

# Monitoraggio comunicazioni ARP

Paolo Prezioso

11 gennaio 2019

# Indice

<b>1</b>	<b>Introduzione</b>	<b>2</b>
<b>2</b>	<b>Protocollo ARP</b>	<b>2</b>
2.1	Struttura pacchetti . . . . .	2
2.2	Funzionamento protocollo . . . . .	3
2.3	Utilizzi alternativi . . . . .	4
2.4	Protocolli inversi . . . . .	4
<b>3</b>	<b>Libreria pyshark</b>	<b>5</b>
<b>4</b>	<b>Implementazione</b>	<b>5</b>
4.1	Risultati . . . . .	5
<b>5</b>	<b>Istruzioni per l'esecuzione</b>	<b>6</b>
5.1	Prerequisiti . . . . .	6
5.2	Esecuzione . . . . .	6
5.3	Interfacce . . . . .	7

# 1 Introduzione

Lo scopo del presente progetto è quello di ottenere una matrice delle comunicazioni ARP che avvengono su di una rete.

## 2 Protocollo ARP<sup>1</sup>

Il protocollo ARP (*Address Resolution Protocol*) è un protocollo di rete usato per associare, ad un indirizzo IPv4, l'indirizzo MAC, di livello datalink, della macchina corrispondente.

Come detto, è un protocollo che lavora solo su IPv4: in IPv6 esiste *Neighbor Discovery Protocol* (NDP), che è un protocollo equivalente.

ARP comunica solamente all'interno della sottorete nella quale viene generata la richiesta e non viene mai instradato al di là di essa. Per questo ARP rientra tra i protocolli di livello datalink dello stack TCP/IP.

### 2.1 Struttura pacchetti

Il protocollo ARP utilizza un sistema di messaggi *richiesta-risposta* dal formato molto semplice. La dimensione dei messaggi dipende in gran parte dalla dimensione degli indirizzi contenuti.

L'*header* dei messaggi contiene informazioni sui livelli di rete in uso sul richiedente, l'indicazione delle dimensioni degli indirizzi e il tipo di messaggio (richiesta/risposta).

Il *payload* del messaggio, invece, contiene quattro indirizzi: quelli di livello di rete e quelli fisici dei due host coinvolti.

Nella *Figura 1* è mostrata la struttura generale di un pacchetto ARP su una rete che usa IPv4 su Ethernet.

**Hardware (HTYPE)** Specifica protocollo di livello datalink (es. Ethernet).

**Protocollo (PTYPE)** Specifica protocollo di livello di rete (es. IPv4).

**Lunghezza indirizzo hardware (HLEN)** Indica lunghezza degli indirizzi fisici.

**Lunghezza indirizzo protocollo (PLEN)** Indica lunghezza degli indirizzi di livello di rete.

**Operazione** Indica se il messaggio è una richiesta (1) o una risposta (2).

---

<sup>1</sup>Fonte: [https://en.wikipedia.org/wiki/Address\\_Resolution\\_Protocol](https://en.wikipedia.org/wiki/Address_Resolution_Protocol)

**Indirizzo hardware mittente (SHA)** Indirizzo fisico del mittente del messaggio: in caso di operazione di risposta, contiene l'indirizzo cercato nella richiesta.

**Indirizzo protocollo mittente (SPA)** Indirizzo di livello di rete del mittente del messaggio.

**Indirizzo hardware destinatario (THA)** Indirizzo fisico del destinatario del messaggio: in caso di operazione di richiesta, in questo campo c'è l'indirizzo di broadcast  $ff:ff:ff:ff:ff:ff$ .

**Indirizzo protocollo destinatario (TPA)** Indirizzo di livello di rete del destinatario del messaggio.

Internet Protocol (IPv4) over Ethernet ARP packet		
octet offset	0	1
0	Hardware type (HTYPE)	
2	Protocol type (PTYPE)	
4	Hardware address length (HLEN)	Protocol address length (PLEN)
6	Operation (OPER)	
8	Sender hardware address (SHA) (first 2 bytes)	
10	(next 2 bytes)	
12	(last 2 bytes)	
14	Sender protocol address (SPA) (first 2 bytes)	
16	(last 2 bytes)	
18	Target hardware address (THA) (first 2 bytes)	
20	(next 2 bytes)	
22	(last 2 bytes)	
24	Target protocol address (TPA) (first 2 bytes)	
26	(last 2 bytes)	

Figura 1: Struttura di un messaggio ARP su una rete che usa IPv4 su Ethernet.

## 2.2 Funzionamento protocollo

Il funzionamento standard del protocollo ARP prevede che un host, che voglia inviare un pacchetto conoscendo solo l'indirizzo IP del destinatario, invii una richiesta ARP in broadcast a tutti i suoi vicini.

Chi riceve una richiesta ARP ha due possibilità:

- non verifica una corrispondenza tra il proprio indirizzo IP e quello indicato nel campo destinatario della richiesta, nel qual caso ignora e scarta il pacchetto;

- verifica una corrispondenza tra il proprio indirizzo IP e quello indicato nel campo destinatario della richiesta, nel qual caso invia una risposta ARP, indicando nell'apposito campo il proprio indirizzo fisico.

Inoltre, chi riceve una richiesta ARP ha in tutti i casi la possibilità di salvare nella propria cache ARP i dati relativi all'host che ha inviato la richiesta.

Il richiedente, una volta ricevuta la risposta ARP, leggerà l'indirizzo fisico del mittente e lo inserirà nel frame Ethernet che doveva inviare in origine.

## 2.3 Utilizzi alternativi

Oltre a quello standard, vi sono altri possibili modi di utilizzare il protocollo.

**Sonda ARP** È possibile inviare una richiesta ARP in broadcast, cioè senza specificare l'IP del destinatario. Un utilizzo del genere del protocollo lo si può riscontrare nella specifica della *IPv4 Address Conflict Detection* (RFC 5227);

**Gratuitous ARP (GARP)** Alcuni sistemi inviano dei *messaggi ARP gratuiti* in modo da aggiornare le cache ARP delle macchine vicine, anche se queste non lo hanno richiesto. Un'operazione del genere può aiutare a risolvere alcuni problemi di indirizzamento, come la sostituzione recente di una scheda di rete, aggiornando l'associazione tra indirizzi IP e MAC, o di gestione delle risorse, come l'indirizzamento verso un'altra scheda di rete cui inviare i pacchetti, in sistemi con più schede di rete cooperanti.

## 2.4 Protocolli inversi

I protocolli che compiono l'operazione inversa rispetto ad ARP, cioè ricavano indirizzi di rete a partire da indirizzi fisici sono:

- *InARP*
- *RARP*

Il primo funziona in modo molto simile ad ARP, in effetti è trattato come un'estensione dello stesso: usa lo stesso formato di messaggi, modificando solo i codici operativi, e lo scambio sulla rete avviene in modo identico.

RARP ha lo stesso scopo di InARP, con la differenza che InARP interroga il livello di rete di macchine remote, mentre RARP ricava l'indirizzo IP locale.

### 3 Libreria pyshark<sup>2</sup>

La libreria `pyshark` è un wrapper Python per *tshark*, un analizzatore di protocolli di rete, versione per linea di comando di *Wireshark*, nella cui distribuzione è inserito.

`pyshark` fornisce tutte le funzionalità messe a disposizione da *tshark* per analizzare i pacchetti di rete sia da un traffico di rete attivo, sia da uno stream precedentemente catturato e salvato.

A differenza di altre librerie Python, `pyshark` non si occupa direttamente del parsing dei pacchetti di rete, ma sfrutta la capacità di *tshark* di esportare dati in formato XML, da cui legge il risultato del parsing, che è pertanto delegato a *tshark*.

### 4 Implementazione

Per l'implementazione del programma di analisi del traffico ARP, l'operazione fondamentale che viene eseguita è la configurazione della struttura dati in cui registrare i dati in ingresso dalla rete, in modo che intercetti solo i pacchetti desiderati:

```
pyshark.LiveCapture(interface=interfaces[interface], bpf_filter='arp')
```

Come si può notare dal codice riportato, è necessario indicare a `pyshark` l'interfaccia dalla quale catturare pacchetti. Questo parametro lo può scegliere l'utente passandolo come argomento al programma al momento dell'avvio; laddove non specificata, l'interfaccia che viene analizzata di default è la *en0*, cioè l'interfaccia per la connessione wifi.

Per filtrare i pacchetti in modo da analizzare solo quelli che viaggiano su protocollo ARP, `pyshark` offre due tipologie di filtri:

**display filters** filtri che permettono più opzioni, ma sono sensibilmente più lenti nell'elaborazione;

**bpf filters** filtri più basilari, ma molto più veloci nell'elaborazione, il che li rende più adatti all'utilizzo su live stream.

Da quanto detto, si spiega perché, nel codice riportato, viene impostato un filtro *bpf* per catturare solo pacchetti ARP.

#### 4.1 Risultati

Per leggere i risultati dell'analisi dei pacchetti, basta interrompere il programma con il comando *Ctrl-C*: l'interruzione viene intercettata (*Figura 2*) e i risultati elaborati, estrapolandone le informazioni di interesse, e mostrati a video (*Figura 3*).

---

<sup>2</sup>Fonte: <https://kiminewt.github.io/pyshark/>

```

try:
    capture.sniff()
except KeyboardInterrupt:
    ...

```

Figura 2: L'interruzione del programma viene intercettata.

## 5 Istruzioni per l'esecuzione

### 5.1 Prerequisiti

**Python 3 (3.7.2)** Assicurarsi di avere Python 3 installato sulla propria macchina (per lo sviluppo è stata usata la versione 3.7.2).

**pyshark** Installare l'ultima versione della libreria **pyshark**.

**Per tutte le piattaforme:**

- installazione diretta da terminale:

```
pip3 install pyshark
```

- installazione da GitHub:

```

git clone https://github.com/KimiNewt/pyshark.git
cd pyshark/src
python3 setup.py install

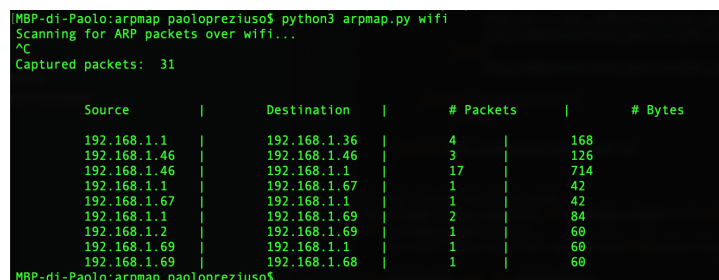
```

In caso di problemi con l'installazione di **pyshark**, si rimanda alla relativa documentazione: <https://github.com/KimiNewt/pyshark>.

### 5.2 Esecuzione

Da terminale, accedere alla directory dov'è situato il file **arpmap.py** ed eseguire

```
python3 arpmap.py <interface>
```



```

MBP-di-Paolo:arpmap paoloprezioso$ python3 arpmap.py wifi
Scanning for ARP packets over wifi...
^C
Captured packets: 31

```

Source	Destination	# Packets	# Bytes
192.168.1.1	192.168.1.36	4	168
192.168.1.46	192.168.1.46	3	126
192.168.1.46	192.168.1.1	17	714
192.168.1.1	192.168.1.67	1	42
192.168.1.67	192.168.1.1	1	42
192.168.1.1	192.168.1.69	2	84
192.168.1.2	192.168.1.69	1	60
192.168.1.69	192.168.1.1	1	60
192.168.1.69	192.168.1.68	1	60

```

MBP-di-Paolo:arpmap paoloprezioso$

```

Figura 3: Stampa risultati dell'elaborazione.

dove `<interface>` va sostituito con la chiave dell'interfaccia da esaminare. È possibile omettere il parametro, in tal caso verrà segnalato l'utilizzo dell'interfaccia wifi (*en0*), impostata come default.

### 5.3 Interfacce

Per aggiungere/rimuovere interfacce da cui catturare pacchetti: aprire il file `interfaces.py`, situato nella stessa directory di `arpmap.py`, e modificare il dizionario *interfaces* tenendo presente che:

- la **chiave** è arbitraria e costituisce il parametro `<interface>` da passare ad `arpmap.py`;
- il **valore** è il codice identificativo dell'interfaccia che si vuole monitorare quando indicata la corrispondente chiave (es. *eth0*, *eth1*, *en0*, ...).

Una lista delle interfacce attualmente disponibili è presente in *Tabella 1*.

Parametro	Interfaccia
<i>wifi</i>	Interfaccia <i>en0</i> , identificativo della connessione wifi
<i>ethernet</i>	Interfaccia <i>eth0</i> , identificativo della interfaccia ethernet primaria

Tabella 1: Interfacce disponibili per analisi ARP.