Esercitazione 2020-10-29

esercizio 6.3

In questo esercizio abbiamo visto come poter realizzare una chat full duplex TCP in java, usando le socket. Il problema è il seguente: Quando un client invia un messaggio, quel messaggio viene recapitato anche al client che l'ha inviato.

soluzione

Invece di usare una lista usiamo una mappa dove vengono memorizzati la socket ed il printstream. Nel momento in cui viene invocata la add la socket che viene usata per estrarre il printstram viene usata come **chiave**. Il metodo broadcast(), invece, prevede come parametri la socket di colui che invia il messaggio ed il messaggio stesso; quando andiamo ad effettuare il broadcast, e scorriamo il keySet della mappa, non inviamo il messaggio quando la chiave corrisponde alla socket passata come parametro.

Problema comune

Un problema comune a questo tipo di appplicazione, è l'interpretazione inesatta dei caratteri; ad esempio se scrivessimo 'è', questa verrebbe interpretata male.

Possiamo risolvere questo problema forzando tutti i client ad usare un determinato set di caratteri nel seguente modo:

```
inFromServer = new Scanner(socket.getInputStream(), "utf-8");
```

Con 'utf-8' forziamo i client ad usare un determinato charset.

Domain Name System - DNS

Assegnare nomi ad host e servizi

Un problema affrontato da questo sistema è quello di consentire la gestione ed indirizzamento delle macchine, in modo diverso rispetto a quanto previsto dal protocollo ip, che prevede degli indirizzi numerici del tipo "192.168.1.1" (notazione decimale a punti); in questa notazione i singoli byte sono separati da punti.

Il problema della notazione IP è che sono difficili da ricordare, e non rappresentativi del servizio che referenziano. Sarebbe più intelligente usare delle stringhe di caratteri, e quindi l'idea è proprio quella di sostituire gli indirizzi IP con delle stringhe alfanumeriche, aventi una struttura ben precisa.

Queste stringhe, però, sono significative solo a livello umano; infatti le macchine continueranno ad usare la notazione IP. Dobbiamo quindi effettuare un mapping tra le stringhe alfanumeriche e gli indirizzi IP.

Soluzione con file locale

Possiamo pensare di avere un file locale nel quale vengono mantenute le associazioni tra Stringa-IP. Con questa soluzione ogni volta che si aggiunge una macchina in internet bisogna aggiornare tutti i file usati dalle altre macchine.

Questa soluzione può si essere usata, ma non in grande scala (ad esempio possiamo usarla all'interno di un'azienda).

Soluzione con database centralizzato

Possiamo avere un database centralizzato in internet, ovvero una macchina che hosta il file citato prima. Sebbene dal punto di vista della manutenzione può essere una soluzione interessante, non lo è dal punto di vista delle prestazioni; questo perchè l'unica macchina sarebbe contattata da **tutte** le altre macchine che hanno bisogno di tradurre un hostname.

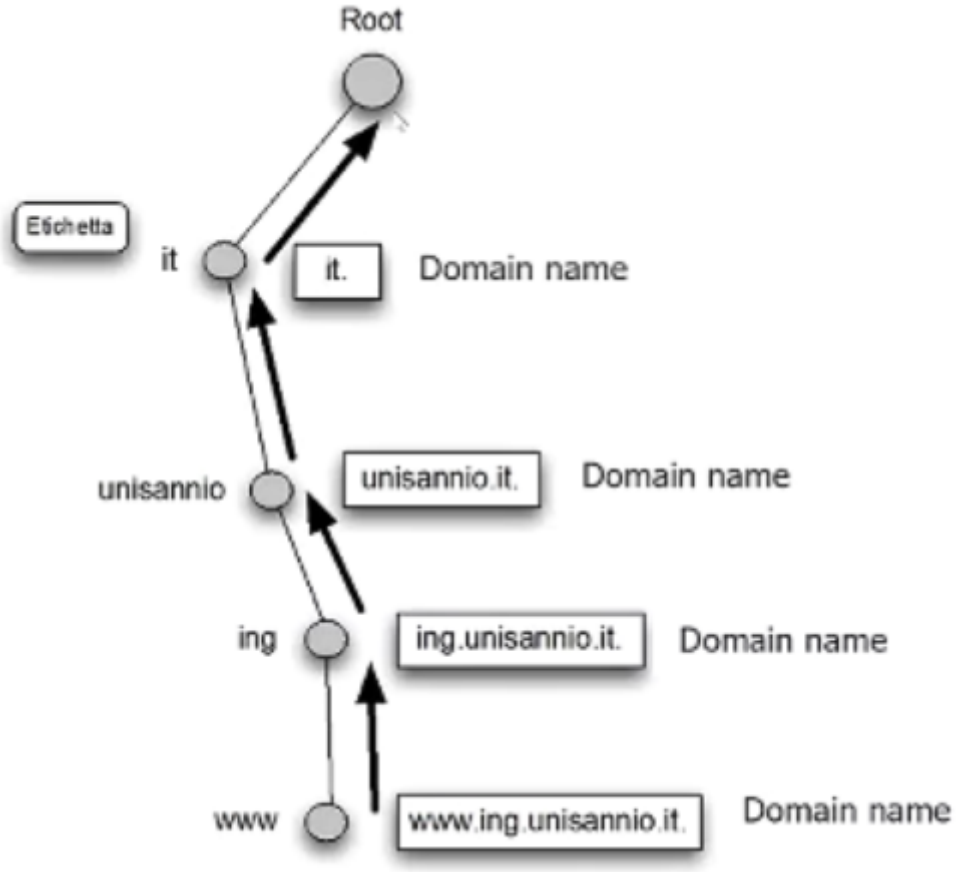
Soluzione adottata - Database distribuito (DNS)

Il sistema DNS è un sistema di database distribuiti; ognuno di questi db conserva una parte delle associazioni tra indirizzi alfanumerici ed indirizzi IP. Questi DB sono hostati da particolari server, chiamati DNS server, che sono in collegamento tra loro secondo dei link che discendono dalla struttura dei nomi che caratterizza il DNS.

Un aspetto importante del DNS è quindi la **struttura dei nomi**: non sono di tipo **flat**, ovvero le stringhe alfanumeriche non sono stringhe semplici, ma sono organizzate secondo una logica tale da dar vita ad una strutturazione di tipo gerarchico.
Un esempio di stringa alfanumerico è `www.unisannio.it`.

Spazio dei nomi DNS

Lo spazio dei nomi previsti dal servizio DNS è di tipo **gerarchico**, quindi possiamo pensare a questo spazio come un albero:



Il numero di livelli disponibile tra la root e le foglie è **127**; ognuno di questi nodi è caratterizzato da un'etichetta, tranne il nodo **root**. Man mano che andiamo verso il basso, fino alla foglia, incontriamo delle etichette; se leggiamo le etichette dalla foglia alla radice, queste danno vita all'indirizzo alfanumerico.

Top Level Domain - TLD

Sono le etichette assegnate ai cosiddetti **domini di livello top**: sono i domini di livello più alto, alcuni esempi sono: **com, edu, gov, mil, net, ecc. ICANN** è una corporazione che ha il compito di gestire i vari domini.

Domini e gestione dei nomi

Il fatto che i domini vengano gestiti da una corporazione, fa si che non ci siano dei **duplicati**, quindi non ci saranno mai due domini con la stessa etichetta. Questo però non nega che un'altra organizzazioni non utilizzi l'etichetta ".unisannio.", ma di certo non avremo due IP che sono referenziati dall'intera sequenza di dominii "www.ing.unisannio.it".

Ad esempio, non è necessario chiedere il consenso da un'ente esterna da "un dominio in poi"; infatti, da unisannio. in poi (da unisannio.it verso il basso), la responsabilità è dell'organizzazione che gestisce il dominio unisannio.it

Questo meccanismo permette la decentralizzazione dei nomi.

Domain names - nomi di dominio

Fully qualified Domain Names (FQDN)

Questo tipo di nome di dominio è del tipo "www.ding.unisannio.it.", ed è una sequenza di labels che termina con una stringa vuota.
Importante: una rappresentazione corretta di un FQDN viene aggiunto anche il punto finale (che rappresenta la root).

Partially qualified Domain Names (PQDN)

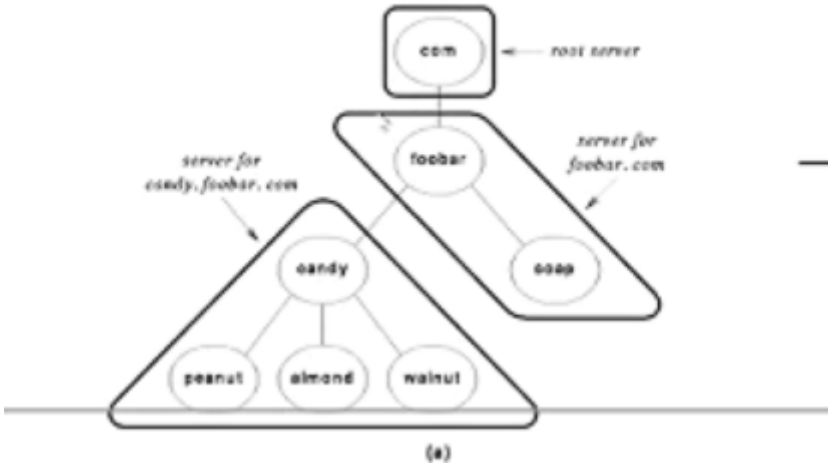
Sono nomi di dominio non completi, che iniziano da un nodo ma non finiscono con la root: "www.ding" questo tipo di nome di dominio non ci permette di arrivare ad una macchina specifica (server che hosta il sito).

DNS e network fisiche

Lo spazio dei nomi non ha nessuna relazione geografica con le macchine che invece gestiscono i nomi; **ad esempio:** il dominio www.google.com individua un'organizzazione (google), ed è accessibile da (quasi) tutto il mondo. Il fatto che si usi il dominio .com, non indica che i server che hostano il servizio sono negli Stati Uniti, ma sono distribuite su tutto il pianeta.

I domini DNS sono quindi **strutture logiche**.

Rappresentazione gerarchica dei server DNS

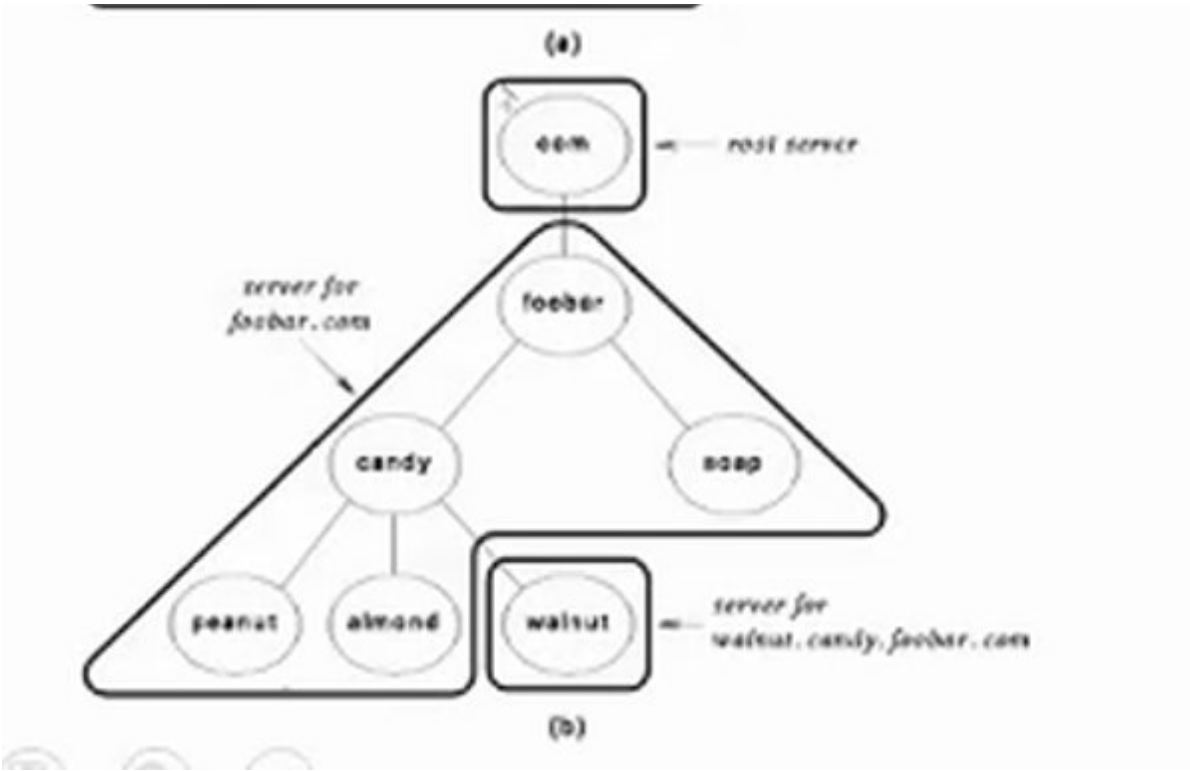


> mi scuso per la risoluzione

Sullo spazio dei nomi possiamo prevedere diversi mapping di server; le forme caratterizzate da una linea scura, rappresentano i server: un server ospita una parte dell'albero, e con la rappresentazione riportata possiamo velocemente dedurre quali sono i nodi di competenza di uno specifico server.

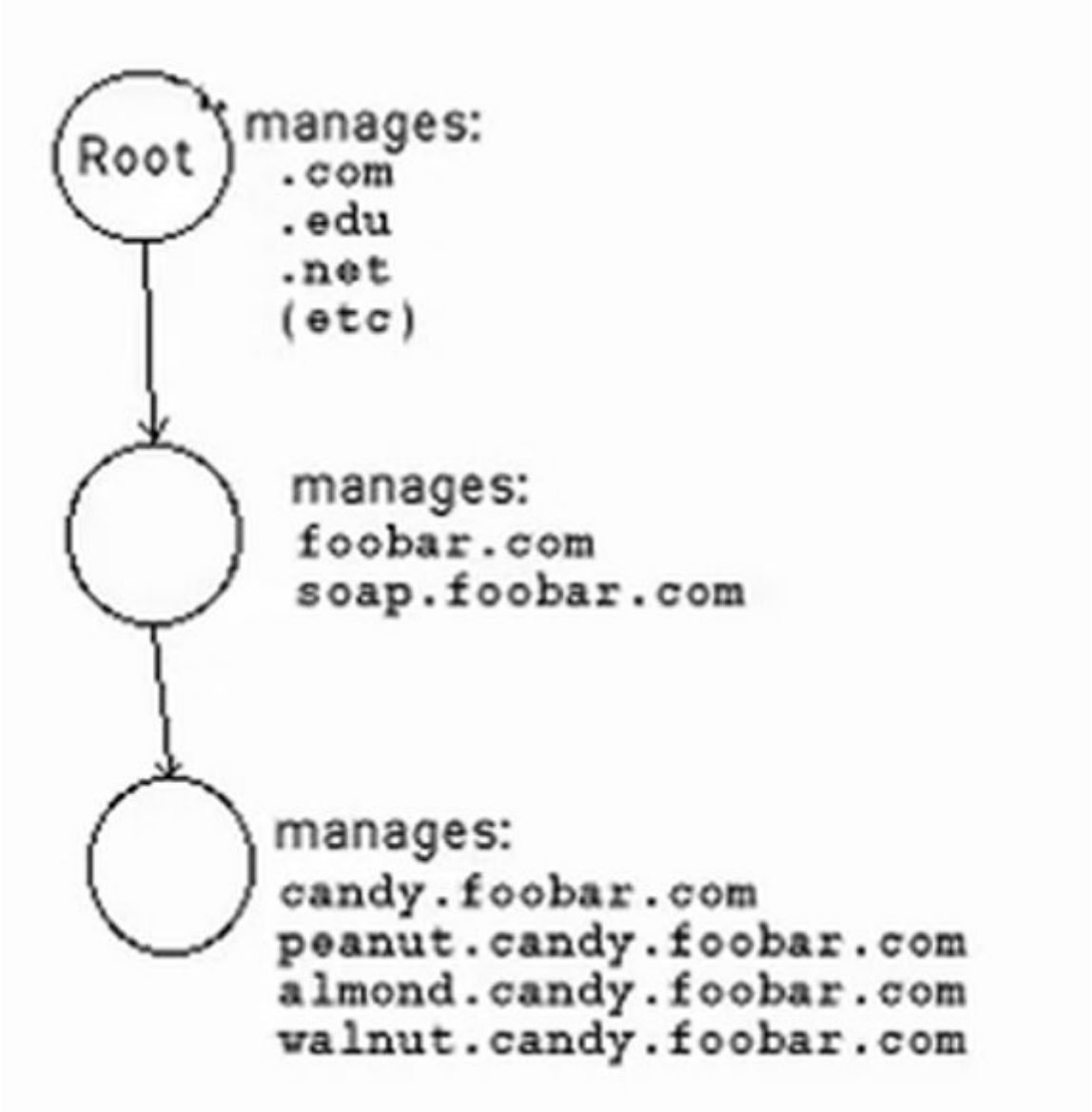
🏁 1:03

Nella figura riportata abbiamo 3 server, così come sono presenti 3 server in questa figura:



La differenza tra queste due figuri è la differenza tra nodi dell'albero dei nomi, e server; abbiamo lo stesso spazio dei nomi, con lo stesso numero di server, ma con responsabilità diverse per ogni server.

Se consideriamo solo i nodi esterni (forme caratterizzate da linee scure) ed ignoriamo i nodi interni (ad esempio com , foobar, ecc), otteniamo un altro albero:



> albero ottenuto

Abbiamo quindi 3 server che gestiscono lo spazio dei nodi, e sono collegati tra loro; il collegamento deriva dal collegamento dal sottostante collegamento nello spazio dei nodi. Il collegamento, **all'atto pratico**, rappresenta il fatto che un server può dialogare con un altro server.

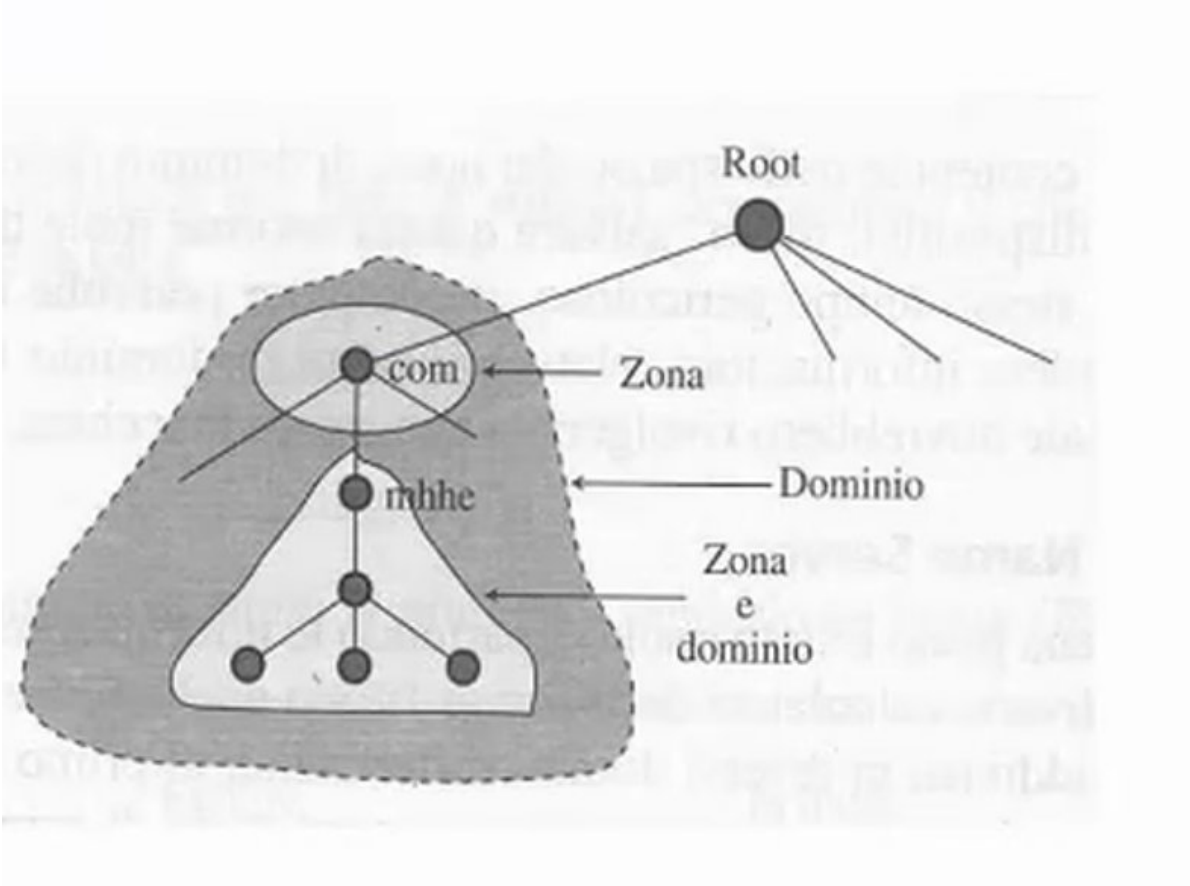
Cosa significa che un server può dialogare con un altro server? Un server può contattarne un altro nel momento in cui ne conosce **l'indirizzo di trasporto**. La comunicazione potrà avvenire, ad esempio, usando uno specifico protocollo di trasporto. Il protocollo applicativo è il DNS, mentre il prtocollo di trasporto usato per inviare messaggi del DNS è UDP.
Un server DNS può comunicare con un altro server DNS usando le socket mediante l'indirizzo di trasporto dell'altro server.

Il **Server Radice** ospita non solo .com, ma **tutti i TLD** del DNS. Questo server radice non è ovviamente in singola istanza nel mondo, ma ha diverse istanze.

Domini e zone

la zona è la competenza diretta di un name server, ed all'interno della zona sono presenti le macchine di cui un name server è responsabile. La zona è caratterizzata da un **file** che racchiude una **partizione del database** introdotto in precedente. Quindi il DB viene partizionato, ed una di queste partizioni è presente all'interno di un file di zona che è hostato da uno specifico server.

In questo file di zona troviamo tutte le macchine di cui quel name server è direttamente responsabile; troviamo quindi l'elenco delle macchine di cui quel name server gestisce l'associazione tra FQDN ed Indirizzo IP.



Ruoli Name Server

I name server in internet sono etichettati a seconda del ruolo che ricoprono durante un'operazione di risoluzione. Sono classificati in:

- **Server root** - Gestiscono i TLD
- **Server locali**
- **Server autorevoli** - sono i server responsabili delle risorse di cui vogliamo la risoluzione, ovvero di cui vogliamo conoscere l'indirizzo IP a partire dall'FQDN; ad esempio il name server è responsabile per la macchina www.unisannio.it, perchè presenta nel suo file di zona l'FQDN.
  Quando ci connettiamo al nostro provider, chiediamo di usare un name server (o lo consideriamo); questo name server sarà il (primo) server che la nostra macchina contatta per entrare nel sistema DNS.
  In un'interrogazione del DNS ci sarà un name server locale (che può anche essere "remoto") nella nostra macchina, ed un name server che vogliamo raggungere, avente nel suo file di zona l'informazione che ci serve.

Scalabilità e affidabilità

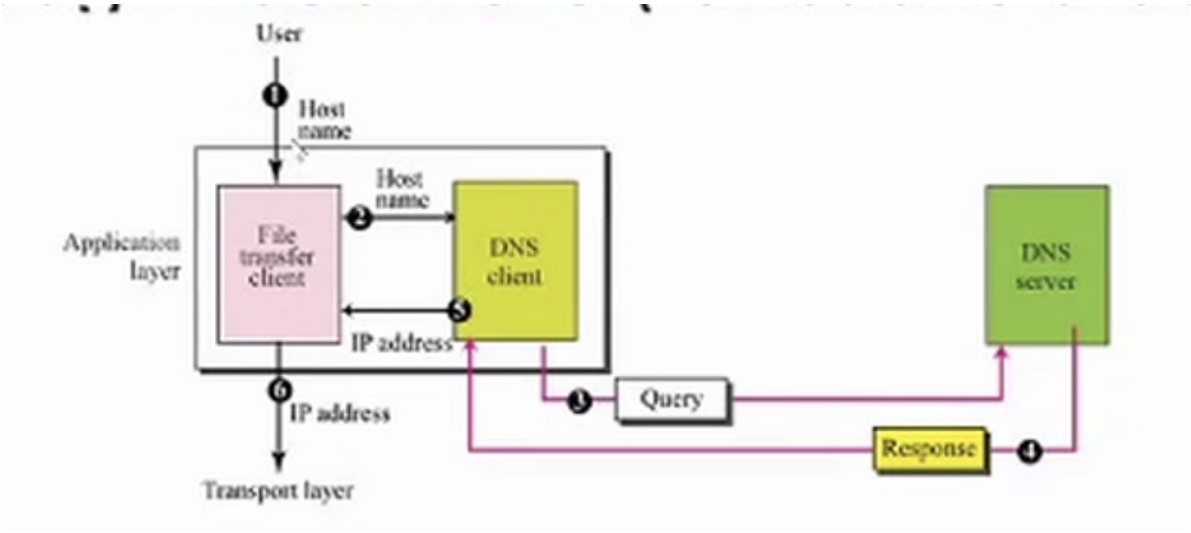
I Name Servers possono essere catalogati come **primari e secondari**; i primari mantengono i file di zona **originali**, mentre i secondari sono usati come backup e comunicano con i privati in modo da tenersi aggiornati in ogni momento.

Clients per la risoluzione dei nomi

Per poter promuovere un'intestazione con un sistema DNS, usiamo i cosiddetti **resolver**.

I **Resolver** sono delle funzioni che operano da client per il sistema DNS; parliamo di funzioni (e non di processi) perchè tipicametne il sistema DNS viene usato da client di altre applicazioni, con lo scopo di trasformare un FQDN in indirizzi IP. Quindi invece di avere un altro processo in esecuzione abbiamo una funzione usata **all'interno del processo** usato come client.

Il meccanismo di risoluzione è rappresentato graficamente:



> Ancora una volta risoluzione imbarazzante

L'utente digita un FQDN per contattare un server; la stringa FQDN viene passata dal client dell'applicazione usata, alla funzione resolver, impiegata per l'implementazione del nostro client.

Se stiamo programmando in C, la funzione che ci consente di effettuare questa operazione è **gethostbyname()**.

Quindi rispetto alle applicazioni implementate finora (dove usavamo degli indirizzi IP), ci basta aggiungere l'invocazione di questa funzione, dove passiamo l'FQDN, per ottenere di ritorno l'indirizzo IP, usato effettivamente per la comunicazione.

All'atto pratico viene effettuata una **query DNS**, dove si chiede al server DNS l'indirizzo IP.

Resolver in C e Java

C

```c
#include <netdb.h>
struct hostent *ptrh;
ptrh = gethostname(host);
```

> ci viene restituito una struttura di tipo hostent vista quì sotto:

```
struct hostent{
  char *h_name;
  char **h_aliases;
  int h_addrtype:
  int h_length;
  char **h_addr_list;
  #define h_addr h_addr_list[0]
};
```

Java

```
InetAddress[] getAllByName(String host);
InetAddress getByName(String host);
```

Passiamo come parametro un FQDN e restituisce un array di InetAddress, dove ogni oggetto rappresenta un indirizzo IP. Ci viene restituito un array perchè è possibile che ad un FQDN corrispondano più indirizzi IP.

Esempio

Con il codice

```
public static void main(String[] args) throws UnknownHostException {
        InetAddress[] googleAddresses =
InetAddress.getAllByName("www.google.com");
        System.out.println("Indirizzi per www.google.com");
        for (InetAddress address : googleAddresses)
            System.out.println(address.getHostAddress());

        InetAddress[] unisannioAddresses =
InetAddress.getAllByName("www.unisannio.it");
        System.out.println("\nIndirizzi per www.unisannio.it");
        for (InetAddress address : unisannioAddresses)
            System.out.println(address.getHostAddress());
    }
```

Otteniamo come output:

```
Indirizzi per www.google.com
142.250.184.68
2a00:1450:4002:806:0:0:0:2004

Indirizzi per www.unisannio.it
35.152.60.177
```

Come possiamo notare, google ha più indirizzi, e quindi più server (in quest'area geografica), mentre l'hostname unisannio.it ne ha solo uno.

---

fine lezione 16