figura:

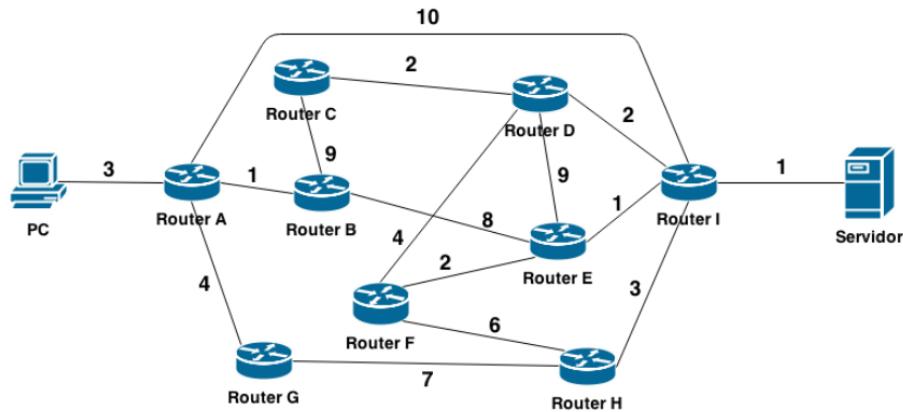


Figura 8: Red de routers

Para este fin, se debe encontrar la ruta más corta para llegar desde el PC al servidor, para lo cual se deben obtener las tablas de costos de los routers, para lo cual se usa el algoritmo vector distancia. Para obtener las tablas de costos de todos los routers mediante el algoritmo vector distancia, debemos realizar una serie de iteraciones. Para el caso de esta experiencia, y solo como a modo de ejemplo ilustrativo, se mostrara la primera inicialización de todas las tablas y se explicara el funcionamiento del algoritmo, para finalmente mostrar como quedan las tablas de costos mediante la implementación en Python del algoritmo vector distancia.

Primera Inicialización Tablas De Costos

Primero se debe inicializar la tabla de costos de cada router de la siguiente manera: Sea x el router al cual se le esta calculando la tabla de costos, en la fila de x se registran los costos para ir de x a todos los demás routers (columnas), si el router es vecino, se registra el costo, si no, se marca como ∞ al igual que todas las demás filas que no correspondan al router x .

A continuación se muestra la primera iteración para cada tabla de costos de la red ilustrada anteriormente, quedando las siguientes tablas:

Como se puede ver en la siguiente tabla, la inicialización inicial solo corresponde a los costos de ir desde el router A a sus vecinos directos. Es importante mencionar que ocurre lo mismo con todos los demás.


```

14 | costosSig=[]
15 | valor_act=[0,0,0,0,0,0,0,0,0]

```

Como se puede observar, *adyacentes* contiene toda la información acerca de las conexiones entre los routers y los costos entre ellos, tal como seria una representación de una matriz de adyacencia. En las variables *costosAct* y *costosSig* se almacenaran las tablas de costos, una para almacenar el estado actual de las tablas y otra para el estado siguiente.

Luego de tener la información base precargada, se procedió a calcular las tablas de costo iniciales, esto mediante la siguiente función:

```

1 | def init():
2 |     cpyAdy=copy.deepcopy(adyacentes)
3 |     tcost=[[[],[],[],[],[],[],[],[],[]]]
4 |     vector=[inf,inf,inf,inf,inf,inf,inf,inf,inf]
5 |     for i in lista:
6 |         for j in lista:
7 |             if i==j:
8 |                 tcost[j].append(cpyAdy[j])
9 |             else:
10 |                 tcost[j].append(vector)
11 |
12 |     return tcost

```

Posteriormente, se procedió a definir el método *dxy*, que es el que se encarga de calcular los costos aproximados $d_x(y)$ mediante el algoritmo de Bellman Ford, cuya implementación se muestra a continuación:

```

1 | def dxy(x,y,costosAct):
2 |     costosVecinos=[]
3 |     vecinos=[]
4 |     costo=0
5 |     dvy=0
6 |     for i in lista:
7 |         if adyacentes[x][i]<inf:
8 |             vecinos.append(0+i)
9 |             for ady in vecinos:
10 |                 costo=0 + adyacentes[x][ady]
11 |                 dvy=0 + costosAct[ady][ady][y]
12 |                 costosVecinos.append(costo+dvy)
13 |
14 |     return menor_lista(costosVecinos)

```

Finalmente, se implementó el método *vectorDist*, cuyo código es:

```

1 | def vectorDist(CA,CS,valor_act):
2 |     vecinos=[]
3 |     for i in range(9):
4 |         for j in range(9):
5 |             valor=dxy(i,j,CA)
6 |             if valor<CA[i][i][j] and valor<CS[i][i][j]:
7 |                 CS[i][i][j]=valor
8 |                 valor_act[i]=1

```

```

9         for k in lista:
10            if adyacentes[i][k]<inf and adyacentes[i][k]>0:
11                vecinos.append(0+k)
12            for v in vecinos:
13                CS[i][v]=copy.deepcopy(CA[v][v])
14        for l in lista:
15            CA[l]=copy.deepcopy(CS[l])
16        continuar=sumalist(valor_act)
17        aux=copy.deepcopy(valor_act)
18        while(continuar>0):
19            for i in lista:
20                v=[]#no hay vecinos
21                valor_act[i]=0
22                for k in lista:
23                    if adyacentes[i][k]<inf and adyacentes[i][k]>0:
24                        vecinos.append(0+k)
25                    for v in vecinos:
26                        if aux[v]>0:
27                            CS[i][v]=copy.deepcopy(CA[v][v])
28                            for j in lista:
29                                valor=dxy(i,j,CA)
30                                if valor<CA[i][i][j] and valor<CS ... :
31                                    CS[i][i][j]=valor
32                                    valor_act[i]=valor_act[i]+1
33            for l in lista:
34                CA[l]=copy.deepcopy(CS[l])
35            continuar=sumalist(valor_act)
36            aux=copy.deepcopy(valor_act)

```

La función anterior funciona de la siguiente forma:

- Primero inicializa los datos de las tablas de costos iniciales, y registra que todos los nodos se han actualizado.
- Luego itera nodo por nodo revisando si sus vecinos se han modificado, de ser así, entonces actualiza los D_v (vector distancia de sus vecinos) en su tabla de costos y calcula si con la nueva información su vector de distancias cambia, de ser efectivo registra que se modificó, y se continua iterando con los otros routers. Así sucesivamente hasta que pase una iteración en que ningún router modifique su tabla de costos.

Finalmente, el algoritmo genera las tablas de costos de todos los routers, las cuales resultaron ser las siguientes:

Python 3.4.1 Shell

Router D	A	B	C	D	E	F	G	H	I
D	0	1	10	12	9	11	4	11	10
B	1	0	9	11	8	10	5	12	9
C	10	9	0	2	5	6	14	7	4
D	12	11	2	0	3	4	12	5	2
E	9	8	5	3	0	2	11	4	1
F	11	10	6	4	2	0	13	6	3
G	4	5	14	12	11	13	0	7	10
H	11	12	7	5	4	6	7	0	3
I	10	9	4	2	1	3	10	3	0

Ln: 592 Col: 6

Figura 12: Tabla de costos final del router D

Python 3.4.1 Shell

Router E	A	B	C	D	E	F	G	H	I
E	0	1	10	12	9	11	4	11	10
B	1	0	9	11	8	10	5	12	9
C	10	9	0	2	5	6	14	7	4
D	12	11	2	0	3	4	12	5	2
E	9	8	5	3	0	2	11	4	1
F	11	10	6	4	2	0	13	6	3
G	4	5	14	12	11	13	0	7	10
H	11	12	7	5	4	6	7	0	3
I	10	9	4	2	1	3	10	3	0

Ln: 592 Col: 6

Figura 13: Tabla de costos final del router E

Python 3.4.1 Shell

Router F	A	B	C	D	E	F	G	H	I
F	0	1	10	12	9	11	4	11	10
B	1	0	9	11	8	10	5	12	9
C	10	9	0	2	5	6	14	7	4
D	12	11	2	0	3	4	12	5	2
E	9	8	5	3	0	2	11	4	1
F	11	10	6	4	2	0	13	6	3
G	4	5	14	12	11	13	0	7	10
H	11	12	7	5	4	6	7	0	3
I	10	9	4	2	1	3	10	3	0

Ln: 592 Col: 6

Figura 14: Tabla de costos final del router F

Router G	A	B	C	D	E	F	G	H	I
G	0	1	10	12	9	11	4	11	10
B	1	0	9	11	8	10	5	12	9
C	10	9	0	2	5	6	14	7	4
D	12	11	2	0	3	4	12	5	2
E	9	8	5	3	0	2	11	4	1
F	11	10	6	4	2	0	13	6	3
G	4	5	14	12	11	13	0	7	10
H	11	12	7	5	4	6	7	0	3
I	10	9	4	2	1	3	10	3	0

Figura 15: Tabla de costos final del router G

Router H	A	B	C	D	E	F	G	H	I
H	0	1	10	12	9	11	4	11	10
B	1	0	9	11	8	10	5	12	9
C	10	9	0	2	5	6	14	7	4
D	12	11	2	0	3	4	12	5	2
E	9	8	5	3	0	2	11	4	1
F	11	10	6	4	2	0	13	6	3
G	4	5	14	12	11	13	0	7	10
H	11	12	7	5	4	6	7	0	3
I	10	9	4	2	1	3	10	3	0

Figura 16: Tabla de costos final del router H

Router I	A	B	C	D	E	F	G	H	I
I	0	1	10	12	9	11	4	11	10
B	1	0	9	11	8	10	5	12	9
C	10	9	0	2	5	6	14	7	4
D	12	11	2	0	3	4	12	5	2
E	9	8	5	3	0	2	11	4	1
F	11	10	6	4	2	0	13	6	3
G	4	5	14	12	11	13	0	7	10
H	11	12	7	5	4	6	7	0	3
I	10	9	4	2	1	3	10	3	0

Figura 17: Tabla de costos final del router I

Como se puede observar, al final de las iteraciones cada router conoce los costos de todos los demás, obteniéndose la misma tabla para cada router, aun cuando los routers solo se intercambiaban información con sus vecinos.

2.3. Tabla de costo de routers en caso de modificación de conexiones

Para esta parte, se estudia el caso en que se corta la conexión entre dos routers, en este caso entre router H y router I. Para estudiar este caso, se creó una versión adaptada de la función *vectorDist*, que elimina la conexión para el arreglo de routers, cuya implementación se muestra a continuación:

```
1 def vectorDistInt(CA,CS,valor_act):
2     vecinos=[]
3     CA[7][7][8]=inf
4     CA[8][8][7]=inf
5     CS[7][7][8]=inf
6     CS[8][8][7]=inf
7     adyH[8]=inf
8     adyI[7]=inf
9     adyacentes=[adyA,adyB,adyC, ...,adyH,adyI]
10    valor_act[7]=1
11    valor_act[8]=1
12    #ahora iteramos nuevamente
13    continuar=sumalist(valor_act)
14    aux=copy.deepcopy(valor_act)
15    while(continuar>0):
16        for i in lista:
17            v=[]
18            valor_act[i]=0
19            for k in lista:#determinamos los vecinos a i
20                if adyacentes[i][k]<inf and adyacentes[i][k]>0:
21                    vecinos.append(0+k)#guardarid vecinos de i
22            for v in vecinos:#para cada vecino ver si se modifco
23                if aux[v]>0:#si se modifco
24                    CS[i][v]=copy.deepcopy(CA[v][v])
25                    for j in lista:
26                        valor=dxy(i,j,CA)
27                        if valor<CA[i][i][j] and valor<CS[i][i][j]:
28                            CS[i][i][j]=valor
29                            valor_act[i]=valor_act[i]+1
30                for l in lista:
31                    CA[l]=copy.deepcopy(CS[l])
32            continuar=sumalist(valor_act)
33            aux=copy.deepcopy(valor_act)
```

Finalmente, el algoritmo entonces genera las tablas de costos de todos los routers al romperse el enlace, que análogamente al caso anterior presentan todos los mismos valores, razón por la cual solo adjuntaremos una en este caso:

Router I	A	B	C	D	E	F	G	H	I
A	0	1	10	12	9	11	4	11	10
B	1	0	9	11	8	10	5	12	9
C	10	9	0	2	5	6	14	7	4
D	12	11	2	0	3	4	12	5	2
E	9	8	5	3	0	2	11	4	1
F	11	10	6	4	2	0	13	6	3
G	4	5	14	12	11	13	0	7	10
H	11	12	7	5	4	6	7	0	9
I	10	9	4	2	1	3	10	5	0

Figura 18: Tabla de costos final del router I con enlace a H roto

3. Conclusión

Como se pudo ver, el algoritmo vector distancia, además de ayudar a encontrar la ruta más corta para el envío de paquetes, también le da robustez a las conexiones, ya que en caso de cortes de conexiones, como fue visto en la sección 2.3, los routers informan a sus vecinos los cambios de topología, lo cual es usado para encontrar rápidamente una nueva ruta óptima. Las conexiones intercontinentales funcionan de igual forma, ya que en caso de cortes de cable, se calcula rápidamente un nuevo camino para el envío de paquetes, así no se deja a continentes enteros sin internet en caso de contratiempos.

Referencias

- [1] Google Careers. Google mountain view (global hq), Junio 2014.
- [2] Alejandro Avilés del Moral. Cables transoceánicos, Abril 2011.
- [3] Wikimedia Foundation. Wikimedia servers, Mayo 2014.
- [4] IT Blog Galido.net. How does the internet connect between continents?, Abril 2012.
- [5] PriMetrica Inc. Submarine cable landing directory, Junio 2014.
- [6] Wilson Rothman. How to fix a mysteriously ruptured undersea cable, Junio 2008.