Avvertimento

Si consiglia di lavorare in un ambiente virtualizzato per evitare di danneggiare il proprio ambiente di lavoro o introdurre problemi di sicurezza.

Si informa che, in generale, le modifiche apportate col mando `ip` sono temporanee, e vengono pertanto perse al riavvio della macchina.

Utilizziamo ip addr per determinare l'indirizzo IP dell'host;

```
$ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
group default qlen 1000
    link/ether 00:15:5d:57:78:fe brd ff:ff:ff:ff:ff:ff
    inet 172.20.4.226/20 brd 172.20.15.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:fe57:7f7c/64 scope link
       valid_lft forever preferred_lft forever
```

E utilizziamo ip route per determinare l'indirizzo ip del router di default:

```
$ip route
default via 172.20.0.1 dev eth0 proto kernel
172.20.0.0/20 dev eth0 proto kernel scope link src 172.20.4.226
```

Tramite arping determiniamo l'indirizzo MAC del router di default:

```
$sudo arping -c 1 172.20.0.1
ARPING 172.20.0.1
```

```
42 bytes from 00:15:5d:95:52:93 (172.20.0.1): index=0 time=302.200
usec

--- 172.20.0.1 statistics ---
1 packets transmitted, 1 packets received,   0% unanswered (0 extra)
rtt min/avg/max/std-dev = 0.302/0.302/0.302/0.000 ms
```

Nel seguito assumiamo:

- Indirizzo IP dell'host: 172.20.4.226
- Indirizzo IP del router di default: 172.20.0.1
- Indirizzo MAC del router di default: 00:15:5d:95:52:93

Ovviamente, negli esempi successivi si assume che usiate i valori appropriati per la vostra macchina!

Sia *ip add show* sia *ip neigh show* permettono di filtrare per interfaccia:

```
$ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
group default qlen 1000
    link/ether 00:15:5d:57:78:fe brd ff:ff:ff:ff:ff:ff
    inet 172.20.4.226/20 brd 172.20.15.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:fe57:7f7c/64 scope link
       valid_lft forever preferred_lft forever

$ip neigh show dev eth0
172.20.0.1 lladdr 00:15:5d:95:52:93 REACHABLE
```

Mandiamo un ping al router di default e verifichiamo di trovarne l'indirizzo nella tabella ARP

```
$ping -c 1 172.20.0.1
PING 172.20.0.1 (172.20.0.1) 56(84) bytes of data.
```

```
64 bytes from 172.20.0.1: icmp_seq=1 ttl=128 time=0.611 ms

--- 172.20.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.611/0.611/0.611/0.000 ms
```

L'opzione `-c` permette di indicare il numero di messaggi `ping` da inviare (nell'esempio 1).

```
$ip neigh
172.20.0.1 dev eth0 lladdr 00:15:5d:95:52:93 REACHABLE
```

Il valore ***REACHABLE*** sull'ultima colonna indica che questa entry è valida fino allo scadere del timeout di raggiungibilità.

Se fossimo stati più lenti avremmo potuto vedere un output diverso:

```
$ip neigh
172.20.0.1 dev eth0 lladdr 00:15:5d:95:52:93 STALE
```

Ora ***STALE*** indica che la entry, benché valida è sospetta, quindi al prossimo uso, la entry dovrà essere verificata. Si faccia riferimento alle slide di approfondimento su ARP per la descrizione delle transizioni di stato delle voci della tabella ARP.

Possiamo ottenere il MAC address da un indirizzo IP

```
$ip neigh get 172.20.0.1 dev eth0
172.20.0.1 dev eth0 lladdr 00:15:5d:95:52:93 STALE
```

Possiamo ottenere le voci della tabella ARP con un certo stato

```
$ip neigh show nud stale dev eth0
172.20.0.1 lladdr 00:15:5d:95:52:93 STALE
```

Si noti che lo stato va scritto in minuscolo.

Possiamo cancellare una specifica voce della tabella ARP:

```
sudo ip neigh del 172.20.0.1 dev eth0
```

Possiamo aggiungere una specifica voce della tabella ARP:

```
$sudo ip neigh add 172.20.0.1 lladdr 00:15:5d:95:52:93 dev eth0
```

Questa voca sarà aggiunta con stato PERMANENT e non sarà mai cancellata (neanche da flush).

A meno che non si indica uno stato diverso (anche nella replace):

```
$sudo ip neigh add 172.20.0.1 lladdr 00:15:5d:95:52:93 nud reachable
dev eth0
```

Possiamo sostituire una specifica voce della tabella ARP:

```
$sudo ip neigh add 172.20.0.1 lladdr 00:15:5d:95:52:93 dev eth0
```

Oppure possiamo cancellare tutte le voci (non permanenti) della tabella ARP

```
sudo ip neigh flush dev eth0
```

Eventualmente possiamo aggiungere un filtro sullo stato,

```
sudo ip neigh flush nud stale dev eth0
```

Usando il comando arping possiamo fare un “ping” con ARP

Utile per debuggare problemi con IP assegnati a più interfacce. Come nel caso di `ping -c`, serve a indicare opzionalmente il numero di messaggi.

```
$sudo arping -c 1 172.20.0.1
ARPING 172.20.0.1
42 bytes from 00:15:5d:95:52:93 (172.20.0.1): index=0 time=328.600
usec

--- 172.20.0.1 statistics ---
```

```
1 packets transmitted, 1 packets received,   0% unanswered (0 extra)
rtt min/avg/max/std-dev = 0.329/0.329/0.329/0.000 ms
```

Si noti che **`arping` non crea una voce nella tabella ARP**: basta aver svuotato la tabella ARP prima di eseguire `arping`, e dopo verificare nuovamente la tabella con `ip neigh`. **Tuttavia, se già c'è una voce stale, l'uso di `arping` ne causa l'aggiornamento**.

Annuncio ARP

```
$sudo arping -A 172.20.4.226
ARPING 172.20.4.226
Timeout
Timeout
^C
--- 172.20.4.226 statistics ---
3 packets transmitted, 0 packets received, 100% unanswered (0 extra)
Usare come destinatario il proprio indirizzo IP.
```

Dobbiamo mettere il nostro indirizzo IP (da annunciare) come destinazione

Consentire la creazione di entry nella tabella ARP in caso di gratuitous arp messages

```
sudo cat /proc/sys/net/ipv4/conf/eth0/arp_accept
```

dovrebbe stampare 0, il valore di default.

Se scriviamo 1, attiviamo il comportamento in oggetto (attenzione alla sicurezza!!!!)

```
sudo su -c "echo 1 > /proc/sys/net/ipv4/conf/eth0/arp_accept"
```

(vedi https://sysctl-explorer.net/net/ipv4/arp_accept/)

Si consideri il caso in cui la tabella ARP è vuota. Quindi eseguiamo:

```
sudo arping -c 1 172.20.0.1
```

A questo punto stampando la tabella ARP dovremmo trovare una entry per 172.20.0.1 (contrariamente a quello che si è visto prima!)

Visualizzare la rotta per un certo indirizzo

Mandiamo un ping a www.google.it e ne registriamo l'indirizzo IP (nell'esempio: **142.250.180.131)**:

```
$ping -c 1 www.google.it
PING www.google.it (142.250.180.131) 56(84) bytes of data.
64 bytes from mil04s43-in-f3.1e100.net (142.250.180.131): icmp_seq=1
ttl=114 time=21.1 ms

--- www.google.it ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 21.080/21.080/21.080/0.000 ms
```

Chiediamo la rotta per l'indirizzo IP di www.google.it (**142.250.180.131**)

```
$ip route get 142.250.180.131
142.250.180.131 via 172.20.0.1 dev eth0 src 172.20.4.226 uid 1000
    Cache
```

Cancellare una rotta (es. la rotta di default)

```
$sudo ip route del default dev eth0
$ip route
172.20.0.0/20 dev eth0 proto kernel scope link src 172.20.4.226
```

In assenza di una rotta di default, non riusciremo più a mandare un ping a www.google.it.

```
$ping -c 1 142.250.180.131
ping: connect: Network is unreachable
```

Infatti, chiedendo la rotta:

```
$ip route get 142.250.180.131
RTNETLINK answers: Network is unreachable
```

Aggiungere una rotta (es. la rotta di default)

$sudo ip route add default via 172.20.0.1 dev eth0
$ip route
default via 172.20.0.1 dev eth0
172.20.0.0/20 dev eth0 proto kernel scope link src 172.20.4.226

Per aggiungere una rotta per una destinazione specifica sarebbe stato sufficiente sostituire *default* con il prefisso di destinazione in formato CIDR.

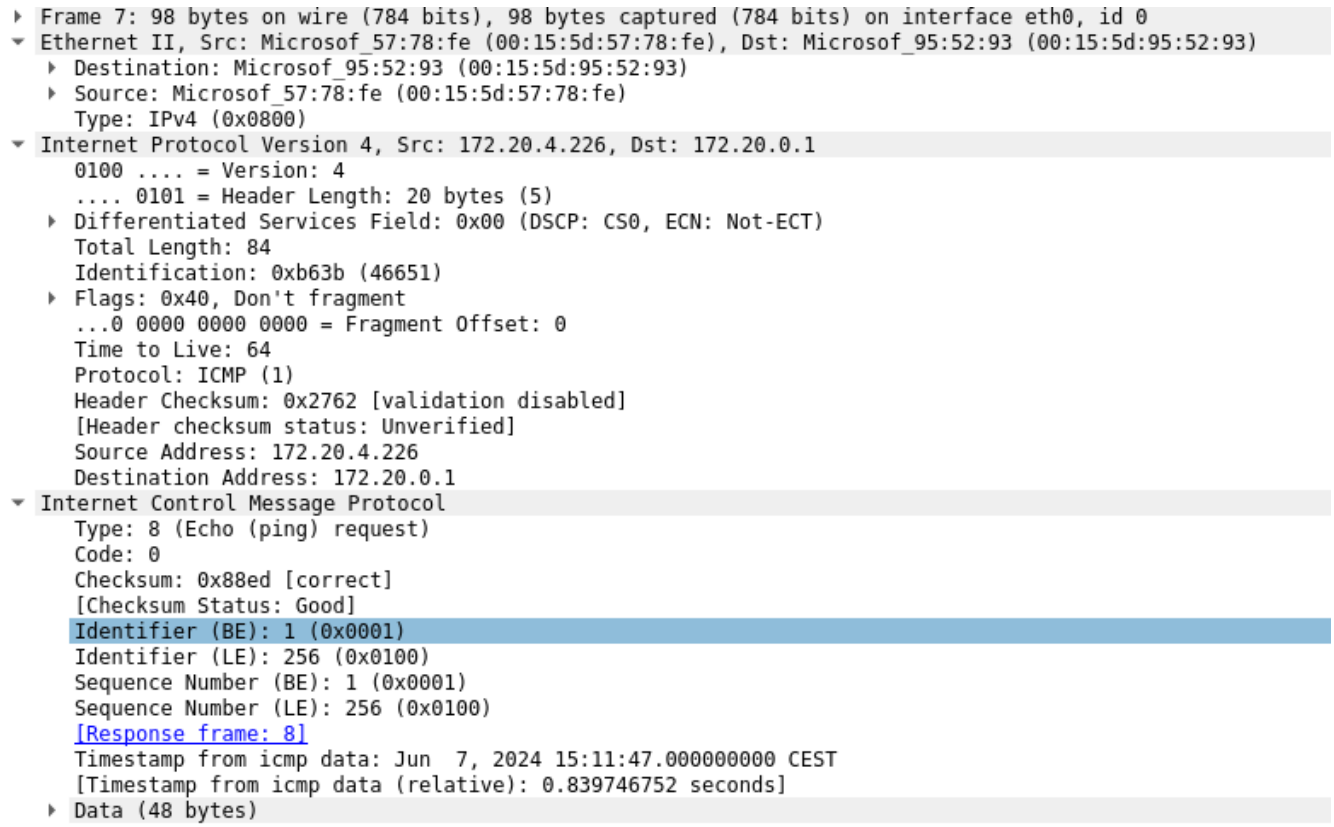
Ora è nuovamente possibile pingare www.google.it

```
$ping -c 1 www.google.it
PING www.google.it (216.58.205.35) 56(84) bytes of data.
64 bytes from mil07s19-in-f3.1e100.net (216.58.205.35): icmp_seq=1
ttl=114 time=19.4 ms

--- www.google.it ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 19.350/19.350/19.350/0.000 ms
```

Cose da verificare con Wireshark:

- Pacchetti inviati/ricevuti con ping al router di default (stessa sottorete).
  Richiesta echo (ping):



Mittente frame: interfaccia dell'host
Destinatario frame: interfaccia router di default
IP mittente: IP dell'interfaccia dell'host
IP destinazione: IP dell'interfaccia del router
Si vedano tipo e codice nell'intestazione ICMP.
Si vedano anche identificatore e numero di sequenza.

Risposta echo (ping):
Nella risposta mittente e destinatario sono invertiti. Il codice e il tipo sono diversi, mentre numeri di sequenza e identificatore sono lo stesso. In effetti, ping successivi avranno stesso identificatore ma numeri di sequenza differenti.

```
▸ Frame 8: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth0, id 0
▾ Ethernet II, Src: Microsof_95:52:93 (00:15:5d:95:52:93), Dst: Microsof_57:78:fe (00:15:5d:57:78:fe)
  ▸ Destination: Microsof_57:78:fe (00:15:5d:57:78:fe)
  ▸ Source: Microsof_95:52:93 (00:15:5d:95:52:93)
    Type: IPv4 (0x0800)
▾ Internet Protocol Version 4, Src: 172.20.0.1, Dst: 172.20.4.226
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0xe313 (58131)
  ▸ Flags: 0x00
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: ICMP (1)
    Header Checksum: 0xfa89 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.20.0.1
    Destination Address: 172.20.4.226
▾ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x90ed [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence Number (BE): 1 (0x0001)
    Sequence Number (LE): 256 (0x0100)
    [Request frame: 7]
    [Response time: 0.554 ms]
    Timestamp from icmp data: Jun  7, 2024 15:11:47.000000000 CEST
    [Timestamp from icmp data (relative): 0.840300552 seconds]
  ▸ Data (48 bytes)
```

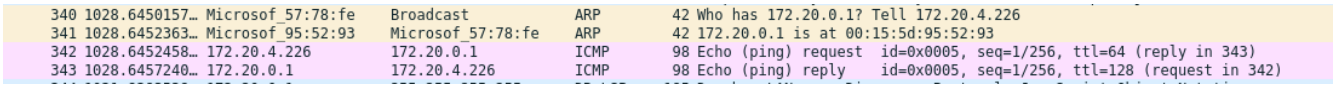
- Pacchetti inviati/ricevuti con ping a www.google.it

  Richiesta echo (ping):

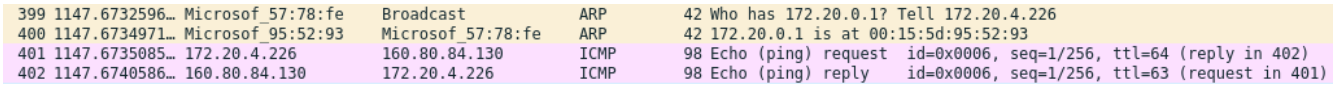
```
▸ Frame 147: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth0, id 0
▾ Ethernet II, Src: Microsof_57:78:fe (00:15:5d:57:78:fe), Dst: Microsof_95:52:93 (00:15:5d:95:52:93)
  ▸ Destination: Microsof_95:52:93 (00:15:5d:95:52:93)
  ▸ Source: Microsof_57:78:fe (00:15:5d:57:78:fe)
    Type: IPv4 (0x0800)
▾ Internet Protocol Version 4, Src: 172.20.4.226, Dst: 142.251.209.3
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0xfd44 (64836)
  ▸ Flags: 0x40, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0x2c6f [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.20.4.226
    Destination Address: 142.251.209.3
▾ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x13e6 [correct]
    [Checksum Status: Good]
    Identifier (BE): 2 (0x0002)
    Identifier (LE): 512 (0x0200)
    Sequence Number (BE): 1 (0x0001)
    Sequence Number (LE): 256 (0x0100)
    [Response frame: 148]
    Timestamp from icmp data: Jun  7, 2024 15:20:20.000000000 CEST
    [Timestamp from icmp data (relative): 0.251065194 seconds]
  ▸ Data (48 bytes)
```

A livello 2, gli indirizzi mittente e destinatario sono gli stessi di prima. Al contrario, a livello 3 il destinatario è ora l'indirizzo IP di www.google.it
Si noti che lo screenshot è stato fatto un giorno successivo e luogo differente, pertanto l'indirizzo IP di www.google.it è diverso.

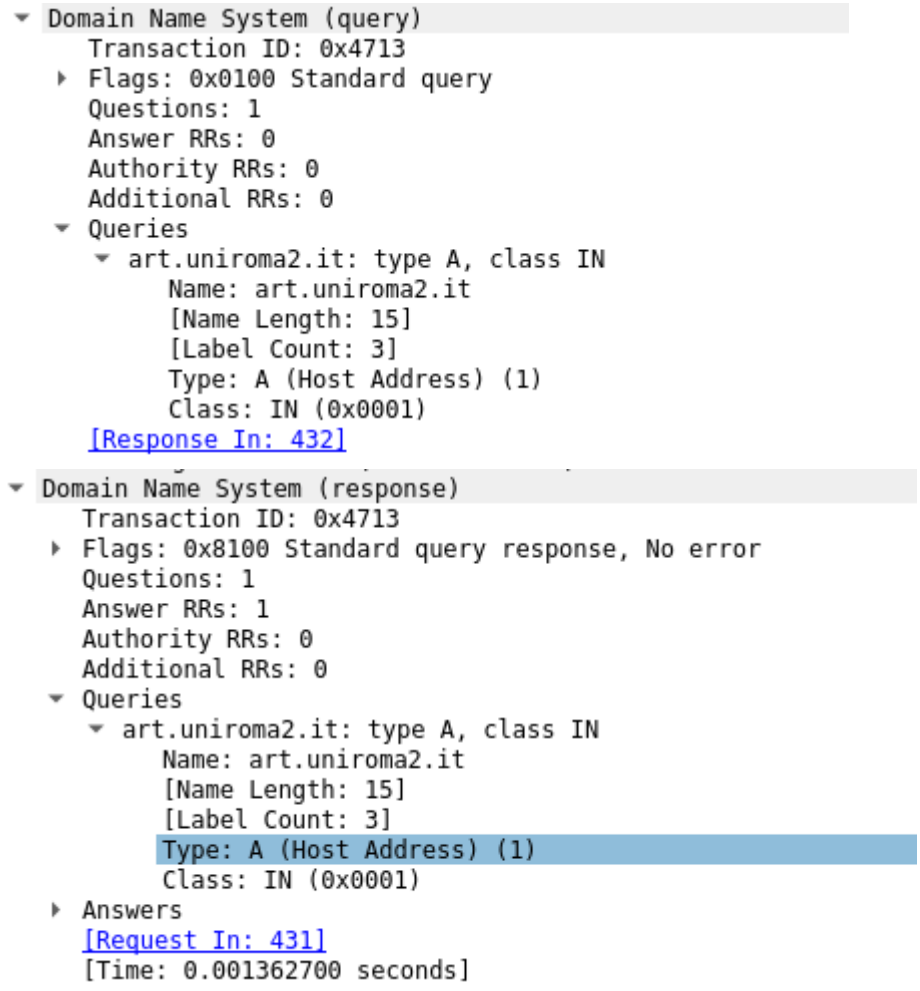
- **Partendo da tabella ARP vuota**, osservare che l'effettivo invio di un ping la cui destinazione è specificata come indirizzo IP nella stessa sottorete (es. router di default) è preceduto da uno scambio col protocollo ARP per la risoluzione dell'indirizzo IP



Nel caso di una destinazione esterna alla sottorete, ci sarebbe stata una richiesta ARP?
Sì, per il router cui viene inoltrato il pacchetto: es. `ping -c 1 160.80.84.130`.

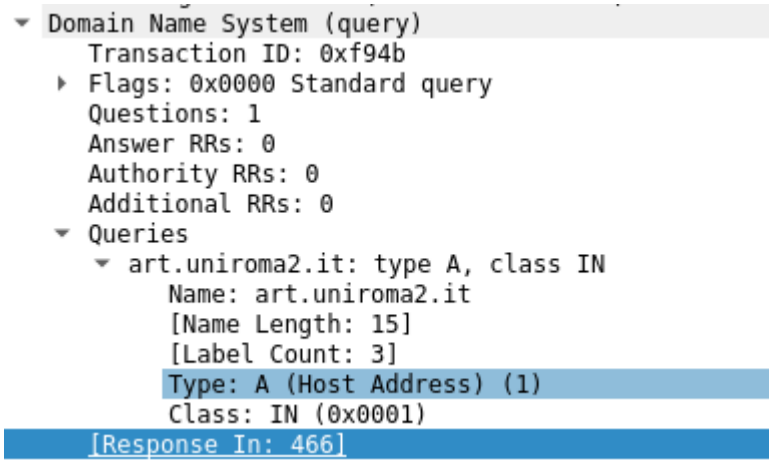


- Analizzare i pacchetti scambiati a seguito dei comandi seguenti. Negli screenshot ci concentriamo sul messaggio DNS, ma non trascurate di vedere l'incapsulamento delle varie PDU!
  - nslookup -debug art.uniroma2.it



- nslookup -debug -norecurse art.uniroma2.it dns.nic.it

  Probabilmente troveremo prima uno scambio per risolvere il dominio dns.nic.it in un indirizzo IP. Poi troveremo lo scambio relativo alla nostra domanda.



  Nella risposta vediamo come sono implementati i referral: non abbiamo Answer RR, bensì uno o più Autority RR che ci forniscono l'host name di un DNS server (sono record NS) cui viene delegato il compito di fornire una risposta. Tra gli Additional Records, troviamo record di tipo A che ci forniscono l'indirizzo IP dei name server indicati.
- nslookup -debug -norecurse art.uniroma2.it dns.uniroma2.it
  Si che nella riposte il server autoritativo per il dominio art.uniroma2.it includerà sia un Answer RR, sia Autority e Additiuonal RRs che descrivono i nameserver per quel dominio.

```
▾ Domain Name System (response)
    Transaction ID: 0xf94b
  ▸ Flags: 0x8000 Standard query response, No error
    Questions: 1
    Answer RRs: 0
    Authority RRs: 3
    Additional RRs: 2
  ▾ Queries
    ▾ art.uniroma2.it: type A, class IN
        Name: art.uniroma2.it
        [Name Length: 15]
        [Label Count: 3]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
  ▾ Authoritative nameservers
    ▾ uniroma2.it: type NS, class IN, ns dns.uniroma2.it
        Name: uniroma2.it
        Type: NS (authoritative Name Server) (2)
        Class: IN (0x0001)
        Time to live: 3600 (1 hour)
        Data length: 6
        Name Server: dns.uniroma2.it
    ▾ uniroma2.it: type NS, class IN, ns dns1.uniroma2.it
        Name: uniroma2.it
        Type: NS (authoritative Name Server) (2)
        Class: IN (0x0001)
        Time to live: 3600 (1 hour)
        Data length: 7
        Name Server: dns1.uniroma2.it
    ▾ uniroma2.it: type NS, class IN, ns ns1.garr.net
        Name: uniroma2.it
        Type: NS (authoritative Name Server) (2)
        Class: IN (0x0001)
        Time to live: 3600 (1 hour)
        Data length: 14
        Name Server: ns1.garr.net
  ▾ Additional records
    ▾ dns1.uniroma2.it: type A, class IN, addr 160.80.5.8
        Name: dns1.uniroma2.it
        Type: A (Host Address) (1)
        Class: IN (0x0001)
        Time to live: 3600 (1 hour)
        Data length: 4
        Address: 160.80.5.8
    ▾ dns.uniroma2.it: type A, class IN, addr 160.80.1.3
        Name: dns.uniroma2.it
        Type: A (Host Address) (1)
        Class: IN (0x0001)
        Time to live: 3600 (1 hour)
        Data length: 4
        Address: 160.80.1.3
    [Request In: 465]
    [Time: 0.027365901 seconds]
```