Il problema del Naming

Necessità di **identificare** (**e ritrovare**) le altre entità (e risorse) nel sistema

Complessità del problema nel distribuito ma anche nel concentrato e difficoltà di soluzioni generali

I nomi sono organizzati in livelli:

- **nome** LOGICO
- **indirizzo** FISICO
- **route** come raggiungere l'entità

Il **nome** specifica *quale oggetto* (entità) si riferisce, denota l'entità:

- nome logico a livello utente
- identificatore come nome (interno) definito dal sistema

L'**indirizzo** specifica *dove* l'oggetto risiede.

La **route** specifica *come* raggiungere l'oggetto.

Livelli di Nomi e mapping

Il passaggio nomi => indirizzi => route avviene attraverso *funzioni di corrispondenza* (**Mapping**):

nomi => indirizzi
indirizzi => route

Indirizzi come tramite tra nomi (statici) e route (dinamiche)

- **nomi** scelti **dall'utente**
- **indirizzi** assegnati **dal sistema**
- **route** gestite dal **protocollo di rete (IP)**

Risoluzione del nome: mapping dal nome all'indirizzo (più in generale, agli attributi corrispondenti al nome)

Binding tra nomi e risorse

Binding: associazione di un nome con un oggetto (es. Server) e il set di relativi attributi (es. l'indirizzo fisico). Il binding è il legame tra un nome e la specifica risorsa (es. Server) a esso associata che il Client userà.

I sistemi di nomi sono l'elemento fondamentale nella gestione del binding tra nomi e risorse in una rete di calcolatori.

L'associazione tra nomi e oggetti / attributi infatti tipicamente cambia nel tempo (es., il servizio viene migrato su un nuovo server con un indirizzo IP diverso, vengono attivate/disattivate repliche del servizio, ecc.).

Molto spesso, inoltre, per motivi di scalabilità o fault tolerance, è desiderabile indicare con lo stesso nome più repliche di una risorsa (Server), anche in modo dinamico (es. aggiungo nuova macchina a un pool di Server già esistenti). Il sistema di nomi "controlla" quale specifica replica della risorsa associata al nome (ovverosia a quale Server) verrà utilizzata dal richiedente (Client).

Binding tra nomi e risorse

Lato Client sono possibili due diverse strategie per la gestione dei binding:

- **binding statico**. Il binding viene eseguito una sola volta e rimane inalterato per tutto il tempo di vita dell'applicazione.
- **binding dinamico**. Il binding ha durata limitata e viene rieseguito periodicamente. Questo abilita l'uso di un pool dinamico di risorse e consente di ottimizzarne l'efficienza, ad esempio ridirigendo le richieste sul gestore più scarico o quello in quel momento disponibile nel sistema, a fronte di un costo maggiore per il binding.

(In teoria si potrebbe pensare di eseguire un nuovo binding per ogni richiesta tra Client e Server, ma ciò sarebbe troppo oneroso: nella pratica si usa lo stesso binding per diverse richieste e chiamate successive allo stesso Server.)

Un sistema di naming per Internet

Internet necessita di un servizio di nomi, con l'obiettivo principale di mantenere e fornire le corrispondenze tra i nomi logici di host e gli indirizzi IP

Prima soluzione (1971-1981): database testuale hosts.txt da scaricare e copiare in /etc/hosts, mantenuto dallo Stanford Research Institute

Qualsiasi database centralizzato rappresenta una soluzione insufficiente e non scalabile per il sistema di naming di Internet:

- **Single point of failure**: se si guastasse il server contenente il sistema di nomi, l'intera Internet ne risentirebbe
- **Prestazioni**: tutte le richieste sarebbero concentrate su un solo nodo, causando significativi problemi di gestione (traffico, computazione)
- **Distanza**: un sistema centralizzato porterebbe a significativi ritardi nel caso di client molto distanti, o che usino collegamenti lenti e congestionati
- **Manutenzione molto frequente e onerosa**: ogni singolo cambiamento (registrazione nuovo nome, cambiamento associazione nome-indirizzo, ecc.), dovrebbe essere riportato sulla base di dati del sistema centralizzato

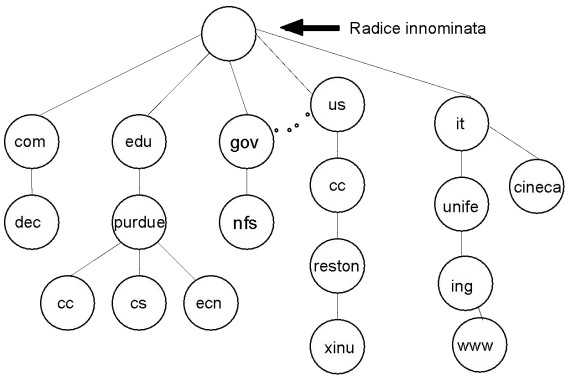
Domain Name System (DNS)

Servizio di nomi decentralizzato e basato sul concetto di **domini** amministrativamente indipendenti

NOMI LOGICI GERARCHICI

Gerarchia di **domini logici**:
- 13 root server il cui indirizzo IP è noto
- Top Level Domains (.com, .it, ecc.)

Esempio: **www.ing.unife.it** a 4 livelli



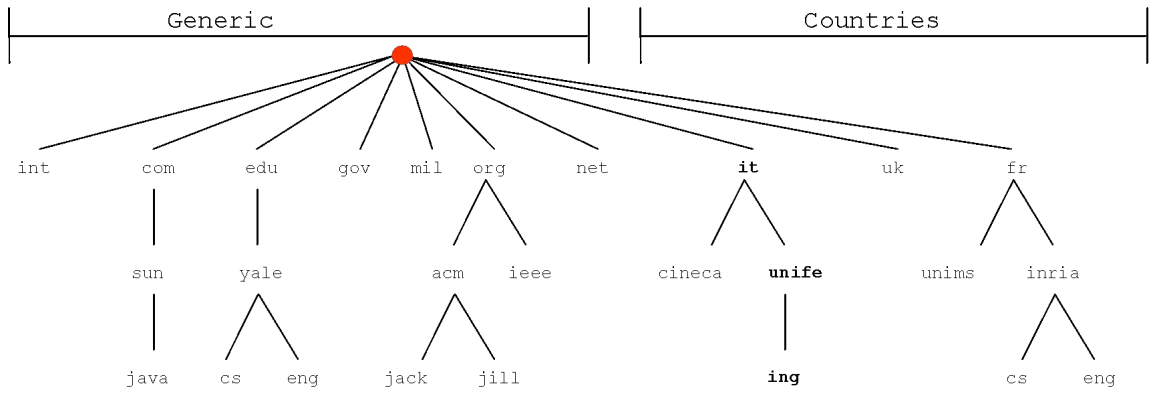
Concetto di delega: un dominio delega i sottodomini a gestori sottostanti (che se ne assumono responsabilità e **autorità**)

Domini non collegati a reti fisiche o organizzazioni (***logico vs. fisico***)

Nomi di DNS

I singoli nomi sono **case insensitive** e al max 63 char, il nome completo al max 255 char

Ogni dominio indicato in modo **relativo** o **assoluto.** Ogni dominio deve fare riferimento al dominio che lo contiene. Es. **ing** è interno a **unife**, che è interno a **it**, che è interno a root.



Risoluzione di un nome

Risoluzione di un nome (conversione tra **nome logico** e **indirizzo fisico)** avviene tramite un servizio di nomi che risponde (dinamicamente) alle richieste.

A livello di API si passa il riferimento da mappare a un **resolver** (Client) che:

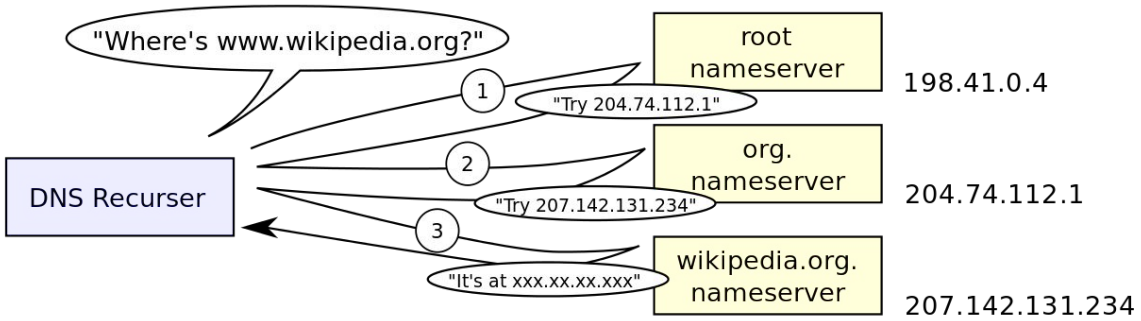
- conosce già la corrispondenza (cache)
- oppure la trova attraverso una richiesta C/S a un **name Server**

Ogni dominio corrisponde al **Name Server** che ha autorità sulla **traslazione degli indirizzi,** che non ha una visione completa, ma solo locale

Per ottimizzare le prestazioni (caching) e ridurre al minimo il traffico generato, ogni richiesta viene fatta al servizio di nomi tramite un **agente specifico** di gestione dei nomi per una località (DNS Resolver).

DNS Resolver

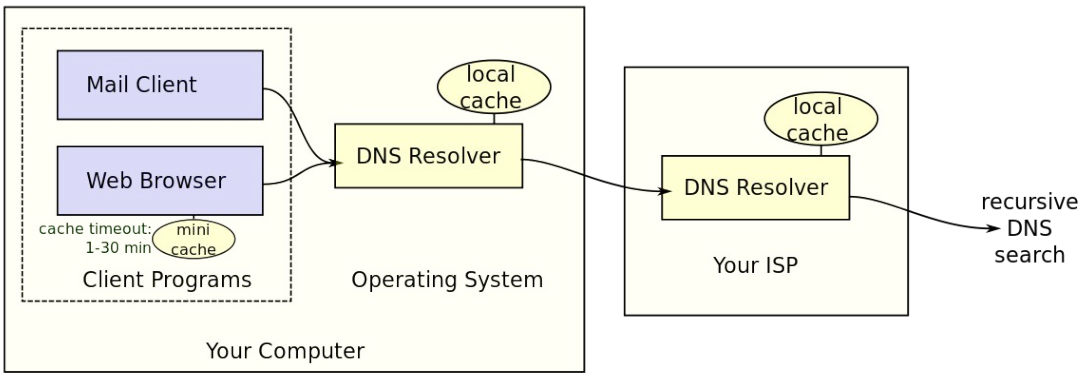
Tipicamente, i client non effettuano le risoluzioni direttamente ma inoltrano le richieste a un componente **DNS Resolver** (o DNS Recurser) installato nella propria rete locale o fornito dal proprio ISP (autoconfigurazione tramite DHCP o PPPoE). Il DNS Resolver a sua volta effettua risoluzioni tramite query ricorsive e salva i risultati nella propria cache => migliori prestazioni.



Caching

Per velocizzare le richeste di risoluzione (successive), i sistemi di nomi fanno spesso uso di meccanismi di caching.

Il DNS prevede un sistema di caching distribuito a più livelli, di cui il DNS Resolver / Recurser visto nel lucido precedente è solo un elemento:



Tempo di validità delle associazioni e caching

Le associazioni tra nome e attributo possono cambiare, senza preavviso. Questo può causare un disallineamento tra le informazioni (aggiornate) nel sistema del nomi e quelle (obsolete) contenute nelle cache.

Necessità di soluzioni specifiche per evitare che i Client usino informazioni invalide.

Nel DNS, i record restituiti alle richieste di risoluzione hanno un tempo di validità esplicito (il cosiddetto "Time To Live" o "TTL"), scaduto il quale essi devono essere eliminati dalle eventuali cache in cui sono stati salvati.

Tempo di propagazione DNS

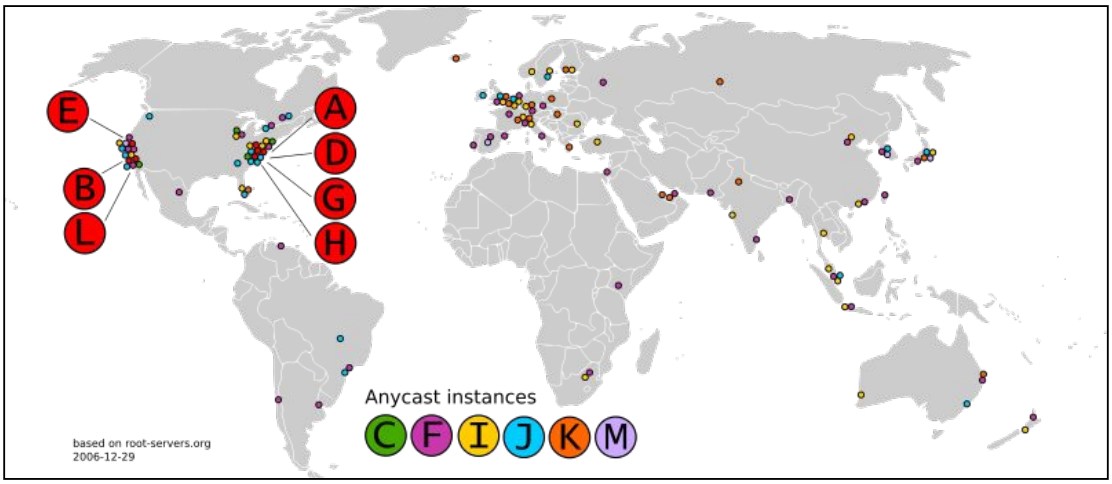
Quando si apporta una modifica (es. nuovo nome, cambio attributo, ecc.) al database del DNS, ci vuole tempo perché essa abbia effetto, per vari motivi:

- Alcuni resolver (bacati) ignorano il valore TTL dei record
- I resolver effettuano il caching dell'assenza di record per un nome (problema per definizione nuovo nomi)
- Browser e OS sono particolarmente aggressivi con il caching
- Il caching di informazioni sui nameserver è tipicamente molto lungo

Anche se un po' a sproposito, si parla di **tempo di propagazione DNS**, che è il tempo necessario perché le informazioni nelle cache sulla rete vengano invalidate e possano essere rimpiazzate. Tipicamente, siamo nell'ordine delle 24 o 48 ore (dato empirico).

Per ulteriori informazioni:
https://jvns.ca/blog/2021/12/06/dns-doesn-t-propagate/

Root Name Server



Per ulteriori informazioni: https://en.wikipedia.org/wiki/Root_name_server

DNS Record Type

DNS salva le informazioni in ***record***

•A ogni nome possono essere associati molti record, sia di tipo diverso (es. sia indirizzo IPv4 che IPv6), che dello stesso tipo (es. per ridondanza)

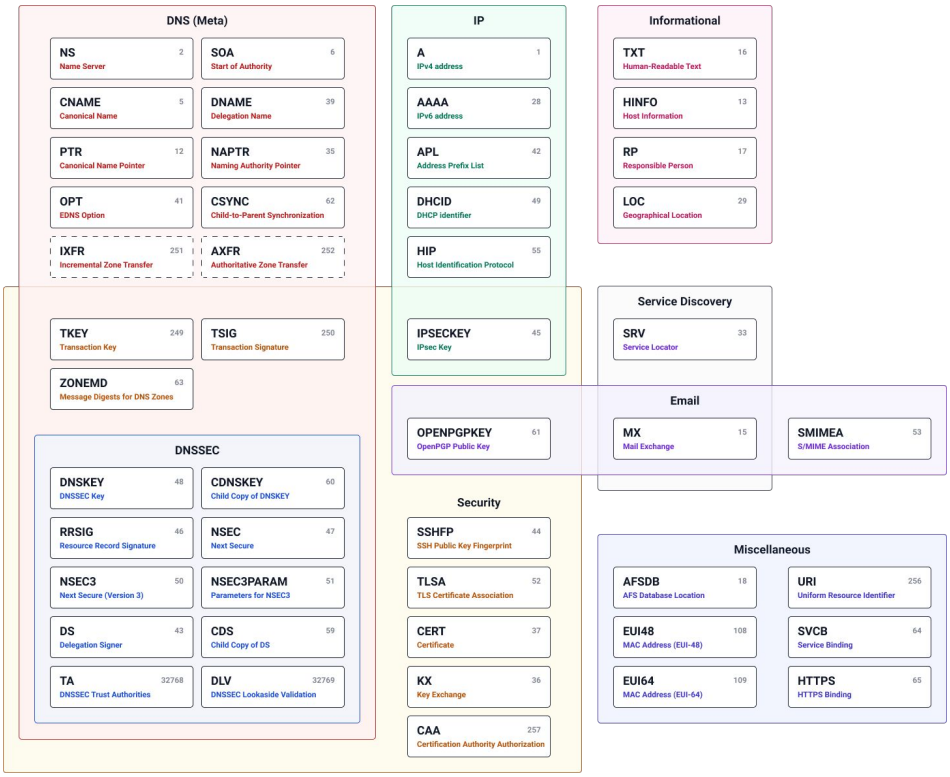
•È possibile effettuare diverse tipologie di interrogazioni al DNS specificando il tipo di record di interesse

Diversi tipi di record:

- *A* per risoluzione da nome host a indirizzo IPv4
- *AAAA* per risoluzione da nome host a indirizzo IPv6
- *MX* per risoluzione da nome dominio a nome dei server e-mail
- *NS* e *SOA* per informazioni su name server e dominio
- *PRT* per risoluzione inversa, ovverosia da indirizzo IP a nome, e DNS-SD

Si veda anche: *http://en.wikipedia.org/wiki/List_of_DNS_record_types*

Nota: la risoluzione inversa è sempre meno usata, per motivi di sicurezza.



Courtesy: https://www.nslookup.io/learning/dns-record-types/

DNS – protocollo di trasporto

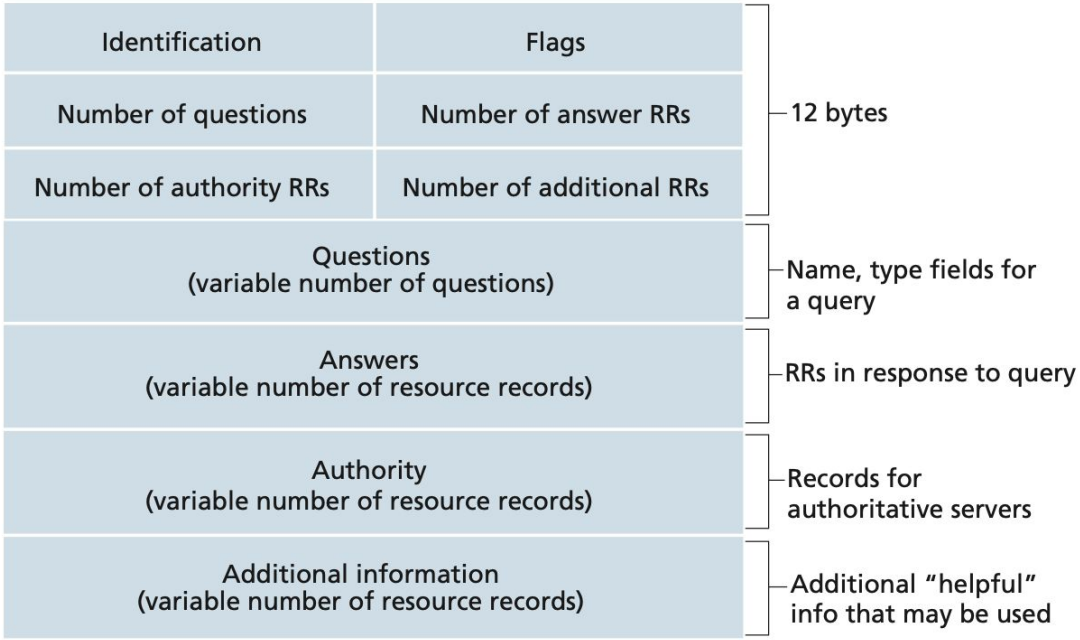
DNS usa un semplice protocollo richiesta-risposta basato su UDP.

UDP scelto perché:

- **Non è necessaria affidabilità.** Il protocollo è talmente semplice che è facile realizzare una forma di affidabilità a livello applicativo.
- **Non è necessario congestion control.** Il protocollo occupa un traffico talmente limitato che è molto improbabile che esso diventi una causa di congestioni.
- **È importante minimizzare la latenza nella risoluzione dei nomi.** Il setup di una connessione TCP rallenterebbe eccessivamente le procedure di risoluzione dei nomi.

Tutto il traffico passa “in chiaro”. Recentemente, in seguito a una cresciuta consapevolezza sui potenziali problemi di sicurezza in ambito DNS si sono sviluppate nuove soluzioni (non prive di alcuni importanti elementi di criticità, a dire il vero): DNSSEC, DNS-over-HTTPS.

DNS – Formato messaggi



dig

dig è una utility per la diagnosi di problemi con il sistema di risoluzione nomi DNS. La sintassi è:

```
dig [@server] nome [ tipo_di_query]
```

Alcuni semplici esempi:

1) Ottieni le informazioni sul dominio unife.it:
```
dig unife.it
```

2) Ottieni i record MX per il dominio unife.it:
```
dig unife.it mx
```

3) Ottieni i record A (ovverosia l'indirizzo IP) per l'host www.unife.it
```
dig www.unife.it
```

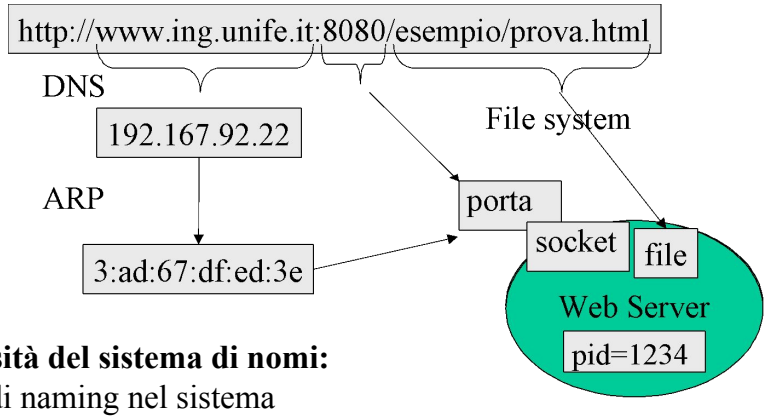
Esempio configurazione miodominio.it in BIND (Server DNS)

```
; File di configurazione BIND per miodominio.it
; (/etc/bind/zones/master/db.miodominio.it)
$TTL    3h   ; TTL di default è di 3 ore
@       IN      SOA     ns1.miodominio.it. admin.miodominio.it. (
                              1          ; Serial
                         3h       ; Refresh after 3 hours
                         1h       ; Retry after 1 hour
                         1w       ; Expire after 1 week
                         1h )     ; Negative caching TTL of 1 day
; Server authoritative per miodominio.it
@       IN      NS      ns1.miodominio.it.
@       IN      NS      ns2.miodominio.it.

; Configurazione MX
miodominio.it.    IN      MX      10      mail.miodominio.it.

; Configurazione nomi e alias
mail              IN      A       192.168.0.10
miodominio.it.    IN      A       192.168.0.10
www               IN      CNAME   miodominio.it.
ns1               IN      A       192.168.0.10
ns2               IN      A       192.168.0.11
miodominio.it      IN      AAAA    2000::1
```


URL: Uniform Resource Locator



Si noti la complessità del sistema di nomi:

- più livelli di naming nel sistema
- più sistemi di naming presenti

con diversi contesti di visibilità

Attenzione che un URL è praticamente un indirizzo. Cosa succede se si sposta una risorsa?

Spazio dei Nomi

NOME è una costruzione linguistica che distingue (**denota**) un oggetto in una collezione di oggetti.

Uno spazio dei nomi (**name space**) è la collezione di tutti i nomi (sintatticamente) validi per un particolare servizio.

Esempi.

DNS. Nomi di host.

Nomi validi: www.ing.unife.it, a.b1.c23.4h
(anche se non esiste, unbound)
Nomi non validi: "…."

Nomi processi Unix.

Nomi validi: 1224, 232, 77777 (anche se non presente)
Nomi non validi: processo34, sette

Spazio dei Nomi - 2

Lo spazio dei nomi può essere:

piatto (flat)

nessuna struttura

partizionato

gerarchia e **contesti (DNS)** (scalabilità e gestione)

`www.ing.unife.it` (ing viene risolto nel contesto unife, etc.)

descrittivo

con riferimento a una struttura di **oggetto**

oggetti hanno attributi (OSI X.500) e classi (OSI X.521)

```
dn: cn=«Mauro Tortonesi», o=unife, c=it
telephoneNumber: +39 0532 97 4888
mail: mauro.tortonesi@unife.it
objectClass: person
objectClass: top
```

Il nome deve anche permettere di ritrovare l'entità che denota, deve facilitare la ricerca.

Name Services – Servizi di Nomi

Un **servizio di nomi** memorizza i binding tra i nomi e gli attributi degli oggetti.

Il principale scopo di un servizio di nomi è di **risolverli** cioè fornire ai Clienti gli attributi associati a un nome (simile a white pages).

Esempio: la risoluzione di un nome logico di un host per avere IP
www.ing.unife.it 192.167.215.12

In generale, la risoluzione fornisce gli attributi legati al nome.

Esempi:
DNS può fornire altri dati, come la validità del binding, etc.
X500 può fornire, per un nome di una persona, l'email, il telefono, etc.

Altre operazioni **Registrazione** delle risorse:
- aggiunta e cancellazione di altre tuple <nome, attributi>
- modifica di tuple esistenti

Un servizio di nomi può essere consultato implicitamente o esplicitamente

Name Services

I Clienti del **name Server** sono:
- sia i **Clienti** che hanno esigenza di **risolvere un nome** per potere riferire una risorsa
- sia le **entità risorse** (Server rispetto ai Clienti di prima, ossia che devono essere riferiti) che devono **registrare** il servizio e diventano Clienti del name Server

Comunicazione Clienti / name Server
Si tendono a ottimizzare le operazioni più frequenti.
I Clienti propongono la maggiore parte del traffico.
Le risorse da registrare fanno operazioni più rare e producono traffico più limitato.

Attenzione, si può cercare un attributo per ottenere un nome, per esempio, cercare un servizio di stampa (servizi **directory** e **discovery**).

Servizi di Directory e Discovery

Un **servizio di nomi** recupera gli attributi associati a un nome.

Un servizio di **directory/discovery** recupera le informazioni disponibili a partire da qualunque attributo (anche i nomi possono essere visti come attributi).

Esempi.

Name service: elenco del telefono (white pages)

Directory service: pagine gialle (yellow pages)

Un servizio di **discovery** dà accesso alle stesse informazioni di un directory service ma è specifico per un ambiente locale dove i servizi e gli host possono cambiare molto dinamicamente. Ha procedure specifiche, pensate per ambienti a elevata dinamicità, per registrare e deregistrare i servizi e per lookup.

Directory	**Discovery**
• soluzioni di **nomi globali** • Cliente deve **conoscere** l'indirizzo della Directory	• soluzioni di **nomi locali** • Cliente recupera il nome del Discovery a runtime (**broadcast**). **Non è richiesta alcuna conoscenza a priori**.

Directory e Discovery

Un utente mobile vuole avere accesso a:

informazioni globali, come
descrizione dei dispositivi,
delle preferenze proprie del suo profilo,
delle sue firme digitali e PKI
delle sue sorgenti di informazioni, ecc.

informazioni locali, come
descrizione delle risorse disponibili localmente,
dei gestori presenti, ecc.

Discovery è fondamentale per arrivare a realizzare servizi location- e context- aware. Esempio: un utente mobile con un laptop che vuole stampare un documento sulla stampante più vicina, nella località in cui si trova.

Directory: esempi

OSI X.500. Servizio di Directory e di Nomi

CCITT definisce X.500 come *"una collezione di sistemi aperti che cooperano per mantenere un database logico di informazioni sugli oggetti del mondo reale. Gli utenti della Directory possono leggere o modificare l'informazione, o parte di essa, solo se hanno i privilegi necessari"*

LDAP (Lightweight DAP)

È un importante directory, molto diffuso nel mondo Internet, è basato su una variazione del Directory Access Protocol (DAP) specificato da X.500

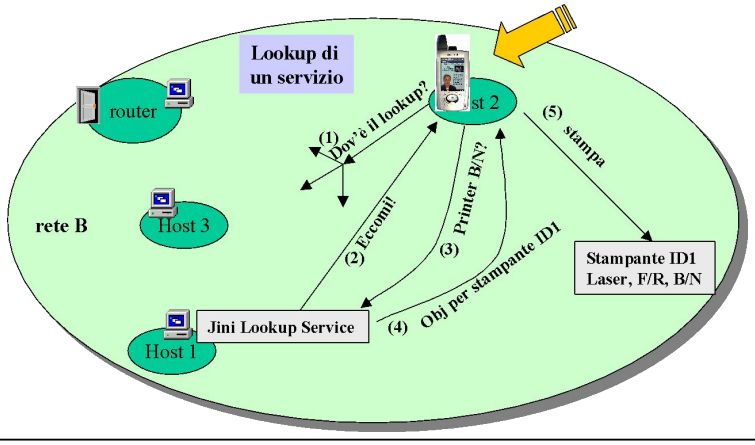
Active Directory

Microsoft propone **Active Directory** come un servizio di direttori integrato nel e per il sistema operativo (anche con interfaccia LDAP)

I Servizi di Discovery

Un servizio di discovery mantiene le associazioni <nome-attributi> come un servizio di directory o di nomi, però memorizza gli attributi delle risorse locali perché serve per facilitare l'accesso a tali risorse **senza conoscenza prestabilita**. Un Cliente usa comunicazioni broadcast (o multicast) per recuperare servizi locali.

Esempio: **JINI**, una delle prime soluzione di discovery (che ha avuto scarsissima diffusione ma è di importanza "storica") in ambiente Java



Multicast DNS (mDNS) e DNS-SD

Multicast DNS è nato come estensione di DNS per il ***la risoluzione di nomi*** dinamica in ambienti di rete locale (LAN)

Fa uso di:
- messaggi UDP (formato compatibile al 99% con DNS tradizionale)
- porta 5353
- comunicazioni multicast (su indirizzi 224.0.0.251 per IPv4 e FF02::FB per IPv6) sia per le richieste che per le risposte
- dominio riservato *.local* per motivi di sicurezza
- codifica UTF-8 per le stringhe dei nomi

Uso di DNS-SD al di sopra di mDNS per il ***discovery di servizi***.

Provate a lanciare sulla vostra rete di casa il comando:
```
dns-sd -B _http._tcp
```


Link Local Multicast Name Resolution (LLMNR)

LLMNR è nato come risposta di Microsoft (quindi ovviamente simile a ma del tutto incompatibile con le altre soluzioni esistenti) a mDNS

Fa uso di:
- messaggi UDP
- porta 5355
- comunicazioni multicast (su indirizzi 224.0.0.252 per IPv4 e FF02::1:3 per IPv6) per le richieste, unicast per le risposte
- i nodi client devono anche mettersi in ascolto sulla porta TCP 5355

Uso di Simple Service Discovery Protocol (SSDP), un protocollo della suite UPnP basato su scelte ingegneristiche discutibili come il trasporto di messaggi HTTP su UDP, al di sopra di LLMNR per il *discovery di servizi*

Per le differenze tra mDNS e LLMNR si veda:
https://www.eiman.tv/blog/posts/lannames/
Pare che Microsoft stia iniziando ad adottare (o quantomeno a supportare) mDNS a partire da Windows 10