Berkeley UNIX Internetworking

Giuseppe Vitillaro

Consiglio Nazionale delle Ricerche

Centro di Studio per il Calcolo Intensivo in Scienze Molecolari

Dipartimento di Chimica

Università degli Studi di Perugia

e-mail: <peppe@unipg.it>

Perugia, 27/11/1999

Breve storia di Internet

- Nel 1970 il Dipartimento della Difesa americano, attraverso la Defense Advanced Projects Research Agency (**DARPA** o **ARPA**), finanzia un progetto per la realizzazione di una rete di comunicazione nazionale.

- La rete, nota come **ARPANET**, ha lo scopo di collegare installazioni militari e laboratori di ricerca.

- ARPANET è stata progettata per essere:
  - eterogena
  - multivendor
  - in grado continuare a funzionare sotto un attacco nucleare.

Breve storia di Internet

- Tra il 1973 ed il 1974 vengono realizzati e si affermano un insieme di protocolli di comunicazione noti come TCP/IP o anche “IP protocol suite”.

- Nel 1980 la rete ARPANET inizia a convertire le prime macchine all’uso del TCP/IP. Nel Gennaio 1983 DARPA adotta il TCP/IP come protocollo “ufficiale” della rete ARPANET: nasce il primo *segmento* della rete **Internet**.

- DARPA decide di rendere il TCP/IP disponibile, a basso costo, a tutti i principale enti di ricerca:

  - all’epoca, quasi tutti i dipartimenti di Computer Science delle principali Università americane, utilizzavano la versione UNIX nota come “Berkeley

Breve storia di Internet

Software Distribution", **BSD**, dell'Università di California;

- DARPA finanzia alla *Bolt Beranek and Newman Inc.* (BBN) e a Berkeley un progetto avente come scopo l'implementazione dei protocolli Internet e la loro integrazione sotto la versione UNIX di Berkeley.

- Nel giro di pochissimo tempo la rete ARPANET interconnette più del 90% dei dipartimenti di Computer Science delle Università americane. La rete ARPANET si scinde in ARPANET e MILNET.

- Nel 1985 la National Science Foundation americana comprende il ruolo *cruciale* del networking per la ricerca scientifica ed inizia

Breve storia di Internet

un programma per la connessione dei sei principali centri di supercomputing. Nel 1986 nasce NSFnet.

- Nel 1990 ARPANET conclude *formalmente* il suo ruolo: nasce la Rete Internet.

- Dai servizi di base (nati con l'IP protocol suite, alla fine degli anni '70) di posta elettronica, file transfer, connessione remota si passa a servizi più *evoluti*:

  - Usenet News System (1981)
  - Internet Gopher (1982)
  - World Wide Web information service (1989).

Breve storia di Internet

- Negli anni ‘90 si assiste ad una rapida espansione e commercializzazione della rete Internet.

- Un evidente segno del suo successo è la confusione nata intorno alla parola *internet*.

  Originariamente veniva utilizzata per indicare una rete costruita sull’**Internet Protocol**.

  Ora la parola *internet* (iniziale minuscola) viene usata per indicare una rete costruita su un protocollo *comune* (IPX, NETBEUI, SNA, etc.).

  Internet (iniziale maiuscola) è invece la collezione mondiale di reti, evolutasi dalla rete ARPANET, interconnesse attraverso l’Internet Protocol: ***la Rete***.

Breve storia di internet

- Molto di moda è anche la parola *intranet*.

  In genere definisce una rete che possiede le seguente caratteristiche:

  - è completamente contenuta all'interno di una organizzazione (una azienda, un dipartimento universitario, un ente, etc.);
  - usa l'Internet Protocol come protocollo di interconnessione;
  - è interconnessa ad Internet attraverso uno o più gateways.

- Le caratteristiche *peculiari* dell'Internet Protocol sono state il propulsore dell'evoluzione dei sistemi di telecomunicazione a cui abbiamo assistito negli ultimi cinque anni.

TCP/IP

- È una sigla che indica un insieme di **protocolli** di comunicazione, cioè di descrizioni *formali* del formato dei messaggi che due o più macchine in rete possono scambiarsi: **Internet Protocol Suite**.

- **Non indica una implementazione software particolare.**

- La sigla proviene dalla composizione di due acronimi, che indicano due diversi sottoinsiemi di protocolli:

TCP	***Transmission Control Protocol***
IP	***Internet Protocol***

TCP/IP

- Le caratteristiche fondamentali del TCP/IP:
  - un insieme *aperto* di protocolli e standards; disponibilità di implementazioni di pubblico dominio;
  - indipendenza dall'hardware per l'accesso fisico alla rete: il TCP/IP può girare su Ethernet, Fast Ethernet, Token Ring, FDDI, ATM, X.25, ISDN, linee dial-up, etc;
  - uno schema di *indirizzamento* strutturato per identificare univocamente le interfacce di rete sull'intera Internet, a livello mondiale;
  - applicazione utente (ftp, telnet, etc.) standardizzate, con interfaccia consistente, su differenti piattaforme.
- Un **protocollo** è un *contratto*: due o più parti si accordano per rispettare un insieme di *regole*

TCP/IP

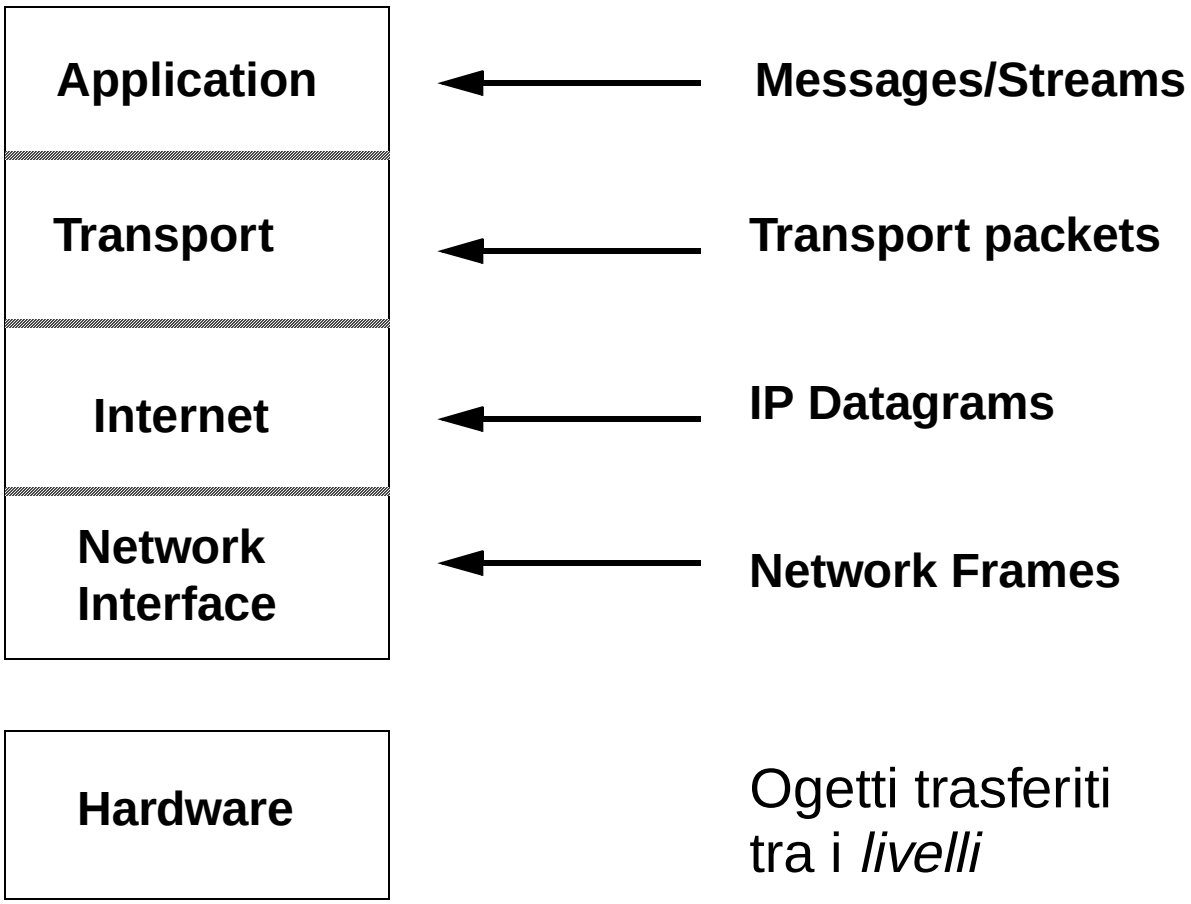
formali di comportamento allo scopo di raggiungere un obiettivo di interesse comune.

- Nella vita di tutti i giorni i contratti (o nella diplomazia i protocolli di intesa) sono utilizzati per permettere la cooperazione tra differenti soggetti: i partecipanti possiedono un *riferimento* comune che permette loro di comunicare ed interpretare correttamente il comportamento degli altri soggetti dell'intesa.

- Nelle telecomunicazioni questi insiemi di regole vengono denominati ***protocolli***. Interfacce di rete progettate da vendor differenti possono utilizzarli per consentire lo scambio di dati tra reti *eterogenee*.

TCP/IP: architettura

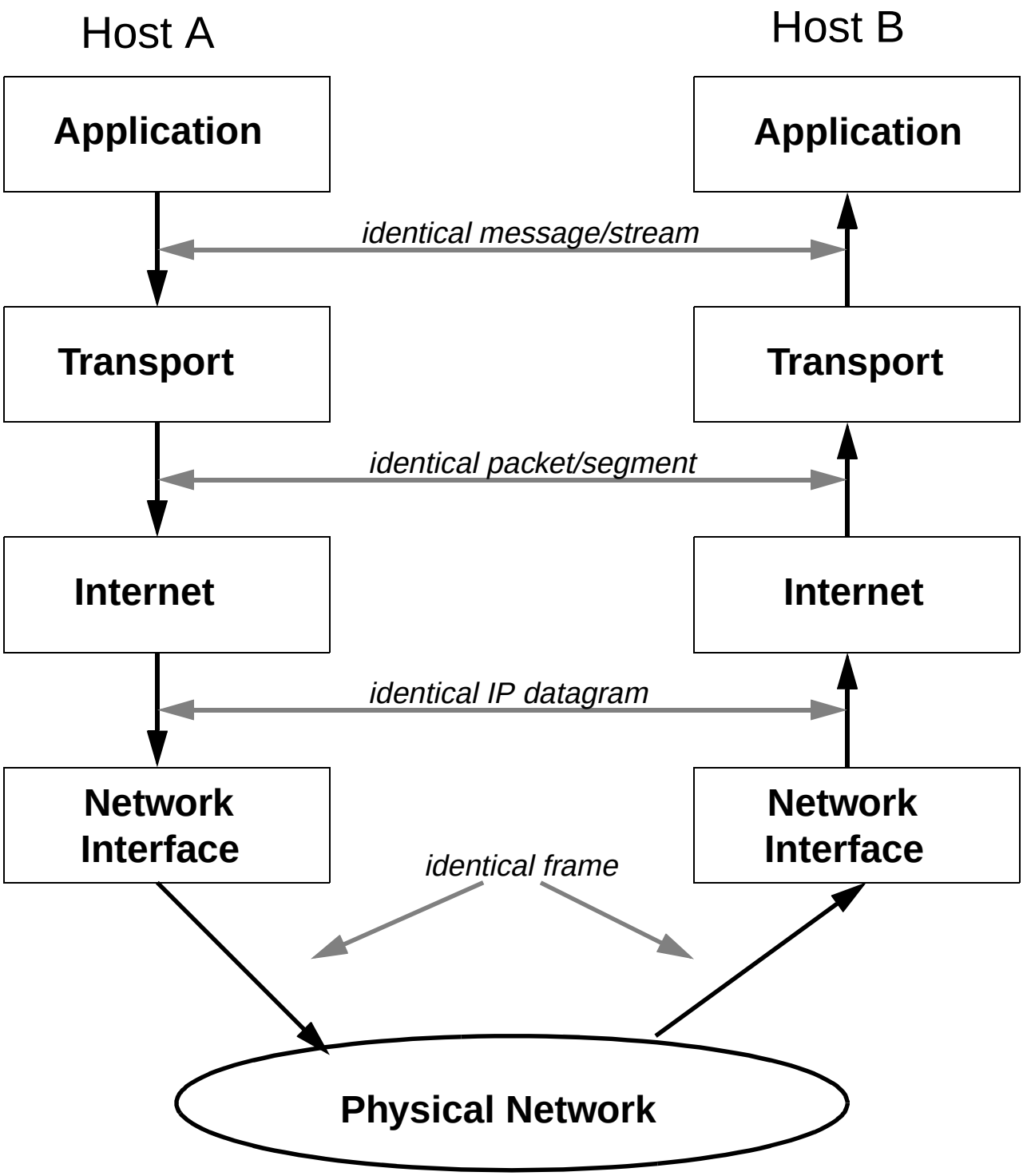
- Nel TCP/IP sono riconoscibili una serie di *strati*. A ciascuno di essi è affidato il compito di eseguire una parte delle operazioni necessarie per trasferire informazioni da una macchina ad un'altra.



TCP/IP: architettura

- Le *responsabilità* degli strati:
  - *Application Layer*. L'applicazione utente interagisce con il livello di trasporto per spedire o ricevere dati come sequenze di messaggi o stream di bytes.
  - *Transport Layer.* Permette la comunicazione tra **applicazioni** e controlla il flusso dei dati, suddividendoli in unità chiamate packets o segments.
  - *Internet Layer.* Gestisce la comunicazione tra macchine. Incapsula i pacchetti IP in datagrams e utilizza le informazioni di routing per istradare le informazioni sulle interfacce fisiche.
  - *Network Interface Layer.* È il livello più basso: ha il compito di convertire gli IP datagrams nelle frames di una rete fisica. È l'unico componente che ha una stretta dipendenza dall'hardware.

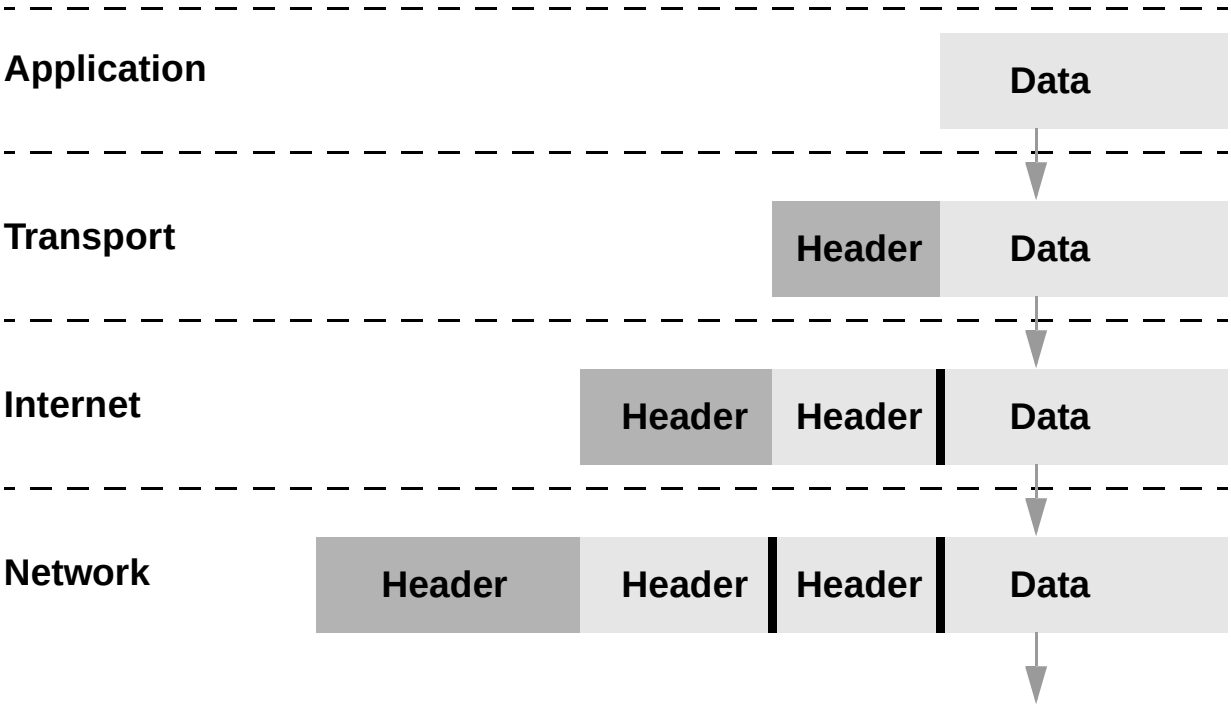
TCP/IP: architettura



TCP/IP: architettura

- Ogni strato scambia informazioni, nello stesso formato, solo con lo strato corrispondente ed utilizza le *primitive* degli strati adiacenti per spedire o ricevere messaggi.

  Ogni volta che avviene uno passaggio di informazioni tra strati differenti, al messaggio in transito, vengono aggiunte una serie di *informazioni di controllo*.



TCP/IP: architettura

- Nel TCP/IP esistono in effetti due differenti *strati* che si occupano del trasporto:
  - *User Datagram Protocol* (**UDP**)

    è orientato alla trasmissione tra applicazioni di singoli messaggi (packets);

    è un protocollo ***connectionless*** che non garantisce nè l'ordine, nè la ricezione dei singoli messaggi.

  - *Trasmission Control Protocol* (**TCP**)

    è un protocollo *affidabile* che garantisce l'ordine e la ricezione dei messaggi, ***connection oriented***, per lo scambio di ***streams*** di bytes tra applicazioni le cui unità vengono denominate *segmenti*.

- Le applicazioni TCP/IP possono scegliere di utilizzare uno dei due protocolli.

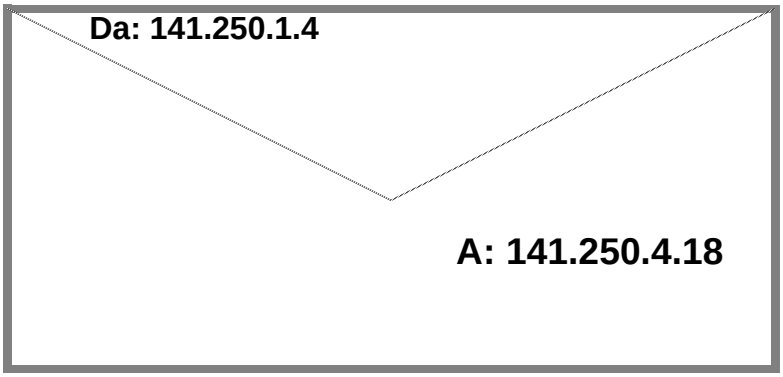
TCP/IP: architettura

- L'unità di trasmissione dell'IP protocol suite è l'**IP datagram**. I datagrams IP non contengono informazioni sul *percorso* che il singolo messaggio dovrà seguire per arrivare a destinazione.

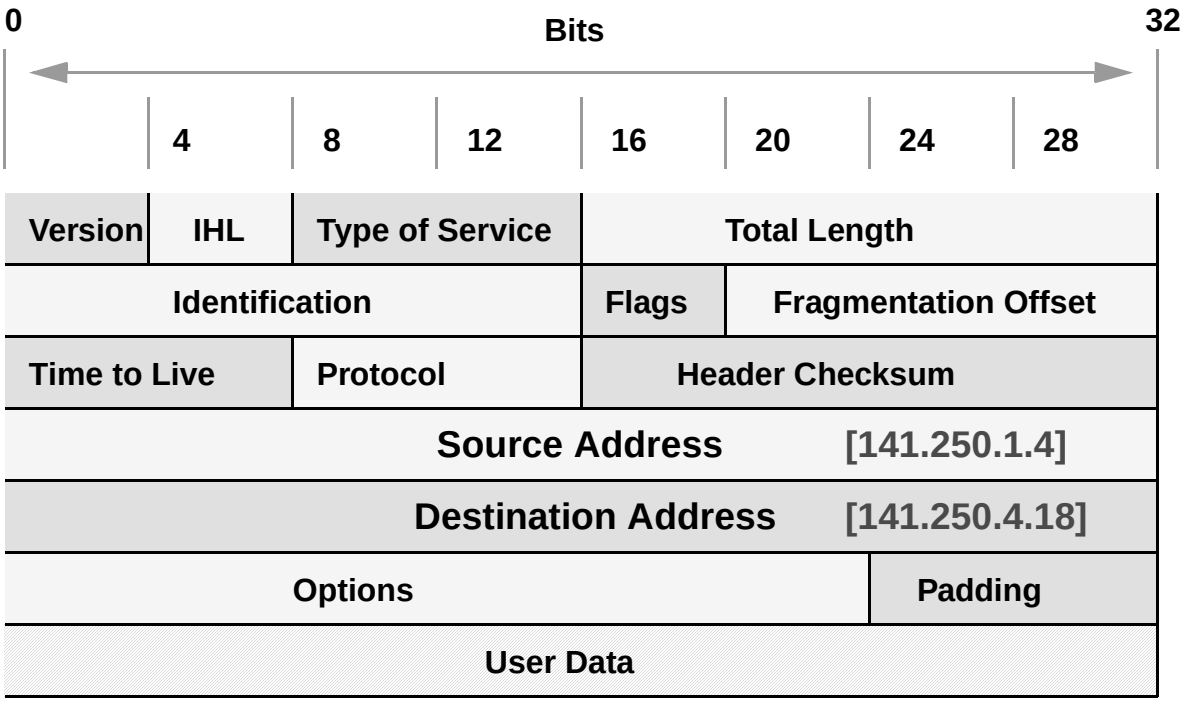
- In pratica ciascun datagram IP è *incapsulato* in una *busta*, un ***header***, nel quale esistono solo i dati relativi all'indirizzo del mittente e del destinatario.

- Ogni *datagram* **attraversa** indipendentemente la rete: pacchetti della stessa sessione applicativa possono seguire strade completamente *differenti*.

TCP/IP: architettura



IP Datagram Format



TCP/IP: architettura

- Ad ogni *host* della rete Internet è assegnato un indirizzo **IP**, lungo 32 bits, 4 bytes.

- L'indirizzo IP è in effetti composto da una coppia di informazioni: (**netid**, **hostid**):

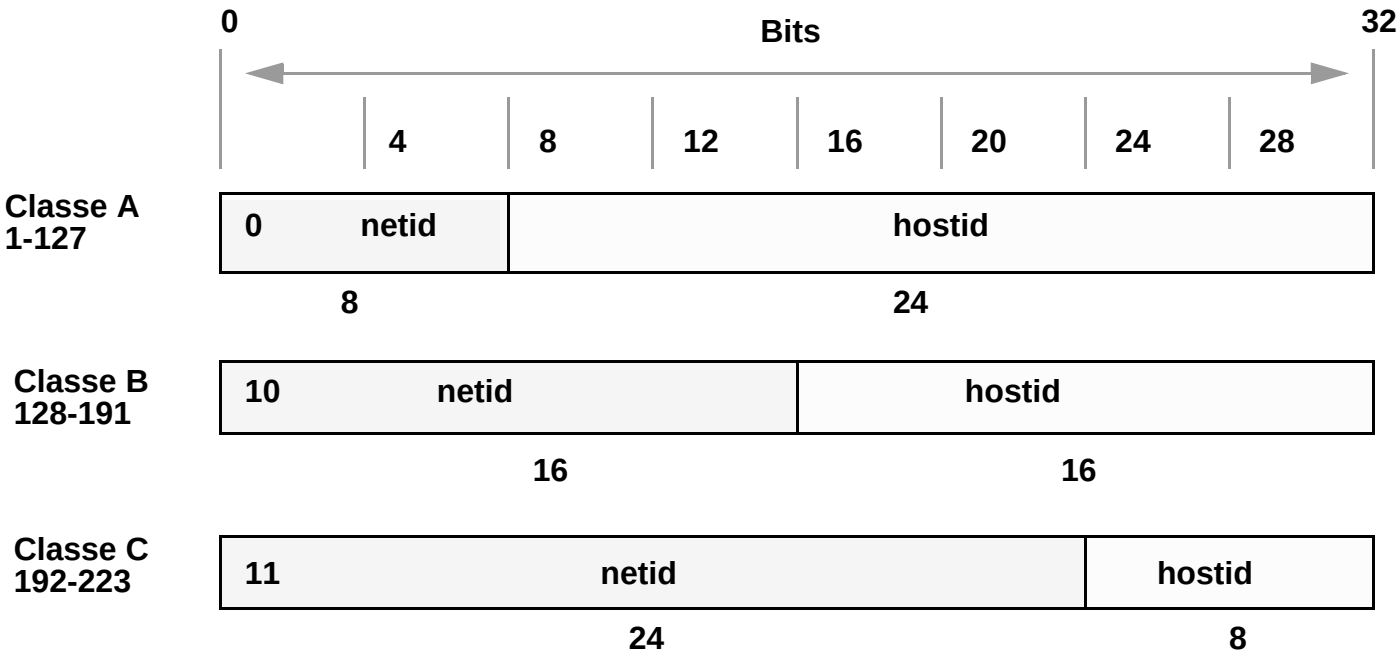
  - **netid** identifica la *rete*
  - **hostid** identifica l'interfaccia di rete

- Il primo byte dell'indirizzo identifica la *classe* della rete:

  - **26**.104.0.18 classe **A**
  - **141.250**.4.18 classe **B**
  - **194.143.128**.33 classe **C**

  ed in pratica *determina* la lunghezza del ***netid***.

TCP/IP: architettura



- **Classe A - 255.0.0.0**

  un centinaio di reti composte da *milioni* di hosts;

- **Classe B - 255.255.0.0**

  qualche migliaio di reti composte da *miglialia* di hosts;

- **Classe C - 255.255.255.0**

  milioni di reti ciascuna composta al più da 254 hosts.

TCP/IP: architettura

- L'interpretazione di un indirizzo IP è meno complicata di quanto possa sembrare:
  - ciascun *ottetto* (byte) dell'indirizzo può assumere un valore tra **0** e **255**;
  - in ***dot notation*** ciascun byte componente l'indirizzo viene scritto in base dieci (notazione *decimale*) e separato da un punto

141	8D	1000	1101
250	FA	1111	1010
4	04	0000	0100
18	12	0001	0010

    equivalenti a: 141.250.4.18.
  - Il valore del primo ottetto determina in modo univoco la classe di *default* della rete, ovvero la *maschera naturale* di bits da sovrapporre all'indirizzo per ottenere ***netid*** e ***hostid***.

TCP/IP: architettura

- L'architettura dell'**IPV4** risale ad un periodo nel quale 32 bits sembravano più che sufficienti per *indirizzare* tutte le macchine che si sarebbero mai potute connettere alla rete ARPANET (forse l'**IPV6** ci porrà rimedio ...).

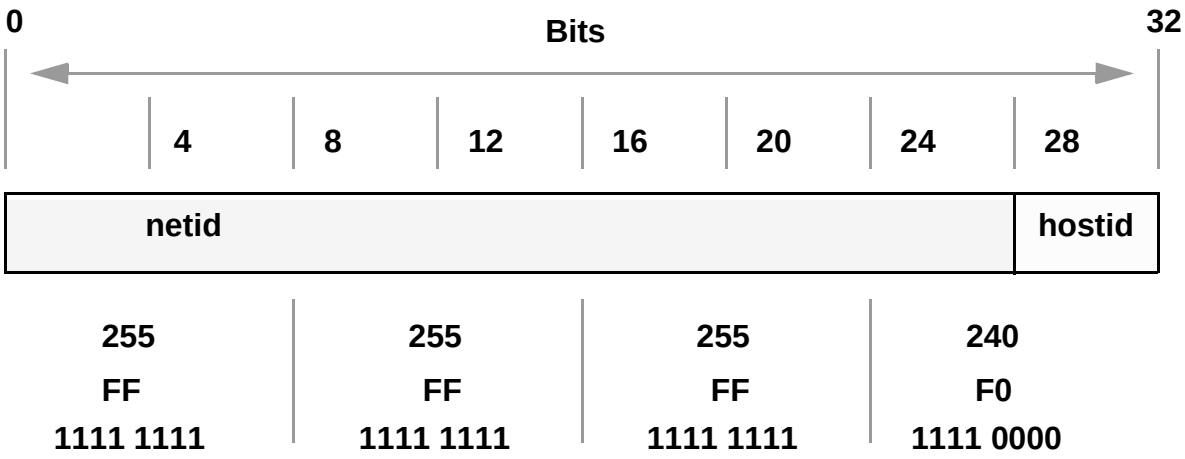
- Il disegno della divisione in classi di default e la rapidissima espansione della Rete ha prodotto una situazione che ha portato all'esaurimento della reti di classe B.

- Nuovi meccanismi per assegnare blocchi contigui di reti di classe C sono emersi nelle architetture dei routers negli ultimi anni.

- All'indirizzo può essere sovrapposta una maschera più corta della maschera *naturale*.

TCP/IP: architettura

- Ad un indirizzo **IP** viene sovrapposta una maschera di bits, nota come ***netmask***, che determina quali bits dell'indirizzo devono essere interpretati come indirizzo di rete (**netid**) e quali come indirizzo dell'host (**hostid**).



Indirizzo: 141.250.4.18
Netid: 141.250.4.1
Hostid: 2

Rete: 141.250.4.16
Host: 141.250.4.18
Broadcast: 141.250.4.31

14 indirizzi disponibili: 17, 18, ..., 30

TCP/IP: architettura

- L'hostid con *tutti* i bits a zero identifica la ***rete*** e l'indirizzo con *tutti* i bits ad uno identifica l'indirizzo di ***broadcast***.

- La maschera di default può essere *accorciata* per raggruppare reti della stessa classe o *allungata* per dividere una rete in sottoreti (**subnets**).

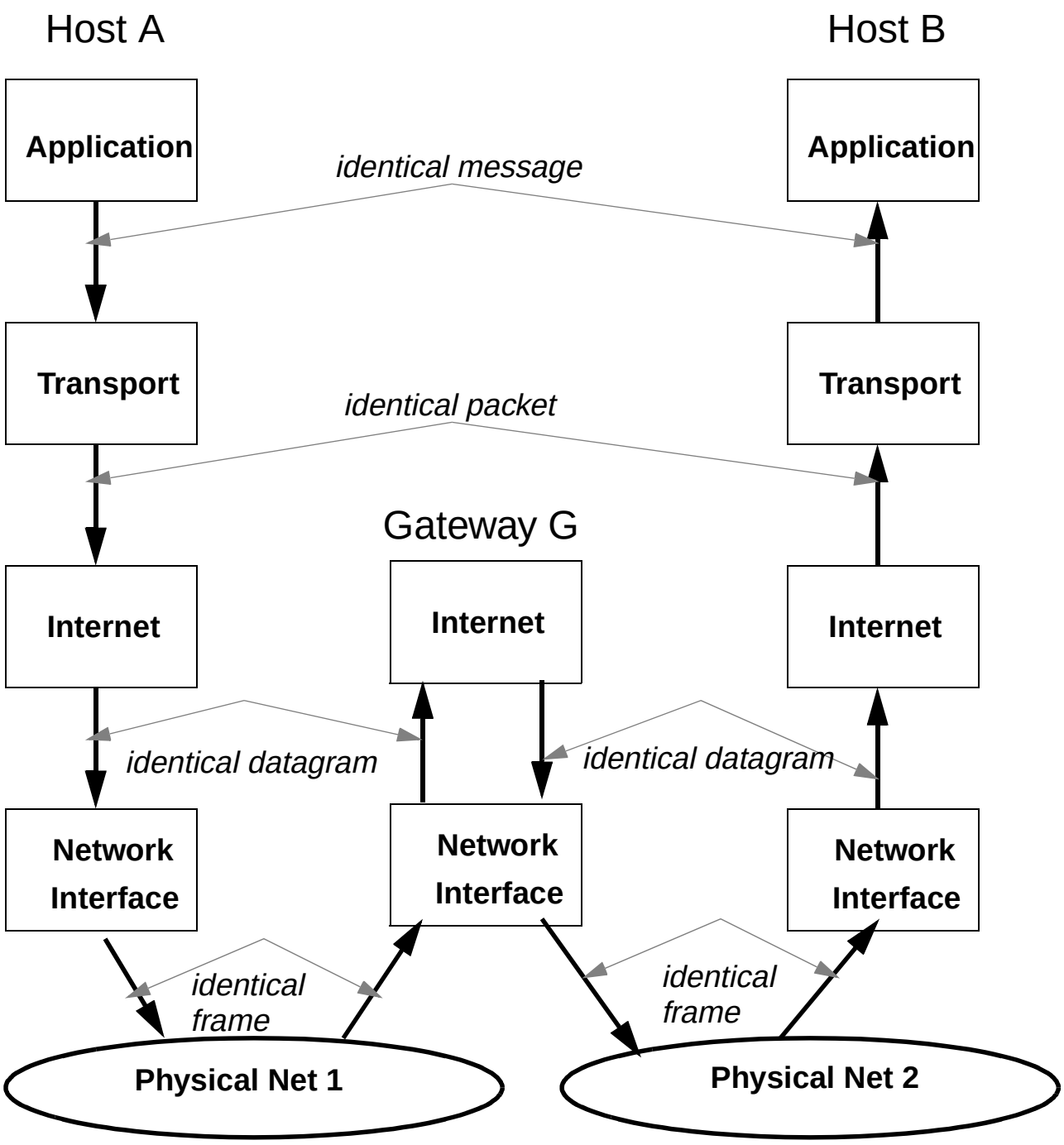
- Il **netid** viene utilizzato per *istradare* i singoli datagram nel loro percorso sulla rete. L'**hostid** identifica in modo univoco una interfaccia di rete su una particolare rete fisica.

- Nella terminologia TCP/IP il processo di istradamento viene definito **routing**.

TCP/IP: architettura

- L'Internet Protocol è il responsabile del *routing* degli IP datagrams. L'istradamento viene fondamentalmente pilotato dal ***netid*** contenuto nel ***Destination Address*** dell'IP datagram:
  - se il netid coincide con quello di una interfaccia di rete presente sull'host, l'IP spedisce il pacchetto alla macchina destinataria attraverso tale interfaccia;
  - altrimenti l'IP cerca di spedire il pacchetto, attraverso una delle interfacce di rete, ad un ***gateway*** che sarà responsabile del routing del datagram verso la destinazione finale.
- I **gateway** sono macchine responsabili del trasferimento dei pacchetti tra reti fisiche *differenti*.

TCP/IP: architettura



TCP/IP: architettura

- In una rete TCP/IP le macchine possono avere due ruoli:
  - ***host***

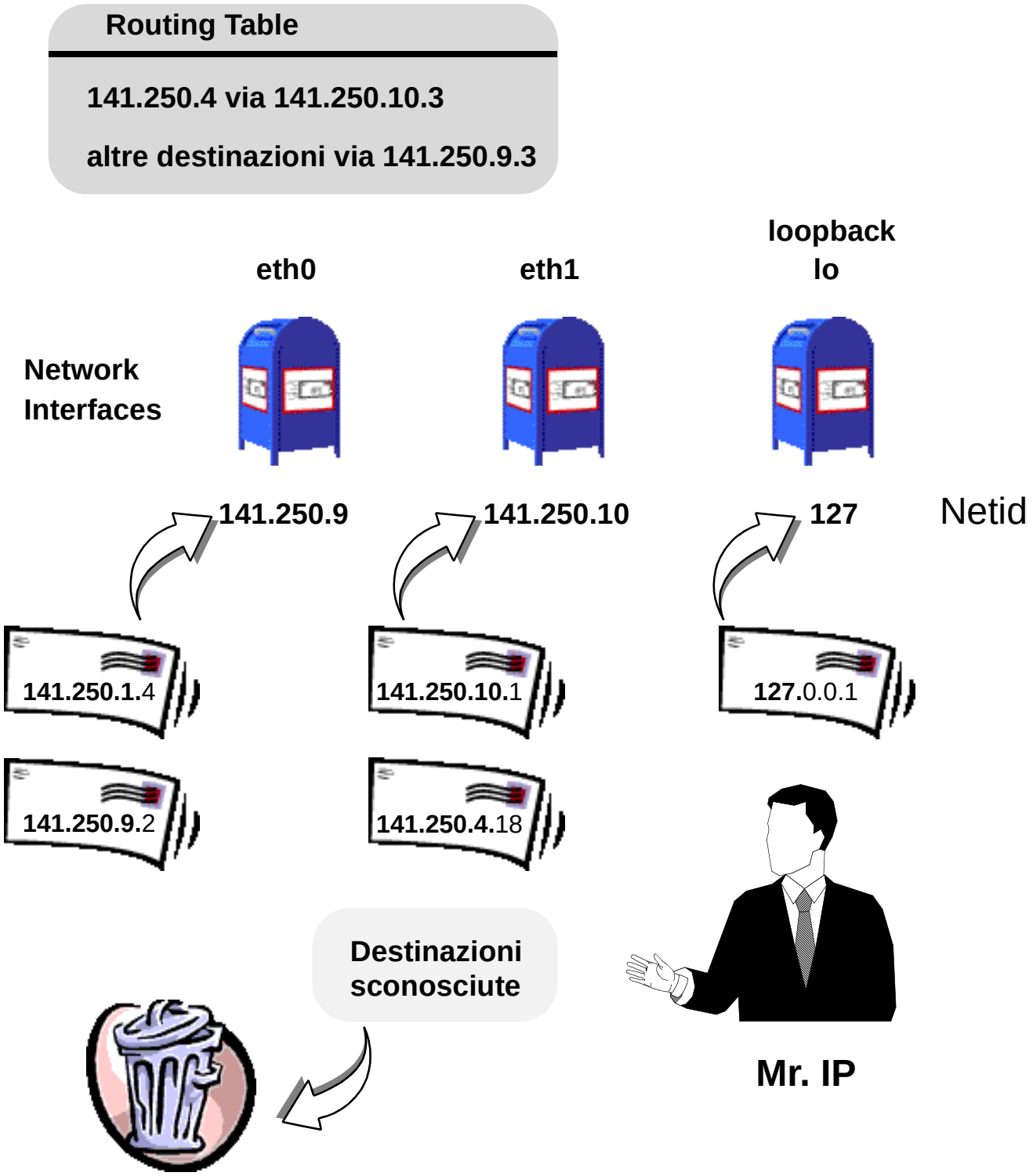
    macchine che non eseguono routing o forwarding di IP datagrams; possono solo ricevere o spedire pacchetti attraverso una o più interfacce di rete;
  - ***gateway*** *o* ***IP routers***

    macchine che eseguono il forwarding e/o il routing di pacchetti tra interfacce fisiche di rete.

- Un IP router si comporta un pò come un *postino* che debba occuparsi della consegna dei pacchetti IP.

  Le *network interfaces* sono le sue *buche delle lettere*. Data una destinazione il postino deve decidere quale *buca* sia la più **appropriata**.

TCP/IP: architettura



TCP/IP: Linux

- Il comando Linux **netstat -ni** permette di determinare quali *Network Interfaces* sono attive sull'host:

```
$ netstat -ni
Kernel Interface table
Iface   MTU Met  RX-OK RX-ERR RX-DRP RX-OVR  TX-OK TX-ERR TX-DRP TX-OVR Flags
lo     3584   0  94146      0      0      0  94146      0      0      0 BLRU
eth0   1500   0 797143      0      0      0 134710      0      0      0 BRU
eth0:0 1500   0      0      0      0      0      0      0      0      0 BRU
```

- Il comando Linux **ifconfig** configura l'indirizzo di una interfaccia di rete:

```
ifconfig eth0 141.250.10.1 netmask 255.255.255.0
broadcast 141.250.9.255
```

assegnando alla Network Interface **eth0** (una scheda Fast Ethernet in questo caso) l'indirizzo di rete 141.250.10.1, il netmask 255.255.255.0 (classe B subnettata) e l'indirizzo di broadcast 141.250.9.255.

TCP/IP: Linux

- Il comando **ifconfig**, sotto Linux, può essere utilizzato anche per ottenere i dettagli relativi alle Network Interfaces

```
$ ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Bcast:127.255.255.255  Mask:255.0.0.0
          UP BROADCAST LOOPBACK RUNNING  MTU:3584  Metric:1
          RX packets:94249 errors:0 dropped:0 overruns:0 frame:0
          TX packets:94249 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0

eth0      Link encap:Ethernet  HWaddr 00:A0:C9:FC:63:38
          inet addr:141.250.10.1  Bcast:141.250.10.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:798240 errors:0 dropped:0 overruns:0 frame:0
          TX packets:135029 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0
          Interrupt:5 Base address:0x1060

eth0:0    Link encap:Ethernet  HWaddr 00:A0:C9:FC:63:38
          inet addr:141.250.9.150  Bcast:141.250.9.255  Mask:255.255.255.0
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0
```

configurate sull'host.

In questo caso **eth0:0** non rappresenta una scheda fisica ma un *alias* sulla scheda **eth0**.

TCP/IP: Linux

- Il comando **netstat -nr** visualizza la ***tabella di routing***:

```
$ netstat -nr
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
141.250.10.0    0.0.0.0         255.255.255.0   U      1500 0          0 eth0
141.250.9.0     0.0.0.0         255.255.255.0   U      1500 0          0 eth0:0
141.250.4.0     141.250.9.3     255.255.255.0   UG     1500 0          0 eth0:0
127.0.0.0       0.0.0.0         255.0.0.0       U      3584 0          0 lo
```

```
$ netstat -nr
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
141.250.9.0     0.0.0.0         255.255.255.0   U      1500 0          0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U      3584 0          0 lo
0.0.0.0         141.250.9.3     0.0.0.0         UG     1500 0          0 eth0
$
$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
141.250.9.0     *               255.255.255.0   U      1500 0          0 eth0
127.0.0.0       *               255.0.0.0       U      3584 0          0 lo
default         cipg03.chm.unip 0.0.0.0         UG     1500 0          0 eth0
```

- La tabella di routing decide l'istradamento degli IP datagrams su una macchina UNIX.

  Al ***default*** gateway vengono inviati tutti i datagrams per cui non esiste un routing.

TCP/IP: architettura

- La tabella di routing può essere:
  - ***statica***

    è il risultato di una serie di comandi e viene modificata solo mediante comandi espliciti
  - ***dinamica***

    la tabella viene aggiornata automaticamente da uno o più *protocolli* di routing.
- Si parla di ***table based routing***: per ogni datagram la tabella viene consultata per decidere l'istradamento del pacchetto sulla più opportuna network interface.
- In genere le macchine UNIX possiedono uno ed un solo ***default gateway***.

TCP/IP: architettura

- Esistono altre implementazioni del routing: nella scelta della Network Interface l'Internet Protocol può valutare altri fattori.

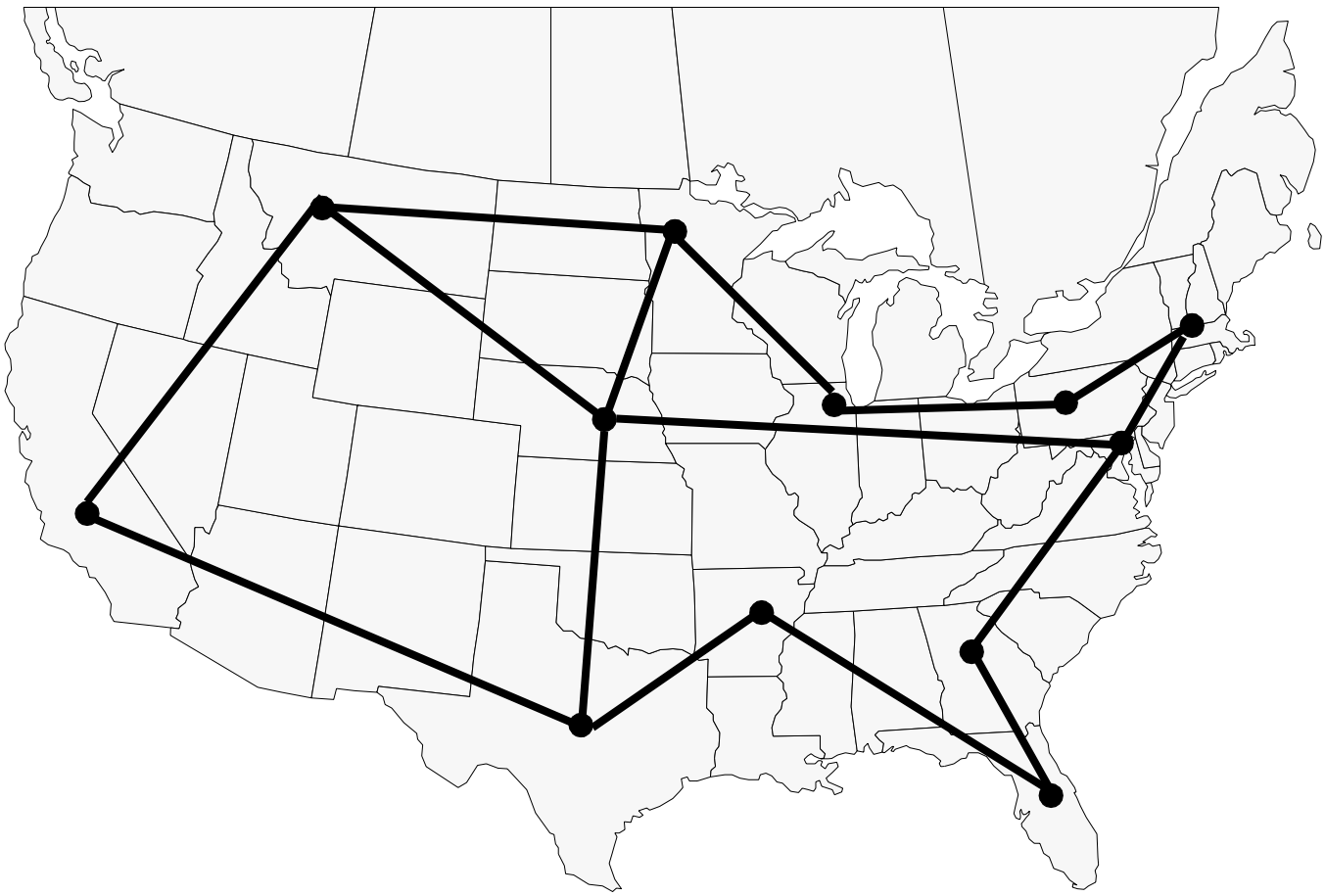
  Macchine dedicate al routing si trovano spesso nella situazione in cui sono possibili percorsi altenativi per raggiungere la stessa destinazione.

  Per una macchina UNIX ciò sarebbe equivalente a possedere default gateways multipli.

  Nella scelta possono entrare in gioco la larghezza di banda dell'interfaccia, pesi definiti in fase di installazione, etc.

  In genere sulle macchine UNIX l'unico meccanismo disponibile è il table based.

TCP/IP: architettura



- Una rete geografica IP può essere un oggetto molto complesso. Per la natura stessa del disegno dell'Internet Protocol esistono percorsi differenti che congiungono due punti del *grafo.*

TCP/IP: Linux

- Nella maggior parte dei sistemi UNIX il comando ***ifconfig***, all'atto della configurazione di una network interface, crea la relativa entry nella tabella di routing.

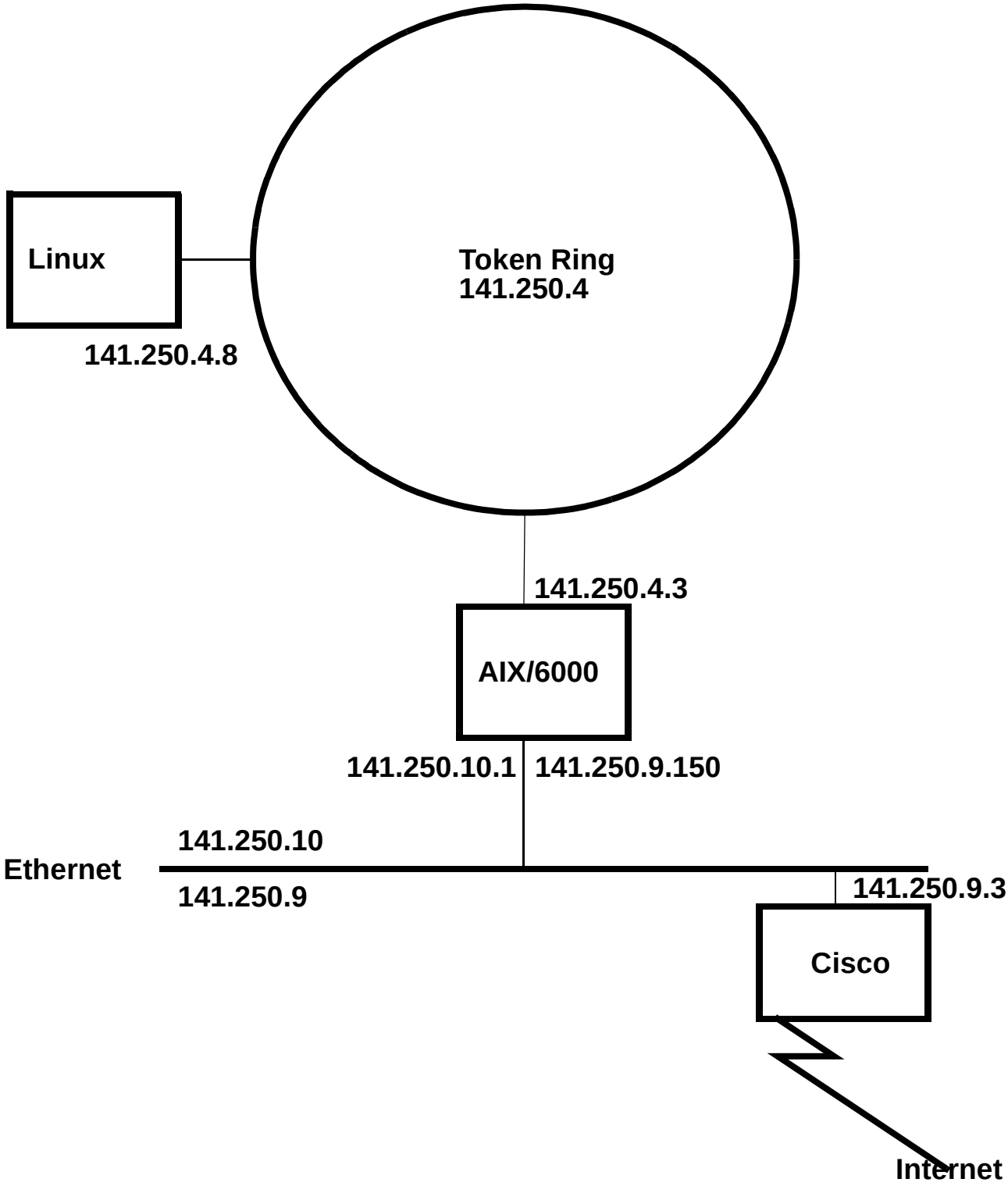
  Nella distribuzione RedHat ciò non avviene: occorre definire ***esplicitamente*** il routing verso l'interfaccia utilizzando il comando **route**:

```
route add -net 141.250.10.0 netmask 255.255.255.0

route add -net 141.250.9.0 netmask 255.255.255.0

route add -net 141.250.4.0 netmask 255.255.255.0 gw 141.250.4.8
```

- Il comando **route** costituisce l'interfaccia tra l'amministratore di rete e la tabella di routing.

  Permette di aggiungere e cancellare entries nella tabella di routing.

TCP/IP: architettura



TCP/IP: architettura

subnet mask: 255.255.255.0

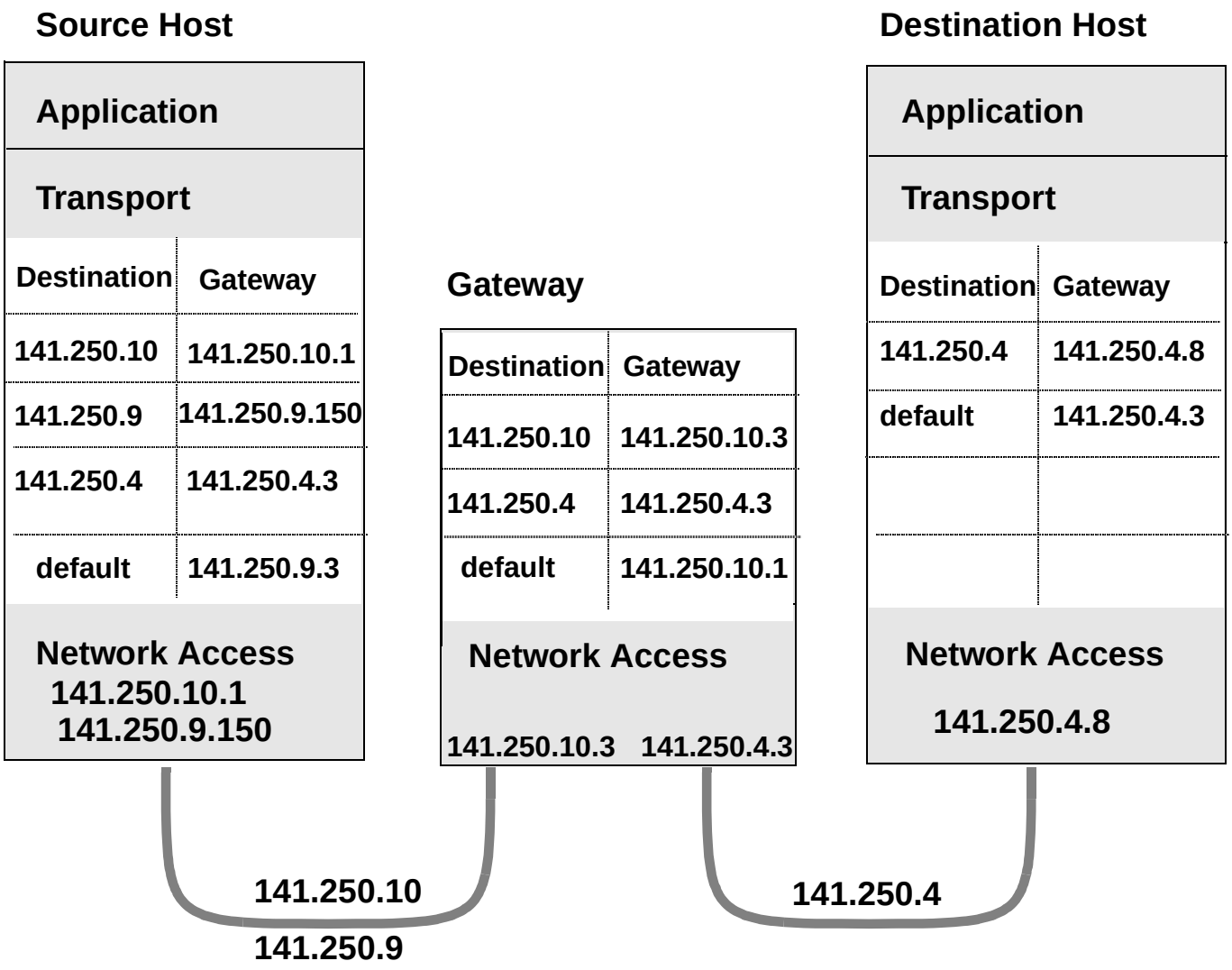


Table Based Routing

TCP/IP: architettura

- Il netid dell'indirizzo IP di destinazione e le tabelle di routing dirigono il datagram verso la destinazione finale: una interfaccia di rete.

- La consegna al destinatario può avvenire solo attraverso l'uso di un protocollo *fisico* di rete.

  Ovviamente questa fase non può essere effettuata dall'Internet Protocol: il datagram IP deve essere a sua volta incapsulato in una *frame* della rete fisica e l'indirizzo IP deve essere tratodotto nell'indirizzo *fisico* della interfaccia di rete del destinatario.

  Questo meccanismo dipende strettamente dal tipo di rete fisica: ethernet, token ring, fddi, etc.

TCP/IP: architettura

- Esamineremo più nel dettaglio uno dei protocolli più utilizzati: l'Address Resolution Protocol (**ARP**).

  In particolare esamineremo l'ARP su una delle reti fisiche più diffuse: Ethernet.

- Su una rete Ethernet ogni interfaccia di rete, ogni scheda che connette una macchina alla rete, possiede un unico indirizzo fisico composto da 48 bits, 6 ottetti:

  `00:A0:C9:FC:63:38`

  che viene assegnato in modo univoco dal vendor e può, in alcuni casi, essere modificato dall'utente (su una rete Ethernet non possono coesistere due schede con lo stesso indirizzo).

TCP/IP: architettura

- Il compito dell'Address Resolution Protocol è proprio quello di *determinare*, in base all'indirizzo IP, l'indirizzo Ethernet del destinatario

Preamble	Destination Address	Source Address	Packet Type	Data	CRC
64 bits	48 bits	48 bits	16 bits	368-12000 bits	32 bits

Ethernet frame format

- Quando l'ARP riceve la richiesta di tradurre un indirizzo IP nell'indirizzo fisico Ethernet, il protocollo consulta una tabella. Se l'indirizzo non esiste già nella tabella, l'ARP invia un ***broadcast*** Ethernet (FF:FF:FF:FF:FF:FF) a tutti gli host della stessa rete fisica.

TCP/IP: architettura

- La frame ethernet contiene la richiesta di identificazione dell'indirizzo IP.

  Se un host sulla stessa rete riconosce il proprio indirizzo IP risponde inviando all'indirizzo ethernet del mittente il proprio indirizzo fisico.

  È facile immaginare cosa può accadere se due host sulla stessa rete fisica condividono lo stesso indirizzo IP.

- Una volta ricevuto l'indirizzo fisico che corrisponde ad un indirizzo IP, l'ARP lo inserisce nella tabella e lo mantiene per un certo periodo di tempo. La ***ARP table*** svolge quindi la funzione di una ***cache*** per gli indirizzi fisici di rete.

TCP/IP: architettura

- Una volta determinato l'indirizzo ethernet del destinatario, la Network Interface costruisce una frame ethernet che contiene, nella parte Data, l'IP datagram e lo trasmette sul supporto fisico.

- La Network Interface destinataria riceve la frame ethernet, elimina l'header fisico e invia l'IP datagram al proprio Internet Protocol che lo trasmetterà agli strati di più alto livello, fino all'applicazione destinataria del messaggio.

- Il protocollo ARP è molto stabile ed in genere non richiede interventi manuali, ma, in caso di malfunzionamenti, il comando UNIX **arp**, può rivelarsi di una certa utilità.

TCP/IP: Linux

- Il comando **arp** permette di visualizzare e modificare la arp table:

```
$ arp -a
rs6.thch.unipg.it (141.250.9.33) at 02:60:8C:2F:7F:3A [ether] on eth0:0
rs5.thch.unipg.it (141.250.9.2) at 02:60:8C:2F:21:89 [ether] on eth0:0
bw02 (141.250.10.2) at 00:A0:C9:FC:61:C0 [ether] on eth0
simbad.thch.unipg.it (141.250.9.61) at 00:A0:C9:95:68:BD [ether] on eth0:0
```

```
$ arp bw02
Address                  HWtype  HWaddress           Flags Mask            Iface
bw02                     ether   00:A0:C9:FC:61:C0   C                      eth0
```

```
$ arp -v
Address                  HWtype  HWaddress           Flags Mask            Iface
rs6.thch.unipg.it        ether   02:60:8C:2F:7F:3A   C                     eth0:0
rs5.thch.unipg.it        ether   02:60:8C:2F:21:89   C                     eth0:0
bw02                     ether   00:A0:C9:FC:61:C0   C                     eth0
simbad.thch.unipg.it     ether   00:A0:C9:95:68:BD   C                     eth0:0
Entries: 4     Skipped: 0      Found: 4
```

- Il comando **arp** può rivelarsi utile anche per inserire *manualmente* un indirizzo nella arp table. È possibile far corrispondere l'indirizzo fisico di una scheda ad altri indirizzi IP che verranno gestiti sulla stessa macchina. Si parla di ***proxy arp***.

TCP/IP: Linux

- Per poter configurare una Network Interface è necessario che il kernel Linux carichi in memoria il *device driver* della scheda di rete che si vuole utilizzare.

- I device drivers delle schede di rete possono essere linkati staticamente nel kernel o caricati dinamicamente come *loadable kernel modules*, mediante il comando ***insmod***.

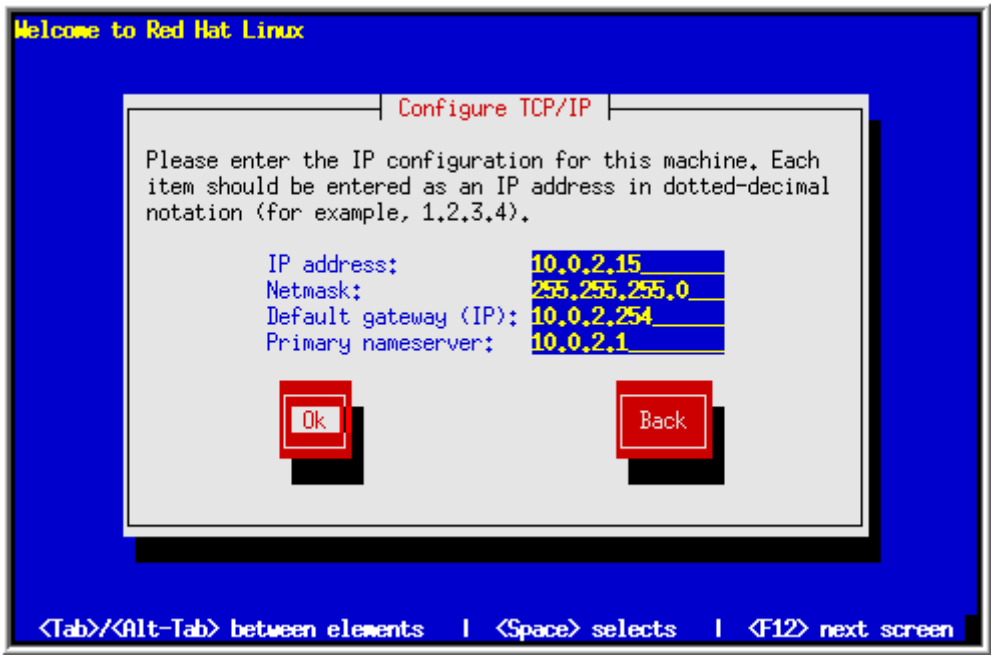
- Linux supporta una grande varietà di hardware di rete. A questo proposito si consiglia di consultare l'elenco dell'hardware supportato dalla distribuzione che si sta utilizzando.

TCP/IP: Linux

- La distribuzione RedHat cerca di identificare la scheda di rete durante la fase di installazione e ne permette la configurazione.

Il programma di installazione propone prima un pannello con l'elenco delle schede di rete individuate e quindi richiede il tipo di configurazione desiderata. Ci porremo nel caso della configurazione di un IP statico, piuttosto comune per un server di rete.

TCP/IP: Linux



Le informazioni richieste permettono di assegnare l'indirizzo IP e il netmask alla Network Interface. Viene inoltre richiesto l'indirizzo del default gateway da inserire nella routing table.

- Il programma di installazione registrerà queste informazioni in un certo numero di files che la

TCP/IP: Linux

script **/etc/rc.d/init.d/network** utilizzerà per inizializzare e configurare il TCP/IP.

- Nel file **/etc/conf.modules** sarà inserita la definizione che identifica il loadable kernel module che supporta l'interfaccia di rete **eth0**

```
alias eth0 eepro100
```

in questo caso una Intel Fast Ethernet Pro 100+.

- Il **kerneld** (kernel daemon) si occuperà di caricare il kernel module **/lib/modules/preferred/net/eepro100.o** quando la script **network** eseguira l'**ifconfig** dell'interfaccia **eth0**.

TCP/IP: Linux

- La script **/etc/rc.d/init.d/network** sarà eseguita quando **init** porterà il sistema al **runlevel 3**. È responsabile della configurazione delle interfacce di rete e dell'aggiornamento della routing table.

- La script legge dal file **/etc/sysconfig/network**

```
NETWORKING=yes
FORWARD_IPV4=true
HOSTNAME=bw01
DOMAINNAME=hpc.thch.unipg.it
GATEWAY=141.250.10.3
GATEWAYDEV=eth0
```

le informazioni sulla configurazione dell'host e passa quindi in esame la directory **/etc/sysconfig/network-scripts**.

In questa directory sono contenuti una serie di files e di scripts che permettono a **network** di

TCP/IP: Linux

eseguire il comando **ifconfig** in modo appropriato sulle corrispondenti interfacce di rete.

Per l'interfaccia **lo** di ***loopback*** (che è presente su tutte le macchine UNIX) il file **ifcfg-lo**

```
DEVICE=lo
IPADDR=127.0.0.1
NETMASK=255.0.0.0
NETWORK=127.0.0.0
BROADCAST=127.255.255.255
ONBOOT=yes
```

Per l'interfaccia **eth0** il file **ifcfg-eth0**

```
DEVICE=eth0
IPADDR=141.250.10.1
NETMASK=255.255.255.0
NETWORK=141.250.10.0
BROADCAST=141.250.10.255
ONBOOT=yes
```

e così via per tutte le interfacce di reti.

TCP/IP: Linux

Al caricamento del kernel module della Network Interface il sistema emette una serie di messaggi di diagnostica che possono essere utili nel caso si evidenzino dei problemi:

```
eepro100.c:v0.99B 4/7/98 Donald Becker linux-eepro100@cesdis.gsfc.nasa.gov
eth0: Intel EtherExpress Pro 10/100 at 0x1060, 00:A0:C9:FC:63:38, IRQ 5.
  Board assembly 677173-001, Physical connectors present: RJ45
  Primary interface chip i82555 PHY #1.
  General self-test: passed.
  Serial sub-system self-test: passed.
  Internal registers self-test: passed.
  ROM checksum self-test: passed (0x24c9f043).
  Receiver lock-up workaround activated.
```

- Utile a questo fino può anche essere il comando **lsmod**

```
$ lsmod
Module         Pages    Used by
ip_alias           1           1 (autoclean)
eepro100           3           1 (autoclean)
ncr53c8xx         12           0
```

per verificare l'avvenuto caricamento della kernel module.

TCP/IP: architettura

- Dopo aver ricevuto un datagram indirizzato all'host locale, l'Internet Protocol elimina il proprio *header* e trasferisce i dati allo strato di *Trasporto* appropriato.

  In questa fase viene esaminato il campo *protocol*: ai protocolli di trasporto è assegnato un valore numerico (unico).

```
ip      0       IP      # internet protocol, pseudo protocol number
icmp    1       ICMP    # internet control message protocol
igmp    2       IGMP    # internet group multicast protocol
ggp     3       GGP     # gateway-gateway protocol
tcp     6       TCP     # transmission control protocol
pup     12      PUP     # PARC universal packet protocol
udp     17      UDP     # user datagram protocol
idp     22      IDP     # WhatsThis?
raw     255     RAW     # RAW IP interface
```

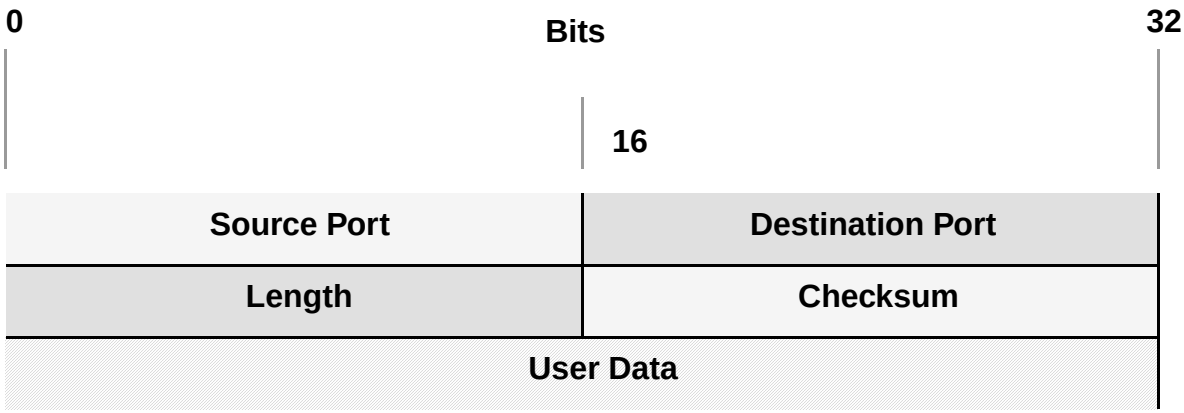
  I Protocol Numbers (raramente modificati) sono elencati, in una macchina UNIX, nel file **/etc/protocol**.

TCP/IP: architettura

- User Datagram Protocol (**UDP**)

  Il protocollo UDP fornisce alle applicazioni un servizio equivalente al servizio fornito dall'Internet Protocol.

  Nel protocollo UDP non esiste il concetto di connessione: il *messaggio* inviato dalla applicazione viene incapsulato in un pacchetto UDP ed inviato allo strato di trasporto UDP di un altro host.



TCP/IP: architettura

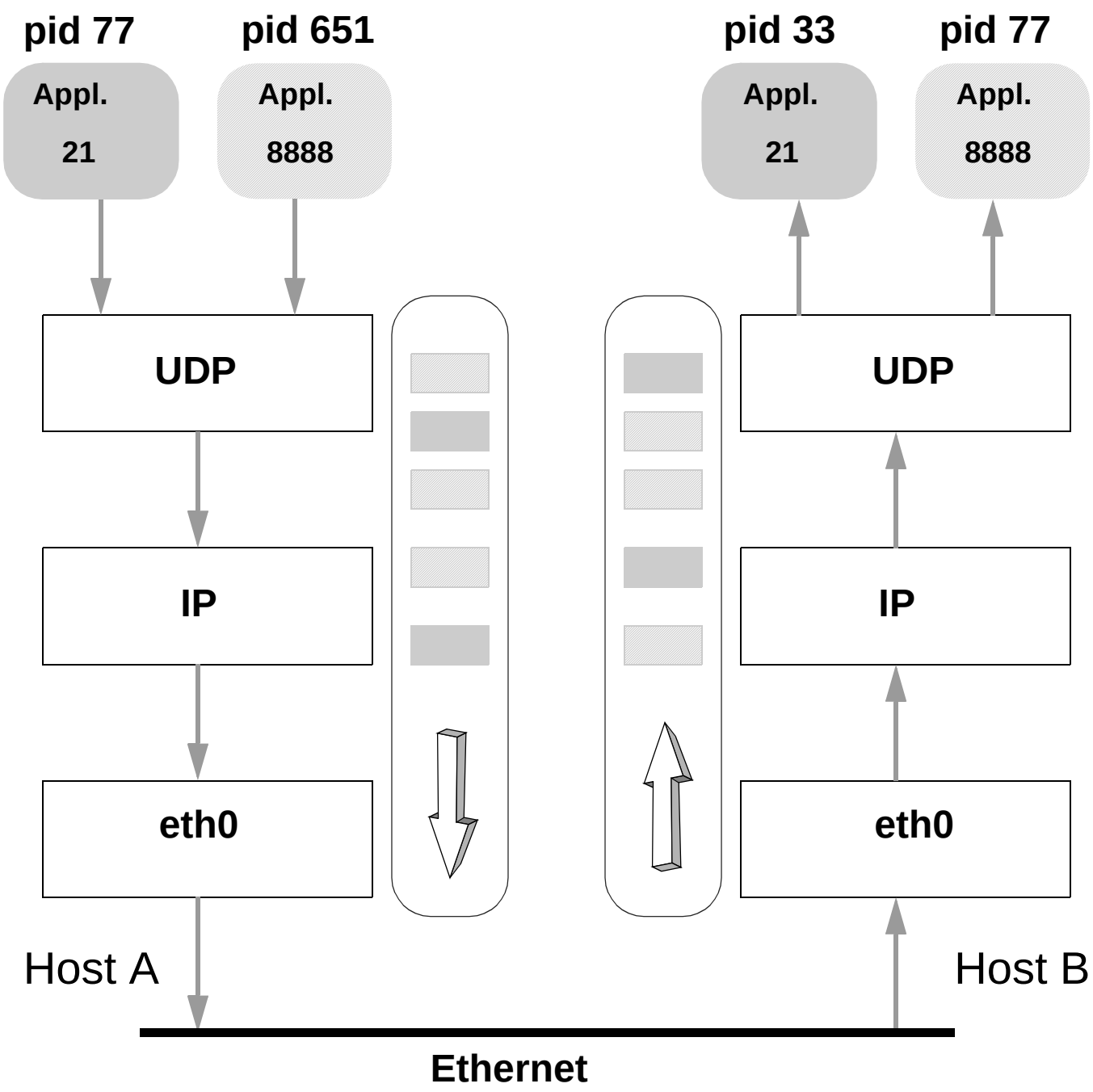
- Il protocollo UDP non garantisce nè la consegna, nè l'ordine di arrivo dei messaggi. È un protocollo *semplice,* adatto ad applicazioni che scambiano piccoli messaggi molto velocemente.

- Due numeri di 16 bits, il *source port number* ed il *destination port number*, identificano in modo univoco le ***applicazioni*** mittente e destinataria del pacchetto.

  È un semplice meccanismo di *multiplexing/demultiplexing* attraverso il quale l'IP può ***moltiplicare*** sè stesso dal punto di vista applicativo.

  Applicazioni differenti possono così utilizzare lo stesso strato di trasporto per comunicare.

TCP/IP: architettura



TCP/IP: architettura

- Le applicazioni, in ambiente UNIX, si identificano con i ***processi***. Più processi possono utilizzare contemporaneamente gli stessi servizi di rete.

- I *protocolli applicativi*, FTP, TELNET, etc. vengono identificati univocamente mediante i *port numbers*.

- La consegna dei dati alla applicazione è una responsabilità dello strato di ***Trasporto***.

  Strati di trasporto (protocolli) differenti (UDP, TCP, etc.) utilizzano contemporaneamente l'Internet Protocol.

TCP/IP: architettura

- Transmission Control Protocol (**TCP**)

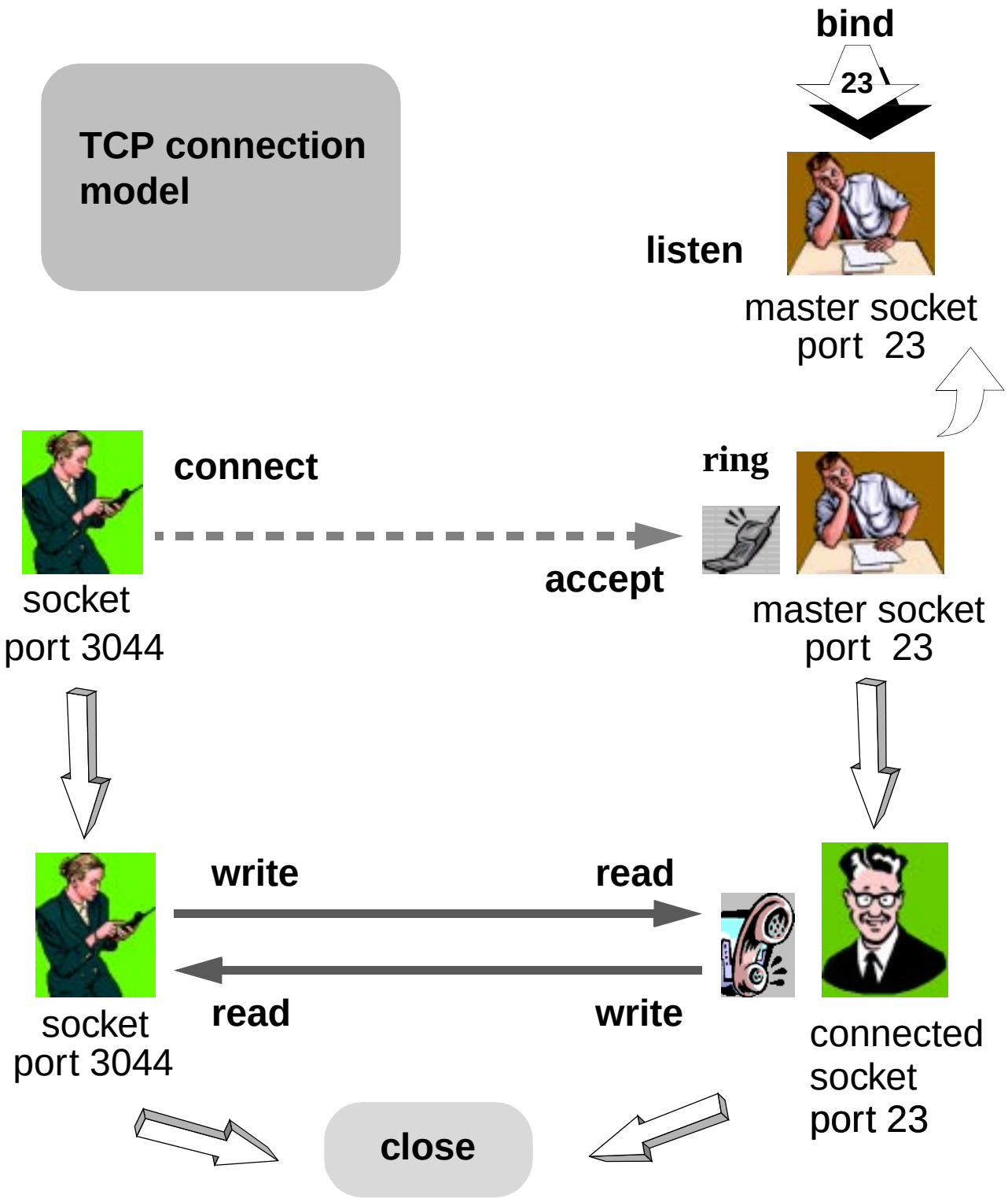
  Il protocollo TCP fornisce un servizio di trasporto ***connection-oriented*** ed *affidabile* alle applicazioni.

  Garantisce la consegna e la sequenza dei dati. Implementa il modello di una *stream* sequenziale di bytes.

  È un protocollo di trasporto più complesso dell'UDP: per poter comunicare due applicazioni devono stabilire una ***connessione***.

  È molto simile a ciò che accade durante una telefonata: una applicazione deve essere in *ascolto* ed un'altra deve richiedere una *connessione*.

TCP/IP: architettura



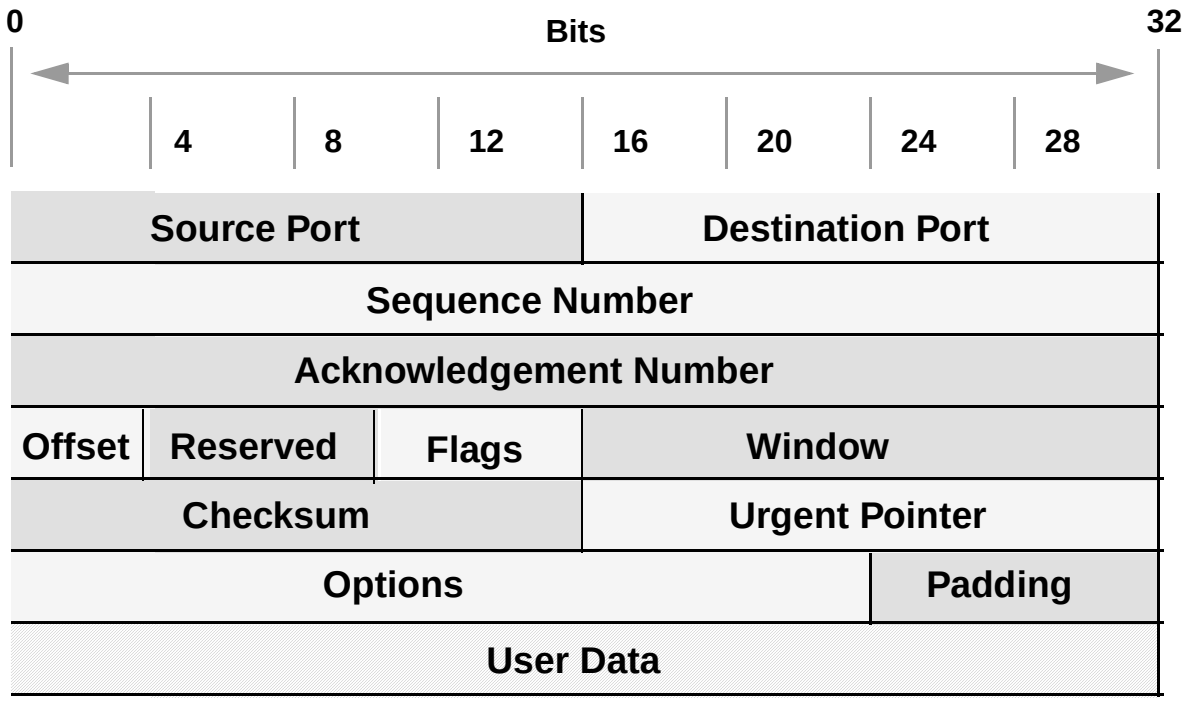
TCP/IP: architettura

- Prima che possa avvenire uno scambio di dati le due parti devono *negoziare* (**handshaking**) una connessione. Vengono scambiati dei dati di *servizio* e raggiunto un accordo sulle modalità del dialogo.

- Nel sistema UNIX le applicazioni scambiano i dati con le stesse primitive utilizzate per accedere i files ***read***() e ***write()***. Dal punto di vista applicativo i **sockets** TCP hanno la struttura di files sequenziali.

- Lo strato di trasporto in effetti incapsula i dati in unità definite ***segmenti***. Ai dati applicativi viene aggiunto l'header TCP.

TCP/IP: architettura



- Il TCP è responsabile della consegna dei dati all'***applicazione***. Il meccanismo è analogo, sebbene più complesso, a quello utilizzato dall'UDP. Il **source port number** identifica il processo mittente ed il **destination port number** il destinatario.

TCP/IP: architettura

- Nel caso del TCP il protocollo distingue tra socket connessi e socket non connessi.

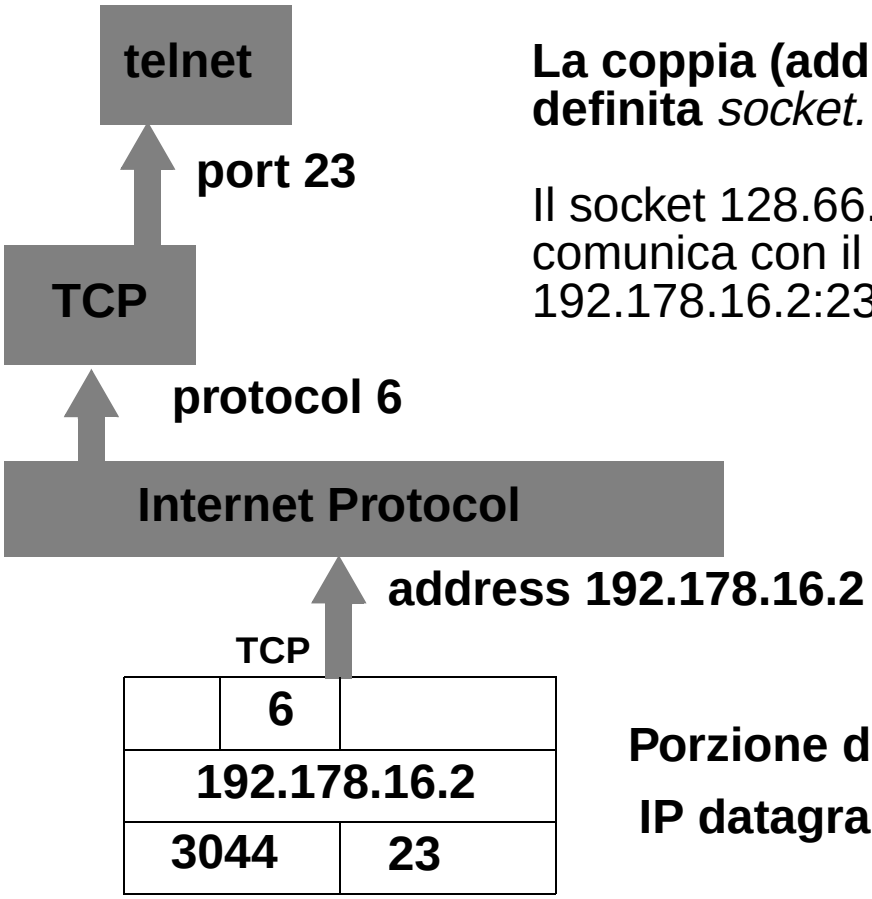
  Un socket *non connesso* è in stato di ***listen***: il processo è in attesa di ricevere una richiesta di connessione.

  Nel momento in cui la connessione viene stabilita, il sistema crea un socket connesso che verrà utilizzato (dallo stesso processo o da un suo figlio) per scambiare i dati.

  Il sistema è ovviamente in grado di distinguere i segmenti TCP diretti ad un ***master socket*** da quelli diretti ad un ***connected socket***.

  Lo scambio di dati applicativi può avvenire solo attraverso l'uso di una ***coppia*** di sockets connessi.

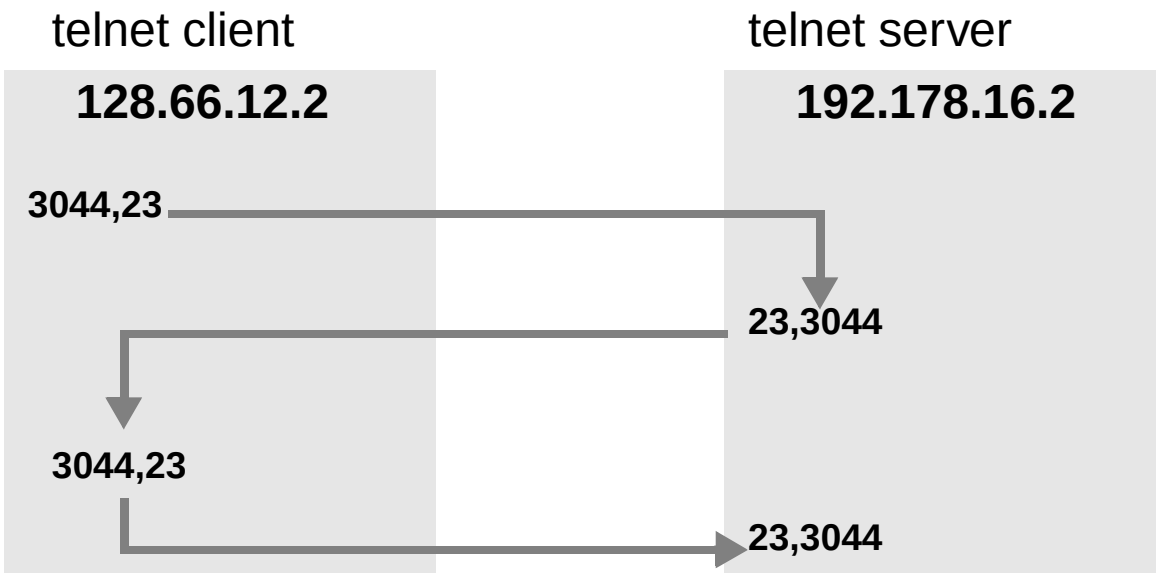
TCP/IP architettura



La coppia (address, port) viene definita *socket.*

Il socket 128.66.12.2:3044 (dinam.) comunica con il socket 192.178.16.2:23 (TELNET).

Porzione di un IP datagram header



TCP/IP: architettura

- Due applicazioni, su due hosts distinti, sono identificate *univocamente* su Internet, dalla coppia (**IP address**, **port number**).

- Ai servizi *fondamentali* sono assegnate delle porte predefinite (FTP tcp/21, TELNET tcp/23) note come ***Well-known*** *ports*.

  I processi server (ftpd, telnetd, etc.) le utilizzano per rimanere in attesa di una connessione da parte dei clients.

- Le applicazioni utente (in genere i clients come telnet, ftp, etc.) utilizzano dei port numbers allocati dinamicamente dal sistema ospite.

TCP/IP: architettura

- Su una macchina UNIX solo i processi con i privilegi del superuser (**root**) possono utlizzare i port numbers inferiori a 1024.

  È uno schema utilizzato spesso per implementare meccanismi di sicurezza: un server che riceve una richiesta da un port number privilegiato può supporre che il processo client stia girando con i privilegi del superuser.

- Sulle macchine UNIX il file **/etc/services** contiene la mappa tra i nomi ed i port numbers dei principali servizi di rete.

  In questo modo le applicazioni possono referenziare i vari port numbers per *nome* invece che per *numero*.

TCP/IP: architettura

- Tutte le informazioni necessarie all'IP ed ai protocolli di trasporto, per la consegna di un pacchetto, sono contenute nei due files **/etc/protocol** ed **/etc/services**.

/etc/protocol

```
ip      0       IP      # internet protocol, pseudo protocol number
icmp    1       ICMP    # internet control message protocol
igmp    2       IGMP    # internet group multicast protocol
ggp     3       GGP     # gateway-gateway protocol
tcp     6       TCP     # transmission control protocol
pup     12      PUP     # PARC universal packet protocol
udp     17      UDP     # user datagram protocol
idp     22      IDP     # WhatsThis?
raw     255     RAW     # RAW IP interface
```

/etc/services

```
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp          mail
...
```

TCP/IP: architettura

- In cima allo stack di protocolli TCP/IP si trova l'**Application Layer**. Contiene le applicazioni utente:
  - TELNET **Network Terminal Protocol**

    login remoto attraverso la rete
  - FTP **File Transfer Protocol**

    trasferimento files
  - SMTP **Simple Mail Transfer Protocol**

    scambio di posta elettronica
  - NFS **Network File System**

    Condivisione di files
  - DNS **Domain Name Service**

    Traduzione di indirizzi IP in nomi simbolici

  e potremmo continuare con HTTP, ...

TCP/IP: architettura

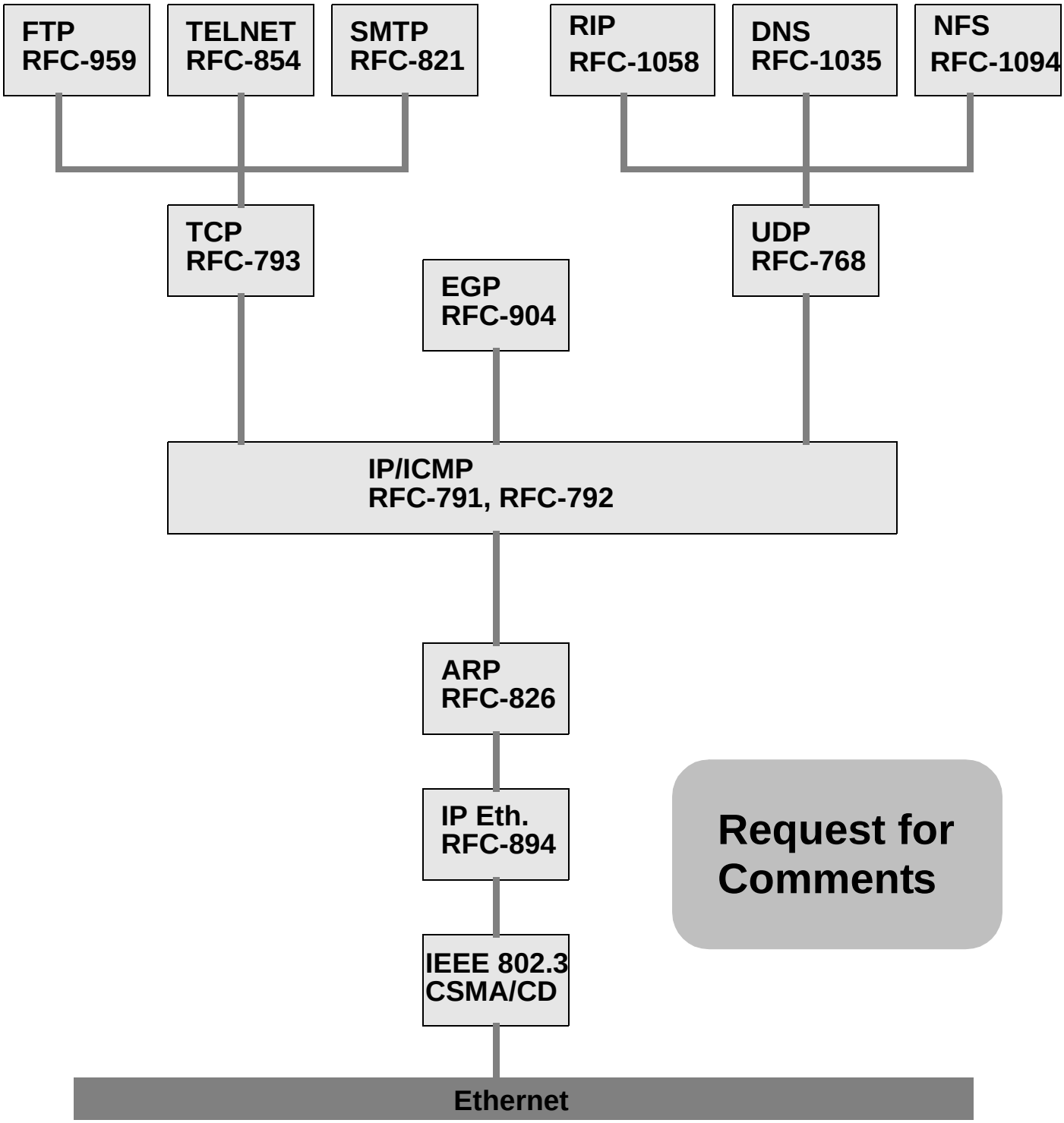
- L'Application Layer è, in effetti, l'Application Programming Interface (**API**) del TCP/IP.

  Il modello dei **Berkeley sockets** è l'unico standard a cui le applicazioni TCP/IP si uniformano.

- Per il resto ogni protocollo applicativo, File Transfer, Terminali di Rete, etc., viene implementato in modo indipendente e per ciascun di essi esiste un RFC (Request for Comments, vedi **http://www.rfc-editor.org**) che ne descrive i dettagli e ne stabilisce lo standard.

- La suite di protocolli TCP/IP è del tutto indipendente dalla implementazione.

TCP/IP: architettura



TCP/IP: Linux

- Ancora una volta il comando **netstat** ci viene in aiuto per visualizzare i ***sockets*** in uso nel sistema:

```
$ netstat -nA inet
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        1      0 127.0.0.1:3141          127.0.0.1:984           TIME_WAIT
tcp        1      0 127.0.0.1:3140          127.0.0.1:984           TIME_WAIT
tcp        0    126 141.250.9.150:23        141.250.9.61:1989       ESTABLISHED
udp        0      0 141.250.9.150:53        0.0.0.0:*
udp        0      0 141.250.10.1:53         0.0.0.0:*
udp        0      0 127.0.0.1:53            0.0.0.0:*
udp        0      0 141.250.9.150:123       0.0.0.0:*
udp        0      0 141.250.10.1:123        0.0.0.0:*
udp        0      0 127.0.0.1:123           0.0.0.0:*
```

- Notiamo una sessione TELNET (port number 23) tra l'host 141.250.9.150 (in cui è stato eseguito il comando) e l'host 141.250.9.61 (da cui è stato fatto il telnet). La coppia di socket connessi in questo caso è:

  141.250.9.150:23 **141.250.9.61:1989**

TCP/IP: Linux

```
$ netstat -A inet
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0    126 bw01.hpc.thch.un:telnet simbad.thch.unipg.:1989 ESTABLISHED
udp        0      0 bw01.hpc.thch.un:domain *:*
udp        0      0 bw01:domain             *:*
udp        0      0 localhost:domain        *:*
udp        0      0 bw01.hpc.thch.unipg:ntp *:*
udp        0      0 bw01:ntp                *:*
udp        0      0 localhost:ntp           *:*
```

Possiamo chiaramente distinguere protocolli, indirizzi e port numbers.

```
$ netstat -aA inet
Active Internet connections (including servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0    132 bw01.hpc.thch.un:telnet simbad.thch.unipg.:1989 ESTABLISHED
tcp        0      0 bw01.hpc.thch.un:domain *:*                     LISTEN
tcp        0      0 bw01:domain             *:*                     LISTEN
tcp        0      0 localhost:domain        *:*                     LISTEN
tcp        0      0 *:time                  *:*                     LISTEN
tcp        0      0 *:finger                *:*                     LISTEN
tcp        0      0 *:login                 *:*                     LISTEN
tcp        0      0 *:shell                 *:*                     LISTEN
tcp        0      0 *:telnet                *:*                     LISTEN
tcp        0      0 *:ftp                   *:*                     LISTEN
```

Il flag ***-a*** ci permette di visualizzare anche i master sockets TCP in ***listen*** per una nuova connessione.

TCP/IP: Linux

- Nel sistema operativo UNIX i servizi applicativi sono gestiti da applicazione classificate, per la maggior parte, con il nome di demoni “daemons”.

- Il loro compito è quello di rimanere in *ascolto* (**listen**) su un master socket, *collegato* (**bind**) ad un well-known port, in attesa della *connessione* (**connect**) di un client.

- Una volta ricevuta una richiesta di connessione, in genere, il server si *biforca* (**fork**): il padre si rimette in ascolto per una nuova connessione sul master socket, il figlio inizia a svolgere il servizio richiesto.

TCP/IP: Linux

- Si parla di:
  - telnetd servizio TELNET
  - ftpd servizio FTP
  - httpd WWW server, HTTP daemon
  - gopherd gopher daemon.
- Alcuni demoni vengono lanciati in fase di avviamento del sistema e gestiscono direttamente il protocollo applicativo. Questa è una strategia adottata per servizi che ricevono un numero elevato di richieste per unità di tempo: **httpd**, **sendmail**, **gopherd**, **DNS named**, etc.
- Per altri servizi (**telnet**, **ftp**, **finger**, etc.) è preferibile una strategia differente.

TCP/IP: Linux

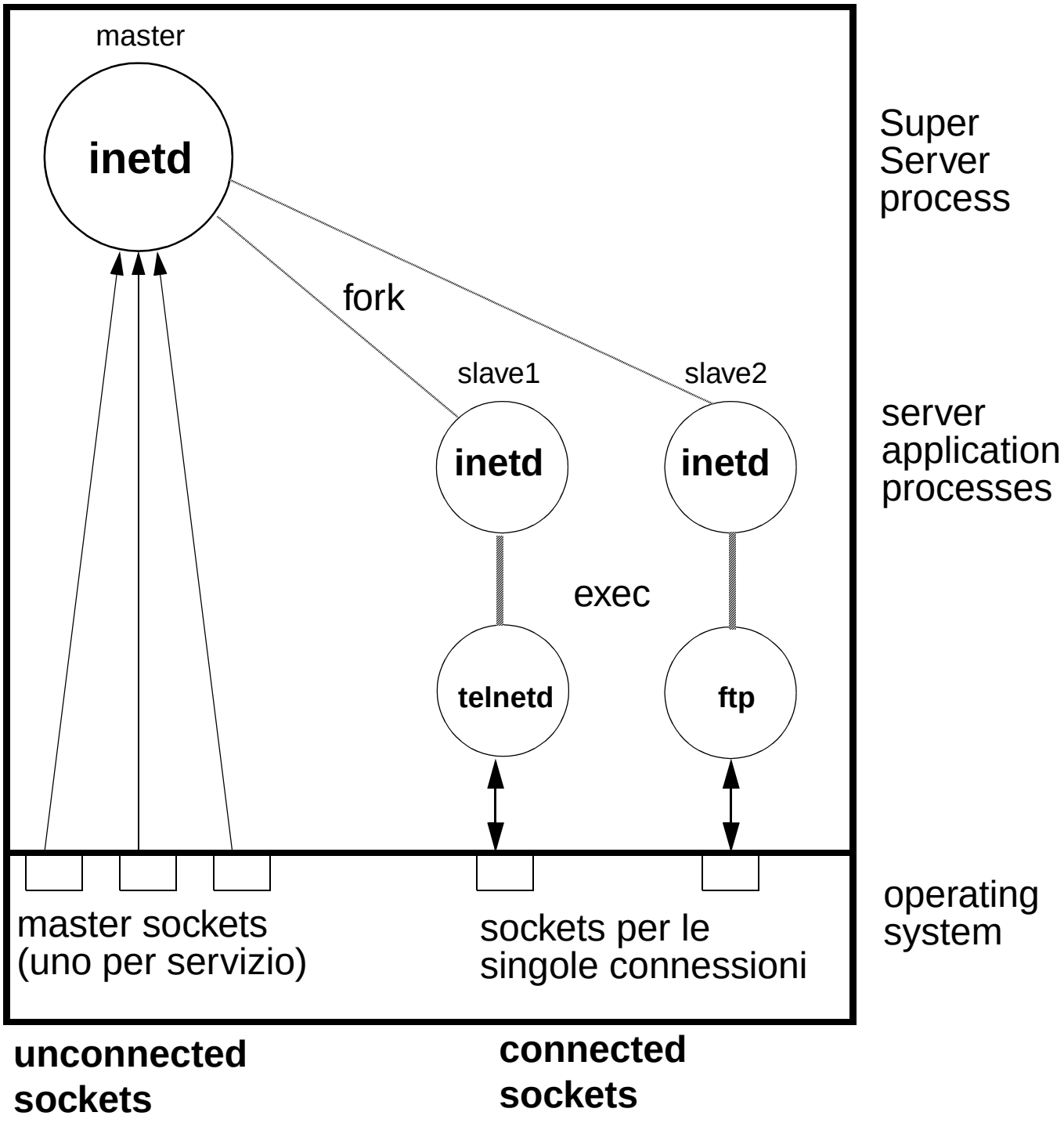
- Esiste un unico demone, noto come ***internet daemon*** o **inetd**, che svolge le funzioni di *centralinistra* smistando un insieme di servizi verso i rispettivi serventi. È un *Super Server*.

- L'**inetd** rimane in ascolto sui master sockets collegati ai port numbers dei servizi che gli sono stati affidati.

  All'arrivo di una richiesta di connessione **inetd** crea il *connected socket*, si biforca e, mediante una ***exec***, carica in memoria il processo che svolgerà la funzione di server per il corrispondente servizio.

  Nell'output di netstat dell'esempio precedente, sui master sockets di **telnet** ed **ftp** era in ascolto, appunto, il processo **inetd**.

TCP/IP: Linux


master
inetd
fork
slave1
slave2
inetd
inetd
exec
telnetd
ftp
master sockets
(uno per servizio)
sockets per le
singole connessioni
Super
Server
process
server
application
processes
operating
system
unconnected
sockets
connected
sockets

TCP/IP: Linux

- L'Internet Daemon viene lanciato alla partenza del sistema dalla script **/etc/rc.d/init.d/inet** quando il sistema si porta al runlevel **3**.

  Il demone legge l'elenco dei servizi di cui si dovrà occupare dal file **/etc/inetd.conf**:

```
ftp     stream  tcp     nowait  root    /usr/libexec/ftpd       ftpd -l
telnet  stream  tcp     nowait  root    /usr/libexec/telnetd    telnetd
shell   stream  tcp     nowait  root    /usr/libexec/rshd       rshd
login   stream  tcp     nowait  root    /usr/libexec/rlogind    rlogind
finger  stream  tcp     nowait  nobody  /usr/libexec/fingerd    fingerd -s
ntalk   dgram   udp     wait    root    /usr/libexec/ntalkd     ntalkd
echo    stream  tcp     nowait  root    internal
discard stream  tcp     nowait  root    internal
chargen stream  tcp     nowait  root    internal
daytime stream  tcp     nowait  root    internal
time    stream  tcp     nowait  root    internal
```

  (da un sistema FreeBSD 2.2).

- Il file inetd.conf può cambiare in modo significativo da un sistema ad un altro, anche della stessa piattaforma.

TCP/IP: Linux

- Ogni riga del file inetd.conf è in effetti il *record* di configurazione di un servizio:

service name	**ftp**
socket type	**stream**
protocol	**tcp**
wait/nowait	**nowait**
user[.group]	**root**
server program	**/usr/libexec/ftpd**
server program arguments	**ftpd -l**

In questo caso **inetd** si metterà in ascolto sul port number **ftp** (23 dal file /etc/services), per una **stream tcp** (6 dal file /etc/protocols).

Il demone si biforcherà e ritornerà in ascolto sul master socket (**nowait**).

Il figlio eseguirà il programma **/usr/libexec/ftpd** con gli argomenti (**ftpd**, **-l**) e comunicherà attraverso il connected socket.

TCP/IP: Linux

- Un estrattto dall'**inetd.conf** di un sistema RedHat 5.2:

```
#daytime        stream  tcp     nowait  root    internal
#daytime        dgram   udp     wait    root    internal
#chargen        stream  tcp     nowait  root    internal
#chargen        dgram   udp     wait    root    internal
#
# These are standard services.
#
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  in.ftpd -l -a
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
#gopher stream  tcp     nowait  root    /usr/sbin/tcpd  gn

# do not uncomment smtp unless you *really* know what you are doing.
# smtp is handled by the sendmail daemon now, not smtpd.  It does NOT
# run from here, it is started at boot time from /etc/rc.d/rc#.d.
#smtp   stream  tcp     nowait  root    /usr/bin/smtpd  smtpd
#nntp   stream  tcp     nowait  root    /usr/sbin/tcpd  in.nntpd
#
# Shell, login, exec and talk are BSD protocols.
#
shell   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rshd
login   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rlogind
#exec   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rexecd
talk    dgram   udp     wait    root    /usr/sbin/tcpd  in.talkd
ntalk   dgram   udp     wait    root    /usr/sbin/tcpd  in.ntalkd
```

Si può notare che in questo caso il server non viene lanciato direttamente dall'inetd, ma prima filtrato da un sistema di controllo di accesso: **/usr/sbin/tcpd**.

TCP/IP: Linux

- Le applicazioni ed i demoni che forniscono i servizi di rete, generalmente, girano in background e disconnessi da un terminale. Non possono quindi avere un dialogo diretto con l'utente.

- I messaggi vengono ridiretti su appositi file di *log,* generalmente gestiti dal demone **syslogd**.

- Le applicazioni che forniscono servizi di rete sono, ***per definizione***, delle esposizioni per la sicurezza del sistema. Ogni volta che ci si mette in ascolto su un *port number* in effetti si apre una **porta** nel sistema da cui potrebbero entrare anche ospiti *indesiderati*. Particolare cura ed attenzione va dedicata alla loro implementazione e configurazione.

TCP/IP: architettura

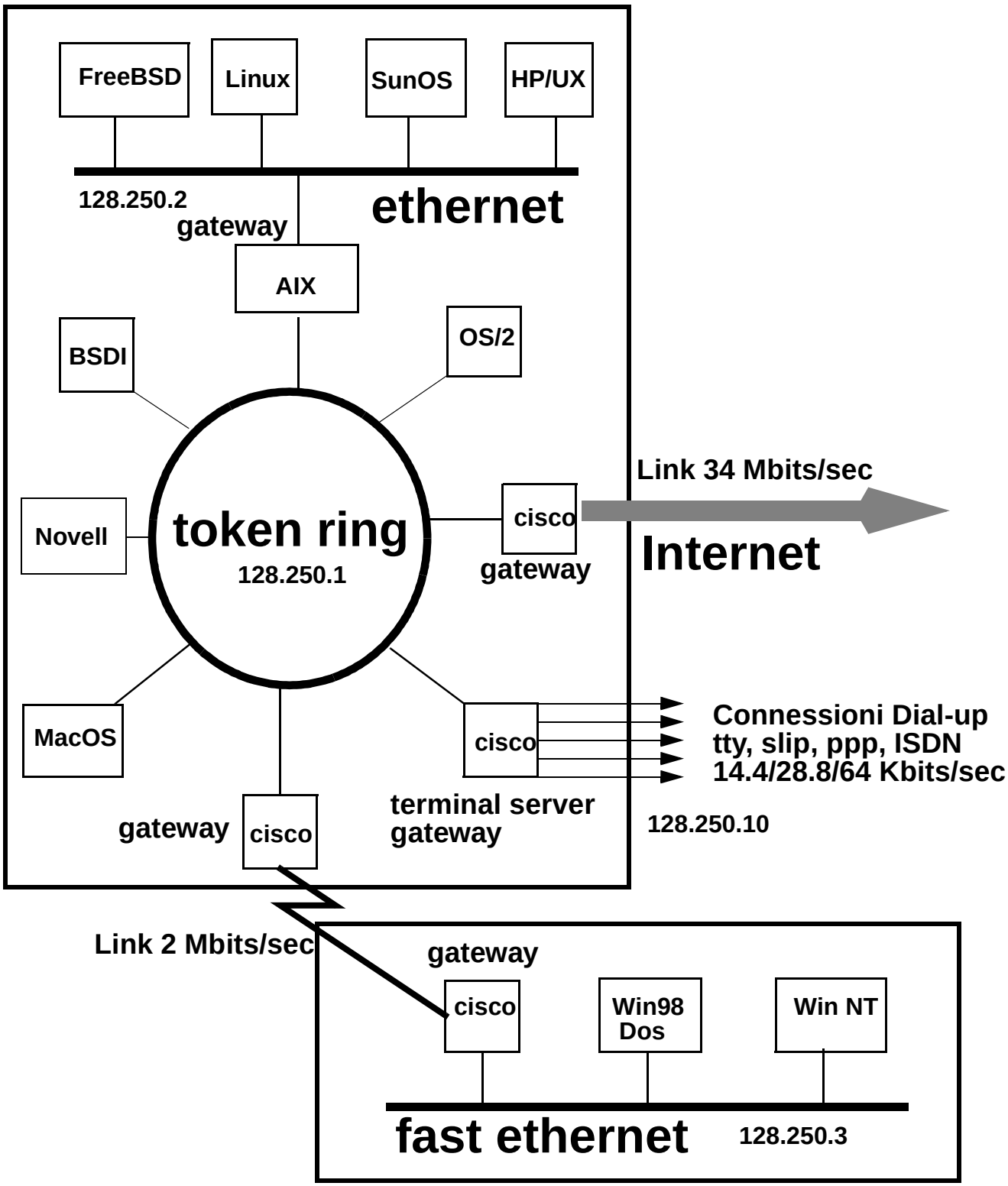
- La storia del TCP/IP spiega almeno in parte il grande successo di Internet: ha permesso di interconnettere rapidamente una grande varietà di reti ed architetture hw/sw.

- In ambienti *eterogenei* il TCP/IP si comporta come il *collante* in grado di far comunicare e cooperare piattaforme apparentemente indipendenti.

- Il TCP/IP è particolarmente adatto alla implementazione di applicazioni *client/server*.

- Nel mondo delle rete TCP/IP è nato il concetto “The machine is the Network”: la rete, non il singolo host, è il centro di tutte le attività.

TCP/IP: architettura



TCP/IP: architettura

- L'Interior Control Message Protocol (**ICMP**), che è parte integrante dell'Internet Protocol, utilizza gli IP datagram per eseguire funzioni di controllo e verifica sulla rete:
  - Flow Control
  - Destinazioni irrangiungibili
  - Routing Redirection
  - Verifica di host remoti
- Il comando **ping** utilizza l'ICMP **Echo Message** per verificare la **raggiungilibità** di un indirizzo remoto. È di uso così comune da aver meritato la *nascita* del verbo ***pingare***.

TCP/IP: architettura

- In pratica il comando invia un Echo Message all'indirizzo che si intende verificare ed informa l'utente sui messaggi ricevuti:

```
$ ping bw02
PING bw02 (141.250.10.2): 56 data bytes
64 bytes from 141.250.10.2: icmp_seq=0 ttl=64 time=0.1 ms
64 bytes from 141.250.10.2: icmp_seq=1 ttl=64 time=0.1 ms
64 bytes from 141.250.10.2: icmp_seq=2 ttl=64 time=0.1 ms
(ctl-C)
--- bw02 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.1/0.1/0.1 ms
```

```
$ ping -s 1024 -c 4 141.250.10.255
PING 141.250.10.255 (141.250.10.255): 1024 data bytes
1032 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.1 ms
1032 bytes from 141.250.10.2: icmp_seq=0 ttl=64 time=0.5 ms (DUP!)
1032 bytes from 141.250.10.1: icmp_seq=0 ttl=64 time=3.1 ms (DUP!)
1032 bytes from 141.250.10.34: icmp_seq=0 ttl=255 time=4.1 ms (DUP!)
(ctl-C)
--- 141.250.10.255 ping statistics ---
1 packets transmitted, 1 packets received, +3 duplicates, 0% packet loss
round-trip min/avg/max = 0.1/1.9/4.1 ms
```

```
$ ping egeo
PING egeo.unipg.it (141.250.1.4): 56 data bytes
ping: sendto: Network is unreachable
ping: wrote egeo.unipg.it 64 chars, ret=-1
(ctl-C)
--- egeo.unipg.it ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
```

TCP/IP: architettura

- **ping** è anche in grado di svolgere funzioni di verifica del percorso (un pò come il comando **tracroute**):

```
$ ping -R egeo
PING egeo.unipg.it (141.250.1.4): 56 data bytes
64 bytes from 141.250.1.4: icmp_seq=0 ttl=252 time=6.9 ms
RR:     bw01 (141.250.9.200)
        cipg03.unipg.it (141.250.7.9)
        unipg-gw-pu.unipg.it (141.250.8.3)
        cipg01.unipg.it (141.250.1.3)
        egeo.unipg.it (141.250.1.4)
        unipg-gw-sl.unipg.it (141.250.8.2)
        cipg02.unipg.it (141.250.7.3)
        cipg03.chm.unipg.it (141.250.9.3)
        bw01 (141.250.9.200)
(ctl-C)
--- egeo.unipg.it ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 6.9/6.9/6.9 ms

$
$ traceroute egeo
traceroute to egeo.unipg.it (141.250.1.4), 30 hops max, 40 byte packets
 1  cipg03.chm.unipg.it (141.250.9.3)  1.020 ms  1.019 ms  1.002 ms
 2  cipg02.unipg.it (141.250.7.3)  1.392 ms  1.390 ms  1.373 ms
 3  unipg-gw-sl.unipg.it (141.250.8.2)  3.509 ms  3.785 ms  6.331 ms
 4  egeo.unipg.it (141.250.1.4)  9.314 ms  4.325 ms  4.378 ms
```

Sono molto utili nel caso si voglia eseguire il debugging di una situazione *anomala*.

Nomi ed Indirizzi

- Ovviamente i calcolatori si trovano perfettamente a loro agio con gli indirizzi IP: trattare informazioni *numeriche* è esattamente il compito per cui sono stati concepiti.

- Per gli essere umani la questione è un pò più complicata. Nella maggior parte dei casi preferiscono di gran lunga avere a che fare con stringhe come **www.ibm.com** piuttosto che con indirizzi come **204.146.18.33** (il telefono fa eccezione?).

- Fin dai *primordi* dell'esistenza della Rete si è posto il problema di far corrispondere, in qualche modo l'*hostname* di una macchina con il suo indirizzo.

Nomi ed Indirizzi

- La prima soluzione proposta è stata, tanto per cambiare, l'uso di una semplice *tabella*, la ***Host Table***.

  È un *semplice* file ASCII che mette in corrispondenza gli *hostnames*, ed i suoi eventuali *aliases*, con l'indirizzo IP.

```
127.0.0.1               localhost loopback
#
141.250.10.1            bw01 bw01.hpc.thch.unipg.it
141.250.10.2            bw02 bw02.hpc.thch.unipg.it
141.250.10.3            bw03 bw03.hpc.thch.unipg.it
141.250.10.4            bw04 bw04.hpc.thch.unipg.it
141.250.10.5            bw05 bw05.hpc.thch.unipg.it
141.250.10.6            bw06 bw06.hpc.thch.unipg.it
141.250.10.7            bw07 bw07.hpc.thch.unipg.it
141.250.10.8            bw08 bw08.hpc.thch.unipg.it
#
141.250.10.254          bwi
```

  Le informazioni vengono registrate nel file **/etc/hosts**.

Nomi ed Indirizzi

- La primitiva **gethostbyname()** legge il file **/etc/hosts** e trasforma il nome nell'indirizzo.

  Ovviamente lo stack di protocolli TCP/IP è in grado di utilizzare solo ed unicamente l'indirizzo numerico: **non esiste spazio per i *nomi* nei datagrams IP!**

  Sono le applicazioni che devono esplicitamente trasformare i nomi in indirizzi.

- Tranne che nei casi più semplici di piccole rete isolate la host table presenta serie complicazioni.

  La tabella deve essere mantenuta congruente su tutte le macchine della rete!

Nomi ed Indirizzi

- Sebbene possa sembrare difficile crederlo, ai primordi della rete ARPANET, era l'unico modo per mappare gli indirizzi in nomi.

  Gli host della rete ARPA aggiornavano periodicamente la host table scaricandola da un sito di riferimento.

  Quando la host table raggiunse dimensioni di migliaia di hosts, fu chiaro che doveva essere implementato uno schema più efficiente.

- Oggi la host table viene in genere utilizzata solo per piccole installazioni e per memorizzare l'indirizzo di host ***indispensabili***: il localhost, l'hostname locale, siti di rifrerimento, etc.

Nomi ed Indirizzi

- Il **Domain Name Service** (**DNS**) supera le limitazioni imposte dalla host table.

  È un database **distribuito**.

  Le informazioni vengono suddivise in ***zone di autorità*** ed un meccamismo di ***delega*** permette a ciascuno di mantenere solo le informazioni di propria competenza.

- Il servizio è implementato con un protocollo relativamente semplice e snello (53/udp e 53/tcp in /etc/services).

  I server mantengono le informazioni su cui hanno ***autorità*** ed i client le utilizzano per risolvere nomi in indirizzi ed indirizzi in nomi.

Nomi ed Indirizzi

- Anche se potrebbe sfuggire ad un'analisi superficiale, lo schema utilizzato è molto simile alla struttura del file system UNIX.

  Il database è ancora una volta strutturato come un albero rovesciato!

- Le *foglie* dell'albero, in questo caso, sono, appunto, gli *hostnames*.

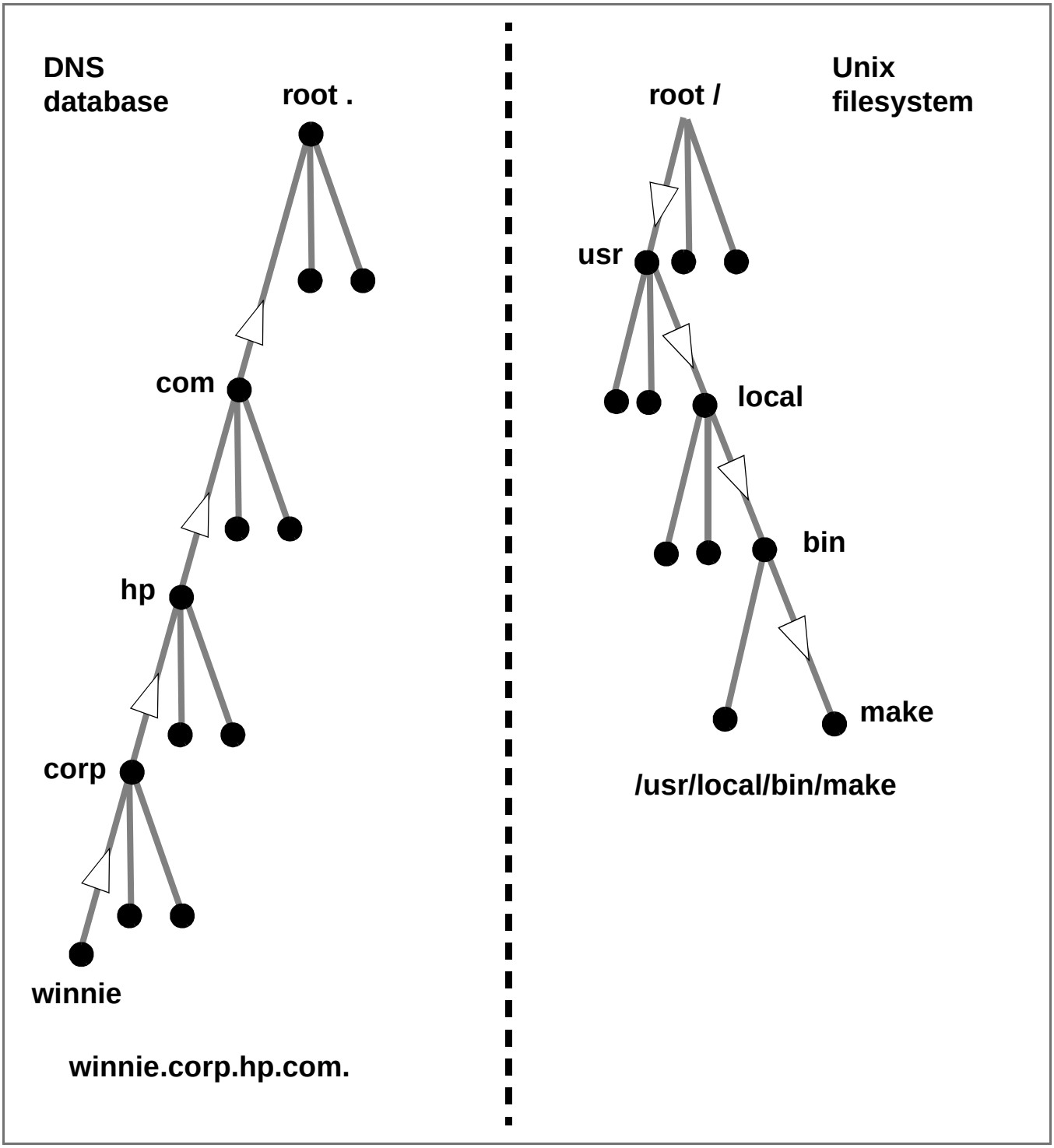
  Come forse vi è già familiare un nome DNS è una stringa della forma:

  winnie.corp.hp.com

  Sostituendo il carattere "**.**" con "***/***", rovesciando il nome ed aggiungendo uno slash si ottiene

  /com/hp/corp/winnie.

Nomi ed Indirizzi



Nomi ed Indirizzi

- Un nome DNS *pienamente qualificato* (Fully Qualified Domain Name **FQDN**), con il punto finale di chiusura, è del tutto analogo ad un pathname assoluto UNIX.

- I nodi dell'albero, i nomi intermedi, sono i ***domini DNS***. I domini possono contenere delle foglie, gli *hostname*, e/o altri domini.

- Ogni dominio è potenzialmente amministrato da una organizzazione differente su un host indipendente. Si potrebbe anzi dire che questa è il paradigma di base: nasce un nuovo dominio tutte le volte che si ravvisa la necessità di *delegare* l'amministrazione di una parte dello spazio dei nomi.

Nomi ed Indirizzi

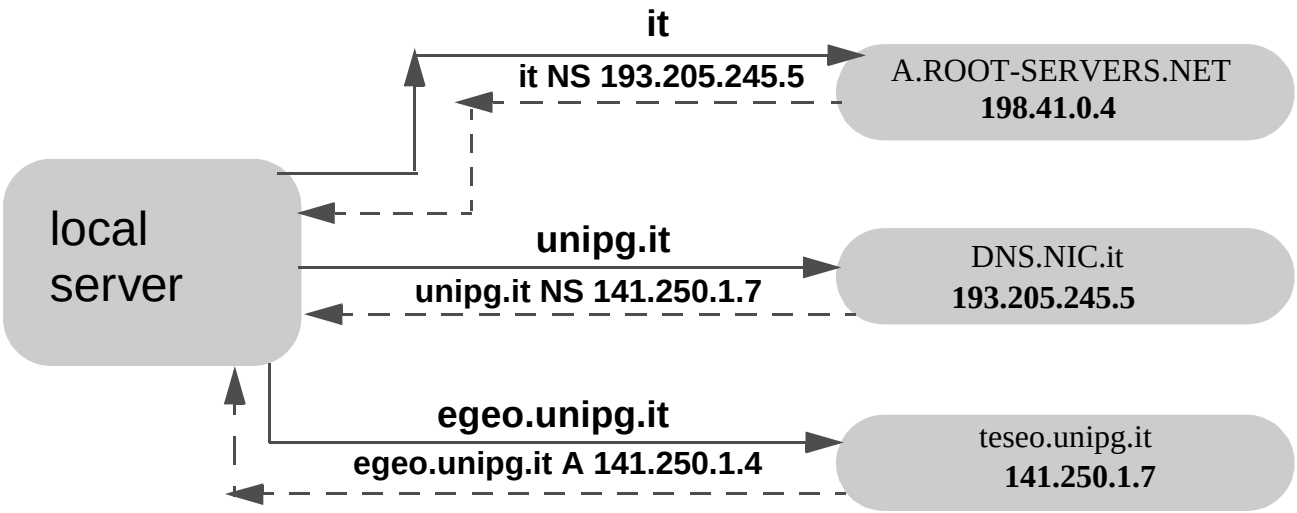
- Al centro di ogni dominio si trova il server, il **Domain Name Server**, che gestisce le informazioni del nodo.

  All'atto della *risoluzione* del nome in un indirizzo viene concettualmente eseguita una scansione dell'albero, **partendo dalla radice**.

  Viene eseguita una *query non ricorsiva* del database:

  - ad uno dei nameservs radice viene richiesto l'indirizzo dei nameservers del dominio **it**;
  - ad uno dei namservers del dominio **it** viene richiesto l'elenco degli indirizzi che hanno autorità per il dominio **unipg**;
  - al nameserver del dominio **unipg** prescelto viene richiesto di risolvere il nome **egeo** nel suo indirizzo 141.250.1.4.

Nomi ed Indirizzi



- Il DNS non può funzionare senza l'accesso ai name servers **radice**: ovviamente i name servers radice possono essere conosciuti solo per indirizzo. Da qualche parte occorre pure cominciare!

- Nel caso il DNS venga attivato su reti *isolate*, sulla rete deve esistere almeno un name server radice locale.

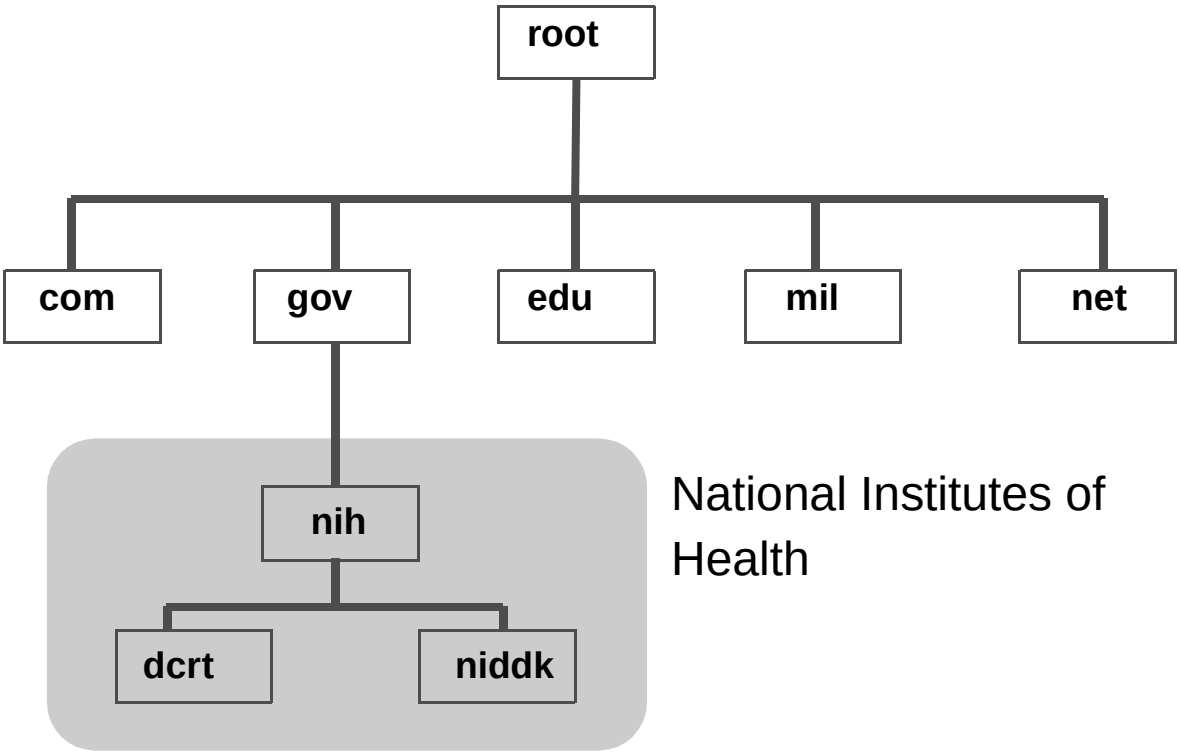
Nomi ed Indirizzi

- L'albero DNS è stato partizionato in sottodomini che si occupano di amministrare tipologie di utenza (per gli USA) o aree nazionali e/o regionali (***top level domains***).

- Negli USA il partizionamento, avvenuto fin dagli albori del DNS, segue lo schema di accomodare settori di utenza:

  - com Organizzazione Commerciali
  - edu Istituzioni Didattiche e/o di Ricerca
  - gov Enti Governativi
  - mil Enti Militari
  - net Supporto di Rete
  - org Organizzazioni diverse.

Nomi ed Indirizzi



National Institutes of Health

- Lo schema persegue il duplice obiettivo di distribuire il carico di lavoro dei name servers su un gran numero di calcolatori e di suddividere le funzioni amministrative fra diverse organizzazioni. Lo scopo è quello di rendere il più possibile snella e veloce la procedura di definzione di un nuovo dominio.

Nomi ed Indirizzi

- Per il resto del mondo (purtroppo o per fortuna il sistema lo hanno implementato per primi gli americani) viene generalmente seguito lo schema di suddivisione per aree nazionali.

  Ad ogni nazione del mondo è stata delegata l'amministrazione del dominio il cui nome è formato da un codice internazionale di due lettere:

  - uk United Kingdom
  - jp Japan
  - us United States
  - it Italia

  e così via. Ogni nazione si è quindi creata una propria gerarchia locale di *top level domains.*

Nomi ed Indirizzi

- Alcune nazioni hanno seguito uno schema identico a quello americano. Per esempio Australia (au) e Taiwain (tw) hanno sottodomini che si chiamano com.au, edu.au, com.tw, edu.tw, mil.tw, etc.

  Oppure hanno implementato uno schema analogo come il Regno Unito: co.uk, ac.uk, etc.

- In altri casi, come quello italiano, la confusione mentale imperante, ha portato a non seguire alcuno schema lasciano che la situazione si evolvesse per proprio conto. In questi casi la frenesia (assurda!) di non delegare la gestione del dominio di livello più alto, ha avuto come conseguenza una gestione del DNS contraria al paradigma software.

Nomi ed Indirizzi

- L'implementazione DNS utilizzata nella stragrande maggioranza dei sistemi UNIX è il *Berkeley Internet Name Domain* (**BIND**): le nostre descrizioni verteranno principalmente su questo pacchetto software (in particolare sulla versione **4.9**).

- Il servizio DNS si può pensare come composto di due componenti software:

  - il ***resolver***

    è una libreria di routines che implementa le principali funzioni per la risoluzione dei nomi (*gethostbyname(), gethostbyaddr(), etc.*);

  - il **name server**

    è un *demone*, il **named**, che implementa il servzio di risoluzione dei nomi.

Nomi ed Indirizzi

- Esistone tre categorie di name server:
  - **primario**

    il *primary server* è il centro nevralgico del sistema DNS; per ogni dominio nello spazio dei nomi DNS dovrebbe esistere uno ed un solo servente primario che gestisce le informazioni del dominio;

    il *primary server* possiede l'*authority* sul dominio: i files che definiscono la *zona di autorità* vengono registrati ed amministrati sul servente primario del dominio.

  - **secondario**

    i *secondary servers* trasferiscono periodicamente il database che definisce il dominio (gli *zone files*) dal primary server; hanno il ruolo di distribuire il carico di lavoro fra diverse macchine e, in genere, costituiscono il *backup* del servente primario.

Nomi ed Indirizzi

- **Caching Only**

  i *caching-only servers* non hanno autorità su una zona; svolgono solo la funzione di mantenere in una ***cache*** i risultati delle queries già eseguite in modo da minimizzare gli accessi sulla rete.

- I dati che definiscono un dominio DNS vengono amministrati sulla stessa macchina su cui gira il *primary name server*: le risoluzioni effettuate dal servente primario di un dominio *devono* essere considerate **accurate** per definizione. Un secondary name server può, in alcuni casi, fornire risoluzioni non corrette.

- Tutti i named mantengono, per un certo periodo di tempo, in una *cache* i risultati delle risoluzioni.

Nomi ed Indirizzi

- Nei sistemi UNIX il resolver è implementato in una libreria (**/usr/lib/libresolv.a** nel RedHat 5.2).

  Le *routines* che si occupano della risoluzione di nomi ed indirizzi leggono alcuni files di configurazione per decidere quale tecnica utilizzare per la risoluzione dei nomi.

  Nella distribuzione Linux RedHat il file **/etc/host.conf** gioca un ruolo essenziale: viene deciso se effetturare le risoluzione mediante la *host table* e/o le quries *DNS* ed in quale ordine:

```
order hosts,bind
multi on
```

  Attenzione all'**ordine**!

Nomi ed Indirizzi

- Le routine del resolver DNS vengono configurate nel file **/etc/resolv.conf**.

  In molte architetture la presenza di questo file è sufficiente per attivare la risoluzione DNS.

- **resolv.conf** è un semplice file ASCII che contiene una serie di definizioni che riguardano le modalità di esecuzione del queruies DNS:

  - **nameserver** *address*

    è l'identificazione, mediante l'***indirizzo IP***, dei nameservers a cui saranno inviate le queries; i nameservers vengono elencati nello stesso ordine in cui appaiono nel file: se il primo non fornisce risposte, dopo un timeout, si passa al secondo e così via; se non esistono definzioni di nameserver le interrogazioni vengono inviate al ***localhost*** ed

Nomi ed Indirizzi

in questo caso il processo ***named*** deve girare sulla stessa macchina;

- **domain** *name*

  domain definisce il dominio di ***default***; è molto simile al concetto di *current directory***;** ai nomi che non contengono il separatore di dominio **dot**, viene appeso il nome del dominio di default; questo meccanismo permette agli utenti di non specificare in continuazione i nomi del dominio e dei sottodomini della loro organizzazione; così il al domain **thch.unipg.it** ed al nome **rs5** corrisponde la query **rs5.thch.unipg.it**;

  se resolv.conf **non contiene** la definizione domain, ma l'hostname è un nome contente il carattere **dot**, la parte di dominio viene utilizzata come dominio di default;

  la variabile di environment **LOCALDOMAIN** può essere usata dagli utenti allo stesso scopo;

Nomi ed Indirizzi

- **search** *domain1, domain2, ..., domainN*

  ha funzione analoga all'entry **domain** e va utilizzata in alternativa; i *domini* elencati vengono appesi ai nomi che non contengono il separatore;

- **resolv.conf** può contenere altre definizioni per cui si rimanda alla pagina del manuale.

```
search          thch.unipg.it unipg.it chm.unipg.it krenet.it
nameserver      141.250.4.11
nameserver      141.250.1.7
```

- Una scelta importante è la quella dell'hostname. Il comando UNIX **hostname** permette di definire (solo il superuser) e visualizzare il nome dell'host.

  Se il sistema è connesso ad una rete IP, ed in particolare usa il DNS, l'hostname, in genere, coincide con il nome di rete della macchina.

Nomi ed Indirizzi

Non è una scelta **obbligata**!

Potrebbero esserci casi, per esempio la presenza di due o più interfacce di rete, in cui l'hostname **non coincide** con **nessuno** degli indirizzi di rete.

Nel caso in cui l'hostname coincida, esiste ancora una scelta da compiere:

- utilizzare l'hostname *semplice* che coincide con l'indirizzo di una delle interfacce di rete;
- utilizzare il nome DNS pienamente qualificato come hostname.

Così per la macchina **egeo** la scelta è tra il semplice egeo e **egeo.unipg.it**.

La *tradizione* ***BSD*** vuole l'uso del nome pienamente qualificato (nel qual caso la

Nomi ed Indirizzi

definizione **domain** nel resolv.conf è ridondante).

- Il comando BSD **host** permette di eseguire delle semplici interrogazioni del DNS:

```
$ host -t a mit.edu
mit.edu has address 18.72.0.100

$ host 141.250.1.4
4.1.250.141.IN-ADDR.ARPA domain name pointer egeo.unipg.it

$ host -t ns unipg.it
unipg.it name server dns2.nic.it
unipg.it name server teseo.unipg.it
unipg.it name server dedalo.unipg.it
unipg.it name server giannutri.caspur.it

$ host -t a rs5
rs5.thch.unipg.it has address 141.250.9.2
rs5.thch.unipg.it has address 141.250.4.11

$ host egeo
egeo.unipg.it has address 141.250.1.4
egeo.unipg.it mail is handled (pri=10) by egeo.unipg.it
egeo.unipg.it mail is handled (pri=50) by ipguniv.unipg.it
```

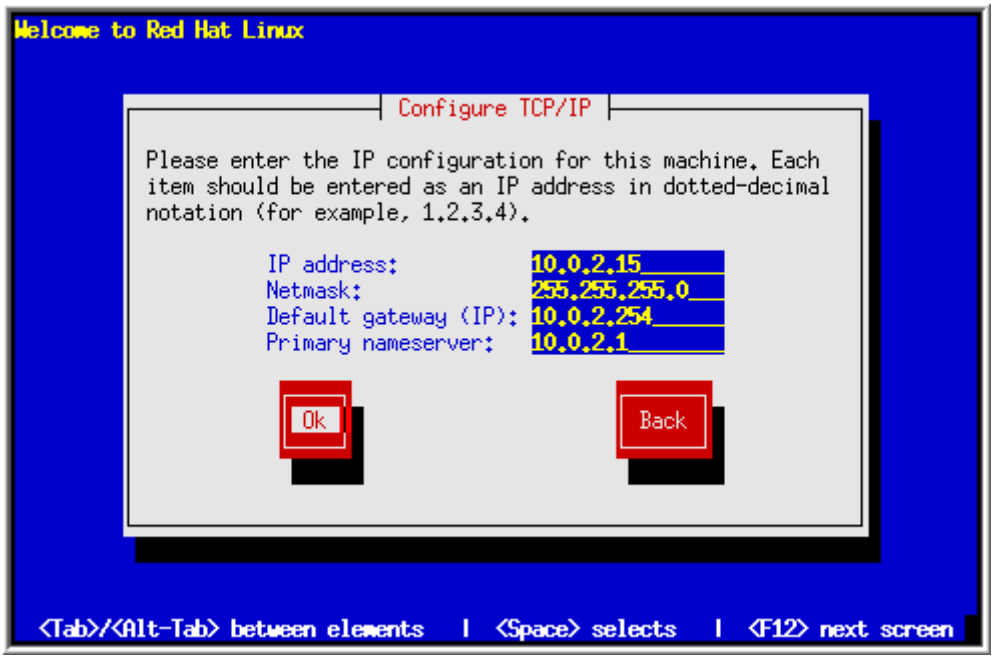
Nomi ed Indirizzi

- Ovviamente tutti i programmi che usano le routine del resolver ricevono lo stesso servizio: per telnet, ftp, ping, etc. la routine **gethostbyname()** tradurrà il nome DNS (o l'hostname nel caso della host table) nel suo indirizzo.

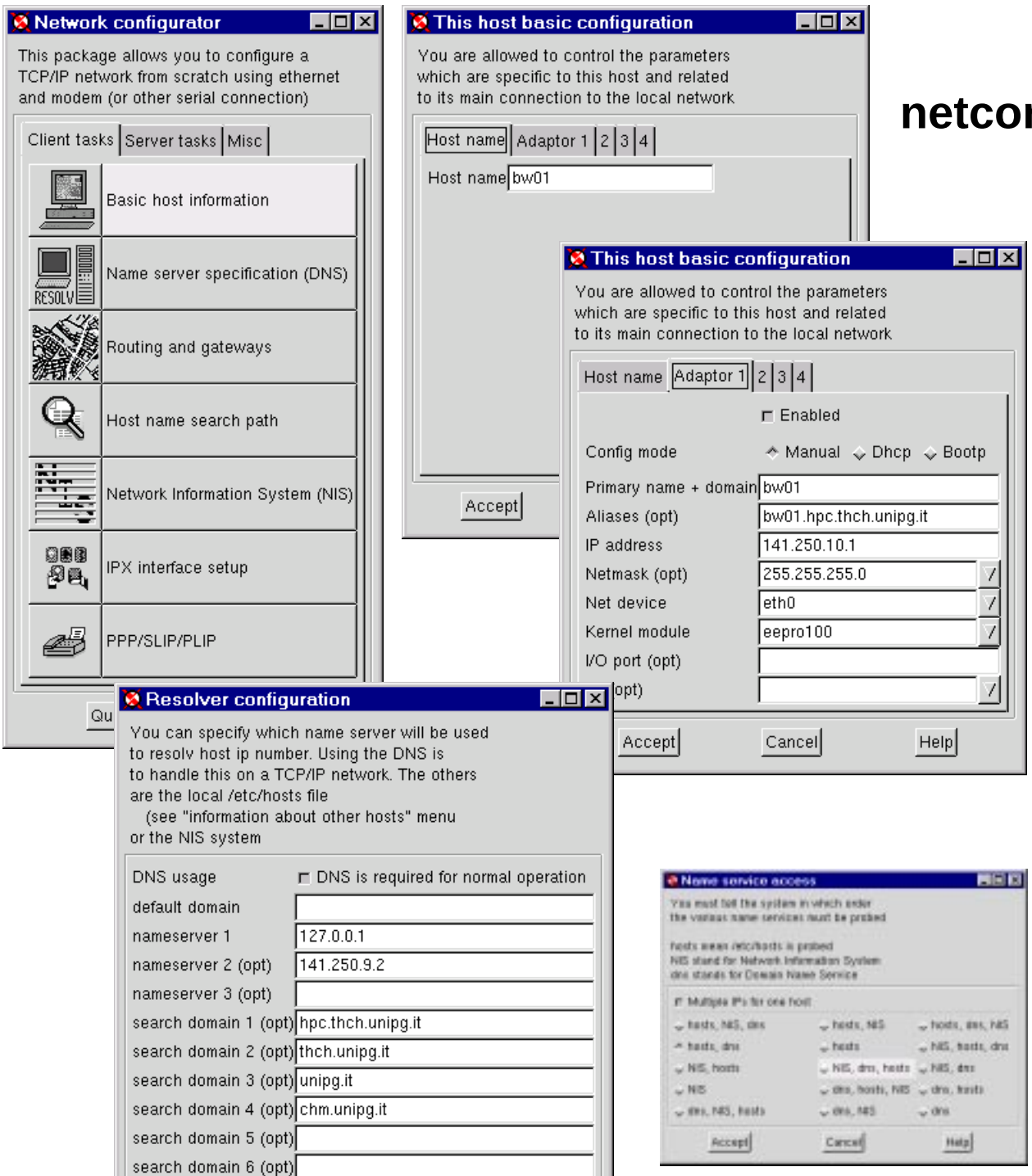
- Una configurazione *resolver-only* non comporta la necessità di configurare un name server: è sufficiente porre le definizioni appropriate nel file ***/etc/resolv.conf*** puntando ai nameserver che hanno l'autorità sul proprio dominio.

- Il programma di installazione del RedHat 5.2 permette di configurare un sistema resolver-only.

Nomi ed Indirizzi



Nomi ed Indirizzi



netconf

Nomi ed Indirizzi

- Un altro programma che può rivelarsi utilile per eseguire interrogazioni DNS è **nslookup**.

  Ha un interfaccia linea di comando molto semplice:

```
$ nslookup
Default Server:  localhost
Address:  127.0.0.1

> set q=a
>
> www.ibm.com
Server:  localhost
Address:  127.0.0.1

Name:    www.ibm.com
Address:  204.146.18.33

>
> set q=ptr
> 4.1.250.141.in-addr.arpa.
Server:  localhost
Address:  127.0.0.1

Non-authoritative answer:
4.1.250.141.in-addr.arpa        name = egeo.unipg.it

Authoritative answers can be found from:
1.250.141.in-addr.arpa  nameserver = teseo.unipg.it
1.250.141.in-addr.arpa  nameserver = dedalo.unipg.it
teseo.unipg.it  internet address = 141.250.1.7
dedalo.unipg.it internet address = 141.250.1.6
```

Nomi ed Indirizzi

- La risoluzione del nome in un indirizzo **IP** è nota come risoluzione ***diretta***. Il problema della risoluzione dell'indirizzo in un nome come risoluzione ***inversa***.

  Sono concettualmente due problemi correlati, ma differenti.

- Nel risolvere l'indirizzo **141.250.1.4** nel nome **egeo.unipg.it** il resolver parte facendo una query al top level domain **arpa**. Il dominio **arpa** è responsabile della delega per il dominio **in-addr** che a sua volta delega la risoluzione degli indirizzi invertendo l'indirizzo di **rete**, secondo la classe di default.

Nomi ed Indirizzi

- Nel caso dell'indirizzo **141.250.1.4** (classe B, netmask di default **255.255.0.0**) il dominio **in-addr.arpa** avrà delegato l'autorità sul dominio **250.141** al nameserver **141.250.1.7**.

  Il nameserver 141.250.1.7 a sua volta fornirà la risposta all'interrogazione fornendo il ***nome*** (il record **PTR**) che corrisponde al nome DNS **4.1.250.141.in-addr.arpa.**, ovvero **egeo.unipg.it**.

- È un meccanismo un pò farraginoso che permette di eseguire la risoluzione inversa utilizzando lo stesso paradigma usato per la risoluzione diretta.

DNS and BIND

- Esamineremo in modo più dettagliato la configurazione del *Berkeley Internet Name Domain* (**BIND**). In particolare analizzeremo la configurazione del **BIND 4.9.6**.

  Le versioni correnti

  - BIND 4.9.7
  - BIND 9.2 (16/3/1999)

  con la relativa *documentazione* possono essere scaricate da

  http://www.isc.org/bind.html

- La versione **4** differisce in modo significativo, sia nei files di configurazione che nelle features disponibili dalla versione **8**.

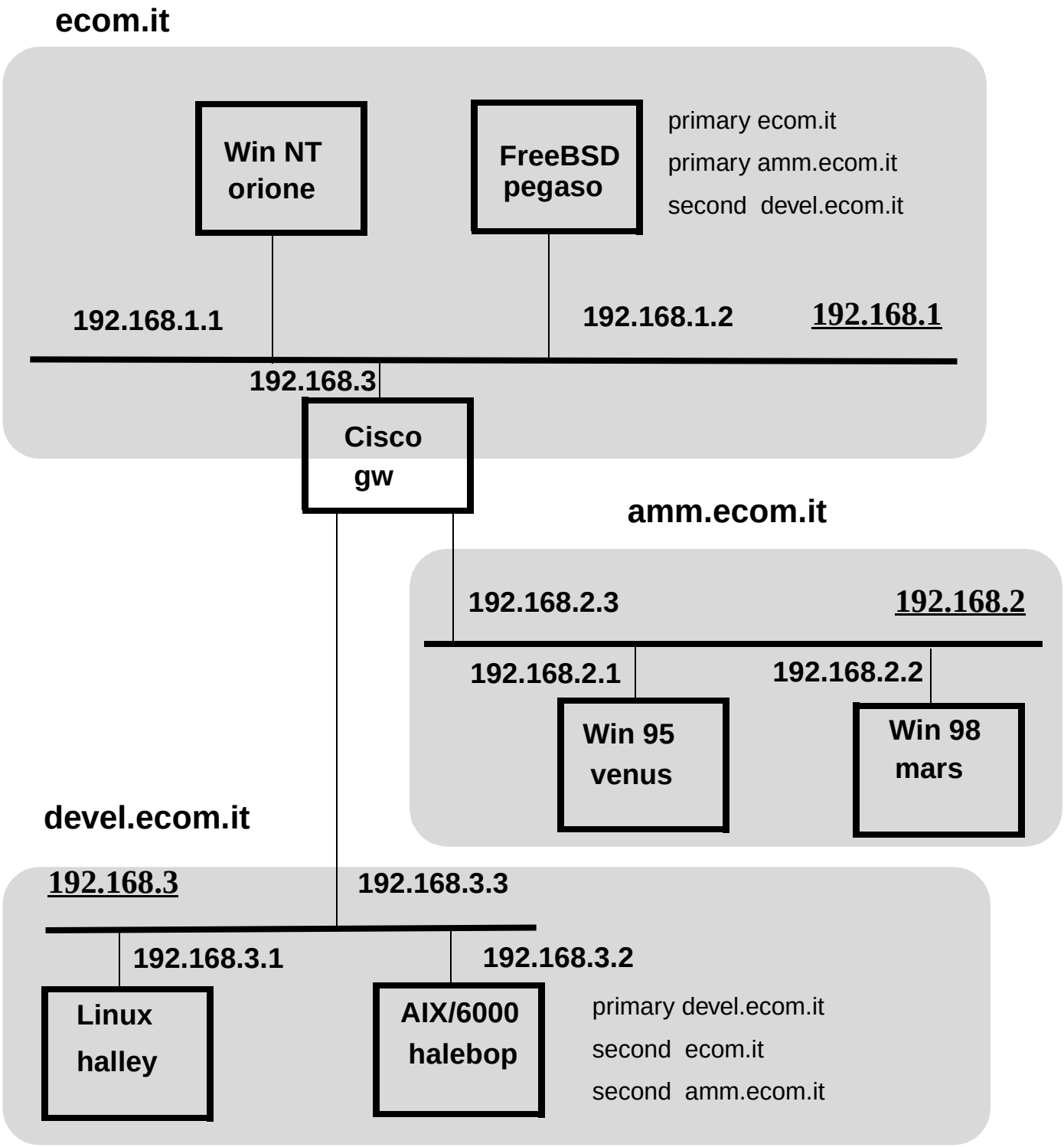
DNS and BIND

- Affronteremo il problema concreto della configurazione di un ***dominio isolato*** ovvero della configurazione del BIND su una macchina UNIX che deve fornire il servizio DNS ad una rete locale ***non connessa*** con Internet.

- Vedremo successivamente come questo *semplice* caso possa essere facilmente trasformato in un ***vero*** dominio della grande Rete.

- La nostra sarà una *piccola* rete Ethernet con un dominio principale **ecom.it** e due sottodomini **amm.ecom.it, devel.ecom.it** per i settori amministrazione e sviluppo.

DNS and BIND


ecom.it
Win NT
orione
FreeBSD
pegaso
primary ecom.it
primary amm.ecom.it
second devel.ecom.it
192.168.1.1
192.168.1.2
192.168.1
192.168.3
Cisco
gw
amm.ecom.it
192.168.2.3
192.168.2
192.168.2.1
192.168.2.2
Win 95
venus
Win 98
mars
devel.ecom.it
192.168.3
192.168.3.3
192.168.3.1
192.168.3.2
Linux
halley
AIX/6000
halebop
primary devel.ecom.it
second ecom.it
second amm.ecom.it

DNS and BIND

- Per prima cosa, dopo aver assegnato un nome ai domini ed agli host, occorre individuare quanti e quali nameserver dovranno essere configurati sulla nostra rete:
  - nella configurazione proposta sarà necessario prevedere un nameserver primario per il dominio **ecom.it** ed il sottodominio **amm.ecom.it** sull'host **pegaso**;

    sarà inoltre utile delegare l'autorità del sottodominio **devel.ecom.it** all'host **halebop** sul quale verrà configurato un altro nameserver primario;
  - inoltre l'host **pegaso** fungerà da nameserver secondario per il dominio **devel.ecom.it** e l'host **halebop** da nameserver secondario per i domini **ecom.it, amm.ecom.it**.

DNS and BIND

- L'host **pegaso** amministrerà le *zone*:
  - **ecom.it**
    - orione.ecom.it 192.168.1.1
    - pegaso.ecom.it 192.168.1.2
    - gw.ecom.it 192.168.1.3
  - **amm.ecom.it**
    - venus.amm.ecom.it 192.168.2.1
    - mars.amm.ecom.it 192.168.2.2
    - gw.amm.ecom.it 192.168.2.3

  e l'host **halebop** amministrerà la *zona:*
  - **devel.ecom.it**
    - halley.devel.ecom.it 192.168.3.1
    - halebop.devel.ecom.it 192.168.3.2
    - gw.amm.ecom.it 192.168.3.3
- L'host **pegaso** avrà autorità anche sulla **radice** del DNS: in una rete isolata almeno un host **deve** avere autorità sulla radice!

DNS and BIND

- Inoltre su **pegaso** verranno registrati gli *zone files* dei domini

  - **1.168.192.in-addr.arpa**

    - 192.168.1.1 orione.ecom.it
    - 192.168.1.2 pegaso.ecom.it
    - 192.168.1.3 gw.ecom.it

  - **2.168.192.in-addr.arpa**

    - 192.168.2.1 venus.amm.ecom.it
    - 192.168.2.2 mars.amm.ecom.it
    - 192.168.2.3 gw.amm.ecom.it

  e su **halebop** gli *zone files* del dominio

  - **3.168.192.in-addr.arpa**

    - 192.168.3.1 halley.devel.ecom.it
    - 192.168.3.2 halebop.devel.ecom.it
    - 192.168.3.3 gw.devel.ecom.it

  per la risoluzione inversa.

DNS and BIND

- Ricordiamo che praticamente tutte le macchine UNIX possiedono l'interfaccia della rete di ***loopback*** configurata sull'indirizzo **127.0.0.1** a cui è uso dare il nome **localhost** o **loopback**.

- Sulla rete Internet nessun nameserver ha autorità sulla zona **127.in-addr.arpa**! È quindi necessario che ogni nameserver gestisca lo zone file del dominio che permette la risoluzione inversa della rete **127**.

- I due **named** che gireranno su **pegaso** e su **halebop** dovranno quindi gestire anche il dominio **127.in-addr.arpa.**

DNS and BIND

- Il BIND **4.9** è implementato nel *demone* **named**: è il demone che fornisce il servizio **DNS** sulla stragrande maggioranza dei sistemi della famiglia UNIX.

- **named** legge dal file di configurazione **/etc/named.boot** la definizione degli *zone files* che costituiscono il database DNS:

  - è un servente di rete che, come tutti i demoni, opera come un processo in background;
  - risponde alle queries DNS sulla porta **UDP 53**;
  - registra i propri messaggi di log, via **syslogd**,sulla facility **daemon**;
  - riceve comandi come segnali inviati dal comando **kill**.

DNS and BIND

- Gli *zone files* sono normali files ASCII che contengono i *records* del database DNS.

  Esistono differenti *tipi* di records che definiscono i differenti tipi di informazione che il resolver può richiedere al nameserver.

- **SOA** records (Start of Authority)

  Questo record definisce l'autorità del nameserver sul dominio

```
@                       172800  IN      SOA     dns root (
                                        1999050601 ; Serial   = yyyymmddxx
                                          86400      ; Refresh  = 1 day
                                          1800       ; Retry    = 30 min
                                          2592000    ; Expire   = 30 days
                                          172800 )   ; Min. TTL = 2 days
```

  ed una serie di informazioni che riguardano il dominio (il cui nome in questo caso, indicato dal carattere @ iniziale, è **ecom.it**).

DNS and BIND

- **NS** records (Name Server)

  Elencano i name servers che hanno autorità sul dominio (ad ognuno di essi deve corrispondere un SOA gestito da un named)

```
;
; Authoritative name servers for this domain: TTL = 5 days
@                       432000  IN      NS      dns
@                       432000  IN      NS      dns.devel.ecom.it.
```

  in questo caso i nameserver DNS del dominio **ecom.it** saranno **dns.ecom.it**, **dns.devel.ecom.it**.

  Tutti i nomi non pienamanente qualificati (cioè che non terminano con il **dot**) vengono completati automaticamente con il nome della zone definito nel file **/etc/named.boot**.

DNS and BIND

Il parametro **TTL** (Time To Live) indica agli altri nameservers quanto tempo l'informazione dovrà essere mantenuta in *cache.*

- **A** records (Name-to-address)

  Sono i records utilizzati per mappare un nome in un indirizzo. Sono i records richiesti dal resolver per trasformare nomi in indirizzi TCP/IP

```
; Information for RR's in this domain. Fake a
; host, so that mail to this domain, and in particular,
; so that mail to postmaster@this.domain works
; even in the case MX records are not read
;
@                    172800  IN      A       192.168.1.2     ;; pegaso

; dns alias for pegaso
dns                  172800  IN      A       192.168.1.2     ; pegaso
```

in questo caso il nome **ecom.it** verrà mappato, così come **dns.ecom.it**, su **192.168.1.2**.

DNS and BIND

- **PTR** records (Address-to-name, Pointer)

  Mappano un indirizzo sul nome DNS. Generalmente compaiono negli zone files che definiscono la risoluzione inversa. Nel caso della zona **1.168.192.in-addr.arpa**

```
; 1.168.192.in-addr.arpa hosts
;
1               172800          IN      PTR     orione.ecom.it.
2               172800          IN      PTR     pegaso.ecom.it.
3               172800          IN      PTR     gw.ecom.it.
```

  permetteranno di risolvere **192.168.1.1** in **orione.ecom.it.**

- **CNAME** records (Canonical name, aliases)

  Permettono di assegnare un *alias* (un nome alternativo) ad un indirizzo IP. Nel nostro caso la macchina **orione.ecom.it**, che ospita un

servizio **http** sarà conosciuta anche come **www.ecom.it**

```
orione                  172800  IN      A       192.168.1.1
                        86400   IN      MX      10 mailhost
www                     172800  IN      CNAME   orione
```

- **MX** records (Mail Exchanger)

  Sono i records che vengono richiesti dal programma ***sendmail***, l'agenti che *trasporta* la posta **SMTP** (Simple Mail Transfer Protocol).

  Nel caso che, per un nome DNS, esistano records **MX** il demone **sendmail** non invierà la posta all'host, ma all'host a cui punta il record MX con la priorità più **bassa**.

```
orione                  172800  IN      A       192.168.1.1
                        86400   IN      MX      10 mailhost
www                     172800  IN      CNAME   orione
```

DNS and BIND

- Esistono altri tipi di records che non analizzeremo nel dettaglio:
  - **TXT** (Textual information)
  - **WKS** (Well-known services)
  - **HINFO** (Host information).
- Sono sempre meno utilizzati per ragioni di *sicurezza* e comunque non costituiscono una parte essenziale della nostra discussione.
- I vari tipi records vengono utilizzati negli zone files per costruire il database del DNS della zona che stiamo definendo.

DNS and BIND

- Il file **/etc/named.boot** dell'host **pegaso**

```
directory       /usr/local/domain
;
primary         .                       root.db
;
primary         0.0.127.in-addr.arpa    p/127
;
primary         ecom.it                 p/ecom.it
primary         1.168.192.in-addr.arpa  p/192.168.1
;
primary         amm.ecom.it             p/amm.ecom.it
primary         2.168.192.in-addr.arpa  p/192.168.2
;
secondary       devel.ecom.it           192.168.3.2      s/devel.ecom.it
secondary       3.168.192.in-addr.arpa  192.168.3.2       s/192.168.3
```

configura il named come nameserver primario per le zone:

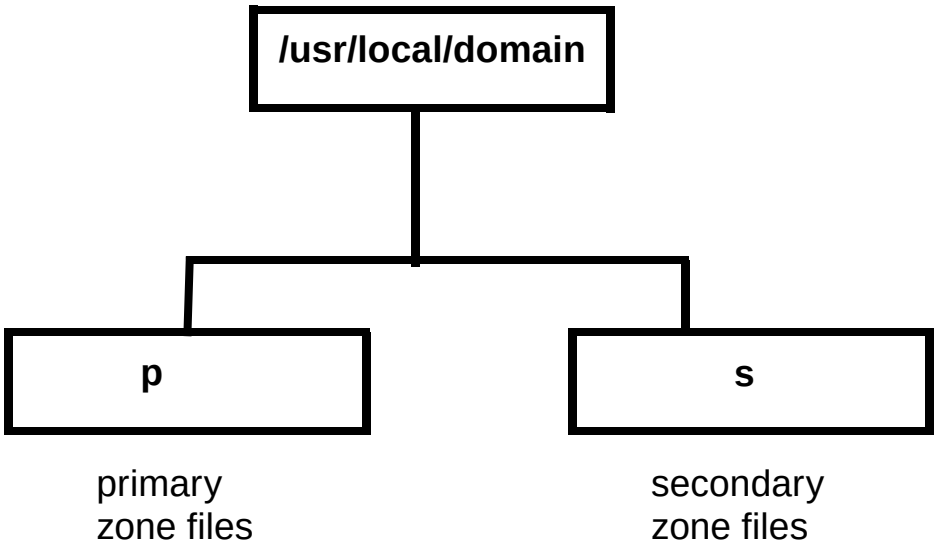
- **.** (**radice**)
- **0.0.127.in-addr.arpa**
- **ecom.it** e **1.168.192.in-addr.arpa**
- **amm.ecom.it** e **2.168.192.in-addr.arpa**

e come nameserver secondario delle zone:

DNS and BIND

- **devel.ecom.it** e **3.168.192.in-addr.arpa**.

- La chiave ***directory*** definisce la directory che il nameserver userà come ***radice***: tutti i files che costituiscone il database si troveranno al di sotto di questa directory



e costituiranno il dababase del BIND.

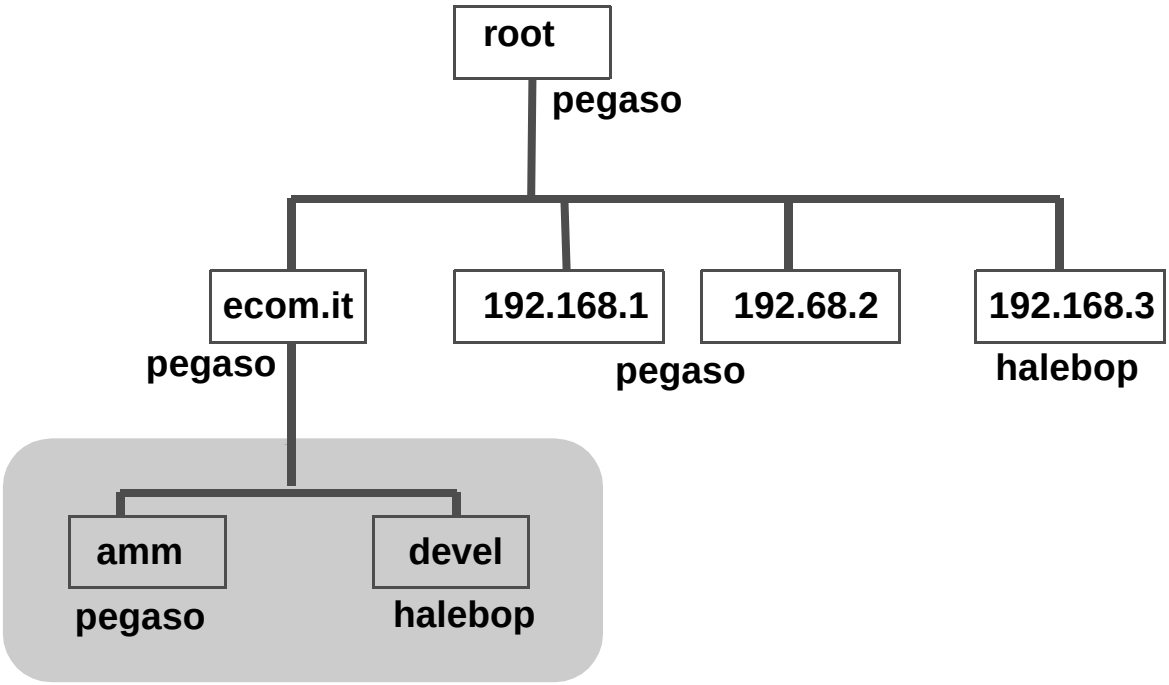
DNS and BIND

- Il nameserver dell'host **pegaso** avrà autorità sulla radice: la rete che stiamo configurando sarà una ***internet*** isolata da **Internet**

```
;       file     : root.db
;       author   : G. Vitillaro (peppe@unipg.it)
;       date     : May 6, 1999
;       scope    : AUTHORITATIVE DATA FOR: root
;       comments : Serial yyyymmddxx will hold 100 changes for day
;                : until 4294123199 (32 long integer) 31/12/4294
;
;
@                       172800  IN      SOA     dns.ecom.it.
root.dns.ecom.it. (
                                      1999050601 ; Serial   = yyyymmddxx
                                        86400      ; Refresh  = 1 day
                                        1800       ; Retry    = 30 min
                                        2592000    ; Expire   = 30 days
                                        172800 )   ; Min. TTL = 2 days
;
; Authoritative name servers for this domain: TTL = 5 days
@                       432000  IN      NS      dns.ecom.it.
;
; Delegations for subdomains: TTL = 5 days
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ecom.it                 432000  IN      NS      dns.ecom.it.
1.168.192.in-addr.arpa  432000  IN      NS      dns.ecom.it.
2.168.192.in-addr.arpa  432000  IN      NS      dns.ecom.it.
3.168.192.in-addr.arpa  432000  IN      NS      dns.devel.ecom.it.
```

e lo *zone file* **root.db** definisce i *top-level* domains.

DNS and BIND



- Lo zone file che definisce l'autorità sul dominio **0.0.127.in-addr.arpa**, per la risoluzione inversa della rete **127**, è praticamente identico per tutti i nameserver.

  Contiene il record **SOA** della zona ed il record **PTR** dell'indirizzo **127.0.0.1** mappato sul nome **localhost.** sotto la radice.

DNS and BIND

```
; 127
;
;       file   : 127
;       author : G. Vitillaro (peppe@unipg.it)
;       date   : May 6, 1999
;       scope  : Data file for local loopback interface
;
@                       172800  IN      SOA     dns.ecom.it.
root.dns.ecom.it. (
                                        1999050601 ; Serial   = yyyymmddxx
                                          86400      ; Refresh  = 1 day
                                          1800       ; Retry    = 30 min
                                          2592000    ; Expire   = 30 days
                                          172800 )   ; Min. TTL = 2 days

;
@       432000                  IN      NS      dns.ecom.it.
;
1       172800                  IN      PTR     localhost.
```

- Il serial number **1999050601** è un *semplice* numero progressivo a 32 bits (non segnato), con un valore tra **0** e **4.294.967.296**.

  Deve essere incrementato di **1** *tutte* le volte che si modifica uno zone file e si *restarta* il **named**.

DNS and BIND

- Si può decidere di incrementarlo partendo da zero, ma la scelta più semplice è quella di utilizzare uno schema del tipo

  yyyy mm dd vv

  dove:

  - **yyyy** anno corrente
  - **mm** mese corrente
  - **dd** giorno corrente
  - **vv** versione del giorno.

- I nameserver secondari utilizzano il serial number per decidere se trasferire lo zone file dal nameserver primario.

  Errori *banali* di incremento del serial number nel SOA record di una zona possono causare gravi malfunzionamenti del DNS.

DNS and BIND

- L'host **pegaso** avrà l'autorità per il *top-level* domain **ecom.it** della nostra rete. Lo zone file **p/ecom.it** contiene i records corrispondenti:

```
;       file     : ecom.it
;       author   : G. Vitillaro (peppe@unipg.it)
;       date     : May 6, 1999
;       scope    : AUTHORITATIVE DATA FOR: ecom.it
;       comments : Serial yyyymmddxx will hold 100 changes for day
;                : until 4294123199 (32 long integer) 31/12/4294
;
;
@                       172800  IN      SOA     dns root (
                                      1999050601 ; Serial   = yyyymmddxx
                                        86400      ; Refresh  = 1 day
                                        1800       ; Retry    = 30 min
                                        2592000    ; Expire   = 30 days
                                        172800 )   ; Min. TTL = 2 days
;
; Authoritative name servers for this domain: TTL = 5 days
@                       432000  IN      NS      dns
@                       432000  IN      NS      dns.devel.ecom.it.
;
; Information for RR's in this domain. Fake a
; host, so that mail to this domain, and in particular,
; so that mail to postmaster@this.domain works
; even in the case MX records are not read
;
@                      172800  IN      A       192.168.1.2     ;; pegaso
;
; MX records for domain: must have canonical names because of the above
;
@                       172800  IN      MX      10 pegaso
```

DNS and BIND

- Sulla zona **ecom.it** avranno autorità i due nameservers:
  - **dns.ecom.it**
  - **dns.devel.ecom.it**

  i cui indirizzi non sono stati ancora definiti.

  dns.ecom.it dovrà *puntare* a **pegaso** con indirizzo **192.168.1.2** e **dns.devel.ecom.it** dovrà puntare ad **halebop** con indirizzo **192.168.3.2**.

- Possiamo notare il record **A** per il dominio: è un modo semplice per definre il cui indirizzo risponde alla risoluzione per il nome del dominio. È particolarmente utile per la gestione della posta elettronica diretta al dominio (indirizzi del tipo **user@ecom.it**).

DNS and BIND

- Poichè il sottodominio **devel.ecom.it** sarà amministrato su un altro host, lo zone file di **ecom.it** dovrà ***delegare*** l'autorità su questa zona all'host **halebop**:

```
;
; Delegations for subdomains: TTL = 5 days
;
; devel
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
devel                   432000  IN      NS      dns.devel

; GLUE : only names that delegates for subdomains
;        where a nameserver not on unipg.it exists
;        TTL = 7 days
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; devel
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
dns.devel               604800  IN      A       192.168.3.2
```

- Notiamo che, poichè l'host **halebop** è una *foglia* di un sottodominio, il nameserver **pegaso** dovrà contenere il suo record A nella zona di **ecom.it**.

DNS and BIND

- Segue una tabella di records A e CNAME che definisce nomi che in genere si vuole siano risolti in tutti i domini in cui operano macchine UNIX:

```
;
; make sure "localhost" is know in this domain.
localhost               432000  IN      A       127.0.0.1
loopback-host           432000  IN      CNAME   localhost
loopback                432000  IN      CNAME   localhost
;
; define loghost (equal loopback for now)
loghost                 432000  IN      CNAME   localhost
;
```

  e che, in genere, riguardano il localhost e l'host che verrà utilizzato come sistema di **log** unificato del dominio.

- Infine la parte dello zone file **ecom.it** che definisce i **nomi** del dominio:

DNS and BIND

```
;
; ecom.it hosts
;
pegaso                  172800  IN      A       192.168.1.2
                        86400   IN      MX      10 pegaso
ftp                     172800  IN      CNAME   pegaso

; MX alias for pegaso - change it as pegaso change
mailhost                172800  IN      A       192.168.1.2     ; pegaso

; dns alias for pegaso
dns                     172800  IN      A       192.168.1.2    ; pegaso
;
; the others in IP address order
; they "may" have the aliased A records in the MX record
;
orione                  172800  IN      A       192.168.1.1
                        86400   IN      MX      10 mailhost
www                     172800  IN      CNAME   orione

gw                      172800  IN      A       192.168.1.3
                        86400   IN      MX      10 mailhost
```

Notiamo la definzione del nome **dns.ecom.it** che punta al nameserver primario **pegaso** del dominio.

Vengono anche definiti gli alias **www, ftp** del dominio per i servizi **http** ed **anonymous ftp**.

DNS and BIND

- **pegaso** avrà anche autorità sui domini **1.168.192|2.168.192.in-addr.arpa** per la risoluzione inversa. Esaminiamo lo zone file **192.168.1**:

```
;       file   : 192.168.1
;       author : G. Vitillaro (peppe@unipg.it)
;       date   : May 6, 1999
;       scope  : Inverse name resolution for 1.168.192.in-addr.arpa
;
;
@               172800          IN      SOA     dns.ecom.it.
root.dns.ecom.it. (
                                        1999031501 ; Serial   = yyyymmddxx
                                          86400      ; Refresh  = 1 day
                                          1800       ; Retry    = 30 min
                                          2592000    ; Expire   = 30 days
                                          172800 )   ; Min. TTL = 2 days
;
;
;
; Authoritative name servers for this domain: TTL = 5 days
;
@               432000          IN      NS      dns.ecom.it.
@               432000          IN      NS      dns.devel.ecom.it.
;
; 1.168.192.in-addr.arpa hosts
;
1               172800          IN      PTR     orione.ecom.it.
2               172800          IN      PTR     pegaso.ecom.it.
3               172800          IN      PTR     gw.ecom.it.
```

DNS and BIND

Notiamo che la struttura è analoga, anche se più semplice, a quella di **ecom.it**.

La tabella configura ancora una volta i nameservers (primario e secondario) per la zona e definisce i records **PTR** (che mappano indirizzi su nomi) degli indirizzi contenuti nel dominio **1.168.192.in-addr.arpa**.

- Del tutto simile è lo zone file del dominio **2.168.192.in-addr.arpa**. Conterrà a sua volta gli indirizzi della rete **192.168.2** che, nel nostro caso, si sovrappone completamente al dominio **amm.ecom.it** (cosa consigliata, ma non sempre possibile).

DNS and BIND

```
;       file   : 192.168.2
;       author : G. Vitillaro (peppe@unipg.it)
;       date   : May 6, 1999
;       scope  : Inverse name resolution for 2.168.192.in-addr.arpa
;
;
@               172800          IN      SOA     dns.ecom.it.
root.dns.ecom.it. (
                                      1999031501 ; Serial   = yyyymmddxx
                                        86400      ; Refresh  = 1 day
                                        1800       ; Retry    = 30 min
                                        2592000    ; Expire   = 30 days
                                        172800 )   ; Min. TTL = 2 days
;
;
;
; Authoritative name servers for this domain: TTL = 5 days
;
@               432000          IN      NS      dns.ecom.it.
@               432000          IN      NS      dns.devel.ecom.it.
;
; 2.168.192.in-addr.arpa hosts
;
1               172800          IN      PTR     venus.amm.ecom.it.
2               172800          IN      PTR     mars.amm.ecom.it.
3               172800          IN      PTR     gw.amm.ecom.it.
```

- Poichè gli zone files sono tutti amministrati dalla stessa autorità le informazioni contenute nel SOA rimangono praticamente identiche.

DNS and BIND

- Poichè la zona **amm.ecom.it** è amministrata sullo stesso host (**pegaso**) che amministra il dominio **ecom.it** non è strettamente necessario delgare esplicitamente questa zona, ma, per completezza e per facilitare futuri cambiamenti della *policy* potrebbe essere necessario inserire nello zone file il records **NS**

```
;
; Delegations for subdomains: TTL = 5 days
;
; amm
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ann                     432000  IN      NS      dns
```

della delega di autorità.

- La zona **amm.ecom.it** sarà configurata, sempre su **pegaso**, da uno zone file del tutto analogo a quello di **ecom.it**:

DNS and BIND

```
;       file     : amm.ecom.it
;       author   : G. Vitillaro (peppe@unipg.it)
;       date     : May 6, 1999
;       scope    : AUTHORITATIVE DATA FOR: amm.ecom.it
;       comments : Serial yyyymmddxx will hold 100 changes for day
;                : until 4294123199 (32 long integer) 31/12/4294
;
;
@                       172800  IN      SOA     dns root (
                                      1999050601 ; Serial   = yyyymmddxx
                                        86400      ; Refresh  = 1 day
                                        1800       ; Retry    = 30 min
                                        2592000    ; Expire   = 30 days
                                        172800 )   ; Min. TTL = 2 days
;
; Authoritative name servers for this domain: TTL = 5 days
@                       432000  IN      NS      dns.ecom.it.
@                       432000  IN      NS      dns.devel.ecom.it.
;
; Information for RR's in this domain. Fake a
; host, so that mail to this domain, and in particular,
; so that mail to postmaster@this.domain works
; even in the case MX records are not read
;
@                      172800  IN      A       192.168.1.2     ;; pegaso
;
; MX records for domain: must have canonical names because of the above
;
@                       172800  IN      MX      10 pegaso.ecom.it.
;
; make sure "localhost" is know in this domain.
localhost               432000  IN      A       127.0.0.1
loopback-host           432000  IN      CNAME   localhost
loopback                432000  IN      CNAME   localhost
;
; define loghost (equal loopback for now)
loghost                 432000  IN      CNAME   localhost
```

DNS and BIND

```
;
;
; amm.ecom.it hosts
;
venus                   172800  IN      A       192.168.2.1
                        86400   IN      MX      10 mailhost.ecom.it.

mars                    172800  IN      A       192.168.2.2
                        86400   IN      MX      10 mailhost.ecom.it.

gw                      172800  IN      A       192.168.2.3
                        86400   IN      MX      10 mailhost.ecom.it.
```

In questo caso alcune definizioni (come quella dell'host **dns**) sono assenti perchè semplicemente coincidono con le definizioni dello zone file **ecom.it**.

- Il **named** che girerà sull'host **pegaso** svolgerà anche le funzioni di secondary nameserver per le zone **devel.ecom.it** e **3.168.192.in-addr.arpa**.

DNS and BIND

- In questo caso gli zone files **s/devel.ecom.it** e **s/192.168.3** verranno trasferiti automaticamente sull'host **pegaso** dall'host **192.168.3.2** (**halebop**).

  Gli zone files vengono trasferiti dal programma **named-xfer** (BIND 4.9) nel momento in cui il named ritiene che gli zone files siano *scaduti*.

  Ciò può essenzialmente avvenire per tre motivi:

  - gli zone file non sono **mai** stati trasferiti;
  - il tempo di **refresh** è scaduto;
  - il **serial number** della zona è stato incrementato.

  I parametri che configurano lo ***zone transfer*** sono essenzialmente contenuti nel record **SOA** della zona.

DNS and BIND

```
@                       172800  IN      SOA     dns root (
                                      1999050601 ; Serial   = yyyymmddxx
                                        86400      ; Refresh  = 1 day
                                        1800       ; Retry    = 30 min
                                        2592000    ; Expire   = 30 days
                                        172800 )   ; Min. TTL = 2 days
```

- **Refresh**

  è il tempo dopo il quale un nameserver secondario tenterà di aggiornare lo zone file eseguendo uno zone transfer dal nameserver primario;

- **Retry**

  è il tempo dopo il quale il nameserver secondario tenterà di ricontattare il nameserver primario nel caso lo zone transfer sia fallito;

- **Expire**

  è il tempo dopo il quale uno zone file verrà considerato *definitivamente* inaccurato e quindi non più valido.

DNS and BIND

- Il **named** daemon dell'host **halebop** svolgerà le funzioni di nameserver primario per le zone **devel.ecom.it** e **3.168.192.in-addr.arpa**.

  Sarà inoltre configurato come nameserver secondario delle zone:

  - **ecom.it**
  - **amm.ecom.it**
  - **1.168.192.in-addr.arpa**
  - **2.168.192.in-addr.arpa**

  Pur girando, in questo caso, su un'altra architettura della famiglia UNIX, cioè AIX/6000, la sua configurazione ed amministrazione sarà del tutto analoga a quella del nameserver FreeBSD di **pegaso**.

DNS and BIND

```
directory       /usr/local/domain
;
cache           .                       root.cache
;
primary         0.0.127.in-addr.arpa    p/127
;
primary         devel.ecom.it           p/devel.ecom.it
primary         3.168.192.in-addr.arpa  p/192.168.3
;
secondary       ecom.it                 141.250.9.56      s/ecom.it
secondary       1.168.192.in-addr.arpa  141.250.9.56      s/192.168.1
;
secondary       amm.ecom.it             141.250.9.56      s/amm.ecom.it
secondary       2.168.192.in-addr.arpa  141.250.9.56      s/192.168.2
```

- Vediamo apparire per la prima volta la direttiva ***cache***. L'host **halebop** non ha autorità diretta sulla **radice**. Non può quindi essere definito come nameserver primario della root.

  D'altra parte non avrebbe alcun modo per risalire all'indirizzo della macchina che gestisce l'autorità sulla radice della nostra piccola *internet.*

DNS and BIND

- Questo è esattamente lo scopo della direttiva ***cache***: definisce lo zone file (senza autorità) che contiene i record NS ed i record A dei nameservers che hanno autorità sulla radice.

  Nel nostro caso si tratta di una sola macchina (un errore nel disegno della rete?), ma nel caso di **Internet** si tratta di una decina di macchine distribuite sull'intero pianeta.

```
.                       3600000  IN  NS    dns.ecom.it.
dns.ecom.it.            3600000      A     192.168.1.2
```

- Nel caso di Internet lo zone file delle radice, da configurare come cache, viene scaricato, via **ftp**, dalla macchina **rs.internic.net** (**198.41.0.6**): **ftp://domain/named.root** ed ovviamente è più complesso:

DNS and BIND

```
;       This file holds the information on root name servers needed to
;       initialize cache of Internet domain name servers
;       (e.g. reference this file in the "cache  .  <file>"
;       configuration file of BIND domain name servers).
;
;       This file is made available by InterNIC registration services
;       under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.RS.INTERNIC.NET
;       -OR- under Gopher at    RS.INTERNIC.NET
;           under menu          InterNIC Registration Services (NSI)
;              submenu          InterNIC Registration Archives
;           file                named.root
;
;       last update:    Aug 22, 1997
;       related version of root zone:   1997082200
;
;
; formerly NS.INTERNIC.NET
;
.                        3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                        3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.      3600000      A     128.9.0.107
;
; formerly C.PSI.NET
;
.                        3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.      3600000      A     192.33.4.12
;
; formerly TERP.UMD.EDU
;
.                        3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.      3600000      A     128.8.10.90
```

DNS and BIND

```
;
; formerly NS.NASA.GOV
;
.                       3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.     3600000      A     192.203.230.10
;
; formerly NS.ISC.ORG
;
.                       3600000      NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.     3600000      A     192.5.5.241
;
; formerly NS.NIC.DDN.MIL
;
.                       3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.     3600000      A     192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
;
.                       3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.     3600000      A     128.63.2.53
;
; formerly NIC.NORDU.NET
;
.                       3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.     3600000      A     192.36.148.17
;
; temporarily housed at NSI (InterNIC)
;
.                       3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.     3600000      A     198.41.0.10
;
; housed in LINX, operated by RIPE NCC
;
.                       3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.     3600000      A     193.0.14.129
;
```

DNS and BIND

```
; temporarily housed at ISI (IANA)
;
.                       3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.     3600000      A     198.32.64.12
;
; housed in Japan, operated by WIDE
;
.                       3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.     3600000      A     202.12.27.33
; End of File
```

- Per collegare la nostra piccola internet con **la Rete** sarà sufficiente sostituire la direttiva

```
;
primary         .                       root.db
```

con la direttiva

```
;
cache           .                       named.root
```

nel file **named.boot** di **pegaso**. Ovviamente occorrerà sostituire gli indirizzi fittizi con gli indirizzi reali assegnati dal proprio network provider.

DNS and BIND

- Gli zone file del database amministrato da **halebop** sono del tutto analoghi a quelli amministrati su **pegaso**.

- Per gli entrambi gli host il demone **named** verrà lanciato in modo automatico durante la fase di inizializzazione del sistema operativo.

- Dopo aver creato le directory **/usr/local/domain** e sistemato gli zone files nelle subdirectory **p**, occorrerà definire i relativi files di configurazione **/etc/named.boot** e testare la configurazione.

- Il demone **named** risponde ad una serie di segnali che possono essere utili per eseguire il debugging:

DNS and BIND

- **kill -1**

  costringe il named a rileggere il named.boot e gli zone files su cui ha autorità

- **kill -2**

  dump del database

- **kill -30**

  trace mode on

- **kill -31**

  trace mode off

- Inoltre il named emette una serie di messaggi di informazione ed errore sulla facility **daemon** del **syslogd**. Un modo semplice per ottenere in un file tutti i messaggi del named è quello di

DNS and BIND

inserire in **/etc/syslog.conf** le seguenti direttive:

```
daemon.err                                          /local/log/daemon.log
daemon.notice                                       /local/log/daemon.log
daemon.info                                         /local/log/daemon.log
daemon.warn                                         /local/log/daemon.log
```

- Un esempio della partenza del named su **pegaso:**

```
May  7 13:00:46 phoenix named[26369]: starting.  named 4.9.6-REL Tue Oct 21 13:08:
04 GMT 1997     jkh@time.cdrom.com:/usr/obj/usr/src/usr.sbin/named
May  7 13:00:46 phoenix named[26369]: primary zone "" loaded (serial 1999050601)
May  7 13:00:46 phoenix named[26369]: primary zone "0.0.127.in-addr.arpa" loaded (
serial 1999050601)
May  7 13:00:46 phoenix named[26369]: primary zone "ecom.it" loaded (serial 199905
0601)
May  7 13:00:46 phoenix named[26369]: primary zone "1.168.192.in-addr.arpa" loaded
 (serial 1999031501)
May  7 13:00:46 phoenix named[26369]: primary zone "amm.ecom.it" loaded (serial 19
99050601)
May  7 13:00:46 phoenix named[26369]: primary zone "2.168.192.in-addr.arpa" loaded
 (serial 1999031501)
May  7 13:00:46 phoenix named[26370]: Ready to answer queries.
May  7 13:00:46 phoenix named[26370]: Ready to answer queries.
May  7 13:00:46 phoenix named[26370]: secondary zone "3.168.192.in-addr.arpa" load
ed (serial 1999031501)
May  7 13:00:46 phoenix named[26370]: secondary zone "devel.ecom.it" loaded (seria
l 1999050601)
```