Detección de enlaces mediante traceroute

Francisco Krause Arnim - LU: 99/19

May 29, 2025

1 Introducción

En este trabajo práctico, estudiaremos si es posible detectar enlaces interoceánicos basándonos únicamente en la medición de los *RTT*[1] (round-trip-time, en castellano: "el tiempo de ida y vuelta"). Para obtener estos *RTT*, vamos a imitar el funcionamiento de la herramienta unix *traceroute*[2]. Dicho funcionamiento consiste en aprovechar el hecho de que en cada paquete del protocolo *IP*[3] hay un campo llamado *TTL* que limita cuántos saltos entre enlaces puede dar un paquete. Si este llega a 0, el paquete debe descartarse y el enlace retornar un mensaje *ICMP*[4] (que forma parte de *IP*) de tipo 11 (Time Exceeded) a la dirección *IP* original que, al igual que *TTL*, es un campo de cada paquete del protocolo *IP*. Una vez obtenidos los resultados, se procederá a compararlos con bases de datos y herramientas que contienen información que relaciona una *IP* dada a su ubicación. Finalmente, usaremos como caso de estudio direcciones *IP* de 5 universidades en distintos continentes:

1. Stanford, Estados Unidos.
2. MIT, Estados Unidos.
3. Universidad de Beijing (o Peking), China.
4. Universidad de Melbourne, Australia.
5. Universidad de Helsinki, Finlandia.

[1] Wikipedia: RTT
[2] Traceroute man page
[3] RFC791
[4] RFC792

2 Experimentación

2.1 Código

2.1.1 Traceroute

El código que se utiliza en esta experimentación (que se encuentra bajo *traceroute.py*) es Python, más en particular, aprovechamos la dependencia *scapy*[5]. Simplificando y obviando boilerplate, el código que se encarga de mandar y recibir mensajes es el siguiente:

```
# max_ttl: el ttl máximo que se usará en el loop.
# times: cuántas muestras por ttl se tomarán.
# dst: ip de destino
def trace_route(max_ttl, times, dst):
    responses = {}
    for i in range(times):
        for ttl in range(1, max_ttl):
            # Objeto IP de scapy
            probe = IP(dst=dst, ttl=ttl) / ICMP()
            t_i = time()
            # Función que manda y recibe paquetes
            ans = sr1(probe, verbose=False, timeout=5)
            t_f = time()
            rtt = (t_f - t_i)*1000
            if ans[ICMP].type == 11:
                responses[ttl] = []
                responses[ttl].append(
                    Measurement(
                        ip=ans.src,
                        ttl=ttl,
                        rtt=rtt,
                        msg_type=11
                ))
    return responses
```

La idea es que vamos enviando paquetes a la IP de destino con TTLs incrementales, arrancamos desde $ttl = 0$ hasta el valor $ttl = max$, si la respuesta ICMP que recibimos es de tipo 11, nos guardamos la *IP* del enlace que respondió, el *ttl* asociado y cuándo se tardó en recibir una respuesta el *rtt*, el cual termina siendo medido en segundos, pues

[5] https://scapy.net

la función time[6] devuelve la cantidad de segundos transcurridos desde la Unix Epoch[7]. Repetimos el proceso tantas veces como lo indique la variable *times*. Esto se debe que para un mismo *ttl* podemos tener varias IPs que nos respondan, ya que es imposible predecir con certeza qué camino tomarán los paquetes que envíemos. Una vez este código termina, nos devuelve un diccionario que asocia un valor de *ttl* con una lista de muestras obtenidas.

2.1.2 Filtro de muestras.

Para filtrar y procesar lo obtenido, utilizamos 3 funciones más que toman la IP que haya respondido más veces por cada TTL, toma el promedio de las mediciones y calcula la diferencia entre cada par TTL_i y TTL_{i+1}.

2.2 Universidades de Ejemplo

[6] https://docs.python.org/3/library/time.html#time.time
[7] https://developer.mozilla.org/en-US/docs/Glossary/Unix_time