Università degli Studi di Roma “Tor Vergata”
Facoltà di Ingegneria

Naming nei Sistemi Distribuiti

Corso di Sistemi Distribuiti

Valeria Cardellini
Anno accademico 2009/10

Ruolo del naming nei SD

- I nomi in un SD sono usati per:
  - condividere risorse, servizi o applicazioni
  - identificare univocamente entità
  - far riferimento alla loro posizione (localizzarle)
- Per risolvere un nome nell’entità a cui si riferisce occorre realizzare un sistema di naming
- Il naming nei SD si differenzia da quello usato nei sistemi non distribuiti per l’implementazione
  - Nei SD anche il sistema di naming è distribuito

Denominazione di entità

- Le entità di un SD devono avere denotazioni che le rendano fruibili
  - Per riferimento, invocazione di servizio, …
- Un nome è una stringa che riferisce entità
- Un punto d'accesso (access point) è un tipo speciale di entità che permette di operare su un'entità
  - Il nome del punto d'accesso è un indirizzo
- Un'entità può avere più punti di accesso
  - Più "punti di vista" simultanei o anche diversi nel tempo
- Esempio di denominazione di entità
  - Persona = entità di sistema
    - Un servizio
  - Il suo telefono = punto di accesso del servizio
    - Il dispositivo (server) che fornisce quel servizio
  - Il numero di telefono = nome (indirizzo) dell'punto di accesso di quel servizio
    - L'indirizzo del server è il suo end point di livello trasporto

Denominazione di entità (2)

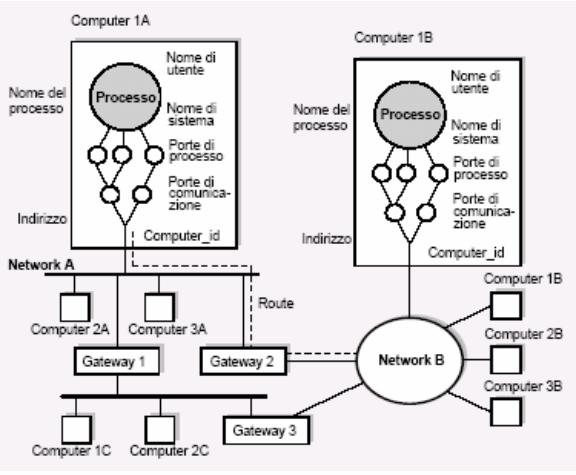
- Trattare gli indirizzi come nomi speciali
  - Decomporre la corrispondenza <entità-indirizzo> in 2 relazioni legate ma distinte
    - Nome di entità → nomi degli attributi dell'entità (tra cui il punto di accesso)
    - Punto di accesso → indirizzo dell'entità
  - Un'entità può cambiare punto di accesso
  - Un punto di accesso può essere riassegnato
  - Una stessa entità può avere più punti di accesso
- In un SD vogliamo nomi di entità indipendenti dai loro indirizzi (indipendenza dalla posizione, location independence)
  - Un nome di entità *E* indipendente dalla posizione è indipendente dagli indirizzi dei punti di accesso offerti da *E*

Identificatori

- Certi nomi designano entità in modo univoco
  - Per questi nomi usiamo il termine identificatori
  - Non tutti i nomi sono identificatori
  - Alcuni nomi devono essere “trattabili” (human-friendly) ma non indirizzi e identificatori
- Un identificatore è un nome con le seguenti proprietà:
  1. Denota al più una singola entità
  2. Una stessa entità non può avere più di un identificatore
     - Identificatori diversi designano entità diverse
  3. Si riferisce sempre alla stessa entità (non è riusabile)
- Es.: un numero di telefono fisso non è un vero identificatore di un’entità “persona” perché può essere riassegnato
- Es.: l’indirizzo MAC è un identificatore della scheda di rete
- Nomi diversi per una stessa entità sono detti alias

Sistemi di naming

- Un sistema di naming (o di nomi) permette di localizzare le entità del SD
  - Mantenendo un collegamento nome-indirizzo tra entità
  - In modo indipendente dalla posizione delle entità
- Spesso nei SD troviamo molti sistemi di naming che hanno numerose proprietà e sono anche molto diversi tra loro



Sistemi di naming (2)

- Complessità del problema del naming e difficoltà di trovare soluzioni generali
- Entità eterogenee → livelli diversi di nomi
  - Più sistemi di naming e più livelli di nomi in un SD
- Con diversi contesti di visibilità
  - Più funzioni di trasformazione (mapping) tra livelli di nomi
- Proprietà di base dei sistemi di naming
  - Generalità
    Varietà dei nomi disponibili e trattati
  - Definizioni multiple della stessa entità (*alias*)
    Varietà di nomi per la stessa entità con mapping capace di traslare tra questi
  - Distribuibilità
    Uso di directory partizionate e/o replicate
  - User-friendliness
    Nomi facili per l'utente

Tipologie di naming

- Tre tipi di nomi da cui derivano altrettanti schemi di naming
  - Nomi **semplici** (o flat)
    - Non strutturati, stringhe di bit casuali
  - Nomi **strutturati**
    - Composti da nomi semplici, leggibili dall'uomo
  - Nomi basati sugli attributi
    - Entità descritte da un insieme di coppie <attributo, valore>
- Naming semplice (o flat)
- Naming gerarchico
- Naming basato sugli attributi

Naming semplice

- Dato un nome non strutturato (ad es. un identificatore) come risolverlo nell'indirizzo dell'entità associata?

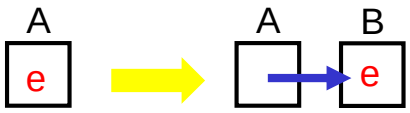
- Approcci semplici
  - Broadcasting o multicasting
  - Puntatori forwarding
- Approcci basati sull'assegnamento di una home
- Hash table distribuite
- Approcci gerarchici

Approcci semplici: broadcasting

- Principio di funzionamento: broadcast di un messaggio contenente l'identificatore dell'entità, richiedendo all'entità di restituire il suo indirizzo
- Ad es. usato in ARP (protocollo di risoluzione degli indirizzi)
  - Mapping tra l'indirizzo IP a 32 bit e l'indirizzo MAC a 48 bit
- Mancanza di scalabilità
  - Adatto solo per reti locali
- Richiede a tutti i processi di ascoltare richieste di localizzazione (cui non sono in grado di rispondere)
  - Soluzione (parziale): multicasting
    - Solo un numero ristretto di macchine riceve il messaggio

Approcci semplici: puntatori forwarding

- Approccio diffuso per localizzare entità mobili
- Ogni volta che un'entità si sposta, lascia un puntatore forwarding alla sua nuova posizione
  - Quindi un client può rintracciare l'entità seguendo la catena di puntatori forwarding

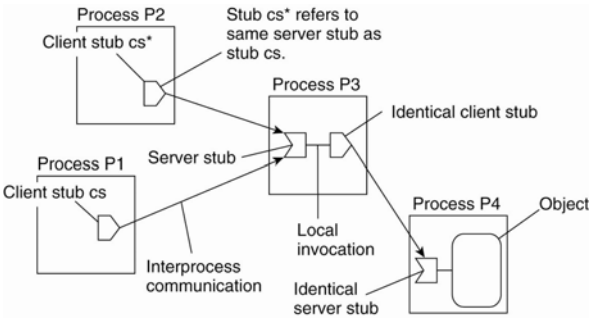


- Svantaggi dei puntatori forwarding:
  - Problemi di scalabilità geografica
    - Catene di forwarding molto lunghe: localizzazione costosa
    - Tempo di latenza per la deferenziazione dei puntatori
  - Ogni punto intermedio della catena deve mantenere le informazioni di forwarding
  - Sistema vulnerabile all'interruzione della catena
- Per mantenere corta la catena di forwarding, si può aggiornare il riferimento sul client non appena viene localizzata la posizione

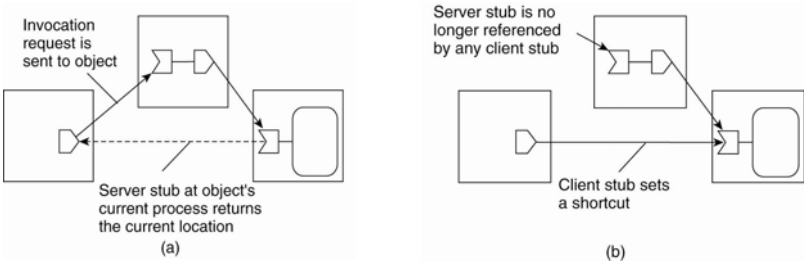
Approcci semplici: puntatori forwarding (2)

- Puntatori forwarding implementati come una coppia (client stub, server stub)

Sistema SSP (*Stub Scion Pairs*) di forwarding



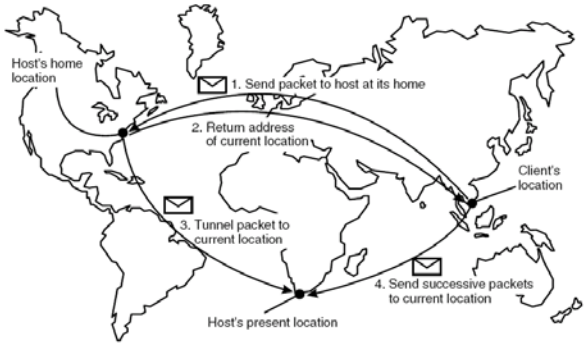
- Redirezione di un puntatore forwarding memorizzando il collegamento in un client stub



Approcci home-based

- Schema ad un **singolo livello**
  - Una home tiene traccia della posizione attuale dell'entità (mobile)
    - Di solito, il posto dove è stata creata l'entità
  - L'home address dell'entità è registrato in un servizio di naming
- La home registra l'indirizzo attuale dell'entità
- I client contattano sempre prima la home e poi continuano usando la posizione attuale dell'entità
- Principio adottato in Mobile IP
  - Home agent
    - Fornisce l'home address
  - Care-of address
    - Indirizzo attuale

  Riferimento: RFC 3344



Approcci home-based (2)

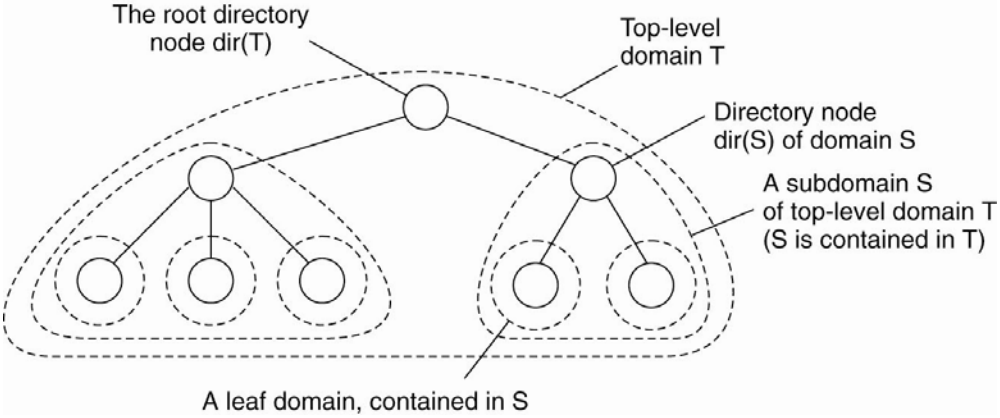
- Schema a **due livelli**
  - Tenere traccia delle entità visitate
  - Controllare prima un registro locale
  - Se la ricerca locale fallisce, contattare la home
- Svantaggi degli approcci home-based
  - L'indirizzo della home deve essere supportato finché esiste l'entità
  - L'indirizzo della home è fisso: problema che si aggrava quando l'entità si sposta in modo permanente in un'altra posizione
  - Scarsa scalabilità geografica (l'entità potrebbe essere vicina al client)

Hash table distribuite

- Meccanismo già esaminato (reti P2P strutturate)
  - Esempio analizzato: Chord
- Come tener conto della vicinanza sulla rete fisica sottostante in un sistema basato su DHT?
  - Assegnamento degli identificatori dei nodi basato sulla topologia (o proximity ID selection)
    - Quando viene assegnato un ID ai nodi, fare in modo che nodi vicini nello spazio degli ID siano anche vicini in rete; può essere molto difficile da realizzare
  - Proximity route selection
    - Ogni nodo mantiene una lista di successori alternativi e sceglie il più vicino tra quelli possibili quando effettua l'instradamento (in realtà è proximity next-hop selection)
  - Proximity neighbor selection
    - Si ottimizzano le tabelle di routing in modo da scegliere come neighbor il nodo più vicino (non possibile in Chord)

Approcci gerarchici

- Idea di base: costruire un albero di ricerca gerarchico
  - La rete è suddivisa in domini non sovrapposti
  - I domini possono essere raggruppati in domini di livello più alto
  - Esiste un unico dominio top-level che copre l'intera rete
  - A qualunque livello, ogni dominio D ha un nodo directory associato dir(D) che tiene traccia delle entità nel dominio

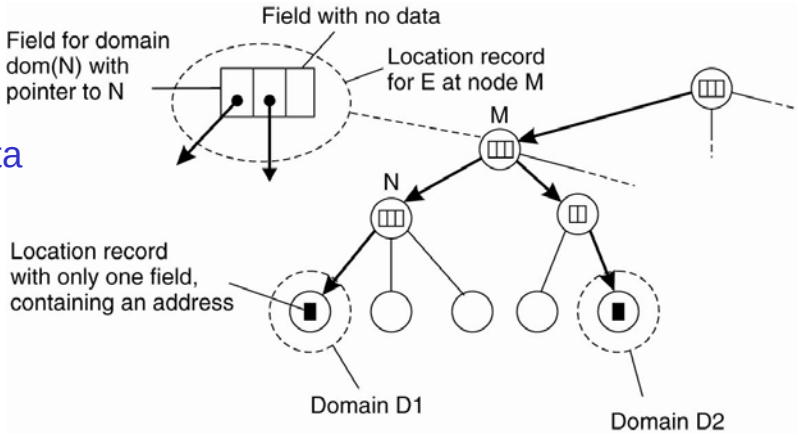


Approcci gerarchici: organizzazione

- L'indirizzo di un'entità è memorizzato solo nei nodi directory foglia all'interno di un location record
- Se un'entità si trova nel dominio D, il nodo directory del dominio di livello superiore D' avrà nel location record un puntatore al nodo directory di D
- Il nodo directory radice conosce tutte le entità
  - Possiede un location record per ogni entità

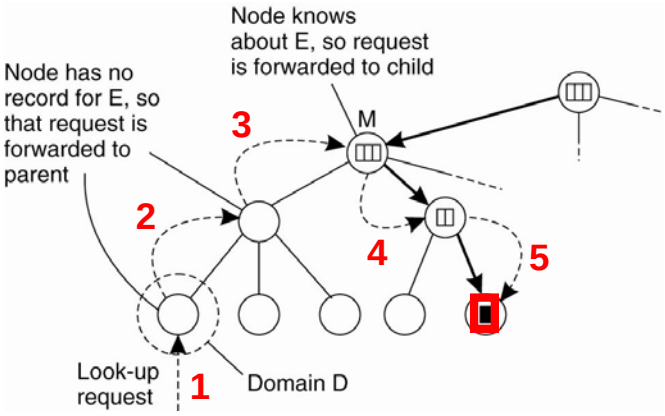
Un'entità può essere replicata

- In figura: nodo M con 2 due puntatori in corrispondenza alle 2 repliche di E (in D1 e D2)

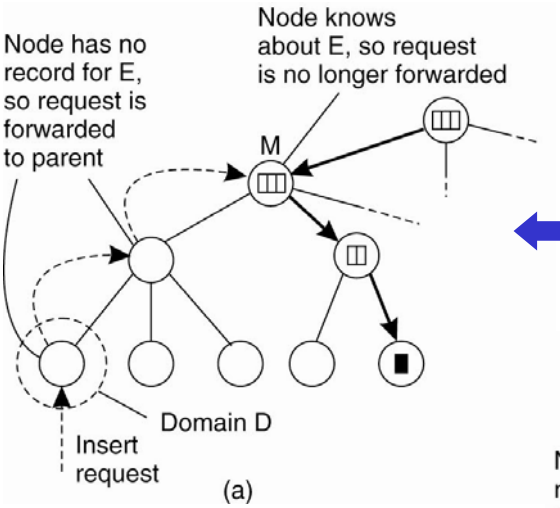


Approcci gerarchici: ricerca

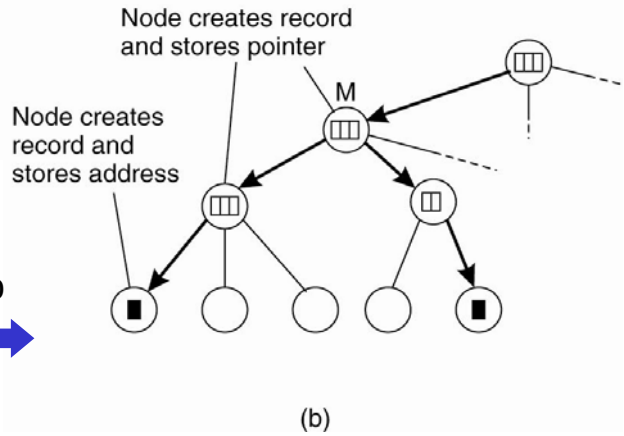
- Operazione di ricerca
  - Si inizia la ricerca dell'entità E dal nodo foglia locale per il client
  - Se il nodo conosce l'entità si segue il puntatore, altrimenti si inoltra la richiesta al nodo padre
  - La ricerca in su termina sempre al nodo radice



Approcci gerarchici: inserimento



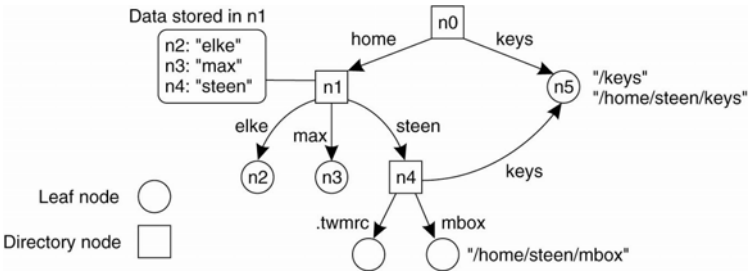
La richiesta di inserimento di una replica dell'entità E è inoltrata fino a raggiungere il primo nodo che conosce l'entità E



La catena di puntatori viene costruita in modo *top-down* verso il nodo foglia (in alternativa: *bottom-up*)

Naming strutturato

- Esempi di nomi strutturati: nomi di file, nomi degli host in Internet
- **Spazio dei nomi**: grafo orientato etichettato dove
  - un nodo foglia rappresenta una data entità
  - un nodo directory è un'entità che fa riferimento ad altri nodi
  - gli archi sono etichettati con un nome
  - il nodo con solo archi in uscita è detto nodo radice (possono esserci più nodi radice in un grafo dei nomi)
  - un nodo directory contiene una tabella (directory) di coppie (*etichetta arco*, *identificatore nodo*)
- I grafi dei nomi sono spesso aciclici, ma non tutti



Risoluzione dei nomi

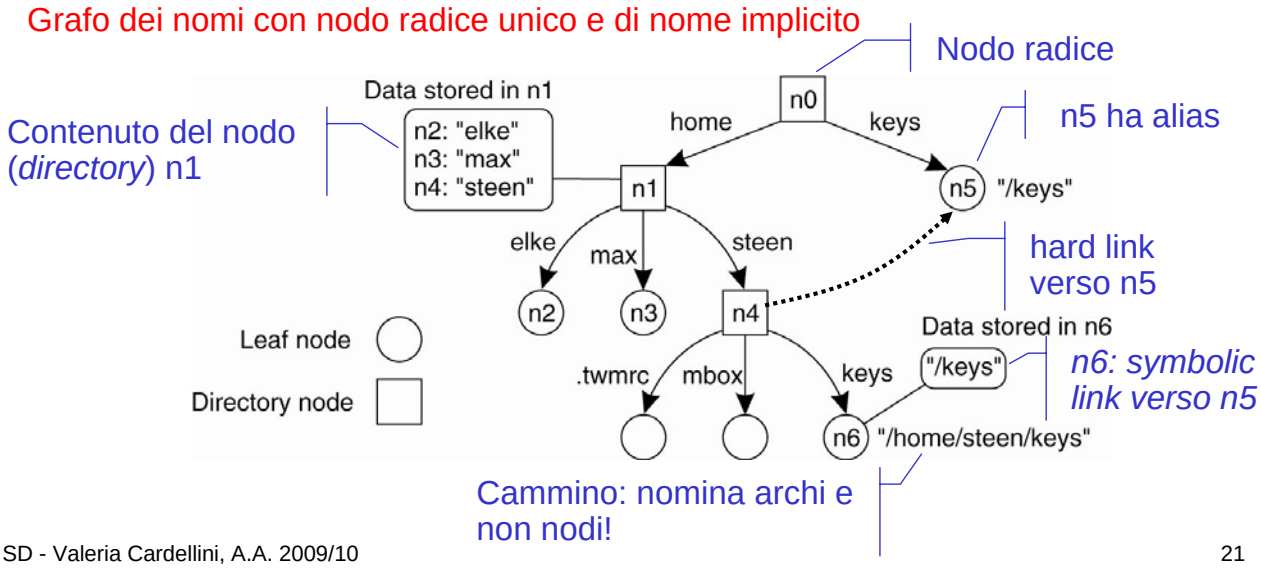
- Ogni cammino N nello spazio dei nomi è denotato dalla sequenza delle etichette degli archi corrispondenti al cammino, detta path name

  N: <$label_1$, $label_2$, .., $label_n$>

  - Cammini *assoluti* o *relativi*
- **Risoluzione dei nomi**: processo di attraversamento del grafo dei nomi cercando, uno per volta, i componenti del path name
- *Problema*: per risolvere un nome occorre un nodo directory → come sapere come e da dove iniziare?
- Meccanismo di chiusura: è il meccanismo per selezionare il contesto implicito dal quale iniziare la risoluzione del nome
  - Es: www.ce.uniroma2.it: inizia da un name server DNS
  - Es: nel grafo dei nomi di un file system in UNIX e GNU/Linux il nodo radice è sempre il primo i-node della partizione che rappresenta il file system

Esempio di grafo dei nomi

- Alias
  - Hard link: più cammini assoluti denotano lo stesso nodo di un grafo dei nomi
  - Symbolic link: il nodo contiene un attributo con il cammino assoluto

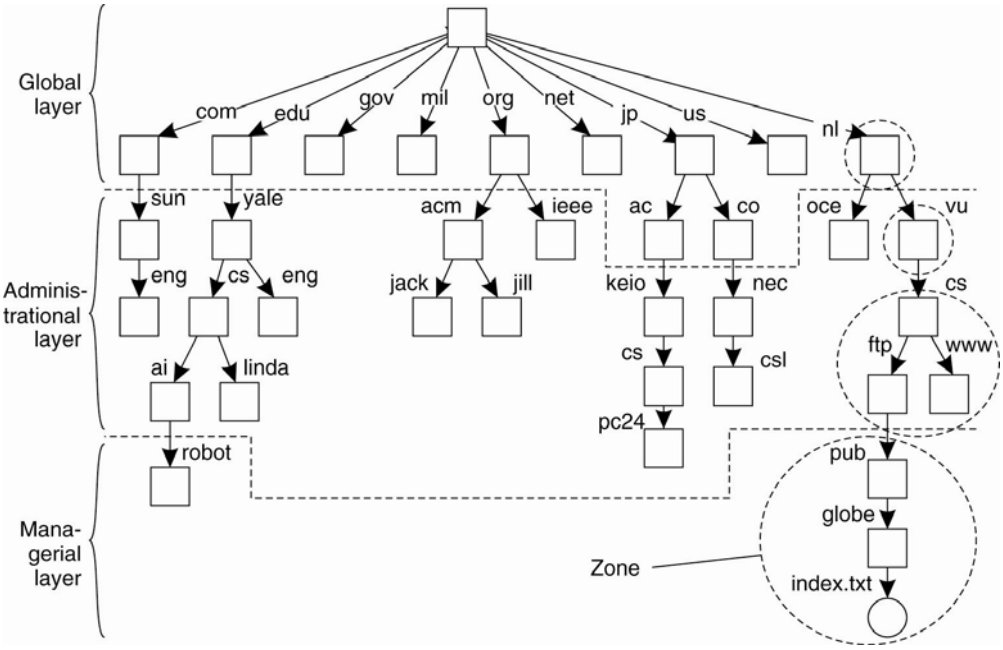


Implementazione di uno spazio dei nomi

- Lo spazio dei nomi è il cuore di un naming service
  - Un naming service consente agli utenti di inserire, eliminare e cercare nomi
  - Un naming service è implementato attraverso uno o più name server
- Occorre distribuire il processo di risoluzione dei nomi e la gestione dello spazio dei nomi su molteplici macchine, distribuendo i nodi del grafo
- Suddividiamo lo spazio dei nomi in 3 livelli logici
- Livello globale
  - Nodo root e nodi directory di alto livello, gestiti congiuntamente da diverse amministrazioni
    - Informazione generalmente stabile
- Livello amministrativo
  - Nodi directory di livello intermedio che rappresentano entità appartenenti alla stessa unità amministrativa o azienda
- Livello gestionale
  - Nodi directory di livello più basso all'interno di una singola amministrazione; possono cambiare frequentemente

Implementazione di uno spazio dei nomi (2)

- Esempio di possibile suddivisione in 3 livelli dello spazio dei nomi del DNS



Requisiti dei name server

- Disponibilità e prestazioni dei name server dipendenti dal livello logico in cui si trovano
- Name server di livello globale
  - Altamente disponibili
  - Caching e replicazione per migliorare le prestazioni
- Name server di livello amministrativo
  - Requisiti simili ai name server di livello globale
  - Buon livello di disponibilità
  - Caching e replicazione per migliorare le prestazioni
- Name server di livello gestionale
  - Possibile temporanea indisponibilità
  - Caching meno efficiente a causa dei continui aggiornamenti

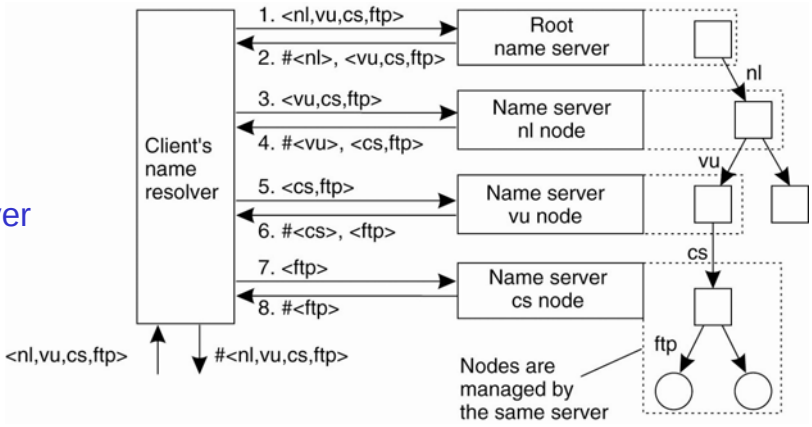
Requisiti dei name server (2)

Elemento	Globale	Amministrativo	Gestionale
Scala geografica	Globale	Azienda	Dipartimento
Numero nodi	Pochi	Molti	Moltissimi
Tempi di risposta alle ricerche	Secondi	Millisecondi	Immediato
Propagazione aggiornamenti	Lenta	Immediata	Immediata
Numero repliche	Molte	Nessuna o poche	Nessuna
Applicazione caching lato client	Sì	Sì	A volte

Risoluzione dei nomi iterativa

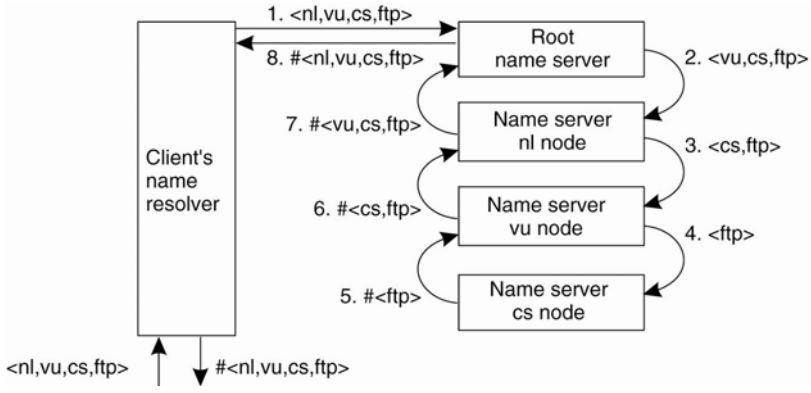
- `resolve(dir,[name1,...,nameK])` è inviato a NameServer0 responsabile per dir
- NameServer0 risolve `resolve(dir,name1)`→`dir1`, restituendo l'identificatore (indirizzo) di NameServer1, che memorizza dir1
- Il client (o il name resolver locale) invia `resolve(dir1,[name2,...,nameK])` a NameServer1, ...

La notazione #< > identifica il name server



Risoluzione dei nomi ricorsiva

- `resolve(dir,[name1,...,nameK])` è inviato a NameServer0 responsabile per dir
- NameServer0 risolve `resolve(dir,name1)`→`dir1` ed invia `resolve(dir1,[name2,...,nameK])` a NameServer1, che memorizza dir1
- NameServer0 attende il risultato della risoluzione da NameServer1 e lo restituisce al client



Caching nella risoluzione ricorsiva

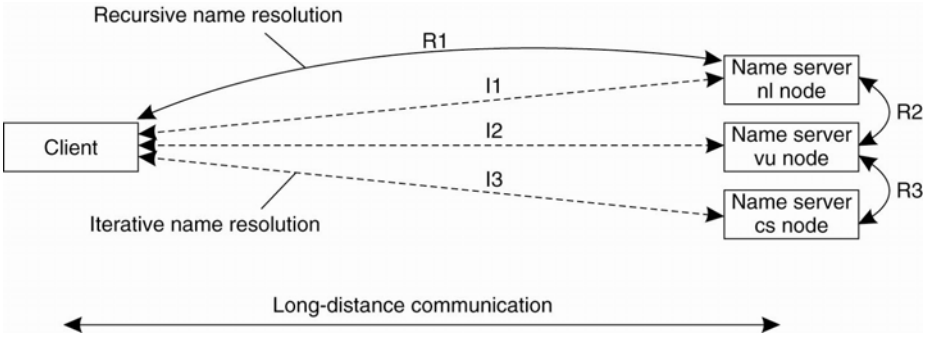
Server for node	Should resolve	Looks up	Passes to child	Receives and caches	Returns to requester
cs	<ftp>	#<ftp>	—	—	#<ftp>
vu	<cs,ftp>	#<cs>	<ftp>	#<ftp>	#<cs> #<cs, ftp>
nl	<vu,cs,ftp>	#<vu>	<cs,ftp>	#<cs> #<cs,ftp>	#<vu> #<vu,cs> #<vu,cs,ftp>
root	<nl,vu,cs,ftp>	#<nl>	<vu,cs,ftp>	#<vu> #<vu,cs> #<vu,cs,ftp>	#<nl> #<nl,vu> #<nl,vu,cs> #<nl,vu,cs,ftp>

Aspetti di scalabilità

- Scalabilità rispetto alla dimensione: occorre assicurare che i name server siano in grado di gestire un elevato numero di richieste per unità temporale
  - Problema per i server di alto livello
- *Soluzione*: si assume (almeno a livello globale ed amministrativo) che il contenuto dei nodi cambi raramente e si applicano meccanismi di replicazione, distribuendo i nodi su molteplici server ed iniziando la risoluzione dal server più vicino, e meccanismi di caching
- *Osservazione*: un attributo importante di molti nodi è l'indirizzo a cui può essere contattata l'entità rappresentata
  - La replicazione dei nodi rende i meccanismi di name server tradizionali inadatti per localizzare entità mobili

Aspetti di scalabilità (2)

- Scalabilità geografica: occorre assicurare che il processo di risoluzione scali su grandi distanze geografiche
  - Risoluzione ricorsiva più scalabile di quella iterativa



- *Problema*: distribuendo i nodi su server che potrebbero, in linea di principio, essere localizzati ovunque, si introduce implicitamente nello schema di naming una dipendenza dalla posizione

Domain Name System

- Il Domain Name System (DNS) è uno dei più diffusi servizi di naming distribuiti
  - Già analizzato nel corso di Reti dei Calcolatori
  - Esaminati esempi di uso del tool a riga di comando **`dig`**
- In letteratura proposta di un'implementazione del DNS *completamente decentralizzata* basata su DHT
- Idea di base: dato un nome DNS completo, trasformarlo tramite l'applicazione di una funzione hash in una chiave *k* ed usare un sistema basato su DHT per effettuare la ricerca della chiave
- Vantaggio principale: scalabilità
- Svantaggio principale: non si possono ricercare tutti i nodi in un sottodominio

Naming basato su attributi

- Sia il naming semplice sia quello strutturato consentono di far riferimento alle entità in modo indipendente dalla loro posizione

- Esiste anche la possibilità di usare informazioni ancora più dettagliate per localizzare le entità
  - Si fornisce una descrizione dell'entità in termini di coppie *<attributo, valore>*
- Sistemi di naming basati sugli attributi: anche noti come directory service

Directory service

- Directory: struttura dati ordinata dove viene memorizzata l'informazione in forma di elementi detti *entry*
- Directory service
  - Realizza un sistema di naming basato su attributi piuttosto che su nomi strutturati (come il DNS)
  - Directory service tratta spazi di nomi come insiemi di coppie <attributo, valore>
    - Alle entità vengono associate informazioni dettagliate che consentono di descrivere le risorse e che vengono usate per effettuare la ricerca dell'esatta localizzazione della risorsa stessa
  - Una sorta di pagine gialle

- Come descrivere le risorse in modo unificato?
  - Possibile soluzione: Resource Description Framework (RDF)

Perché LDAP?

- Le operazioni di ricerca in un directory service distribuito possono essere molto costose
  - Richiedono il matching con i valori degli attributi richiesti rispetto ai valori reali degli attributi (ricerca esaustiva tra tutti i descrittori)
- Soluzione: implementare un directory service essenziale e combinarlo con un tradizionale sistema di naming strutturato
- Da tale combinazione ha origine **LDAP** (Lightweight Directory Access Protocol)
  - Derivante da OSI X.500, ma con miglioramenti nelle prestazioni
- L'architettura complessiva LDAP risulta molto simile a quella del DNS, però anche più sofisticata e potente

LDAP

- Protocollo di accesso a informazioni condivise in rete
- Accesso client-server a collezione di informazioni: operazioni di ricerca, lettura, inserimento, modifica e cancellazione
  - Ma LDAP non è un database, essendo il suo scopo la ricerca di informazioni e non la gestione delle stesse
    - Informazioni mantenute in un server LDAP soprattutto lette, soltanto occasionalmente scritte o aggiornate (no meccanismi di roll-back e sincronizzazione)
    - Organizzazione delle informazioni di tipo descrittivo e non relazionale
- Il protocollo LDAP è costituito dalle seguenti parti:
  - Definizione dell'organizzazione dei dati, per

LDAP (2)

- Server LDAP: implementa LDAP, le informazioni sono mantenute al suo interno
- Gateway LDAP: implementa LDAP, utilizza altri server per reperire le informazioni

- Alcune implementazioni di LDAP
  - OpenLDAP (open source)
  - Sun Directory Service
  - Microsoft Active Directory

Applicazioni di LDAP

- LDAP è la soluzione più diffusa per l'implementazione di servizi di directory *general purpose*
- Fra le possibili applicazioni di LDAP:
  - White pages (elenco telefonico e indirizzario)
    - Ad es. i principali client di posta elettronica prevedono la connessione ad un server LDAP per le funzioni di rubrica
  - Autenticazione e autorizzazione
  - Routing dei messaggi di posta elettronica
  - Distribuzione di certificati X.509 e CRL
  - Persistenza di oggetti e classi Java (via JNDI)
  - Backend per altri servizi di directory
  - Memorizzazione di profili utenti

Organizzazione dell'informazione

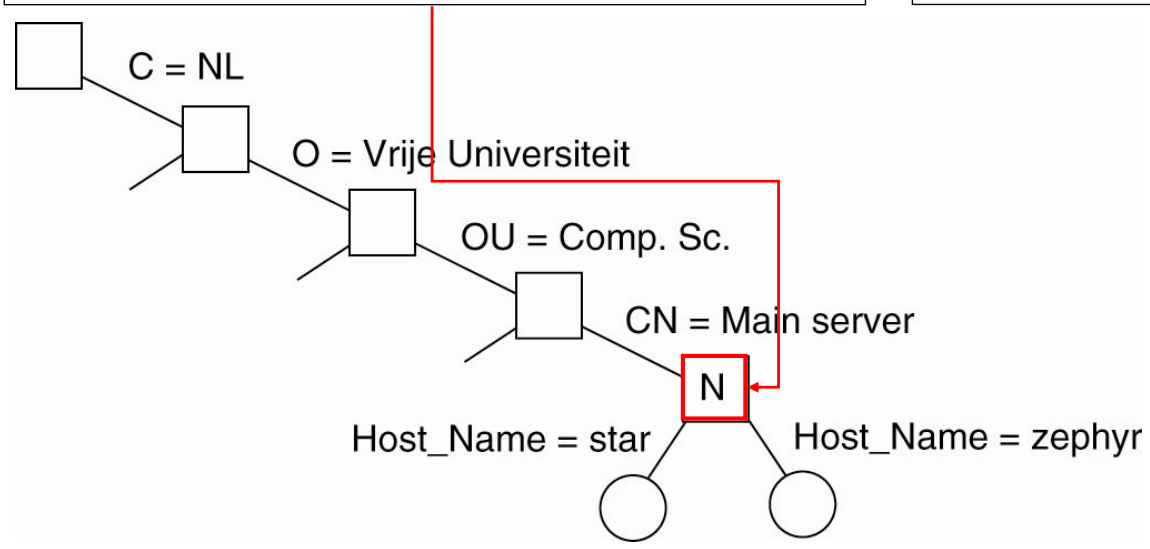
- Directory service LDAP consiste di entry della directory (paragonabili ai resource record nel DNS)
  - Entry costituita da un insieme di coppie <attributo, valore> in cui ogni attributo ha un tipo associato

Attribute	Abbr.	Value
Country	C	NL
Locality	L	Amsterdam
Organization	O	Vrije Universiteit
OrganizationalUnit	OU	Comp. Sc.
CommonName	CN	Main server
Mail_Servers	—	137.37.20.3, 130.37.24.6, 137.37.20.10
FTP_Server	—	130.37.20.20
WWW_Server	—	130.37.20.20

- Insieme di tutte le entry in un directory service LDAP: DIB (Directory Information Base)
  - Ogni entry nel DIB ha un nome univoco globale
- Informazioni organizzate con una struttura gerarchica ad albero: DIT (Directory Information Tree)

Organizzazione dell'informazione (2)

LDAP: /C=NL/O=Vrije Universiteit/OU=Comp. Sc. → **DNS**: nl.vu.cs




```
answer = search("&(C = NL)
                 (O = Vrije Universiteit)
                 (OU = *)
                 (CN = Main server)")
```

Organizzazione dell'informazione (3)

- In caso di directory di larga scala, DIT suddiviso e distribuito su più server, detti directory service agent
  - Simile al name server nel DNS
- Un server LDAP è in grado di propagare le proprie directory ad altri server LDAP, fornendo accesso globale all'informazione
- Quando un client LDAP si connette a un server LDAP può ricercare in una directory oppure modificare informazioni al suo interno
  - In caso di ricerca, il server LDAP risponde oppure delega il flusso dell'interrogazione ad un altro server
  - In caso di modifica, il server LDAP verifica che l'utente abbia il permesso di attuare la modifica, poi inserisce, aggiorna o cancella l'informazione