

Taller de Capa de Red

Teoría de las Comunicaciones

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

06.05.2013

Agenda

- 1 Introducción
- 2 ICMP: el protocolo de control de internet
 - ICMP desde Scapy
- 3 Traceroute: construyendo la ruta que siguen los datagramas
 - Implementaciones
 - traceroute desde Scapy

Agenda

1 Introducción

2 ICMP: el protocolo de control de internet

- ICMP desde Scapy

3 Traceroute: construyendo la ruta que siguen los datagramas

- Implementaciones
- traceroute desde Scapy

Objetivos

- Estudiar el protocolo de control de internet.
- Y algunas herramientas que se apoyan sobre esta tecnología.
- Implementar algunas de ellas.
- Analizar cómo funcionan y sacar conclusiones al respecto.
- En sí, ponernos las botas y dar una vuelta por la capa de red.

Agenda

1 Introducción

2 ICMP: el protocolo de control de internet

- ICMP desde Scapy

3 Traceroute: construyendo la ruta que siguen los datagramas

- Implementaciones
- traceroute desde Scapy

El protocolo ICMP

- Protocolo de control que forma parte del núcleo de la arquitectura TCP/IP.
- La sigla: *Internet Control Message Protocol*.
- Objetivo: proveer mensajes de error y de control. No intercambia datos!
- Especificado en el RFC 792.

Cómo y dónde se usa

- Del RFC: ICMP **debe** ser implementado por cada módulo IP.
- Pueden ser enviados tanto por routers como por hosts arbitrarios.
- Son generados a causa de:
 - ▶ Errores en los datagramas IP.
 - ▶ Necesidad de comunicar información de diagnóstico.
 - ▶ Necesidad de comunicar información de ruteo.
- Siempre se envían a la dirección source del datagrama IP que motivó el mensaje.

Formato de los paquetes

- Los paquetes constan de un header de 8 bytes y una sección de datos variable.
- **Header:**
 - ▶ Type (1 byte): indica el tipo del mensaje y define el formato de lo que sigue.
 - ▶ Code (1 byte): especifica el subtipo.
 - ▶ Checksum (2 bytes): usa el algoritmo de IP sobre el header más los datos del paquete ICMP.
 - ▶ Los restantes 4 bytes dependen del tipo.

Ejemplo: Echo Request (PING)

- La herramienta de diagnóstico ping usa estos mensajes (y el respectivo *Echo Reply* - tipo 0).
- En este caso, los 2 bytes restantes del header indican:
 - ▶ Identifier (1 byte): permite asociar solicitudes con respuestas.
 - ▶ Sequence Number (1 byte): ídem anterior.
- Y la sección de datos puede contener información arbitraria que debe ser devuelta en la respuesta.

Demo

Implementación de ping

Armando y enviando un Echo Request

```
>>> packet = IP(dst='www.google.com') / ICMP()  
>>> sr(pkt)  
>>> res[0][ICMP].display()  
0000 IP / ICMP 192.168.0.105 > 173.194.42.211  
      echo-request 0 ==> IP / ICMP 173.194.42.211 > 192.168.0.105  
                          echo-reply 0
```

Jugando con el TTL

Armando un paquete con TTL bajo

```
>>> sr(IP(dst='www.dc.uba.ar', ttl=1))
```

```
>>> res[0][ICMP].display()
```

```
0000 192.168.0.105 > 157.92.27.21 ip ==> IP / ICMP 192.168.0.105 > 157.92.27.21 : 0
```

Agenda

- 1 Introducción
- 2 ICMP: el protocolo de control de internet
 - ICMP desde Scapy
- 3 Traceroute: construyendo la ruta que siguen los datagramas
 - Implementaciones
 - traceroute desde Scapy

¿Qué es traceroute?

- Es una herramienta de diagnóstico para averiguar las rutas que atraviesan los paquetes en Internet.
- La mayoría de los sistemas operativos actuales proveen alguna implementación. Ejemplos:
 - ▶ `tracert` en Windows.
 - ▶ `traceroute` en *nix.
- Al correr la herramienta, se debe indicar hacia qué host destino se desea trazar la ruta.
- La salida obtenida suele mostrar las direcciones IP de los hops sucesivos y el respectivo tiempo de respuesta esperado.

Los distintos sabores

- Existen varias maneras de implementar traceroute.
- Usualmente consisten en enviar paquetes IP donde se incrementa progresivamente el campo TTL.
- El efecto colateral de esto es recibir respuestas ICMP sucesivas informando que el tiempo de vida del paquete acaba de expirar.
- En lo que sigue describiremos dos implementaciones de traceroute:
 - ▶ Enviando paquetes ICMP de tipo *Echo Request* ajustando el TTL.
 - ▶ Utilizando las opciones de los datagramas IP (RFC 1393).

traceroute sobre ICMP

- Implementa (esencialmente) el siguiente algoritmo:
 - ➊ Sea h la IP del host destino y sea $\text{ttl} = 1$.
 - ➋ Repetir los siguientes pasos hasta obtener una respuesta ICMP de tipo *Echo Reply* por parte de h :
 - ➌ Enviar un paquete ICMP de tipo *Echo Request* al host h cuyo campo TTL en el header IP valga ttl .
 - ➍ Si se recibe una respuesta ICMP de tipo *Time Exceeded*, anotar la IP origen de dicho paquete. En otro caso, marcar como desconocido (*) el hop.
 - ➎ Incrementar ttl .

traceroute sobre ICMP: observaciones

- Usualmente suele enviarse una serie de paquetes por cada valor de `ttl` (por lo general tres).
- A través de esto, puede estimarse el tiempo medio de respuesta.
- El host origen define un timeout para esperar por cada respuesta. Pasado este intervalo, el hop actual se asume desconocido.
- Observar que las rutas no necesariamente serán siempre iguales!

traceroute utilizando opciones IP

- Problemas del enfoque anterior:
 - ▶ Se generan muchos paquetes: $\geq 2n$, siendo n la cantidad de hops.
 - ▶ La ruta puede cambiar en el transcurso del algoritmo.
- El RFC 1393 especifica un algoritmo nuevo de traceroute que utiliza las opciones IP.
- Es más eficiente: genera $n + 1$ paquetes y no sufre del cambio de rutas dado que el origen envía un único paquete.

El algoritmo básico

- La idea: enviar un paquete arbitrario con la opción IP de traceroute adjuntada.
- Cada hop intermedio notará su presencia y devolverá un paquete ICMP de tipo 30 (*Traceroute*) con información apropiada.
- Desventaja: los routers deben implementar esta nueva funcionalidad.

Formato de la opción IP

- La opción de traceroute definida en el RFC esencialmente contiene estos campos:
 - ▶ ID Number: valor arbitrario para identificar las respuestas ICMP.
 - ▶ Hop Count: número de routers a través de los cuales pasó hasta el momento el paquete original.
 - ▶ Originator IP Address: dirección IP del host que origina el traceroute. Los routers utilizan este campo para devolver las respuestas ICMP.

Formato de los paquetes ICMP (tipo 30)

- El RFC también define el formato de los paquetes ICMP de tipo 30.
- Éstos corresponden a los paquetes intermedios que los routers van enviando al host origen.
- Los campos más relevantes son los siguientes:
 - ▶ ID Number: el identificador copiado del paquete original.
 - ▶ Hop Count: el valor (actualizado) de la cantidad de hops atravesados.
- También indica la velocidad del enlace y la MTU respectiva.

La implementación nativa de Scapy

- Scapy provee una implementación propia de traceroute.
- Utiliza conceptos de nivel de transporte (puntualmente TCP).

```
>>> traceroute('www.dc.uba.ar')
*****Finished to send 30 packets.
  157.92.27.21:tcp80
1  192.168.0.3      11          10 190.220.179.1    11
2  190.246.18.1    11          11 190.220.176.34   11
6  200.89.165.117  11          12 190.220.179.122  11
7  200.89.165.1    11          14 157.92.47.13     11
8  200.89.165.250  11          15 157.92.18.21     11
9  200.49.69.165   11          16 157.92.27.21     SA
```

- 11 indica el tipo ICMP: *Time to Live Exceeded*.
- SA indica la contestación positiva del destino (SYN-ACK).

Detrás de las bambalinas

```
>>> res = sr(IP(dst="www.google.com", ttl=range(7))/ICMP(), timeout=10)
Begin emission:
****Finished to send 7 packets.
**.....
Received 14 packets, got 6 answers, remaining 1 packets
>>> res
(<Results: TCP:0 UDP:0 ICMP:6 Other:0>, <Unanswered: TCP:0 UDP:0 ICMP:0 Other:0>)
>>> help(res)

>>> help(sr)

>>> res
(<Results: TCP:0 UDP:0 ICMP:6 Other:0>, <Unanswered: TCP:0 UDP:0 ICMP:0 Other:0>)
>>> res[0].display()
0000 IP / ICMP 192.168.0.105 > 173.194.42.211 echo-request 0 ==>
0001 IP / ICMP 192.168.0.105 > 173.194.42.211 echo-request 0 ==>
0002 IP / ICMP 192.168.0.105 > 173.194.42.211 echo-request 0 ==>
0003 IP / ICMP 192.168.0.105 > 173.194.42.211 echo-request 0 ==>
0004 IP / ICMP 192.168.0.105 > 173.194.42.211 echo-request 0 ==>
0005 IP / ICMP 192.168.0.105 > 173.194.42.211 echo-request 0 ==>
```