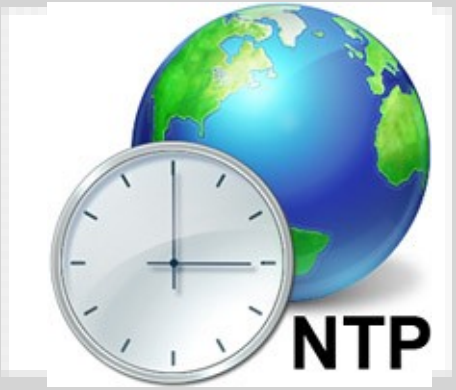
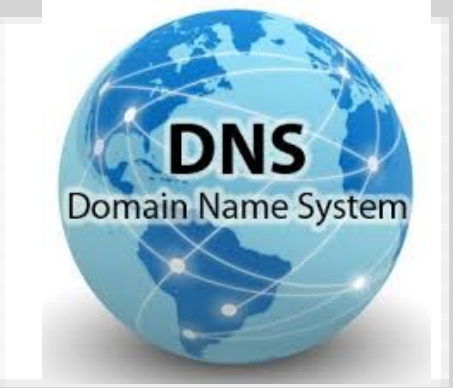
SERVIZI DI RETE

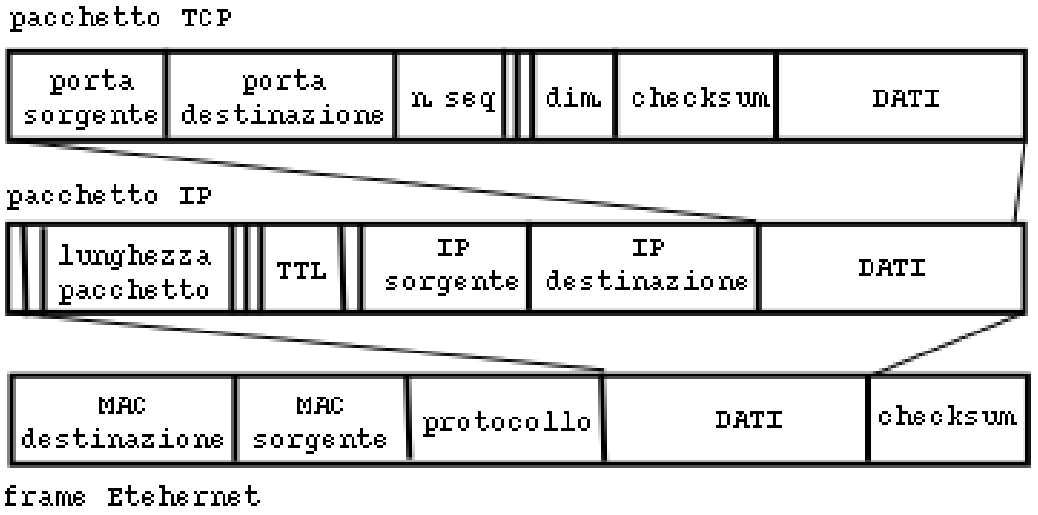




Premessa: Porte & Socket

PORTE

- Ogni processo locale che comunica con uno remoto viene identificato in una connessione TCP/IP tramite una **porta**.
- Una **porta** è rappresentata, all'interno di un pacchetto TCP o UDP, da un campo a 16 bit che può assumere un valore tra 0 e 65535.



PORTE

E' possibile suddivere le **porte** in tre categorie:

- **Well Known Ports**: il cui valore va da **0** a **1023** sono assegnate a specifici protocolli dalla **Internet Assigned Number Authority (IANA)**.
- **Registered Ports**: il cui valore va da **1024** a **49151**, sono registrate a nome delle società che hanno sviluppato specifiche applicazioni.
- **Dynamic and/or Private Ports**: il cui valore va da **49152** a **65535**, non sono gestite da nessun organo di controllo, e vengono assegnate dinamicamente, dal sistema operativo, quando un client si connette ad un host remoto.

SOCKET

- **La combinazione tra indirizzo IP, protocollo di trasporto e numero di porta prende il nome di Socket**.
- Le condizioni per instaurare una connessione tra due socket TCP/UDP sono due:
  - **apertura passiva lato server**, la quale indica al sistema operativo su quale porta vengono accettate le connessioni;
  - **apertura attiva lato client**, che richiede al sistema operativo l'assegnamento di una porta per connettersi all'host remoto;

WELL KNOWN PORTS TCP

- Sebbene generalmente un'applicazione utilizzi solamente un protocollo tra TCP e UDP, vi sono dei casi, come per il servizio DNS, in cui vengono utilizzati entrambi i protocolli. In quest'ultimo caso si avrà il medesimo numero di sia per quanto riguarda TCP che per quanto riguarda UDP. Di seguito alcune tra le **Well Known Port** (porte note) più comuni:
- **Porte TCP**
  - 7 ECHO - Servizio Echo;
  - **20** FTP DATA - File Transfer Protocol Dati;
  - **21** FTP - File Transfer Protocol Controllo;
  - **22** SSH - Secure Shell Remote Login Protocol
  - 23 TELNET - Telnet Protocol;
  - **25** SMTP - Simple Mail Transfer Protocol;
  - **53** DNS - Server dei nomi di dominio;
  - 67 BOOTPS - (Dhcp) Bootstrap Protocol Server;
  - 68 BOOTPC - (Dhcp) Bootstrap Protocol Client;

WELL KNOWN PORTS TCP

- Continua…
  - **80** HTTP - Hypertext Transmission Protocol;
  - **110** POP3 - Post Office Protocol 3;
  - 111 SUNRPC - Sun RPC Portmap;
  - 113 AUTH - Servizio autenticazione;
  - 119 NNTP - Network News Transfer Protocol;
  - 137 NETBIOS-NS - NETBIOS Name Service
  - 138 NETBIOS-DGM - NETBIOS Datagram Service
  - 139 NETBIOS-SSN - NETBIOS Session Service
  - **143** IMAP - Internet Mail Access Protocol;
  - **389** LDAP - Lightweight Directory Access Protocol;
  - **443** HTTPS - http protocol over TLS/SSL;
  - 515 PRINTER - Spooler;

WELL KNOWN PORTS UDP

Porte UDP

- 7 ECHO - Servizio Echo;
- **53** DNS - Server dei nomi di dominio;
- 67 BOOTPS - (Dhcp) Bootstrap Protocol Server;
- 68 BOOTPC - (Dhcp) Bootstrap Protocol Client;
- 69 TFTP - Trivial File Transfer Protocol;
- 111 SUNRPC - Sun RPC Portmap;
- **123** NTP- Network Time Protocol;
- 137 NETBIOS-NS - NETBIOS Name Service;
- 138 NETBIOS-DGM - NETBIOS Datagram Service;
- 139 NETBIOS-SSN - NETBIOS Session Service;
- 161 SNMP - Simple Network Management Protocol (SNMP);
- 162 SNMP - TRAP Simple Network Management Protocol Trap;
- **515** PRINTER - Spooler;

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

void error(const char *m){

  perror(m);

  exit(0);

}



int main(void) {

const char *s_addr = "157.138.7.88";

const char i_port = 80;

int s;

struct sockaddr_in server_addr;

char *message, server_reply[10000];

printf("Create socket\n");

s = socket(AF_INET, SOCK_STREAM, 0);

if (s < 0) { error("socket()"); }

memset(&server_addr, 0, sizeof(server_addr));

server_addr.sin_family = AF_INET;

server_addr.sin_addr.s_addr = inet_addr(s_addr);

server_addr.sin_port = htons(i_port);

printf("Trying connect\n");
```

```c
if (connect(s, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {

        error("connect()");

}

printf("Connect success!\n");

/* Send some data */

message = "GET /?st=1 HTTP/1.1\r\nHost: www.unive.it\r\n\r\n";

if( send(s , message , strlen(message) , 0) < 0) error("Send failed");

printf("Data Send\n");

/* Receive a reply from the server */

if( recv(s, server_reply , 10000 , 0) < 0)error("recv failed");



printf("Reply received\n");

printf("%s",server_reply);



close(s);

return 0;

}
```

Servizi di rete

Avvio....ancora!?!?

- Tradizionalmente, i sistemi UNIX, dopo l'avvio del kernel, lanciano il processo **init**, che:
  - si occupa di gestire tutte le successive fasi di avvio come:
    - il montaggio di file system aggiuntivi,
    - l'avvio di programmi per fornire servizi.
  - si occupa infine di lanciare i gestori dei login da terminale.

- Esistono molte varianti di questo programma, ma quella tradizionalmente più usata nei sistemi Linux prevede che il sistema passi attraverso una serie di stati (o **runlevel**), all'inizio e al termine dei quali vengono eseguiti degli script, residenti in `/etc/init.d`) che si occupano di lanciare o uccidere determinati processi.

Avvio....ancora!?!?

- Recentemente il sistema "***init+runlevel***" è rimpiazzato da altri, tipicamente basati su eventi, a seconda della distribuzione (ad es. **systemd**).
- Tuttavia al momento il consenso sembra essere quello di adottare il sistema **systemd**.
- Tutti i vari sistemi descritti sopra si basano, per quanto riguarda l'avvio e lo spegnimento manuale di servizi, sul comando **`service`** **(e** **`systemctl`**).

```
# service nfs-kernel-server restart
# systemctl restart nfs-kernel-server
```

Operazioni periodiche

- Tra i servizi lanciati dal sistema in avvio vi può essere un gestore delle operazioni periodiche, che negli Unix prende il nome di **Cron**.
- **Cron permette di lanciare periodicamente dei comandi**, specificando anche l'utente con cui debbano essere eseguiti. La configurazione di **cron** è suddivisa tra il file `/etc/crontab` e i file presenti nelle directory`/etc/cron.*/`

  Digitate da terminale: `# cat /etc/crontab`

- In Ubuntu/Debian vi sono tre directory in contententi gli script che devono essere eseguiti periodicamente dal sistema:
  - *`/etc/cron.daily/`: esecuzione giornaliera.*
  - *`/etc/cron.weekly/`: esecuzione settimanale.*
  - *`/etc/cron.monthly/`: esecuzione mensile.*

Operazioni periodiche

- Vi è poi una quarta directory, `/etc/cron.d`, che contiene script che vengono eseguiti arbitrariamente (a discrezione di chi li ha creati) o che si "schedulano da soli", come ad esempio `rsnapshot`.
- **Ogni utente del sistema ha la sua crontab**, editabile con il comando `crontab -e`, che segue la sintassi di `/etc/crontab`.
- Per operazioni programmate una tantum si può invece usare il comando `at` (`batch`, `atq`, `atrm`..)

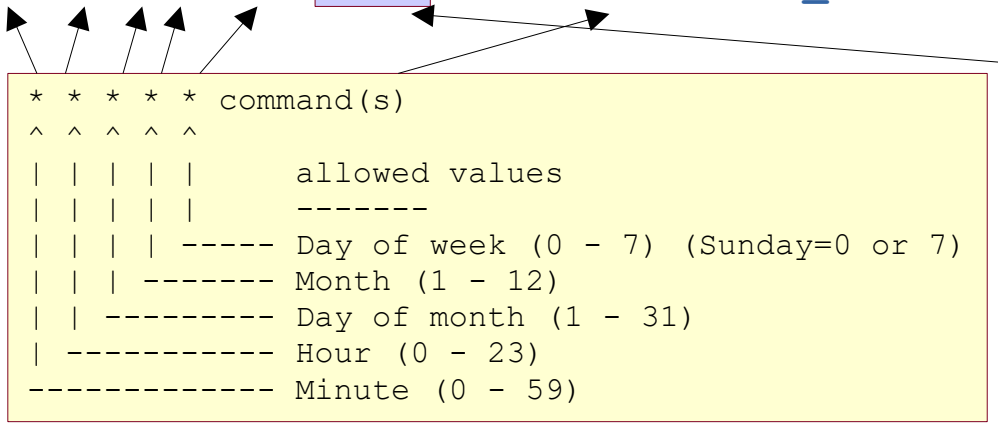
```
# at 6 am sep 9
```

/etc/crontab - esempio

```
# commenti e altre cose varie che ho tagliato per restare nei limiti della slide.
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
                        /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
                        /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
                        /etc/cron.monthly )
#
# Procedura di shutdown automatico laboratori

35 19 * * 0-5 root /root/sbin/mess_popup /root/sbin/messaggio_chiusura_15min
35 19 * * 0-5 root /sbin/shutdown -h 19:50
45 19 * * 0-5 root /root/sbin/mess_popup /root/sbin/messaggio_chiusura_5min
```



Utente che esegue lo script. Va specificato solo in `/etc/crontab` perchè di sistema.

Servizi di base - `inetd`

- Nei sistemi UNIX, il demone **`inetd`** **è spesso il modo più semplice di mettere in opera un servizio**. Questo "*superserver*", infatti, resta in ascolto sulle porte configurate, e, qualora riceva una richiesta (di connessione nel caso di protocolli *stream-oriented*, come **TCP**, o un semplice pacchetto nel caso di protocolli *datagram-oriented*, come **UDP**), si preoccupa di lanciare il programma desiderato, nonché di reindirizzare standard input, output ed error dello stesso sulla rete.
- Vi sono diverse implementazioni di questo demone, da quelle più tradizionali (ad es. nel pacchetto Debian/Ubuntu **`openbsd-inetd`**) a quelle che aggiungono funzionalità, ma che presentano modalità di configurazione più complesse (ad es. nel pacchetto **`xinetd`**). Nel primo caso la configurazione risiede nel file `/etc/inetd.conf`.

Servizi di base - inetd/tcpd

- Spesso abbinato all'uso di **inetd** vi è quello del tcp wrapper **tcpd**, (`apt install tcpd`) che si occupa di registrare le richieste in entrata e di accettarle o rifiutarle in base ad una serie di semplici regole, definite nei file `/etc/hosts.allow` e `/etc/hosts.deny`.
- Per approfondire:
- `http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-it-4/s1-tcpwrappers-access.html`

Servizi di base - `inetd/tcpd`

- Esempio (blocchiamo tutto tranne il servizio `ftp`) `hosts.deny`:

```
# /etc/hosts.deny: list of hosts that are _not_
# allowed to access the system.

ALL EXCEPT ftpd: 192.168.0.
```

- Esempio (accettiamo `ssh`) di `hosts.allow`:

```
# /etc/hosts.allow: list of hosts that are allowed to
# access the system.

sshd: 157.138.22.5 gundam.dsi.unive.it
```

- NB: Di solito è preferibile bloccare tutto in `hosts.deny`, ad esempio usando la clausola `ALL: ALL` e aprire i servizi che ci interessano in `hosts.allow`.

Servizi di base - `inetd/tcpd`

- Quindi la situazione precedente diventa:
  - Esempio di `hosts.deny`:

```
# /etc/hosts.deny: list of hosts that are _not_
# allowed to access the system.

ALL: ALL
```

  - Esempio di `hosts.allow`:

```
# /etc/hosts.allow: list of hosts that are allowed to
# access the system.

sshd: 157.138.22.5 gundam.dsi.unive.it

ftpd: 192.168.0.
```

.

Servizi di base – installazione `inetd`

- Tra i servizi di base possiamo elencare `ftp, tftp, telnet, rsh`.
- In Debian/Ubuntu per installare il pacchetto basta lanciare il comando:

```
# apt install openbsd-inetd tcpd
```

- Il pacchetto è poi configurabile editando il file `/etc/inetd.conf`

```
# nano /etc/inetd.conf
```

  - Provate ad attivare i servizi `daytime` ed `echo`:

```
daytime stream tcp nowait root internal
echo stream tcp nowait root internal
```

  - Dopo di che li testate con:

```
# service inetd restart
# telnet localhost daytime
# telnet <ipvostravm> echo
```

Sincronizziamo le lancette!

- All’interno di un **sistema distribuito** è **fondamentale** che client e server abbiano tutti la stessa ora (fuso orario?).

- Ciò è essenziale per **autenticazione**, **condivisione**, **sincronizzazione**, **esecuzione di job periodici**, ecc.

- Il protocollo che ci **permette di mantenere data e ora sincronizzati** si chiama **NTP (Network Time Protocol).**

NTP

- **NTP** permette ai client di sincronizzarsi con dei server, chiamati **Time Server**, che possono essere collegati a:
  - un dispositivo di misurazione del tempo, quale un orologio atomico o un ponte radio,
  - una stazione di misurazione,
  - oppure possono ottenere le informazioni temporali da un altro server **NTP**.

- I dispositivi di misurazione diretta del tempo (ad esempio un *orologio atomico*) vengono detti **strato 0**, mentre i server **NTP** direttamente collegati ad essi vengono detti server allo **strato 1**, o **primari**. I dispositivi che ottengono le informazioni da uno o più server allo strato **n** sono allo strato **n+1**.

NTP – installazione e configurazione

- In debian/ubuntu esiste il pacchetto `ntp`:

```
# apt install ntp
```

- Per la configurazione è sufficiente editare il file `/etc/ntp.conf` inserendo nella direttiva *`server`* il server **ntp** di riferimento :

```
server ntp.dsi.unive.it #nei pc dei laboratori
```

- Nel caso di **ntp.dsi.unive.it**:

```
server algol.unive.it
```

- Esiste anche `chrony` (`https://chrony-project.org/`)

-

chrony

- **chrony** è un'implementazione versatile del Network Time Protocol (NTP). Può sincronizzare l'orologio di sistema con server NTP, orologi di riferimento (ad esempio ricevitore GPS) e input manuale tramite orologio da polso e tastiera. Può anche funzionare come server NTPv4 (RFC 5905) e peer per fornire un servizio orario ad altri computer nella rete.
- È progettato per funzionare bene in un'ampia gamma di condizioni, tra cui connessioni di rete intermittenti, reti fortemente congestionate, temperature variabili (i normali orologi dei computer sono sensibili alla temperatura) e sistemi che non funzionano in modo continuo o funzionano su una macchina virtuale.
- La precisione tipica tra due macchine sincronizzate su Internet è di pochi millisecondi; su una LAN, la precisione è in genere nell'ordine delle decine di microsecondi. Con il timestamp hardware o un orologio di riferimento hardware, potrebbe essere possibile una precisione inferiore al microsecondo.

Altri comandi per la gestione del clock

- **`rdate`**: permette la sincronizzazione con un server che abbia a disposizione il servizio **time**:

```
# rdate -s ntp.dsi.unive.it
```

- **`hwclock`**: gestisce il clock harware del pc.

```
# hwclock -w //scrive l'ora del SO
    // nel bios del computer
```

NB: Sono comandi di amministrazione… quindi funzionano solo come se siete **super user** (**root**) o tramite i comandi `sudo` o `su`!!!

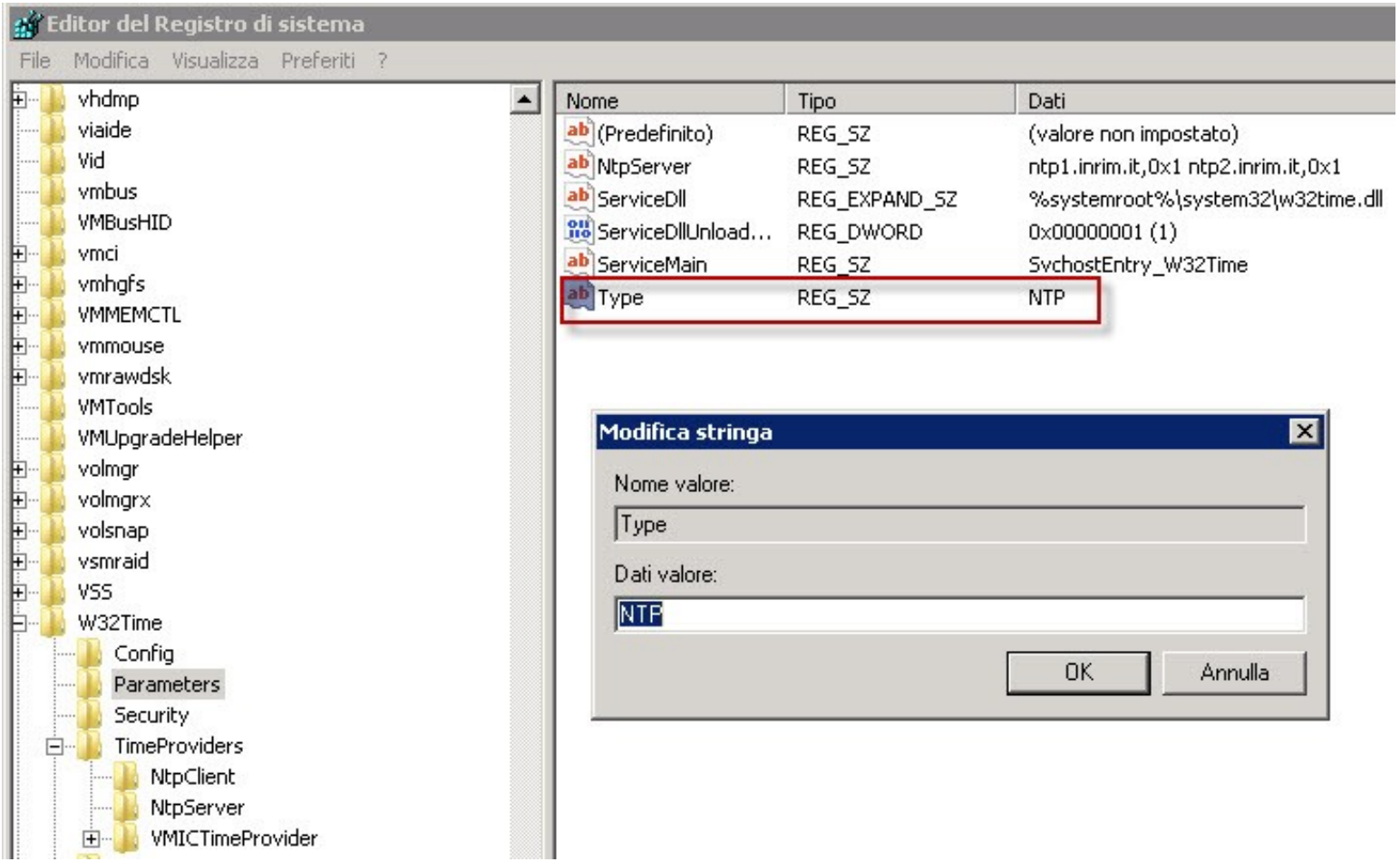
NTP su Windows

- **Windows Server** non dispone in modo automatico di questo servizio, per cui è necessario attivarlo manualmente modificando una chiave di registro tramite l'applicazione `regedit`, seguendo il percorso qui sotto:

```
HKEY_LOCAL_MACHINE\SYSTEM\
CurrentControlSet\services\W32Time\
Config
```

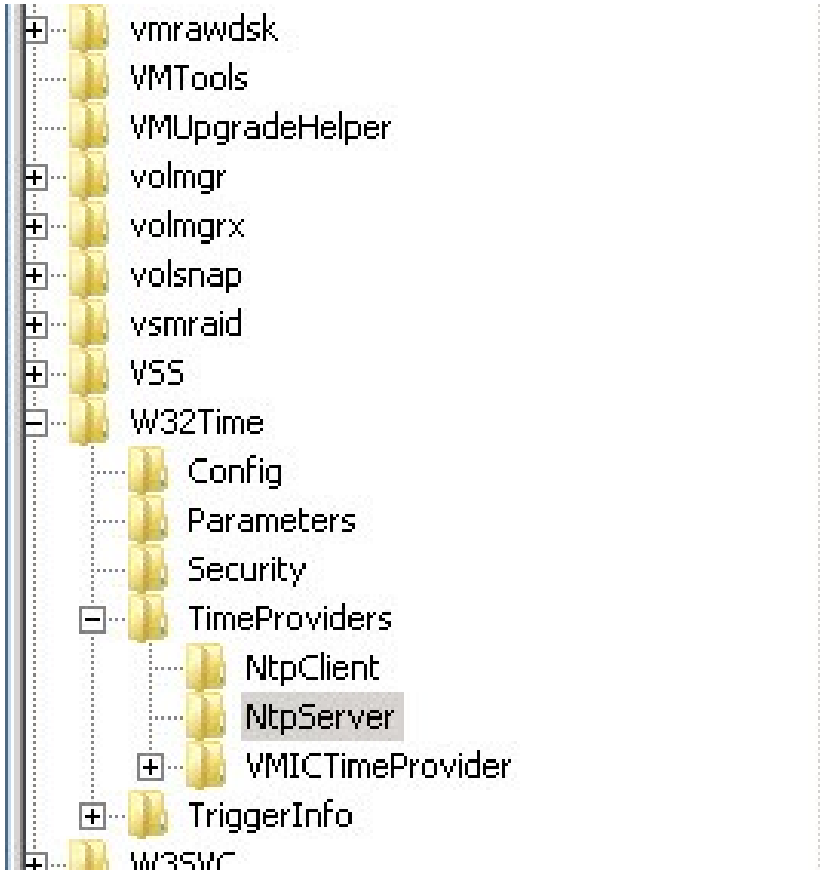
NTP su Windows

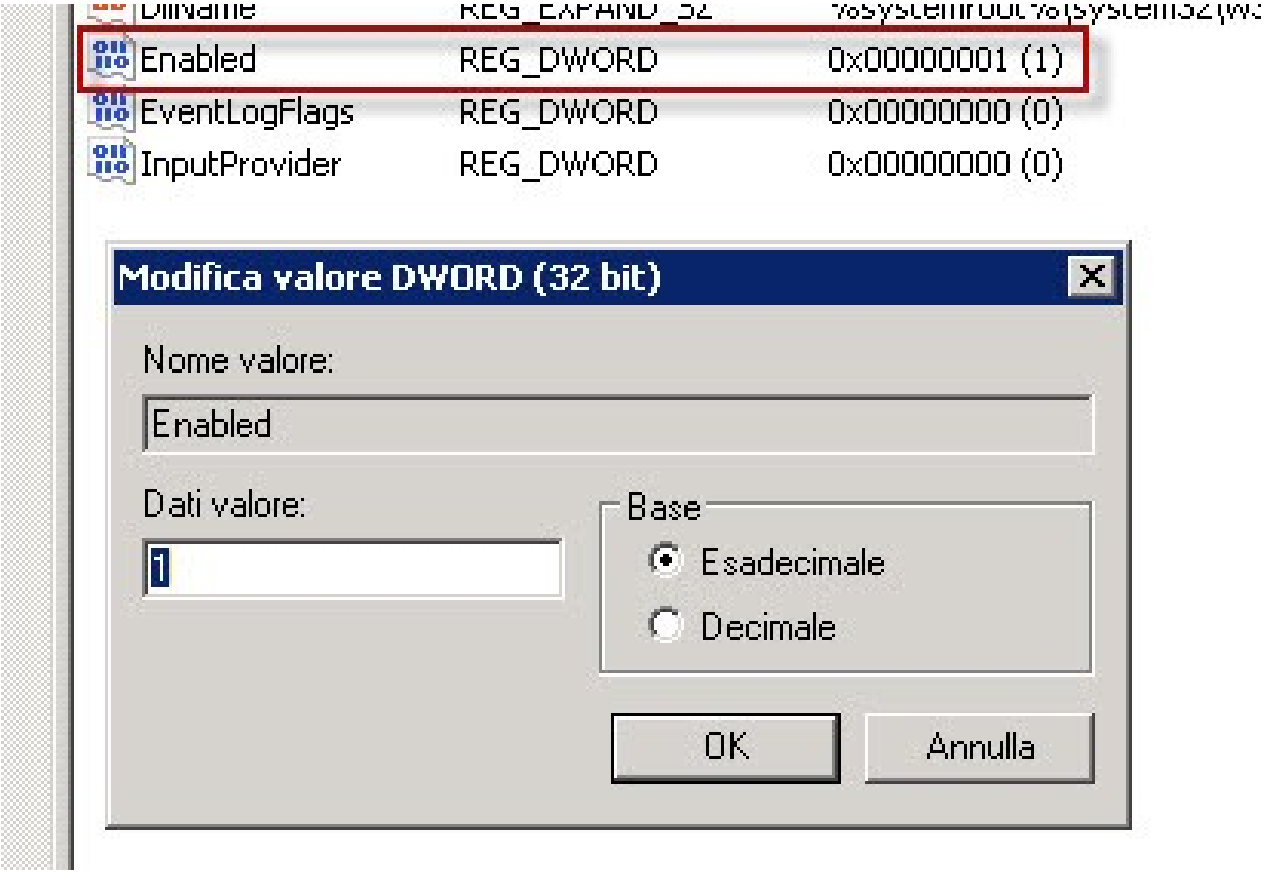
- Dovete andare nella voce `Parameters`, inserire gli `NtpServer` preferiti e specificare **NTP** sulla voce `Type`....



NTP su Windows

- Dopo di che sotto la voce `TimeProviders` cercate `NtpServer` e impostate a **1** la chiave `Enabled`...





NTP su Windows

- Infine, dopo aver aperto il `prompt dei comandi` in modalità **Amministratore** si riavvia il servizio:

  ```
  C:\Windows\System32> net stop w32time && net start w32time
  ```

- OK, ma come si configurano i client?
  Se siete in un **server di dominio**, per segnalare ai client l'esistenza del server **NTP** dovrete utilizzare l'**Editor dei Criteri di Gruppo:** È sostanzialmente un programma che definisce le **policy** (regole) per la gestione di un **Dominio Windows**.
  - NB: Un **Dominio** è un insieme di Workstation, Utenti e risorse che operano all'interno di uno stesso ambiente condividendo risorse gestite attraverso regole e permessi (**policy**).
- In generale (fuori da un dominio) anche per impostare un client Windows all'uso del server NTP bisogna agire sui registri di sistema (del client ovviamente) similmente a quanto visto.

DHCP – Configurazione di rete automatica

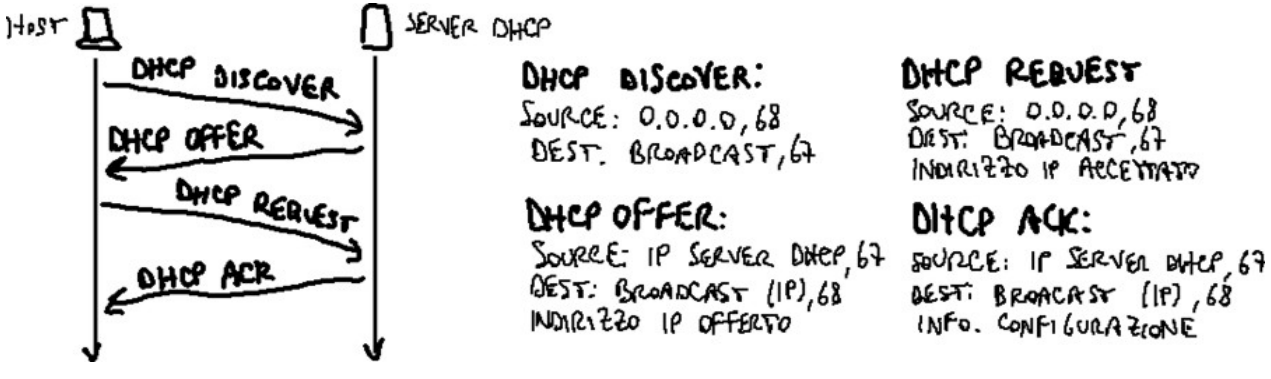
- Nell’ambito di sistemi distribuiti e reti può essere utile, al fine di velocizzare il lavoro dei sistemisti e di rendere meno difficile la vita all’utenza della rete, utilizzare degli strumenti per la configurazione automatica dei parametri di rete: indirizzo ip, netmask, dns e gateway.
- Lo strumento principale per fornire la **configurazione di rete** ai client he ne fanno richiesta è il protocollo **DHCP**.

DHCP – Configurazione di rete automatica

- **DHCP (Dynamic Host Configuration Protocol)**: permette ad un server (detto **server DHCP**), di assegnare dinamicamente **IP, gateway, hostname**, una lista dei **DNS** ed altre eventuali informazioni aggiuntive a degli host che ne fanno richiesta.
- È inoltre possibile fare in modo di **assegnare ad uno stesso host** (o meglio, ad una stessa scheda di rete con il medesimo **MAC** address) **sempre il medesimo IP**.
- Con lo stesso meccanismo si possono assegnare indirizzi IP solo a MAC address conosciuti, come avviene in laboratorio.

DHCP – come funziona

- Semplicemente il client all'avvio della configurazione di rete invia in *broadcast (255.255.255.255)* un pacchetto (**DHCPDISCOVER**) che ricerca un eventuale **server dhcp**.
- Se è presente un **server dhcp** nella rete, esso riceve la richiesta, controlla eventualmente il ***mac address*** della scheda di rete che ha inviato la richiesta, e invia l'indirizzo IP al client (**DHCPOFFER**).
- Se il client accetta la configurazione, invia un pacchetto di *richiesta* (**DHCPREQUEST**) a cui il server risponderà con un pacchetto di *ack* (**DHCPACK**) contente le altre informazioni sulla configurazione di rete.



DHCP - installazione

- L'implementazione di **DHCP** più usata nei sistemi derivati da UNIX è **ISC DHCP.**

- In debian/ubuntu si installa con i comandi:

```
# apt install isc-dhcp-server
# apt install isc-dhcp-client
```

- Per la configurazione del server riferirsi al file in `/etc/dhcp/dhcp.conf`
- Nei client, invece, se in fase di avvio non si ottiene nessuna configurazione ip, o se non si era ancora installato il pacchetto client, è sufficiente usare il comando dhclient:

```
# dhclient -v <networkcard=eth0|eno1|...>
```

DHCP- Esempio: file /etc/dhcp/dhcp.conf

```
option domain-name "dsi.unive.it";

option domain-name-servers oink.dsi.unive.it, ihoh.dsi.unive.it;

default-lease-time 600;

Max-lease-time 7200;


subnet 157.138.22.0 netmask 255.255.255.0 {

  option domain-name-servers     157.138.22.12, 157.138.22.11;

  option netbios-name-servers    157.138.22.12, 157.138.22.11;

  option ntp-servers ntp.dsi.unive.it;

  option routers 157.138.22.1;

  option subnet-mask 255.255.255.0;

  option root-path "/pxelab/";

  next-server 157.138.22.250;

  filename "pxelinux.0";

}
```

Queste righe di configurazione permettono ai client del laboratorio di fare il boot da rete.

DHCP in Windows

- In Windows esiste un **ruolo DHCP** da installare tramite l'applicazione **Server Manager**.
- Una volta installato il **ruolo DHCP**, per la gestione del DHCP Server si fa riferimento ad un'interfaccia semplificata detta **Console DHCP**.
- Tramite **Console DHCP** è possibile specificare:
  - quale rete viene servita dal DHCP Server,
  - qual'è range di indirizzi IP assegnabili,
  - attraverso quale/i connessione/i di rete fornire il servizio.

Domain Name System - DNS

- Ad *ogni indirizzo IP può essere assegnato un nome simbolico.* Ad esempio:

  `157.138.20.97 → gundam.dsi.unive.it`

- Per far questo è necessario gestire in qualche modo **una tabella che indica *la corrispondenza tra nome → ip (diretta) e ip → nome (inversa).***

- Questo è il compito del **Domain Name System** che viene gestito dai **Domain Name (System) Server.**

Domain Name System - DNS

Un **DNS** è sostanzialmente **un database distribuito, basato su modello C/S che traduce il nome di una macchina in un indirizzo internet (ip) e viceversa.**

- Permette di:
  - Amministrare la relazione tra nomi ed indirizzi del proprio dominio in maniera indipendente.
  - Risolvere i nomi esterni al proprio dominio accedendo alle informazioni gestite da altre organizzazioni.

Domain Name System - DNS

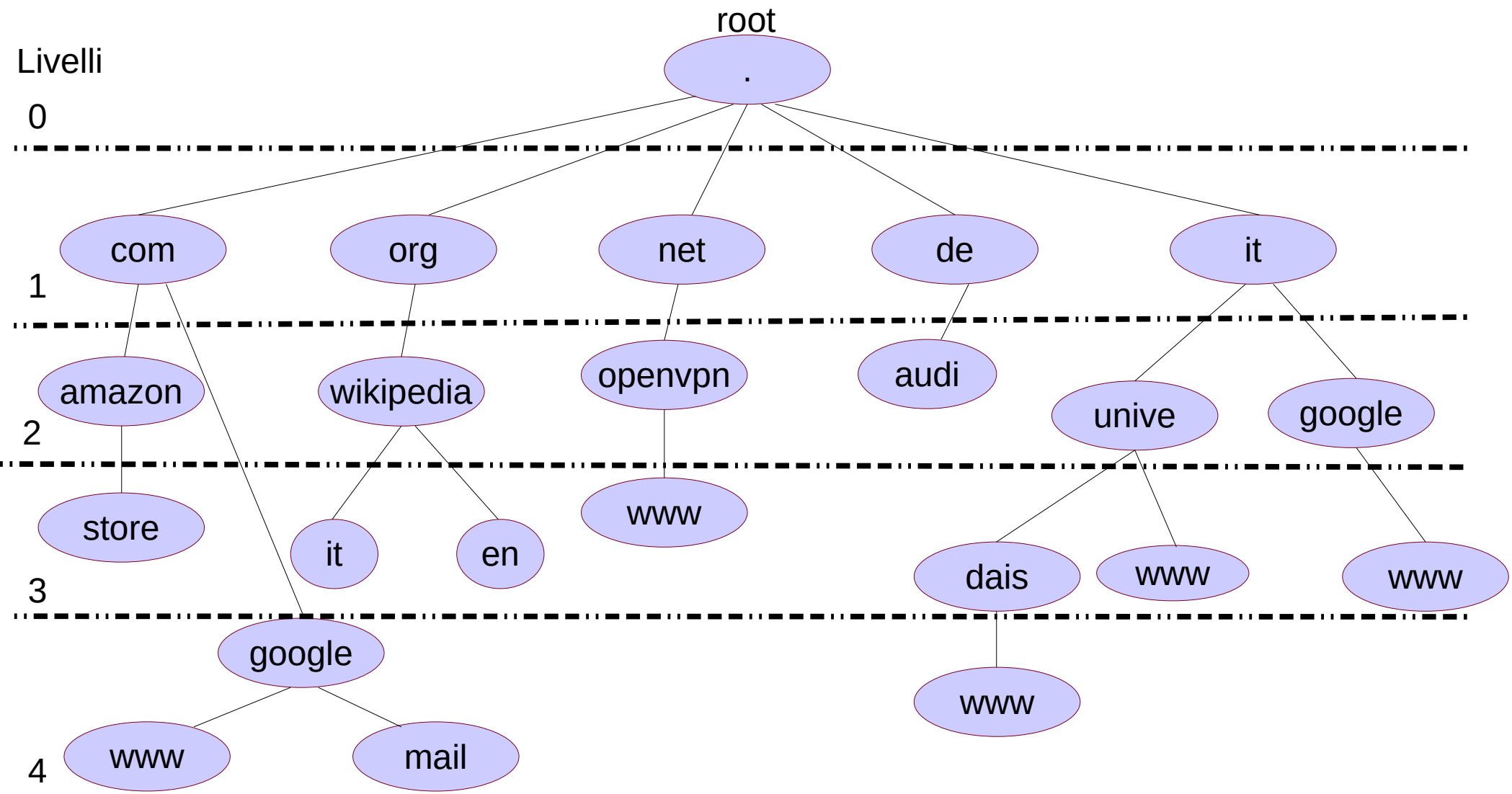
- Possiede una **duplice gerarchia**:
  1) **dei nomi di dominio**.
  2) **dei server che ne forniscono i dati relativi**.

1) Per i nomi di dominio: posizione più alta occupata dai **Top Level Domain**: *com*, *it*, *uk*, *edu,* ecc.
   Al di sotto di essi vi sono altri domini: unive.it, dais.unive.it, google.it, gazzetta.it, amazon.it, ecc..

2) Lato server: le informazioni sono suddivise in **zone**.

- **Una zona non coincide necessariamente con un dominio o un sottodominio e quindi è concettualmente diversa da esso.**
- La **gerarchia** dei name server parte dai **root** name server e si realizza tramite **deleghe** sulle **zone**, aventi come radice la zona "**.**", detta anche **root** o **radice**.

DNS Gerarchia dei Nomi di Dominio

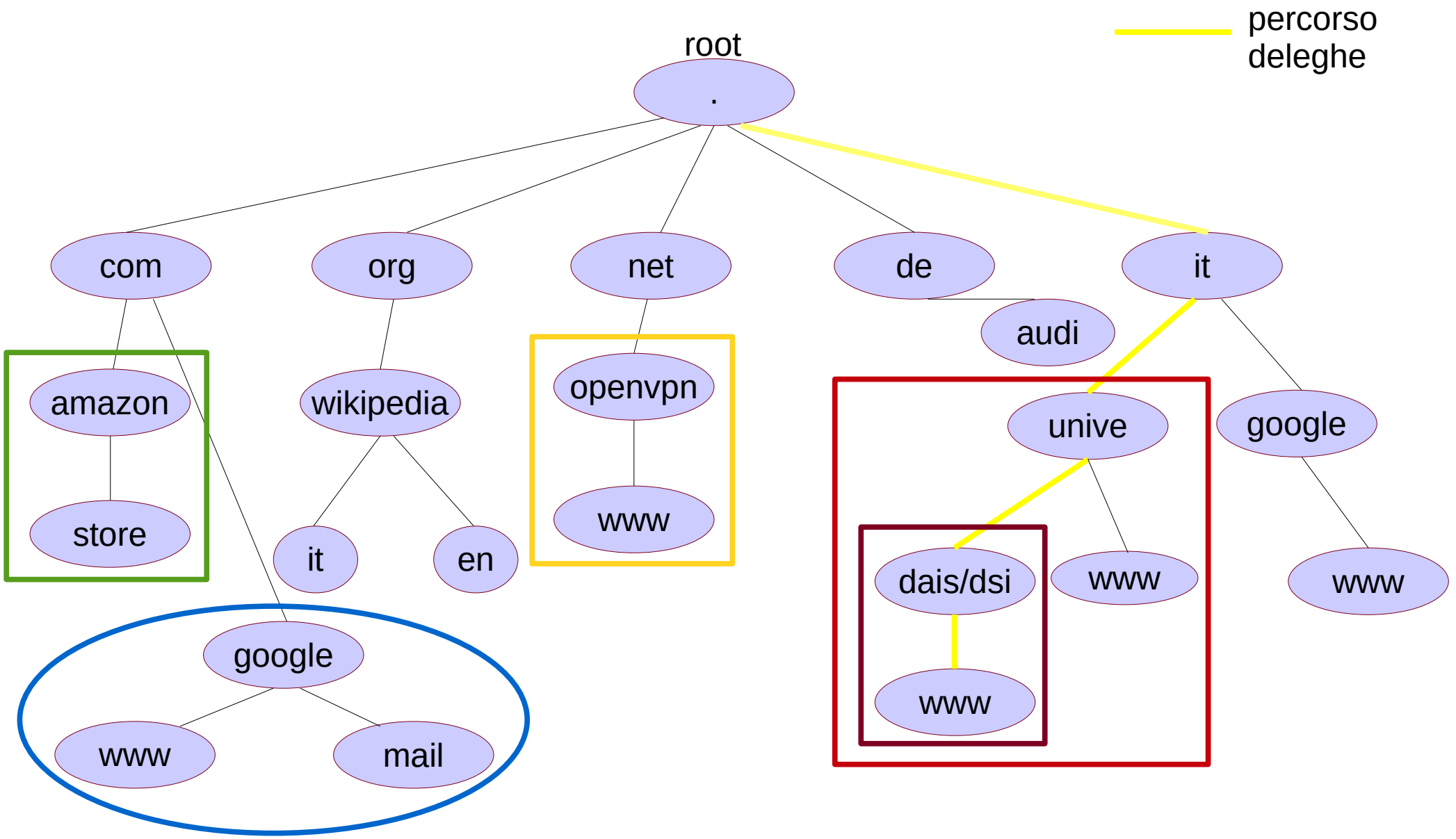


DNS Gerarchia dei server

- Si basa sul concetto di **zona DNS** cioè una parte dello spazio dei nomi, costituita da un dominio e i suoi sottodomini che non sono a loro volta delegati, posta sotto una s*tessa gestione amministrativa* e quindi gestita da uno o più server.
- La gestione di una **zona** è delegata dalla **zona** superiore tramite dei **record** di tipo **NS**. Ad esempio nella zona `.it` vi sarà una delega per la zona `unive.it` ai server DNS che la gestiscono (`157.138.1.8` e `157.138.1.9`). Per la zona `dais.unive.it` e `dsi.unive.it` vi è una delega nella zona `unive.it` per i dns server del DAIS (`157.138.20.12` e `157.138.20.17`).
- Per ridondanza **ogni zona è replicata su più server**, vi sono quindi più record **NS** che specificano quali server siano **autoritativi** per quella zona.
- **Ricorsione**: per ottenere la risoluzione di un nome è necessario partire dalla radice, interrogare uno dei root server nel dominio di primo livello, ottenere il server che lo gestisce, interrogarlo nel dominio di secondo livello e così via fino a raggiungere il server autorevole per il nome desiderato. Questa tecnica è detta "**ricorsione**".

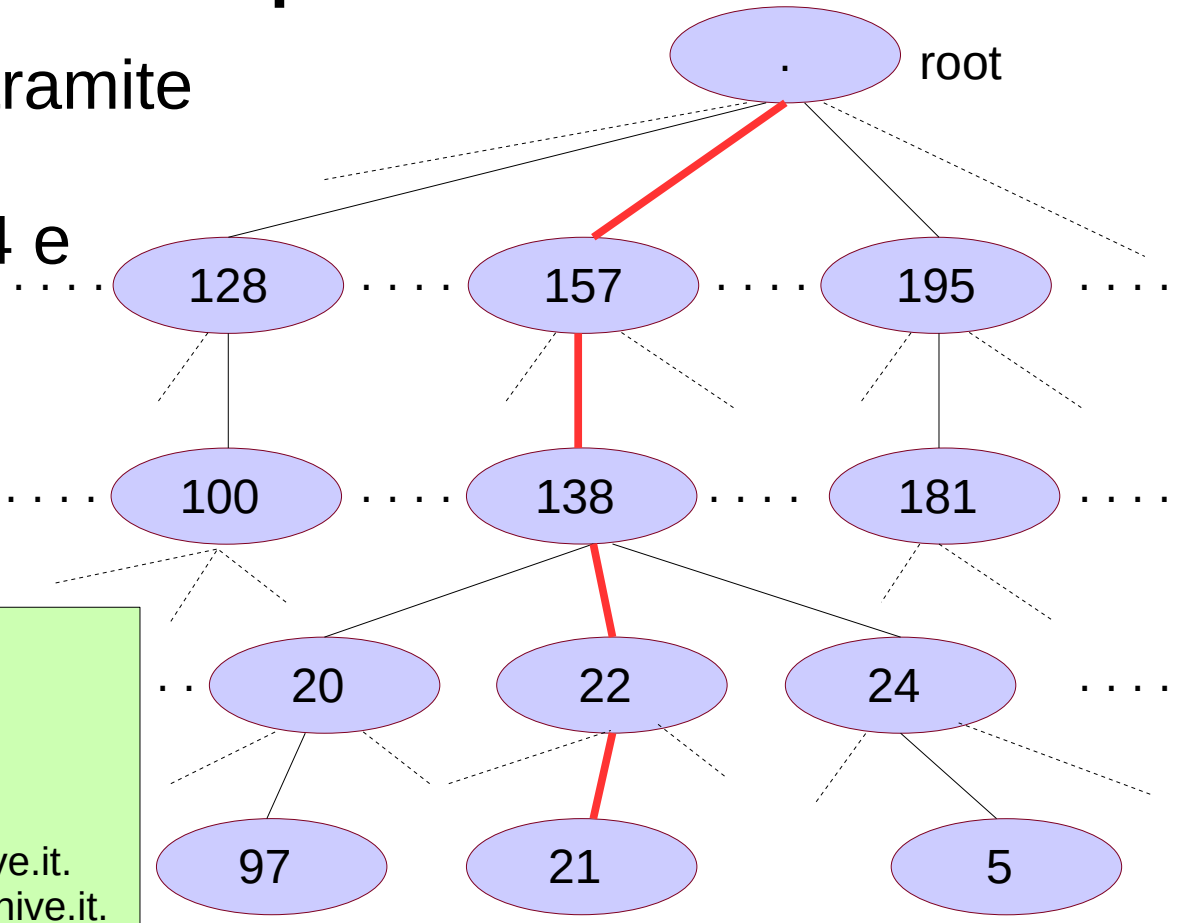

AVRÀ SICURAMENTE UN'ALTRA!
QUAL'È L'IP DI WWW.DAIS.UNIVE.IT...

DNS Gerarchia dei Server



DNS - reverse

- I **DNS server**, oltre alla risoluzione **nomi → ip**, provvedono anche alla risoluzione inversa **ip → nomi**.
- Quest'ultima avviene tramite la **gerarchia di zone** **in-addr.arpa**. per IPv4 e **ip6.arpa.** per IPv6.



- 138.157.in-addr.arpa → unive.it.
- 22.138.157.in-addr.arpa → dsi.unive.it.
- 24.138.157.in-addr.arpa → dais.unive.it.
- **21.22.138.157.in-addr.arpa → lab050101.dsi.unive.it.**
- 5.24.138.157.in.addr-arpa → jenny.dais.unive.it.
- 97.20.138.157.in.addr.arpa →gundam.dsi.unive.it.

DNS - record

- Le **zone** contengono diversi tipi di informazioni, memorizzate in **record**. Ogni **record** contiene solo una certa tipologia di informazioni:
  - **Record SOA**, **Start Of Authority,** fornisce le informazioni generali sulla zona, tra cui un identificativo sequenziale della versione della definizione (**serial number**), l'indirizzo email del responsabile e il tempo di vita (**TTL, time to leave**) della definizione della zona.
    - **serial number:** di solito una sequenza di numeri in cui compare la data del giorno in cui si modifica il file e un progressivo di due cifre nella forma `YYYYMMGGxx`. Ad esempio `2019031101`. Va aggiornato ad ogni modifica effettuata alla configurazione del DNS.
    - **TTL**: indica il tempo, in secondi, per il quale un altro server DNS può ritenere valida l'informazione ricevuta.
  - **Record NS**, che definisce quali siano i DNS responsabili per le richieste iterative riguardanti la zona.
  - **Record MX**, che definisce quali siano gli hostname dei server di posta in entrata responsabili per il dominio.

DNS - record

- .
  - **Record A**, che rappresenta traduzioni nella forma nome → ip. Per **IPV6** il tipo corrispondente è **AAAA**.
  - **Record CNAME**, che rappresenta dei nomi alternativi (alias) per un host, nella forma alias → nome.
  - **Record TXT**, che permette di associare stringhe arbitrarie ad un nome, nella forma nome → stringa.
  - **Record SRV**, che permette di associare delle informazioni strutturate al nome di un servizio, nella forma nomeservizio.protocollo → priorità peso porta nome host.
  - Il **Record PTR** che, nelle zone per la risoluzione inversa, rappresenta le traduzioni nella forma ip → nome.

DNS - Linux

- Nei sistemi Unix/Linux (ma anche in Windows e MacOS) un sistema **DNS** può essere realizzato in due modi a seconda della dimensione della zona da gestire e dei servizi offerti:

  1) Inserendo nel file `/etc/hosts` l'elenco di tutti le coppie ip ↔ nome della propria rete con la sintassi:

```
# IP              FQDN              ALIAS
157.138.20.1      rex.dais.unive.it  rex
```

  2) L'uso di un **server DNS**, in Linux **BIND**.

  - Comandi utili per la gestione: `host` e `dig`.

```
# host lab051001.dsi.unive.it
```

File /etc/hosts - esempio

```
127.0.0.1 localhost

157.138.22.12  broot.dsi.unive.it  broot

157.138.22.11  bau.dsi.unive.it    bau

157.138.22.10  roar.dsi.unive.it   roar

157.138.22.56  lab051001.dsi.unive.it lab051001

# The following lines are desirable for IPv6
capable hosts

::1     ip6-localhost ip6-loopback

fe00::0 ip6-localnet

ff00::0 ip6-mcastprefix

ff02::1 ip6-allnodes

ff02::2 ip6-allrouters
```

DNS Server - categorie

- Nella stessa zona **posso esserci più server DNS**, di solito riconosciuti come:
  - **master:** (primario) contiene la copia principale del dabase dns e propaga agli altri le modfiche,
  - **slave:** (secondario) che possiede una replica,
  - **caching:** che tiene solo una cache delle informazioni.
- In un sistema, di qualsiasi dimensione è sempre preferibile avere almeno un **DNS master** e un **DNS slave** o almeno **caching**. In caso contrario, nel caso di fallimento dell'unico DNS disponibile, il flusso di lavoro potrebbe bloccarsi…..Sarebbe preferibile una replica per avere un backup della configurazione.
- **DNS** è un servizio fondamentale per moltissimi altri servizi, come l'autenticazione, la posta elettronica, e il semplice accesso ad internet!

DNS - Installazione

- Per installare **BIND** su debian/ubuntu:

```
# apt install bind9
```

- I file di configurazione li trovate in `/etc/bind`. In particolare:
  - `named.conf`, `named.conf.*`, che indicano a **BIND** com'è strutturata la configurazione del DNS.
  - vi sono altri file, `db.*` che indicano la configurazione inziale.

BIND – Esempio: `named.conf`

```
// This is the primary configuration file for
the BIND DNS server named.

include "/etc/bind/named.conf.options";

// zone "com" { type delegation-only; };
// zone "net" { type delegation-only; };

include "/etc/bind/named.conf.local";
```

BIND – Esempio: named.conf.options

```
options {
        directory "/var/cache/bind";
        auth-nxdomain no;    # conform to RFC1035
        allow-transfer { 157.138.20.17; 157.138.1.8;
          157.138.1.9; 157.138.1.34; ... 157.138.20.97;};
        allow-recursion { 157.138/16; 10/8; 192.168/16; 127/8;
    };
        check-names master ignore;
        check-names slave ignore;
        check-names response ignore;
        querylog no;
        transfers-out 100;
};
```

BIND – Esempio: named.conf.local

```
// Definzione zone di default
 . . . . . .
// Definizione zone locali

// Reverse mapping per la rete 157.138.22.*
zone "22.138.157.in-addr.arpa" {
     type master;
     file "/etc/bind/etc/hosts.22.db";
};

//altre definzioni di zone
 . . . . . .
// Siamo master per dais.unive.it
zone "dais.unive.it" {
     type master;
     file "/etc/bind/etc/dais.unive.it.db";
     allow-update { 127.0.0.1; 157.138.20.12; };
     allow-transfer { 157.138.20.17; 157.138.1.8; 157.138.1.9;
       157.138.1.34;..; 157.138.22.12.. };
     also-notify { 157.138.20.10; 157.138.20.17; 157.138.1.8; 157.138.1.9; };
};
 . . . . . .
```

BIND – Esempio:
dais.unive.it.db

```
$ORIGIN .
$TTL 604800 ;1 week
dais.unive.it              IN SOA  algol.unive.it. marino.unive.it. (
                                   2024031301 ; serial
                                   1800       ; refresh (30 minutes)
                                   900        ; retry (15 minutes)
                                   1209600    ; expire (2 weeks)
                                   43200      ; minimum (12 hours)
                                   )
                           NS      oink.dais.unive.it.
                           MX      8 oink.dais.unive.it.
coccode                    A       157.138.20.11
www                        CNAME   coccode
oink                       A       157.138.20.12
smtp                       CNAME   oink
imap                       CNAME   oink
. . . . . .
```

BIND – Esempio: `dais.unive.it.db`

- Record speciali per Kerberos+ldap, configurazione diretta, file `dais.unive.it.db`:

```
. . . . . .
_ntp._udp                IN SRV 0 100 123        oink
_kerberos._tcp           IN SRV 0 100 88         will
_kerberos._udp           IN SRV 0 100 88         will
_kerberos-master._tcp    IN SRV 0 100 88         will
_kerberos-master._udp    IN SRV 0 100 88         will
_kpasswd._tcp            IN SRV 0 100 464        will
_kpasswd._udp            IN SRV 0 100 464        will
_ldap._tcp               IN SRV 0 100 389        will
```

BIND Esempio: 157.138.20.db

- Configurazione inversa, file coinvolti: 157.138.20.db:

```
$TTL 604800 ; 1 minute 40 seconds

$ORIGIN .

20.138.157.in-addr.arpa IN SOA   algol.unive.it. marino.unive.it. (

                2024031301 ; serial

                1800       ; refresh (30 minutes)

                900        ; retry (15 minutes)

                1209600    ; expire (2 weeks)

                43200      ; minimum (12 hours)

)

NS oink.dsi.unive.it.

11 PTR   coccode.dsi.unive.it.

12 PTR   oink.dsi.unive.it.
```

$ORIGIN (direttivs RFC 1035)

- `$ORIGIN` definisce un nome di base da cui vengono effettuate le sostituzioni dei nomi "non qualificati" (quelli senza punto finale) durante l'elaborazione del file di zona. Deve terminare con un punto.

```
$ORIGIN example.com.
@  IN NS    ns1.example.com
@  IN NS    ns2.example.com.
@  IN MX  5 mail
```

- La presenza di @ trasforma la definizione come segue:

```
example.com.  IN  NS   ns1.example.com.example.com.
example.com.  IN  NS   ns2.example.com.
example.com.  IN  MX 5 mail.example.com.
```

Errore!!!!

- Definire $ORIGIN come “ . “ evita errori ma non permette l’uso di @ (troveremo un punto al posto di `example.com`) .

```
$ORIGIN .
example.com   IN  NS  ns1.example.com.
example.com.  IN  NS  ns2.example.com
```

```
example.com.  IN  NS  ns1.example.com.
example.com.  IN  NS  ns2.example.com.
```

DNS - Windows

- In Windows vi è un **ruolo DNS** da installare tramite l'applicazione **Server Manager**. La gestione è affidata all'applicazione **DNS** che fornisce un'interfaccia semplicata per la gestione dei record. I file di configurazione generati dall'interfaccia seguono le stesse regole di quelli visti con BIND.
- Oltre al sistema DNS trovate anche:
  - **WINS** (Windows Internet Naming Service).
  - **NetBEUI** (NetBIOS extended User Interface).
  - **NetBIOS** (Network Basic Input Output System): avete presente quando nella rete di casa vostra sul file explorer scrivete \\pc\cartella?
- Comandi utili: `netsh, nslookup`

```
# nslookup 157.138.22.56
```

Abbiamo aggiunto iservizi di rete