

# Utilities

Insegnamento di Sistemi Operativi di Rete  
Master Universitario in Tecnologie Internet

**Domenico De Guglielmo**

E-mail: [domenicodegu@virgilio.it](mailto:domenicodegu@virgilio.it) Telefono: 050 2217468  
Pervasive Computing & Networking Lab (PerLab) <http://www.perlab.it>  
Dipartimento di Ingegneria dell'Informazione, Università di Pisa

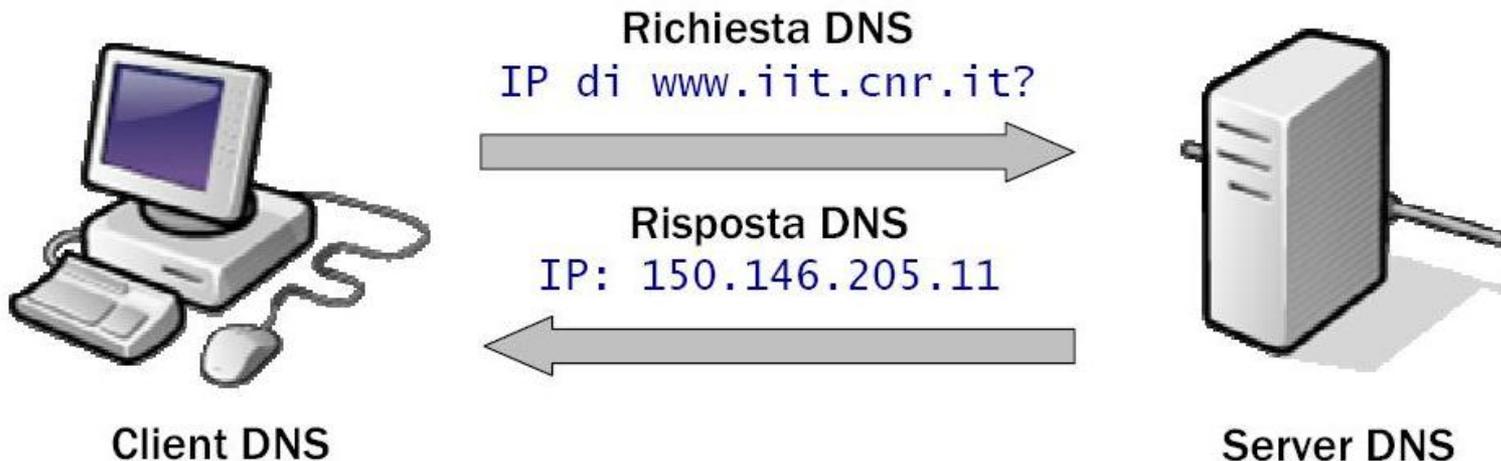
# **Nomi di host e risoluzione degli indirizzi**

- La gestione diretta degli indirizzi IP in forma numerica può essere utile in fase di progetto di una rete, ma a livello di utente è una pretesa praticamente inaccettabile.
- Per questo, agli indirizzi IP numerici si affiancano quasi sempre dei nomi.
- Ogni volta che si fa riferimento a un nome, il sistema è (o dovrebbe essere) in grado di convertirlo nel numero IP corrispondente.
- Ci sono due metodi per trasformare un nome in un indirizzo IP e viceversa:
  - un elenco contenuto nel file `/etc/hosts`
  - server DNS

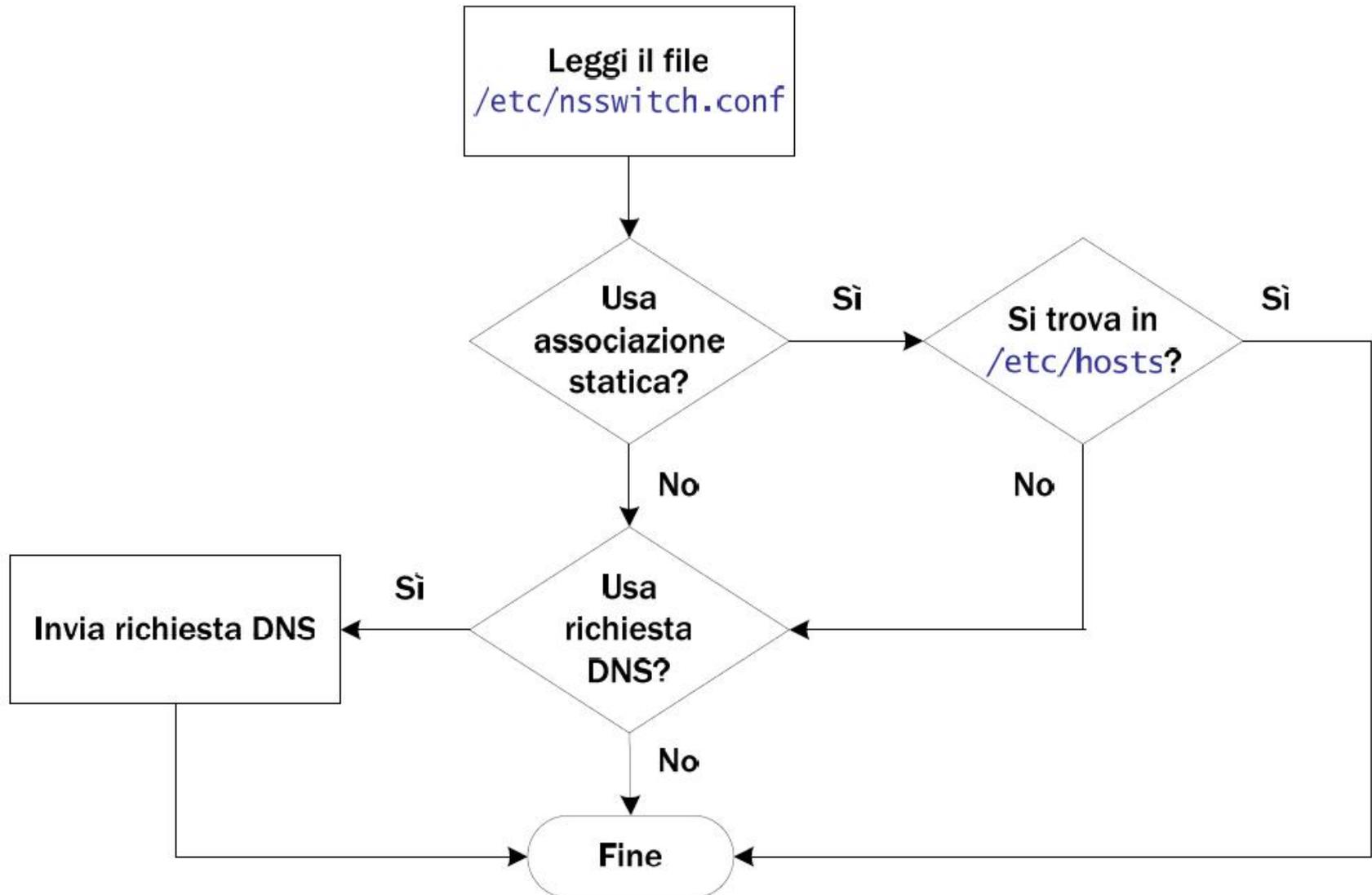
# Servizio di risoluzione dei nomi



- Architettura di riferimento
  - client/server
    - ⇒ sistema gerarchico
- Interazione richiesta/risposta



# Risoluzione di una richiesta DNS



/etc/nsswitch.conf



- **Il file del Network Services Switch determina l'ordine delle ricerche effettuate quando viene richiesta una certa informazione**

```
hosts: files nis dns
```

- **specifica che le funzioni di ricerca degli host dovrebbero prima guardare**
  - **nel file locale** `/etc/hosts`,
  - **di seguito fare una ricerca NIS**
    - ⇒ **Network Information Service: servizio che fornisce informazioni che devono essere conosciute in ogni parte della rete, a tutte le macchine che vi prendono parte.**
  - **infine utilizzare il servizio dei nomi di dominio**  
`/etc/resolv.conf`
- **A quel punto, se nessuna corrispondenza è stata trovata, viene riportato un errore.**

- Viene usato per convertire i nomi degli elaboratori in numeri IP e viceversa.
- `/etc/hosts` accetta il formato:

```
[Indirizzo Internet ] [nome host ufficiale] [alias1] [alias2] ...
```

- **Esempio:**

```
10.0.0.1 myRealHostname.example.com myRealHostname foobar1 foobar2
```

# /etc/hosts



```
#  
# Necessario per il "loopback" IPv4.  
#  
127.0.0.1          localhost.localdomain    localhost  
#  
# Indirizzi IPv4.  
#  
192.168.1.1        dinkel.brot.dg            dinkel  
192.168.1.2        roggen.brot.dg            roggen  
#  
192.168.2.1        weizen.mehl.dg           weizen
```

- **Comando che permette di visualizzare o impostare l'hostname ovvero il nome del computer su cui esso viene lanciato.**
- **Se il comando viene lanciato senza passare nessun argomento verrà solamente visualizzato il nome dell'host.**

- **Sintassi:**

```
hostname [opzioni] [argomenti]
```

- Name Server Lookup è uno strumento presente in tutti i sistemi operativi che utilizzano il protocollo TCP/IP
- Consente di effettuare delle query ad un server DNS per la risoluzione di indirizzi IP o hostname.
- Si usa per poter ottenere da un dominio il relativo indirizzo IP o nome host e viceversa.
- Si può utilizzare in due modi: interattivo e non interattivo.

```
nslookup [opzioni] [host_da_indagare | -DNSserver]
```

# nslookup interattivo



- Questo modo permette di effettuare più query e visualizza i singoli risultati.
- Viene abilitato in modo automatico quando:
  - il comando non è seguito da argomenti
  - se il primo argomento è un trattino (-) seguito dal secondo argomento che corrisponde all'host name o all'ip del name server.
- `exit`: permette di uscire dalla shell interattiva

```
$ nslookup
> www.ing.unipi.it
Server:          131.114.21.15
Address:         131.114.21.15#53
www.ing.unipi.it canonical name =www.web.ing.unipi.it.
Name:   www.web.ing.unipi.it
Address: 131.114.28.27
>
```

# nslookup non interattivo



- **Permette di effettuare una sola query e ovviamente visualizza il risultato della singola query.**

```
$ nslookup www.ing.unipi.it
Server:          131.114.21.15
Address:         131.114.21.15#53
www.ing.unipi.it      canonical name
                    =www.web.ing.unipi.it.
Name:   www.web.ing.unipi.it
Address: 131.114.28.27
$
```

- **host è considerato insieme a dig il sostituto ufficiale di nslookup.**
- **Viene utilizzato per risolvere i nomi in indirizzi numerici e viceversa.**
- **Se lanciato senza parametri ci presenta una lista delle opzioni possibili e una breve spiegazione della loro funzione.**

```
host [opzioni] host [server]
```

- `-f nome_del_file`
  - Invia l'output ad un file specificato oltre che allo standard output.
- `-T`
  - Utilizza TCP anzichè UDP.
- `-v` e `-vv`
  - Rispettivamente *verbose* e *very verbose*.
  - Il primo include tutti i campi del record delle risorse, compresi Time-to-Live e classe, oltre ai campi "additional information" e "authoritative nameservers".
  - Il secondo aggiunge anche informazioni riguardo l'host richiesto (CPU, sistema operativo) ovviamente se presenti nel database DNS.

- **Strumento per l'analisi del traffico che avviene nella rete fisica a cui si è collegati.**
- **È uno strumento di sniffing particolarmente flessibile**
- **Si setta in modalità promiscua cioè non vede solo i pacchetti diretti a lui ma tutto il traffico**
- **NON visualizza il contenuto dei pacchetti ma solo le loro intestazioni (protocollo, IP sorgente, destinazione, porte ecc.) per cui si presta bene alla diagnostica di problemi di networking**
- **Sintassi**

```
tcpdump [opzioni] [espressioni]
```

# tcpdump: opzioni



- -a
  - Permette di convertire indirizzi broadcast e ip in nomi
- -c
  - Esce dopo aver ricevuto un tot di pacchetti (es. `c100`)
- -e
  - Stampa il MAC address in ogni pacchetto catturato
- -i
  - Definisce l'interfaccia di rete. (es. `-i eth0` o `-i lo`)
- -p
  - L'interfaccia locale non viene settata in modalità promiscua
- -q
  - Visualizza il minor numero di informazioni possibili

# tcpdump: opzioni



- `-r`
  - Utilizza il file specificato come input per i dati da filtrare
- `-S`
  - Indica la quantità in byte di un pacchetto catturato
- `-t`
  - Non stampa il contrassegno temporale per ogni pacchetto catturato
- `-tt`
  - Stampa il contrassegno temporale (timestamp) non formattato su ogni linea
- `-v`
  - **Verbose**
- `-w file`
  - Scrive su file il risultato dello sniffing
- `-X`
  - Stampa i pacchetti in formato HEX e ASCII

# tcpdump: espressioni



`tcpdump [opzioni] [espressioni]`

- **Composte da primitive che possono essere raggruppate per mezzo delle parentesi tonde**
- **Connesse attraverso operatori booleani**
- **Solo i pacchetti che soddisfano la condizione espressa vengono presi in considerazione**
- **Se l'espressione manca, vengono catturati tutti i pacchetti.**

# Operatori booleani



- **!** **o** not
  - **negazione logica**
- **&&** **o** and
  - **and logico**
- **||** **o** or
  - **OR logico**
- **tcpdump** **accetta anche operatori come:**
  - **<**
    - ⇒ **es. < 40**
  - **>**
    - ⇒ **es. > 50**

# tcpdump: entità



- `type`
  - **Dice a cosa l'ID (numerico o nome) si riferisce.**
    - ⇒ `host`
    - ⇒ `net`
    - ⇒ `port`
- `dir`
  - **entità che specifica una particolare direzione di trasferimento per/da un ID.**
    - ⇒ `src`
    - ⇒ `dst`
    - ⇒ **Default** `src` or `dst`
    - ⇒ **es.** `'src foo', 'src or dst port ftp-data'`
- `proto`
  - **entità che restringe la cattura ad un particolare protocollo.**
    - ⇒ `ether, fddi, ip, arp, rarp, decnet, lat, mopr, mopdl, tcp` e `udp`
    - ⇒ **es.** `'ether src foo', 'tcp port 21'`
  - **Se il protocollo non è selezionato vengono considerati tutti i protocolli**

# tcpdump: esempio



```
tcpdump -vvv -i lo
```

```
15:46:49.086648 localhost.1029 > localhost.ftp: P  
3015740888:3015740901(13) ack 3004264398 win 31072  
<nop,nop,timestamp 578305 549639> (DF) [tos 0x10]  
15:46:49.086648 localhost.1029 > localhost.ftp: P  
0:13(13) ack 1 win 31072 <nop,nop,timestamp 578305  
549639> (DF) [tos 0x10]  
15:46:49.086739 localhost.ftp > localhost.1029: . ack  
13 win 31059 <nop,nop,timestamp 578305 578305> (DF)  
[tos 0x10]  
15:46:49.086739 localhost.ftp > localhost.1029: . ack  
13 win 31059 <nop,nop,timestamp 578305 578305> (DF)  
[tos 0x10]
```

# tcpdump: esempio



```
15:46:49.086648 localhost.1029 > localhost.ftp: P
3015740888:3015740901(13) ack 3004264398 win 31072
<nop,nop,timestamp 578305 549639> (DF) [tos 0x10]
```

- 15:46:49.086648
  - tempo in cui il pacchetto è stato catturato.
- localhost.1029 > localhost.ftp
  - il pacchetto è stato spedito da un'applicazione su localhost associata alla porta 1029 ad un'applicazione ancora su localhost associata alla porta riservata ai servizi ftp.
  - sniffing dal loopback
- **Ci sono altri dettagli di TCP come i numeri di sequenza della connessione (3015740888) o il valore di particolari bit che non ci interessano.**
- **Con queste opzioni tcpdump \*non\* prende in considerazione la parte dati.**

# tcpdump: esempio



```
tcpdump -i lo -s 200 -x -q
```

- **-s 200** specifica di catturare 200 byte di ogni pacchetto
- **-x** specifica di stampare (in esadecimale) l'inizio di ogni pacchetto
- **-q** fornisce una stampa meno invadente dell'output (mancano i numeri di sequenza e altri dettagli)

```
15:59:35.695801 localhost.1031 > localhost.ftp: tcp 13
(DF) [tos 0x10]
    4510 0041 014e 4000 4006 3b57 7f00 0001
    7f00 0001 0407 0015 f47b f62d f429 54a3
    8018 7960 c764 0000 0101 080a 0009 fe76
    0009 f7c5 5553 4552 2067 6975 6d6f 6e0d
    0a
```

```
15:59:35.695801 localhost.1031 > localhost.ftp: tcp 13
(DF) [tos 0x10]
    4510 0041 014e 4000 4006 3b57 7f00 0001
    7f00 0001 0407 0015 f47b f62d f429 54a3
    8018 7960 c764 0000 0101 080a 0009 fe76
    0009 f7c5 5553 4552 2067 6975 6d6f 6e0d
    0a
```

# tcpdump: esempi



```
tcpdump -i eth0 -w file tcp port 23
```

- **cattura solo i pacchetti** tcp
- **interfaccia di rete** eth0
- **destinati alla** porta 23 (telnet)
- **salva tutto l'output su un file**

# tcpdump: esempi



```
tcpdump port 80
```

- **Visualizza solo i pacchetti che hanno come sorgente o destinazione la porta 80**

```
tcpdump host 192.168.0.150
```

- **Visualizza solo i pacchetti che hanno come IP sorgente o destinazione 192.168.0.150.**

```
tcpdump host 10.0.0.150 and not port 22
```

- **Visualizza solo i pacchetti relativi all'host 10.0.0.150 che non usino la porta ssh (and not port 22).**

```
tcpdump net 10.0.0.0/24 and port 22
```

- **Visualizza tutti i pacchetti per la rete 10.0.0.0/24 relativi al protocollo ssh (and port 22)**

# tcpdump: considerazioni



- `tcpdump` è molto potente e versatile
- Interpretare direttamente il suo output non è pratico.
- Se si è interessati a visualizzare e raggruppare i dati sniffati si possono usare dei tool che facilitano queste operazioni.

# **Connessione a Internet tramite gateway**

# route



```
route -n get www.google.it
```

```
route to: 209.85.135.99
```

```
destination: default
```

```
mask: default
```

```
gateway: 131.114.31.254
```

```
interface: r10
```

```
flags: <UP,GATEWAY,DONE,STATIC>
```

# route: campi



- destination
  - **indica l'host o la rete che si vuole raggiungere**
- mask
  - **indica la maschera di rete**
- gateway
  - **indica, dove presente, il gateway, ovvero il computer della rete locale tramite cui connettersi ad un'altra rete**
- interface
  - **l'interfaccia di rete tramite cui connettersi**

## ■ Sintassi

```
route [-dnqtv] command [[modifiers] args]
```

### ■ command

#### ■ add

⇒ Aggiunge una route.

#### ■ flush

⇒ Rimuove tutte le route

#### ■ delete

⇒ Elimina una specifica route

#### ■ Change

⇒ Cambia le caratteristiche di una route, ad esempio il gateway

#### ■ get      Lookup and display the route for a destination.

⇒ Cerca e mostra una route per una destinazione

#### ■ monitor

⇒ Riporta continuamente i cambiamenti alla tabella di routing

- -n
  - Per indicare il valore numerico degli host nell'output
- -v
  - Verbose
- -q
  - Mostra un output piu' scarno per add, change, delete e flush

# route: GNU/LINUX



- **Permette di vedere e modificare la tabella di routing.**

```
route -n
```

```
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
10.114.228.145	0.0.0.0	255.255.255.255	UH	0	0	0	tun0
10.114.228.128	10.114.228.145	255.255.255.128	UG	0	0	0	tun0
<b>131.114.9.0</b>	<b>0.0.0.0</b>	<b>255.255.255.0</b>	<b>U</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>eth0</b>
10.114.0.0	10.114.228.145	255.255.0.0	UG	0	0	0	tun0
<b>169.254.0.0</b>	<b>0.0.0.0</b>	<b>255.255.0.0</b>	<b>U</b>	<b>1000</b>	<b>0</b>	<b>0</b>	<b>eth0</b>
0.0.0.0	131.114.9.29	0.0.0.0	UG	100	0	0	eth0

- **Il computer dove è stato lanciato il comando si connette tramite l'interfaccia eth0 direttamente alle reti 131.114.9.0 e 169.254.0.0 (il campo gateway è indicato con 0.0.0.0)**
- **per tutte le altre destinazioni (0.0.0.0) si connette mediante il gateway 131.114.9.29**

# route: esempi



```
route add -net 192.168.101.0 netmask
255.255.255.0 gw 192.168.101.102
```

- **Aggiunge la route 192.168.101 alla routing table indicando la maschera di rete 255.255.255.0 e il gateway 192.168.101.102**

```
route add default gw 192.168.12.1
```

- **Aggiunge l'host 192.168.12.1 come gateway di default**

```
route add 10.41.0.0 mask 255.255.0.0 10.27.0.1
if 0x3
```

- **Aggiunge una route alla destinazione 10.41.0.0 con la subnet mask 255.255.0.0, l'indirizzo di hop successivo 10.27.0.1 e utilizza l'indice di interfaccia 0x3**

- **Invia una successione di pacchetti ad una stazione per verificarne la raggiungibilità**
- **Ricorre al protocollo ICMP.**

- ***Internet Control Message Protocol*** (Protocollo per i Messaggi di Controllo in Internet).
- Il protocollo IP fornisce un meccanismo di trasferimento dei pacchetti dal mittente al destinatario secondo un approccio *best-effort*.
- Questo vuol dire che l'IP non è in grado di garantire la consegna dei pacchetti al destinatario.
- ICMP segnala solamente errori e malfunzionamenti, ma non esegue alcuna correzione

# Esempio



- Supponiamo che dalla **stazione A** si voglia controllare l'integrità della connessione fino alla **stazione B**.
- Si esegue il comando `ping`, passandogli come argomento l'indirizzo della **stazione B**.
- Il programma manda una serie di messaggi ICMP `ECHO_REQUEST` (generalmente uno al secondo) dalla **stazione A** verso la **stazione B**.
- Quando la **stazione B** riceve un pacchetto `ECHO_REQUEST`, risponde con un nuovo datagramma `ECHO_REPLY`, che viene mandato indietro alla **stazione A**.
- Il programma `ping` userà le informazioni così collezionate (esistenza dei pacchetti di ritorno, tempo intercorso per ogni pacchetto, etc.) per calcolare dei valori statistici sulla bontà della connessione e presentarli all'utente.

# ping: output



```
ping www.google.it
PING www.l.google.com (64.233.183.103) 56(84) bytes of data.
64 bytes from nf-in-f103.google.com (64.233.183.103):
  icmp_seq=1 ttl=232 time=91.6 ms
64 bytes from nf-in-f103.google.com (64.233.183.103):
  icmp_seq=2 ttl=232 time=94.6 ms

--- www.l.google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time
 4001ms
rtt min/avg/max/mdev = 69.754/87.351/101.647/11.455 ms
```

- **dimensione del pacchetto** ECHO\_REPLY
- **indirizzo IP di DEST**
- **numero di sequenza della risposta**
- **“time-to-live” (TTL)**
- **“round-trip time” (RTT)**
- **risultati statistici**

# ping: opzioni



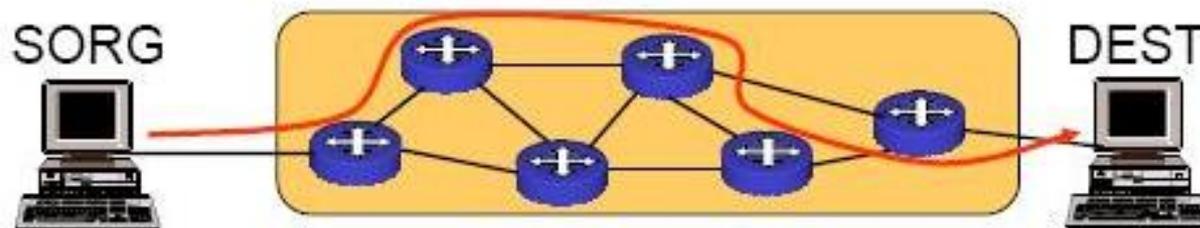
- `-c count`
  - **Stop l'invio (e ricezione) di pacchetti count ECHO\_RESPONSE**
- `-i wait`
  - **Aspetta wait secondi tra gli invii dei pacchetti.**
  - **Il default è di aspettare per un secondo tra ciascun pacchetto.**
- `-n`
  - **Solo output numerico. Non verrà fatto nessun tentativo di cercare nomi simbolici per gli indirizzi dell'host.**
- `-q`
  - **Output silenzioso. Non è visualizzato nulla tranne le linee di sommario all'avvio e quando termina.**
- `-s dimensione pacchetto`
  - **Specifica il numero di byte di dati da inviare.**
  - **Il default è 56, che si traduce in 64 byte di dati ICMP quando combinato con gli 8 byte dei dati di intestazione di ICMP.**

- Permette di conoscere il percorso seguito dai pacchetti inviati da una sorgente e diretti verso una destinazione

```
traceroute [<opzioni>] <destinazione> [<lunghezza>]
```

- `traceroute` inizia la trasmissione di pacchetti (utilizzando il protocollo ICMP) con un TTL molto basso.
- Si aspetta di ricevere un messaggio di errore dall'host in cui il TTL raggiunge lo zero.
- Incrementando lentamente il valore del TTL, `traceroute` riesce a conoscere gli indirizzi dei nodi attraversati, purché tutto funzioni come previsto (cioè che i vari nodi generino correttamente i pacchetti ICMP di errore).
- Quando tutto funziona come previsto, `traceroute` genera un elenco di host a partire da primo nodo attraversato, fino all'ultimo che rappresenta la destinazione richiesta.
- Se in alcuni punti non si ottiene risposta, i nodi ipotizzati vengono segnalati con degli asterischi.

# traceroute: esempio



- SORG invia a DEST una serie di pacchetti con un TIME-TO-LIVE (TTL) progressivo da 1 a 30 (per default)
- Ciascun nodo intermedio decrementa TTL
- Il nodo che decrementa TTL a 0 invia a SORG un pacchetto ICMP di tipo TIME\_EXCEEDED
- SORG costruisce una lista dei nodi attraversati fino a DEST

# traceroute: output



```
traceroute www.ing.unipi.it
```

```
traceroute to www.ing.unipi.it (131.114.28.27), 30 hops max,  
40 byte packets
```

```
1  man-gate.iet.unipi.it (131.114.9.29)  0.706 ms  0.688 ms  
   0.929ms  
2  131.114.186.9 (131.114.186.9)  3.929 ms  4.494 ms  13.325  
   ms  
3  www.web.ing.unipi.it (131.114.28.27)  3.908 ms  !H  3.905  
   ms !H  5.469 ms !H
```

- **TTL**
- **nome DNS**
- **indirizzo IP dei nodi intermedi**
- **ROUND-TRIP TIME (RTT)**

- **Address Resolution Protocol (ARP) è un protocollo che fornisce la "mappatura" tra l'indirizzo IP e il suo MAC address**
- **ARP viene utilizzato per ottenere l'indirizzo MAC quando un pacchetto deve essere inviato ad un calcolatore nella stessa sottorete**
- **Se il pacchetto deve essere inviato ad un calcolatore di un'altra sottorete, ARP viene utilizzato per scoprire il MAC address del gateway (non può attraversare router).**
- **Il protocollo ARP tiene traccia delle risposte ottenute in una apposita cache, per evitare di utilizzare ARP prima di inviare ciascun pacchetto.**
- **Le voci della cache ARP vengono cancellate dopo un certo periodo dall'ultima occorrenza, tipicamente dopo 5 minuti.**

# Meccanismo ARP



- L'host che vuole conoscere il mac address di un altro host, di cui conosce l'indirizzo IP, invia in broadcast una richiesta `ARP_REQUEST` contenente l'indirizzo IP dell'host di destinazione ed il proprio indirizzo MAC.
- Tutti i calcolatori della sottorete ricevono la richiesta.
- In ciascuno di essi il protocollo ARP verifica se viene richiesto il proprio indirizzo IP.
- L'host di destinazione che riconoscerà il proprio IP nel pacchetto di `ARP_REQUEST`, provvederà ad inviare una risposta `ARP_REPLY` in unicast all'indirizzo MAC sorgente, contenente il proprio MAC.
- Ogni host può scoprire l'indirizzo fisico degli altri host sulla stessa sottorete.

- -v
  - **Abilita il verbose mode**
- -n
  - **Non esegue il DNS lookup degli indirizzi ip**
- -i
  - **Specifica l'interfaccia**

- **Permette di visualizzare e manipolare le voci di `arp` nella cache del sistema.**

```
arp [opzioni] -a [hostname]
```

- **Visualizzazione del contenuto di tutta la cache, oppure specificando l'host solo l'`arp` del suddetto host**

```
arp [opzioni] -d hostname
```

- **Cancellazione dell'`arp` di uno specifico host**

```
arp [opzioni] -s hostname hw_addr [opzioni]
```

- **Creazione manuale di uno specifico `arp` di un host**
- **Trova l'indirizzo MAC della macchina da raggiungere attraverso una tabella di cache**

# arp: esempio



```
arp -a
```

```
ns.ing.unipi.it (131.114.28.5) at  
 00:0e:2e:02:c7:3a on r10 [ethernet]  
docenti.ing.unipi.it (131.114.28.20) at  
 00:04:75:77:5f:ae on r10 [ethernet]  
studenti.ing.unipi.it (131.114.29.9) at  
 00:e0:4c:39:21:15 on r10 [ethernet]  
e-ing.ing.unipi.it (131.114.29.11) at  
 00:04:96:34:7e:d6 on r10 [ethernet]  
? (131.114.31.0) at 00:48:54:01:1b:40 on r10  
 [ethernet]  
juniper.ing.unipi.it (131.114.31.254) at  
 00:19:e2:98:04:21 on r10 [ethernet]
```

# Network Filesystem (NFS)



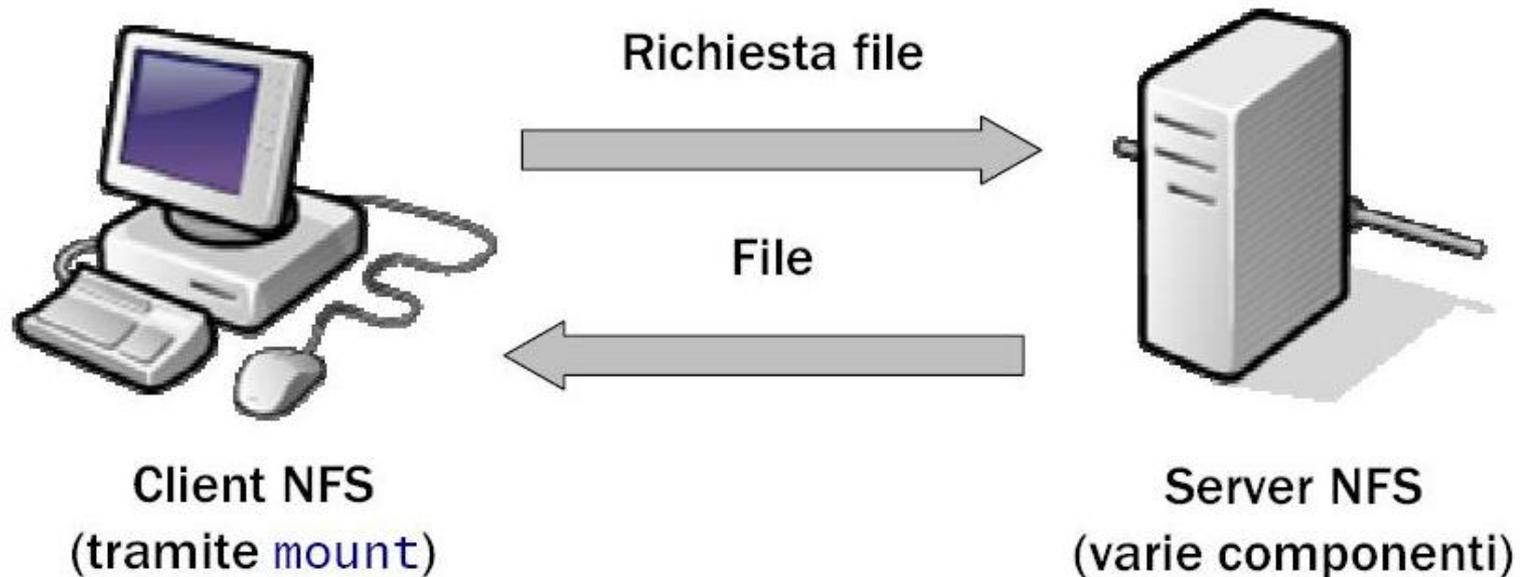
- Il servizio di NFS permette la condivisione dei file system in rete.
- Usando NFS, utenti e programmi possono accedere a file su sistemi remoti quasi come se fossero file locali.
- **Funzionamento**
  - **Server**
    - ⇒ Devono essere configurate le directory da condividere
  - **Client**

- **Possibilità di concentrare i dati a cui spesso accedono tutti gli utenti su un host centrale e di renderli disponibili ai client al momento del boot**
- **Possibilità di conservare su un unico host tutti i dati ed i programmi che occupano molto spazio su disco.**

# NFS: architettura di riferimento



- Architettura di riferimento
  - client/server
  - Interazione tramite RPC
  - Protocollo stateless



## 1. Installazione pacchetti:

⇒ *sudo apt-get install portmap nfs-kernel-server nfs-common*

## 2. Configurare le directory da condividere:

⇒ *file /etc/exports*

## 3. Avviare i demoni:

⇒ */etc/init.d/nfs-kernel-server  
start | stop | restart | status*

⇒ */etc/init.d/portmap start | stop | restart*

# Lato server: demoni NFS



- `mountd`
  - gestisce le richieste di mount dei client alle cartelle condivise
  - usa il file di configurazione `/etc/exports`
  - Parte in automatico quando il NFS è attivo
- `nfsd`
  - realizza il servizio di trasferimento file vero e proprio

# Lato server: /etc/exports



- Contiene l'indicazione delle porzioni di file system locale da concedere in condivisione alla rete NFS.
- Se il file manca o è vuoto, non viene concesso l'utilizzo di alcuna parte del file system locale all'esterno.
- Ogni record del file è composto da:
  - l'indicazione di una directory a partire dalla quale si concede la condivisione
  - una serie di nodi o reti cui viene concesso l'utilizzo di questa directory con l'eventuale specificazione di opzioni di accesso.
- Quando si fanno modifiche a questo file, è necessario riavviare il sistema di gestione del servizio NFS.

**Lato server:** /etc/exports



- **Formato del file**

`cartella_base [host] (<opzioni>)`

# Lato server: /etc/exports



`cartella_base [host] (<opzioni>)`

- Host singolo (nome o indirizzo IP)
- Possono essere utilizzati i caratteri jolly \* e ? per indicare un gruppo di nomi di elaboratore con una sola notazione, tenendo presente che questi simboli non possono sostituirsi ai punti di un nome di dominio
- Con `indirizzo_ip/maschera_di_rete` è possibile indicare simultaneamente tutti gli elaboratori collocati all'interno della rete o della sottorete a cui si fa riferimento.

`-network <netaddr> -mask <mask>`

# Lato server: /etc/exports



- `ro`
  - **Accesso in sola lettura**
- `rw`
  - **Accesso in lettura e scrittura**
- `root_squash`
  - **Impedisce l'accesso come utente root**

# Lato server: /etc/exports



- **Le cartelle possono comparire in più di una riga**
  - ogni riga si riferisce ad uno o più client
  - se a client distinti si applicano opzioni diverse devono esserci righe distinte per la cartella
- **Si può inserire una lista di cartelle come primo campo**
  - nella forma di percorsi assoluti
  - separate da spazi
- **Non è possibile esportare allo stesso client punti diversi dello stesso filesystem con opzioni diverse**
  - non si possono avere due righe in cui le cartelle si trovano sullo stesso file system

# Lato server: aggiornamento



- Il demone `mountd` deve essere forzato a rileggere il file `/etc/exports` ogni volta che viene modificato
- Questo può essere ottenuto:
  - inviando un segnale `HUP` al processo `mountd`  
`kill -HUP `cat /var/run/mountd.pid``

# Lato client: `mount`



- Il montaggio di un file system di rete avviene in modo analogo a quello di una normale unità di memorizzazione, con una sintassi leggermente diversa:

```
mount -t nfs host_remoto:dir_remota dir_locale[options]
```

- **Esempio**

- l'elaboratore `mio.server.it` concede l'utilizzo della directory `/usr/` e successive
- l'elaboratore `mio.client.it` potrebbe connettersi attraverso

```
mount -t nfs mio.server.it:/usr /usr
```

- Sul lato client si può aggiungere una riga nel file `/etc/fstab` in modo da automatizzarne la connessione.

```
mio.server.it:/usr /usr nfs defaults
```

- **Per verificare l'utilizzo effettivo del servizio da parte dei client**

```
showmount [opzioni] [host]
```

- **Sul lato client serve per conoscere le directory esportate da un server NFS**
- **Opzioni**
  - **-a**
    - ⇒ **Elenca i client che utilizzano il proprio servizio e le directory che questi hanno montato.**
  - **-e**
    - ⇒ **Elenca le directory esportate dal server locale o dal server remoto**