Redes de Computadores: Tarea 3

20 de junio de 2014

Oscar Encina

Juan Pablo Escalona G.
juan.escalonag@alumnos.usm.cl
201073515-k

Pablo Albornoz N.
pablo.albornoz@alumnos.usm.cl
201073560-5

Pregunta 1. Open Visual Traceroute

A continuación se muestra una serie de pantallazos mostrando los resultados entregados por Open Visual Traceroute

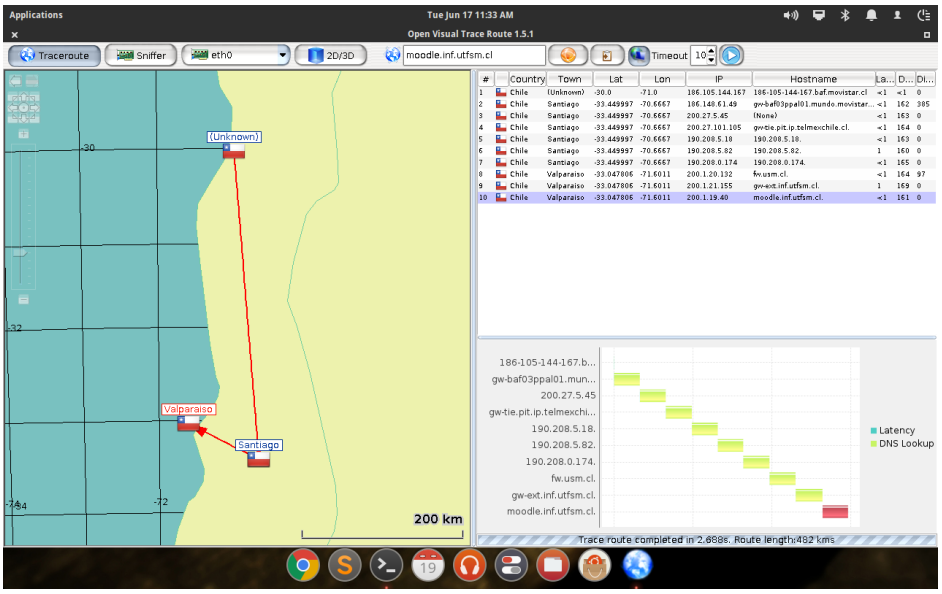

Figura 1: traceroute a http://moodle.inf.utfsm.cl

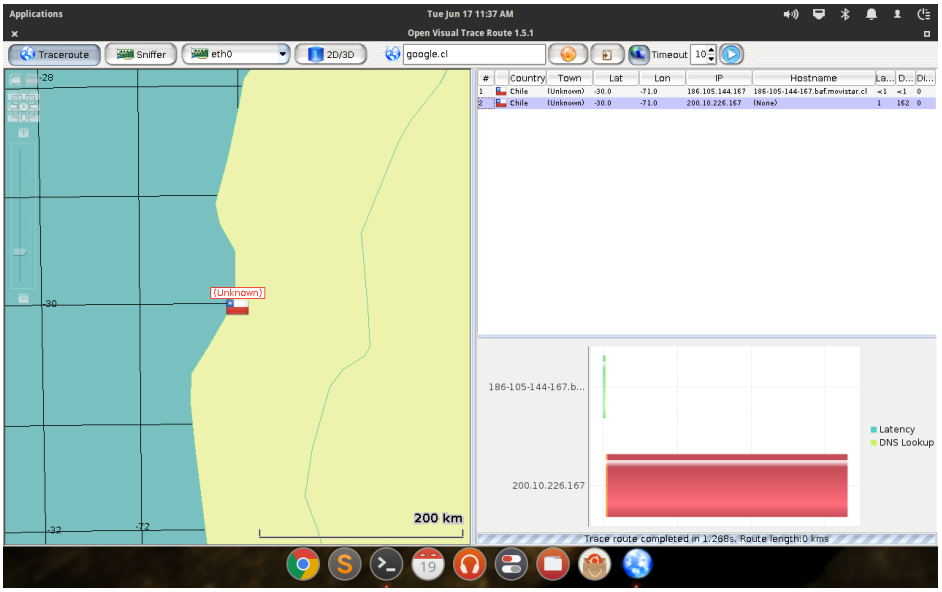

Figura 2: traceroute a http://google.cl

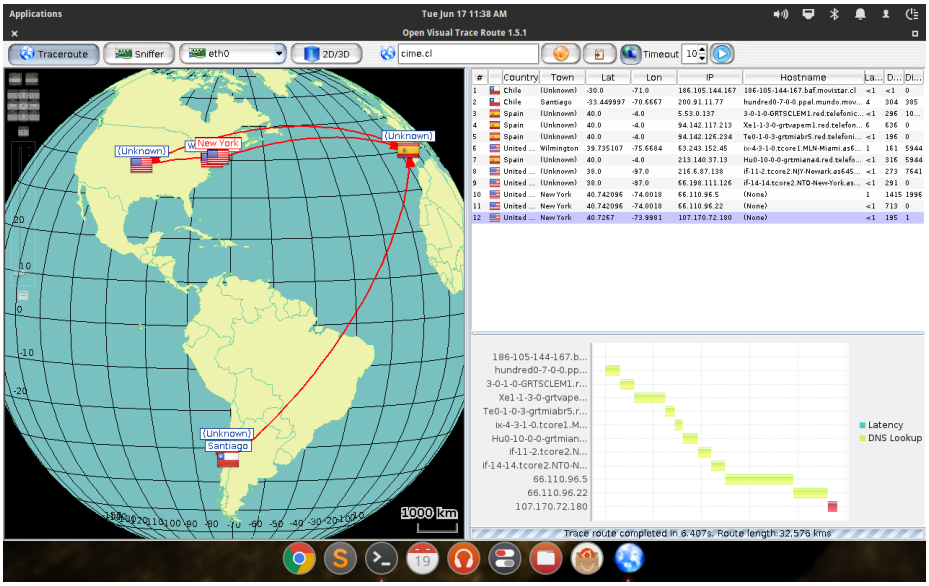

Figura 3: traceroute a http://cime.cl

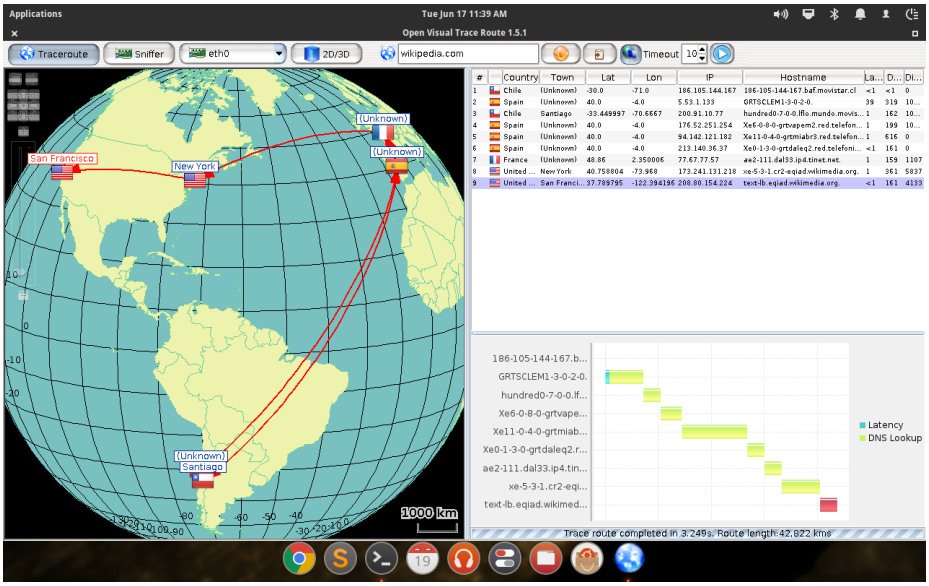

Figura 4: traceroute a http://wikipedia.com

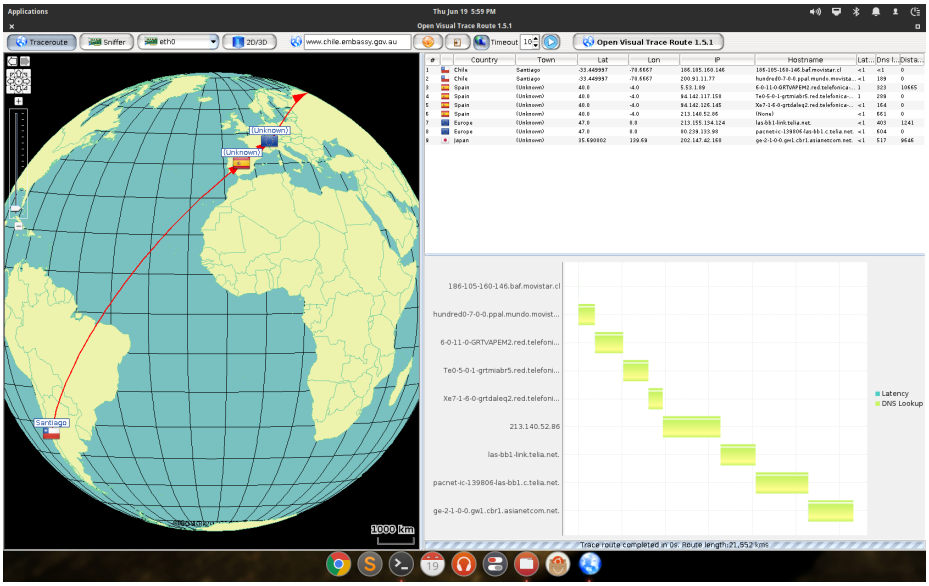

Figura 5: traceroute a http://www.chile.embassy.gov.au - Parte I

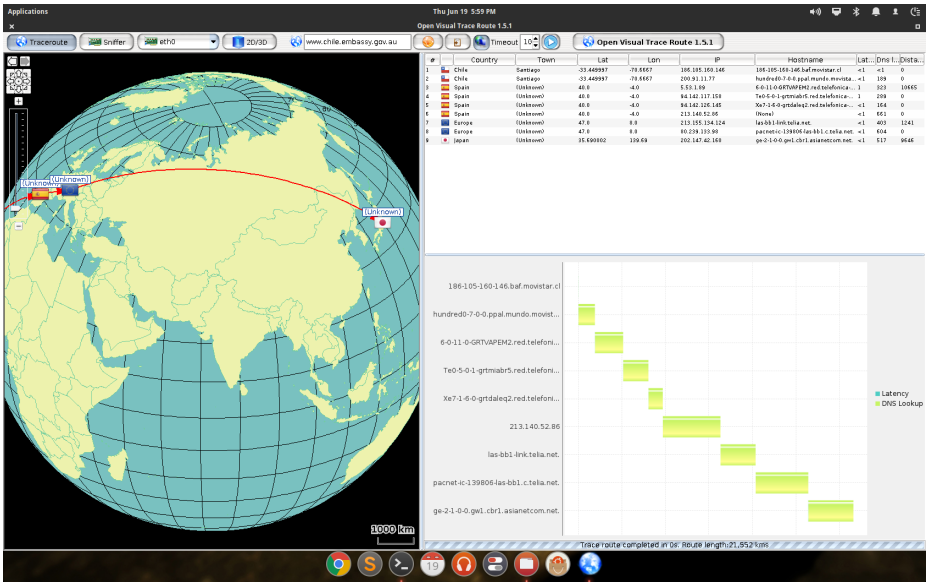

Figura 6: traceroute a http://www.chile.embassy.gov.au - Parte II

Los paquetes toman diferentes rutas según el algoritmo de vector distancia. Cada *router* sabe los mejores caminos a sus vecinos. Es por eso que en la mayoría de los casos a servidores internacionales los paquetes van a España para luego devolverse a USA.

Chile se conecta internacionalmente utilizando Cables Submarinos. Este se conecta primero a Brazil el que tiene un enlace con Europa [1]. Luego los paquetes pueden llegar fácilmente a USA como es el caso de cime.cl y wikipedia.com. En el caso de http://www.chile.embassy.gov.au/ los paquetes primero van a Brazil, luego España, USA, cruzan a Asia hasta llegar a Japón, En este punto el traceroute deja de funcionar, probablemente es bloqueado por el host japones.

En el caso de servidores nacionales como http://moodle.inf.utfsm.cl/ y http://google.cl/ se nota una

[1] http://www.submarinecablemap.com/#/submarine-cable/brazil-europe

diferencia entre la cantidad de viajes que realizan los paquetes. Para llegar a google.cl solo debe pasar por 2 hosts, a diferencia de moodle que visita 10 hosts para llegar a su destino. Se podría decir entonces que la rapidez que google es superior pues es un enlace mas directo.

Los enlaces internacionales de Chile son:

1. South American Crossing (SAC)/Latin American Nautilus (LAN) (Valparaíso)
2. Panamericano (PanAm) (Arica)
3. South America-1 (SAm-1) (Arica y Valparaíso)

Preguntas 2 y 3.

Script 1: Distance

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-

"""
Enrutamiento de Vector-Distancia con el algoritmo de Bellman-Ford
"""


class Node(object):
    """
    Representa un nodo dentro del grafo
    """
    def __init__(self, name):
        self.distance = {}
        self.next = {}
        self.name = name

    def to_string(self):
        response = "Nodo {}:[ ".format(self.name)
        for key in sorted(self.distance):
            distance = self.distance[key]
            next = self.next[key]
            response += "{}: {}, {};\t".format(
                key,
                "∞" if distance == float('inf') else distance,
                "∅" if next is None else next)
        response += "]"
        return response


class Graph(object):
    """
    Reprentación del Grafo.

    El API del grafo con que se inicializa tiene que ser la siguiente:
    iter(graph) itera sobre cada nodo del grafo
    iter(graph[u]) itera sobre cada vecino del nodo u
    graph[u][v] entrega la distancia entre el nodo u y v
    """
    def __init__(self, graph):
        self.graph = graph
        self.nodes = {}
        for node in self.graph:
```

```
            self.nodes[node] = Node(node)
            for other in self.graph:
                self.nodes[node].distance[other] = float('inf')
                self.nodes[node].next[other] = None
            self.nodes[node].distance[self.nodes[node].name] = 0

    def measure(self, node, neighbour, obj):
        """
        Mide la distancia entre nodo y su vecino y
        la almacena en la representación del nodo
        """
        new_distance = obj.distance[node] + self.graph[node][neighbour]
        if obj.distance[neighbour] > new_distance:
            obj.distance[neighbour] = new_distance
            obj.next[neighbour] = node if node != obj.name else None

    def step(self):
        """
        Itera sobre todos los nodos del grafo y hace un
        intercambio de mediciones entre los nodos una vez
        """
        for node in self.nodes:
            for u in self.graph:
                for v in self.graph[u]:
                    self.measure(u, v, self.nodes[node])

    def node_string(self):
        """
        Devuelve una cadena con la representación en pantalla
        de cada nodo. Se usa para comparar.
        """
        response = ""
        for node in sorted(self.nodes):
            response += self.nodes[node].to_string() + "\n"
        return response

    def break_link(self, one, two):
        """
        Rompe el enlace entre one y two, para luego recalcular
        las distancias.
        """
        if one in self.graph:
            if two in self.graph[one]:
                del self.graph[one][two]
        if two in self.graph:
            if one in self.graph[two]:
                del self.graph[two][one]
        if one in self.nodes:
            if two in self.nodes[one].distance:
                self.nodes[one].distance[two] = float('inf')
                self.nodes[one].next[two] = None
        if two in self.nodes:
            if one in self.nodes[two].distance:
                self.nodes[two].distance[one] = float('inf')
                self.nodes[two].next[one] = None
        for node in self.nodes:
            if self.nodes[node].next[one] == two:
                self.nodes[node].distance[one] = float('inf')
                self.nodes[node].next[one] = None
```

```
103             elif self.nodes[node].next[two] == one:
104                 self.nodes[node].distance[two] = float('inf')
105                 self.nodes[node].next[two] = None
106 
107 if __name__ == "__main__":
108     graph = {
109         'A': {'B': 1, 'G': 4, 'I': 10},
110         'B': {'A': 1, 'C': 9, 'E': 8},
111         'C': {'B': 9, 'D': 2},
112         'D': {'C': 2, 'E': 9, 'F': 4, 'I': 2},
113         'E': {'B': 8, 'D': 9, 'F': 2, 'I': 1},
114         'F': {'D': 4, 'E': 2, 'H': 6},
115         'G': {'A': 4, 'H': 7},
116         'H': {'F': 6, 'G': 7, 'I': 3},
117         'I': {'A': 10, 'D': 2, 'E': 1, 'H': 3},
118     }
119 
120     g = Graph(graph)
121 
122     print "Estado Inicial"
123     last = g.node_string()
124     print last
125     print "Comenzando Enrutamiento..."
126     breaking = False
127     i = 0
128     while True:
129         g.step()
130         now = g.node_string()
131         if last == now:
132             if not breaking:
133                 print "Primera convergencia"
134                 g.break_link('H', 'I')
135                 now = g.node_string()
136                 #   H ------/  /------ I
137                 #.  o ..
138                 #    o . o o.o
139                 #          ...oo
140                 #            __[]__
141                 #         __/_o_o_o\__
142                 #         \"""""""""/
143                 #          \. ..  . /
144                 #    ^^^^^^^^^^^^^^^^^^^
145                 print ("Nos acaban de informar que un transatlántico"
146                        " cortó el enlace H-I, recalculando...")
147                 breaking = True
148             else:
149                 break
150         last = now
151         print last
152     print "Convergencia final"
```

En el método *break_link()* del objeto *graph* definido en la línea 80 del Codigo python Distance se implementa la pregunta 3. Solo se actualizan los vecinos de H e I. A continuacion se analiza un problema con el algoritmo vector distancia.

Es importante destacar que en la pregunta 3, al eliminar el enlace entre H e I, se debe re-calcular todos los costos que utilizaban este enlace. Pero por la forma asíncrona del algoritmo se puede caer en un estado de *slow convergence to infinity* (convergencia lenta al infinito.) Pues el nuevo enlace tiene valor ∞ y al momento de re calcular las distancias, se compara con el estimado que se tenia anteriormente. Se genera el problema

entonces de seleccionar el mínimo como: $min(a, \infty)$ donde a es el valor que se tenia previo a la eliminación del enlace.Obviamente se utiliza a por se menor, pero esto induce a un error pues se esta utilizando el antiguo valor cuando el enlace estaba disponible.

Soluciones encontradas para el slow convergence to infinity:
http://intronetworks.cs.luc.edu/1/html/routing.html#slow-convergence-fixes

Referencias

1. Submarine Cable Landing Directory