Naming nei Sistemi Distribuiti

Naming (1)

- I nomi sono elementi primari di ogni sistema composto da diversi entità per riferirci ad esse.
- La ***risoluzione dei nomi*** permette ad un processo di accedere a una entità in un sistema distribuito.
- Un sistema di naming è necessario per avere un modello comune di identificazione delle risorse.
- In un sistema distribuito il naming system è distribuito, per ragioni di scalabilità, efficienza, affidabilità, ecc.

Naming (2)

- In un sistema distribuito
  - Un **nome** è una stringa (di bit o caratteri).
  - Una **entità** è una risorsa generica.

  - Un **access point** è una entità speciale.
  - Un nome di un access point è chiamato **address**.

- Esempi: *telefono – numero,*
  canale – frequenza,
  router – indirizzo IP

Naming (3)

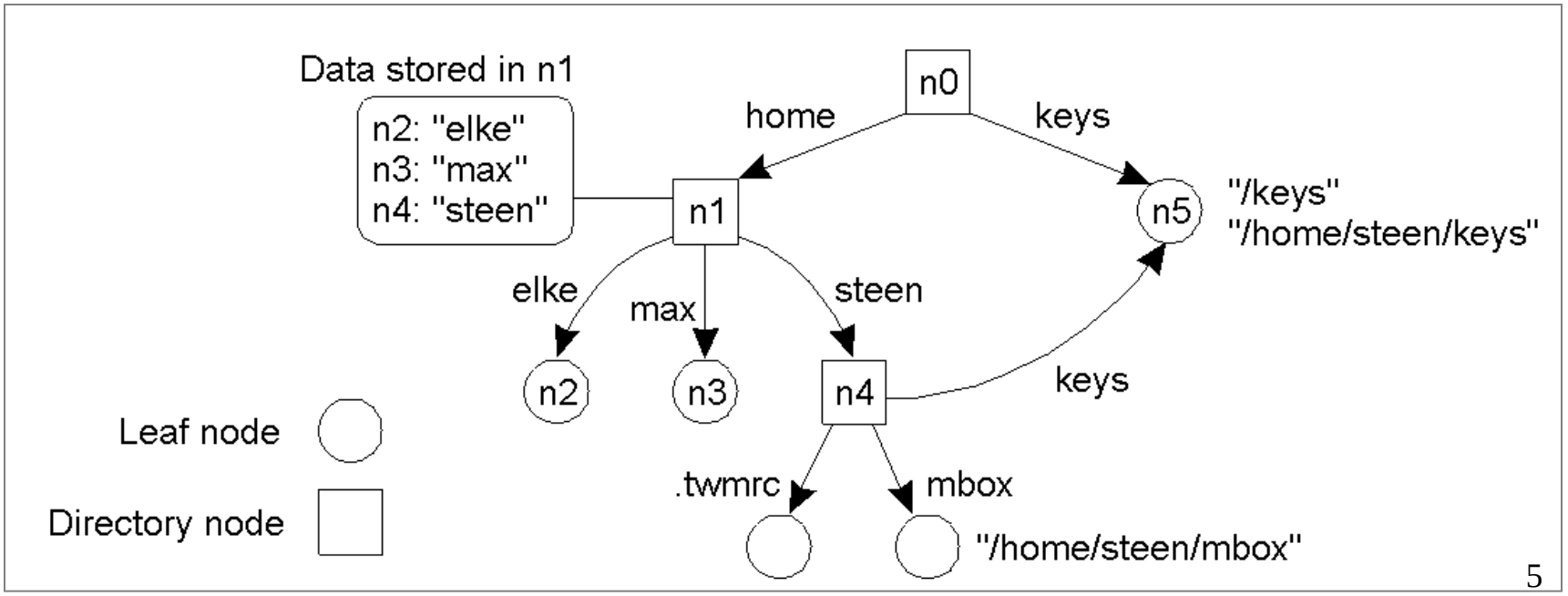
- Una entità può avere più di un access point.
- Un nome di entità può essere **indipendente dalla locazione**.
- Quando
  - Un nome si riferisce ad una sola entità,
  - Ogni entità è riferita al più da un nome,
  - Un nome fa riferimento sempre ad una stessa entità (no riuso)

  Il nome è detto **identificatore** (**identifier**).

Name Spaces (1)

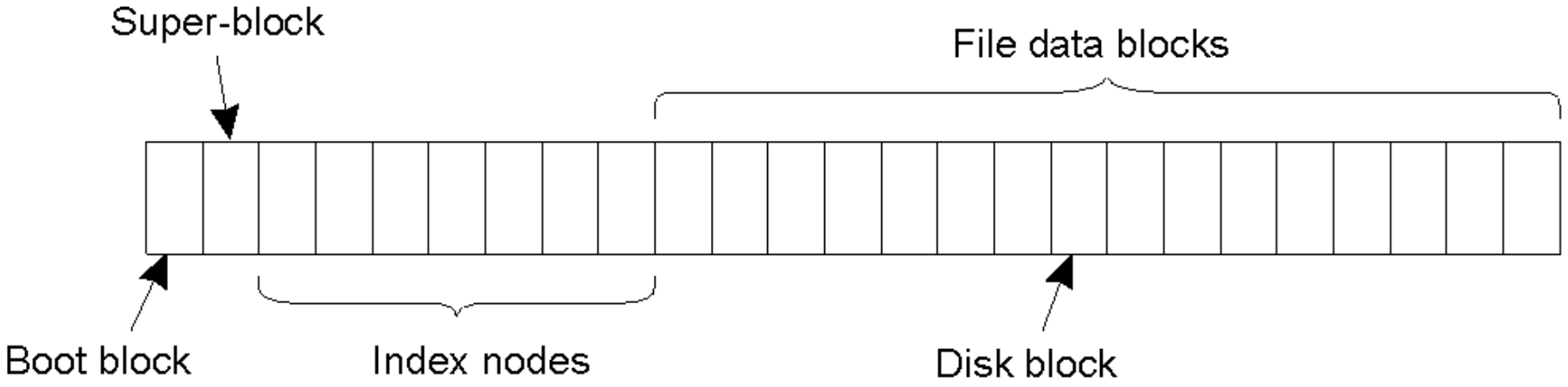
- Un **name space** è una struttura che organizza i nomi e definisce le relazioni tra essi (e quindi tra le entità).

Esempio 1: Un grafo di naming generale con un solo nodo radice.



Name Spaces (2)

Esempio 2: L'organizzazione generale della implementazione del file system di UNIX su un disco logico di blocchi contigui che include il blocco di boot, il superblocco, gli i-node e i blocchi dei dati.



Name resolution

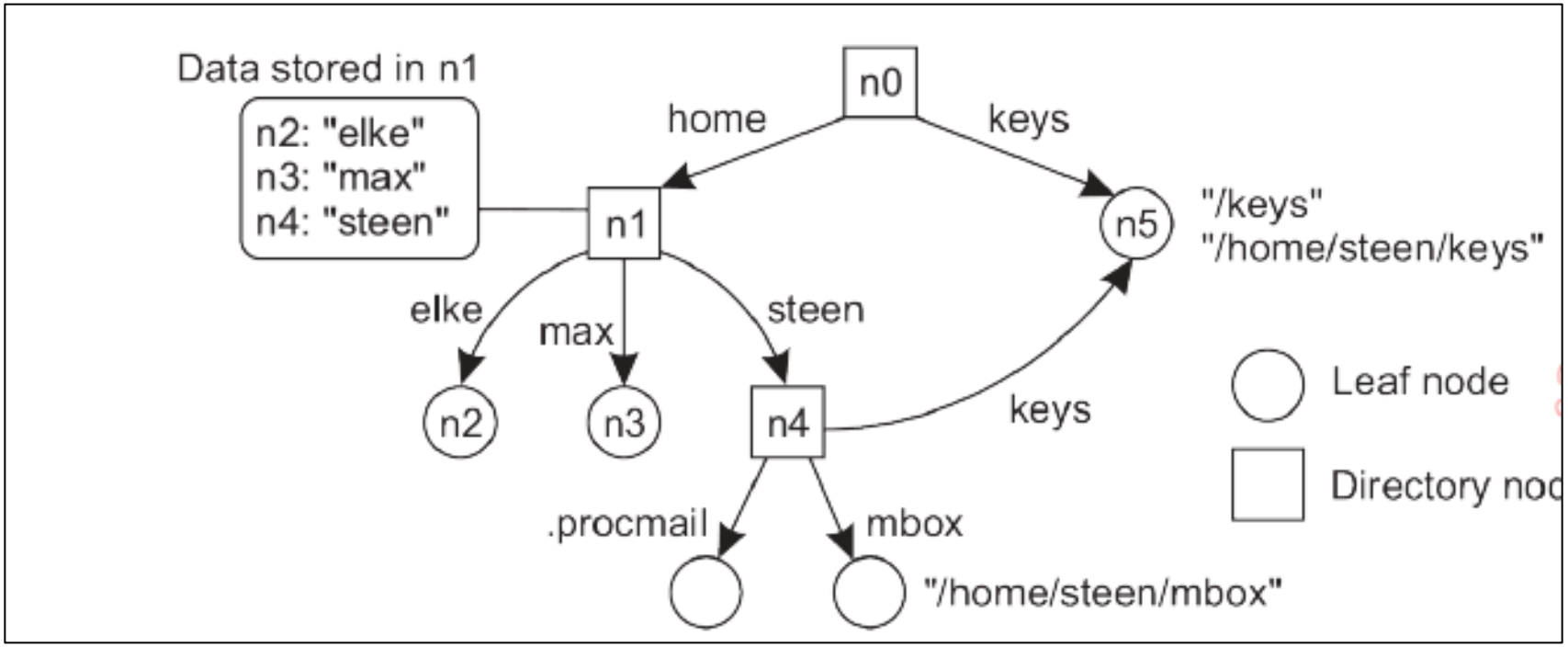
- Risoluzione dei nomi distribuita :

$$N:<l_1, l_2, \ldots, l_n>$$

- **Closure mechanism**: conosce da dove la risoluzione di un nome ha inizio.

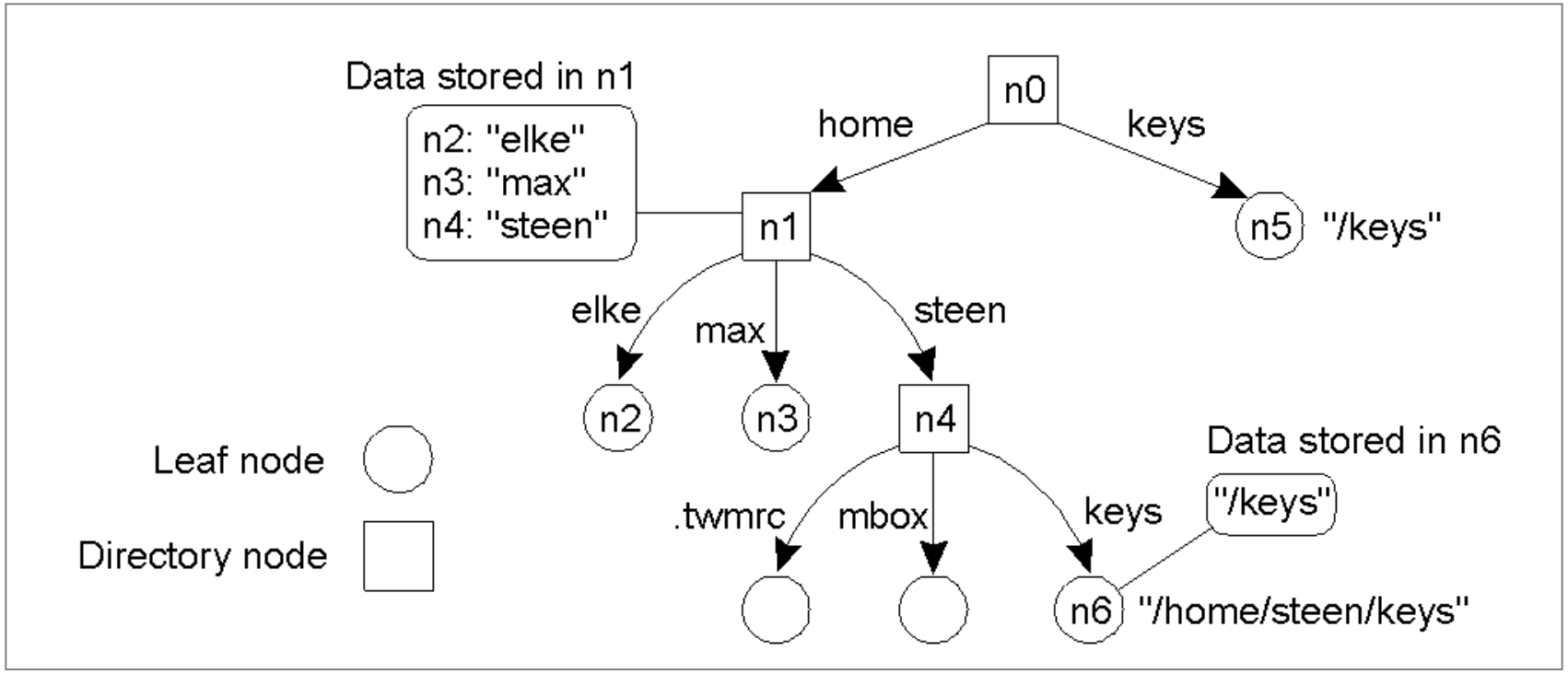
- Vengono usati gli alias (altri nomi per una entità):
  - hard links
  - symbolic links.

Linking e Mounting (1)



Il concetto di hard link spiegato in un grafo di naming:
(/keys e /home/steen/keys sono degli hard ink)

Linking e Mounting (1)



Il concetto di link simbolico spiegato in un grafo di naming:
(n6 è un link simbolico)

Linking e Mounting (2)

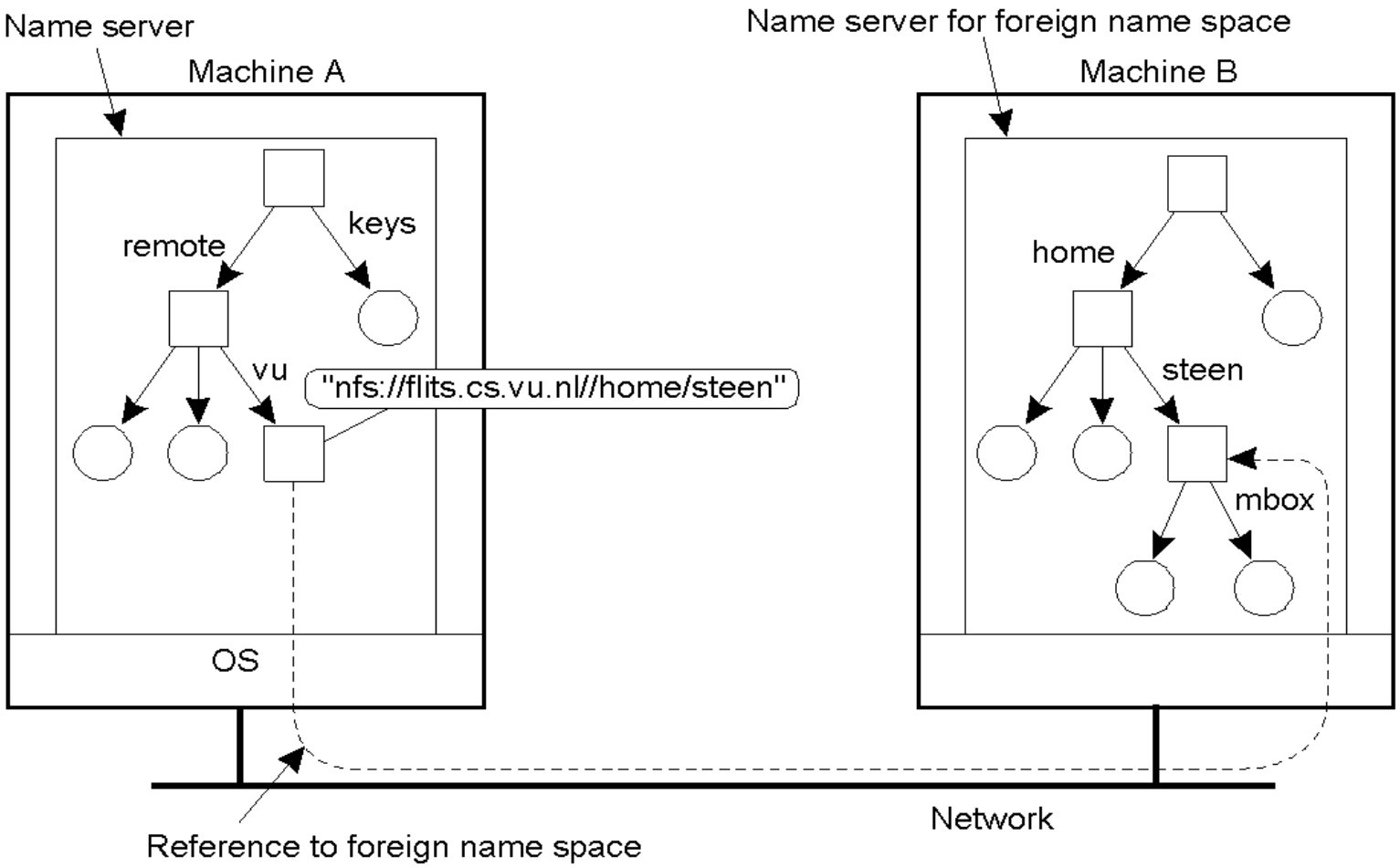
- Per il montaggio di un name space remoto in un sistema distribuito è necessario risolvere:

1. il nome del protocollo di accesso,
2. il nome del server,
3. il nome del punto di mounting nel name space remoto.

Esempi: nfs://flits.cs.vu.nl/home/steen

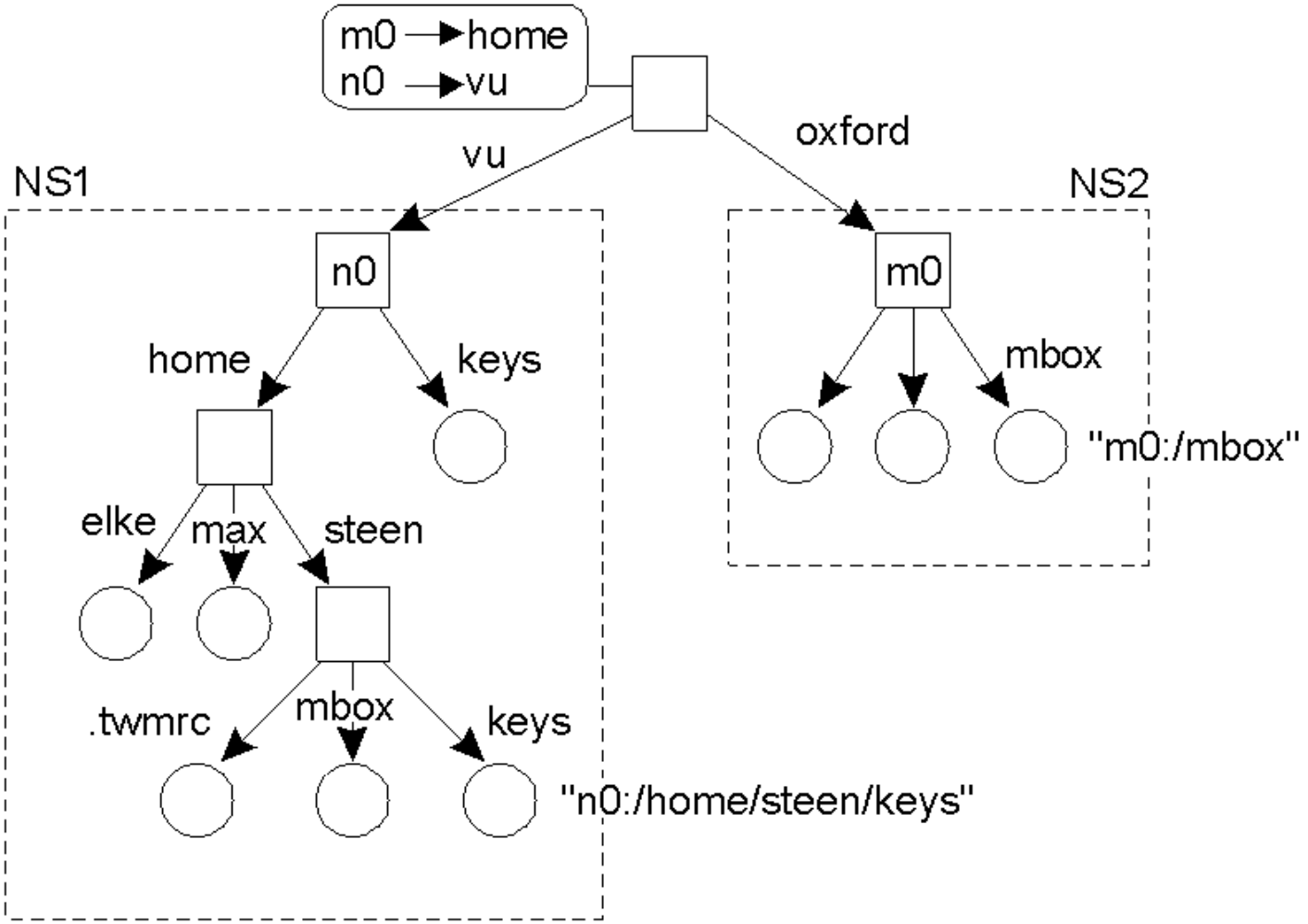
http://www.unical.it/didattica/

Linking e Mounting (3)



Mounting di un name space remoto attraverso un protocollo specifico.

Linking e Mounting (3)



Organizzazione del DEC Global Name Service

Name Space Distribuito

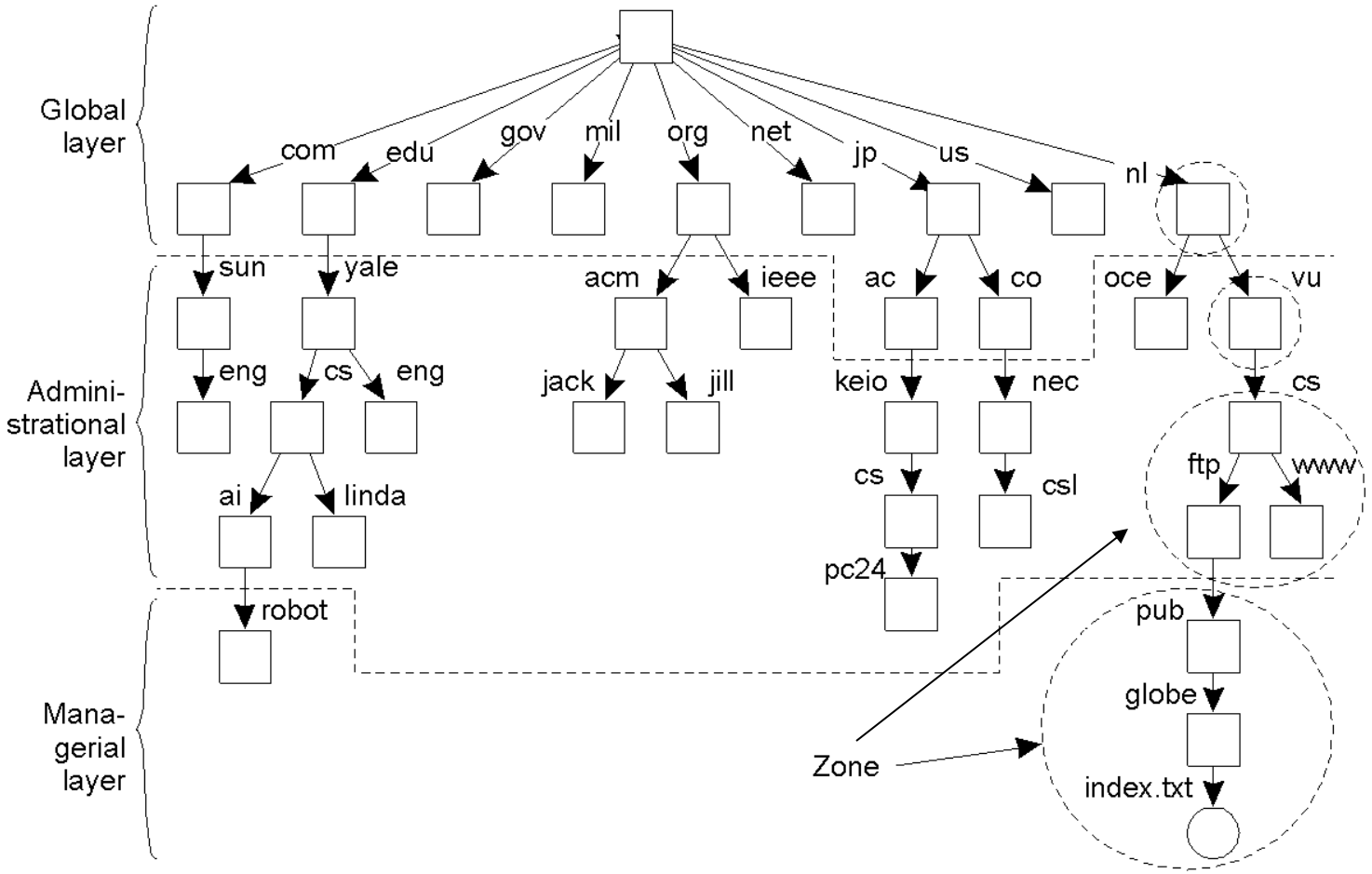
I sistemi distribuiti di grandi dimensioni usano **name server gerarchici**.

La replicazione dei name server può essere utile.

I name space possono essere suddivisi in più livelli logici:

- Livello globale,
- Livello di amministrazione,
- Livello di gestione.

Distribuzione del Name Space (1)



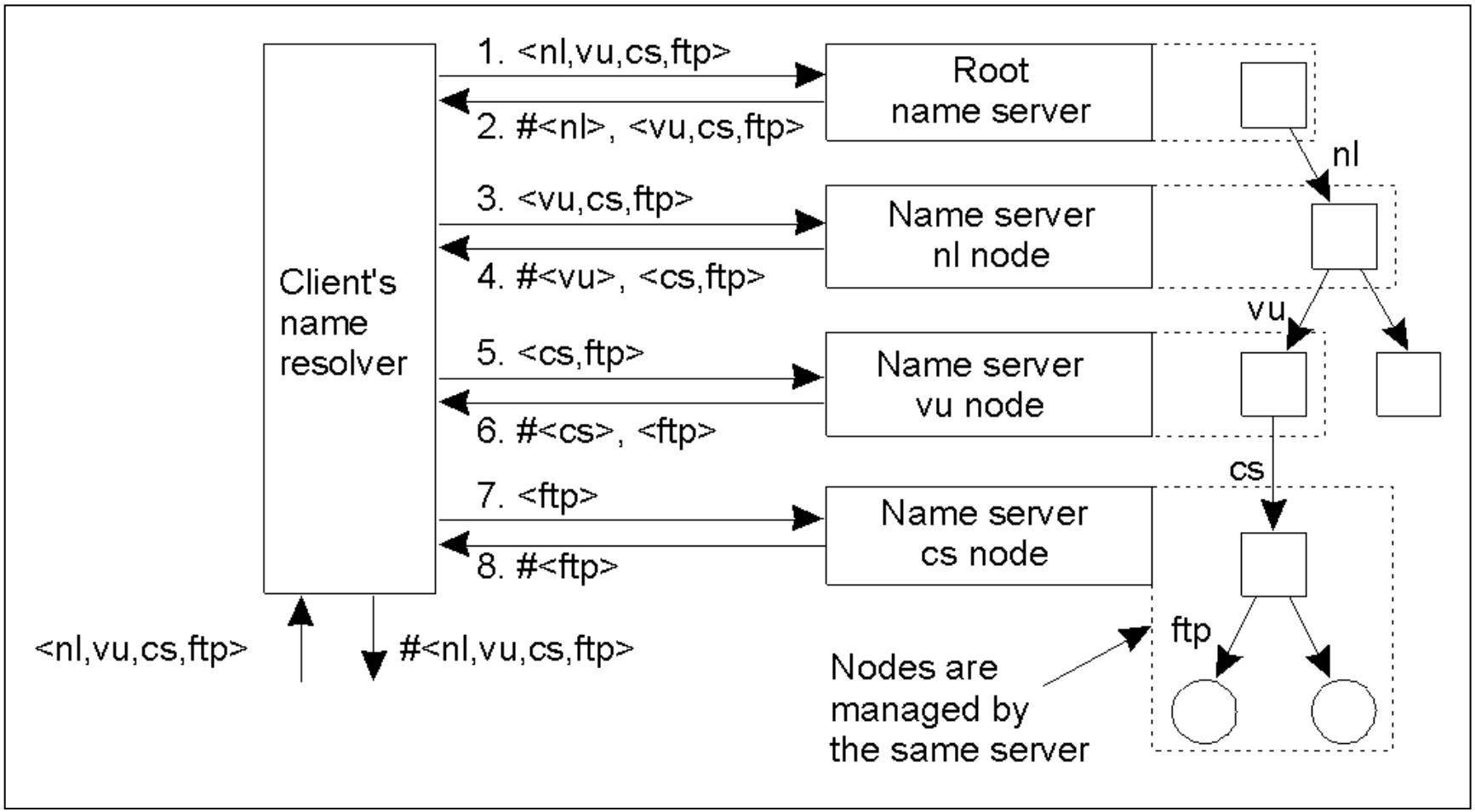
Un esempio di partizionamento del DNS name space a tre livelli che include file accessibili via Internet.

Distribuzione del Name Space (2)

	Globale	Amministrativo	Gestionale
Scala geografica della rete	Mondiale	Organizzazione	Dipartimento
Numero totale dei nodi	Pochi	Molti	Molti
Tempi di risposta	Secondi	Millisecondi	Immediati
Propagazione degli update	Lenta	Immediata	Immediata
Numero di repliche	Molte	Poche	Almeno una
E' usato caching nel client?	Si	Si	Talvolta

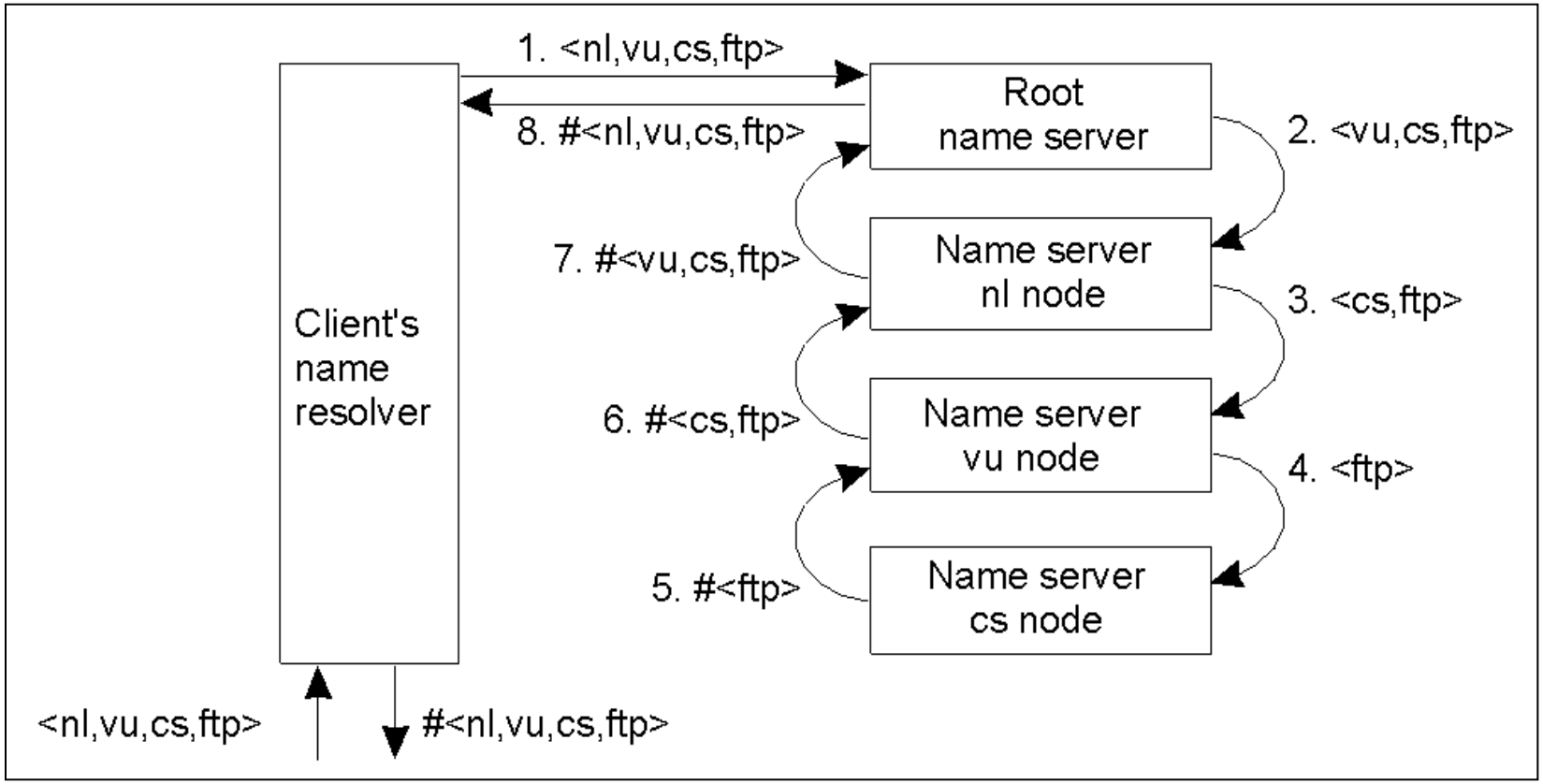
Un confronto tra le caratteristiche delle organizzazione dei name servers a diversi livelli di scala.

Name Resolution Iterativa



Lo schema di name resolution iterativa. Il cliente invia richieste ad ognuno dei name server. Ogni server svolge una parte della risoluzione e il client effettua le singole richieste.

Name Resolution Ricorsiva



Lo schema di name resolution ricorsiva. Il cliente invia richieste al name server radice. Ogni server svolge una parte della risoluzione e effettua le richieste ai server sottostanti.

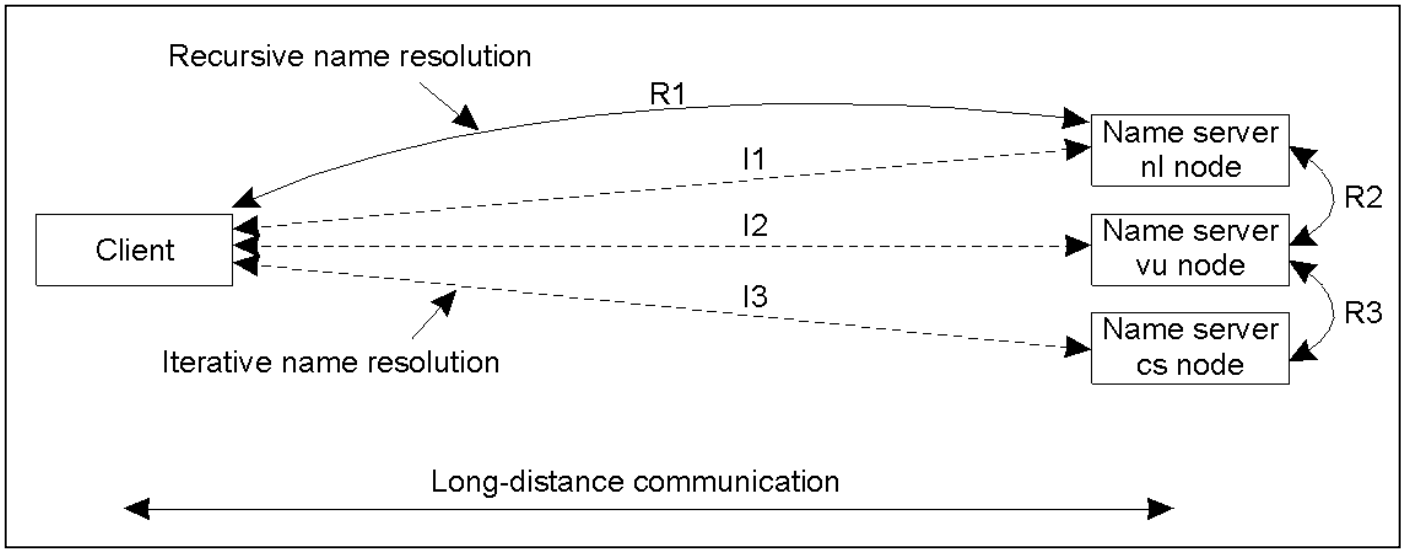
Name Resolution Ricorsiva

Server per nodo	Deve risolvere	Looks up	Passa al figlio	Riceve e memorizza	Ritorna al richiedente
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	<ni,vu,cs,ftp>	#<nl>	<vu,cs,ftp>	#<vu> #<vu,cs> #<vu,cs,ftp>	#<nl> #<nl,vu> #<nl,vu,cs> #<nl,vu,cs,ftp>

Risoluzione dei nomi ricorsiva di<*nl, vu, cs, ftp*>. I name server memorizzano nella cache i risultati intermedi per accessi successivi.

Implementazione della Risoluzione dei Nomi

Confronto tra name resolution ricorsiva e iterativa in relazione ai costi di comunicazione



Aspetti positivi della risoluzione ricorsiva:

- Minori costi di comunicazione
- Benefici dal caching dei nomi nei server

Il Name Space del DNS

- Il Domain Name System (**DNS**) di Internet è il più grande (?) servizio di naming (distribuito) esistente oggi.
- Il DNS ha una struttura gerarchica. Le risorse sono identificate da **pathname** che sono composti da **etichette**.
- Ogni etichetta è composta al massimo da 63 caratteri e un pathname da 255 caratteri.
- Un sottoalbero è chiamato **domain** e la sua radice **domain name**.
- Ogni nodo contiene una sequenza di **resource records**.

Il Name Space del DNS

I tipi più importanti di record delle risorse formano i contenuti dei nodi nel name space del DNS.

Tipo di record	Entità Associata	Descrizione
SOA	Zona	Contiene informazioni sulla zona rappresentata
A	Host	Contiene un IP address dell'host che questo nodo rappresenta
MX	Dominio	Indica un mail server per gestire gli indirizzi di email del nodo
SRV	Dominio	Indica un server per gestire un servizio specifico del nodo
NS	Zona	Indica un name server che implementa la zona interessata
CNAME	Nodo	Link Simbolico con il nome primario della zona rappresentata
PTR	Host	Contiene il nome canonico dell'host
HINFO	Host	Contiene infomazioni sugli host che il nodo rappresenta
TXT	Ogni tipo	Contiene informazioni ritenute utili sull'entità

Implementazione del DNS (1)

Un estratto del database del DNS per la zona *cs.vu.nl*

Name	Record type	Record value
cs.vu.nl	SOA	star (1999121502,7200,3600,2419200,86400)
cs.vu.nl	NS	star.cs.vu.nl
cs.vu.nl	NS	top.cs.vu.nl
cs.vu.nl	NS	solo.cs.vu.nl
cs.vu.nl	TXT	"Vrije Universiteit - Math. & Comp. Sc."
cs.vu.nl	MX	1 zephyr.cs.vu.nl
cs.vu.nl	MX	2 tornado.cs.vu.nl
cs.vu.nl	MX	3 star.cs.vu.nl
star.cs.vu.nl	HINFO	Sun Unix
star.cs.vu.nl	MX	1 star.cs.vu.nl
star.cs.vu.nl	MX	10 zephyr.cs.vu.nl
star.cs.vu.nl	A	130.37.24.6
star.cs.vu.nl	A	192.31.231.42
zephyr.cs.vu.nl	HINFO	Sun Unix
zephyr.cs.vu.nl	MX	1 zephyr.cs.vu.nl
zephyr.cs.vu.nl	MX	2 tornado.cs.vu.nl
zephyr.cs.vu.nl	A	192.31.231.66
www.cs.vu.nl	CNAME	soling.cs.vu.nl
ftp.cs.vu.nl	CNAME	soling.cs.vu.nl
soling.cs.vu.nl	HINFO	Sun Unix
soling.cs.vu.nl	MX	1 soling.cs.vu.nl
soling.cs.vu.nl	MX	10 zephyr.cs.vu.nl
soling.cs.vu.nl	A	130.37.24.11
laser.cs.vu.nl	HINFO	PC MS-DOS
laser.cs.vu.nl	A	130.37.30.32
vucs-das.cs.vu.nl	PTR	0.26.37.130.in-addr.arpa
vucs-das.cs.vu.nl	A	130.37.26.0

DNS Implementation (2)

Parte della descrizione per il dominio *vu.nl* che contiene il dominio *cs.vu.nl*

Name	Record type	Record value
cs.vu.nl.	NS	solo.cs.vu.nl.
cs.vu.nl.	NS	star.cs.vu.nl.
cs.vu.nl.	NS	ns.vu.nl.
cs.vu.nl.	NS	top.cs.vu.nl.
ns.vu.nl.	A	130.37.129.4
top.cs.vu.nl.	A	130.37.20.4
solo.cs.vu.nl.	A	130.37.20.5
star.cs.vu.nl.	A	130.37.24.6
star.cs.vu.nl.	A	192.31.231.42

Naming e Localizzazione di Entità

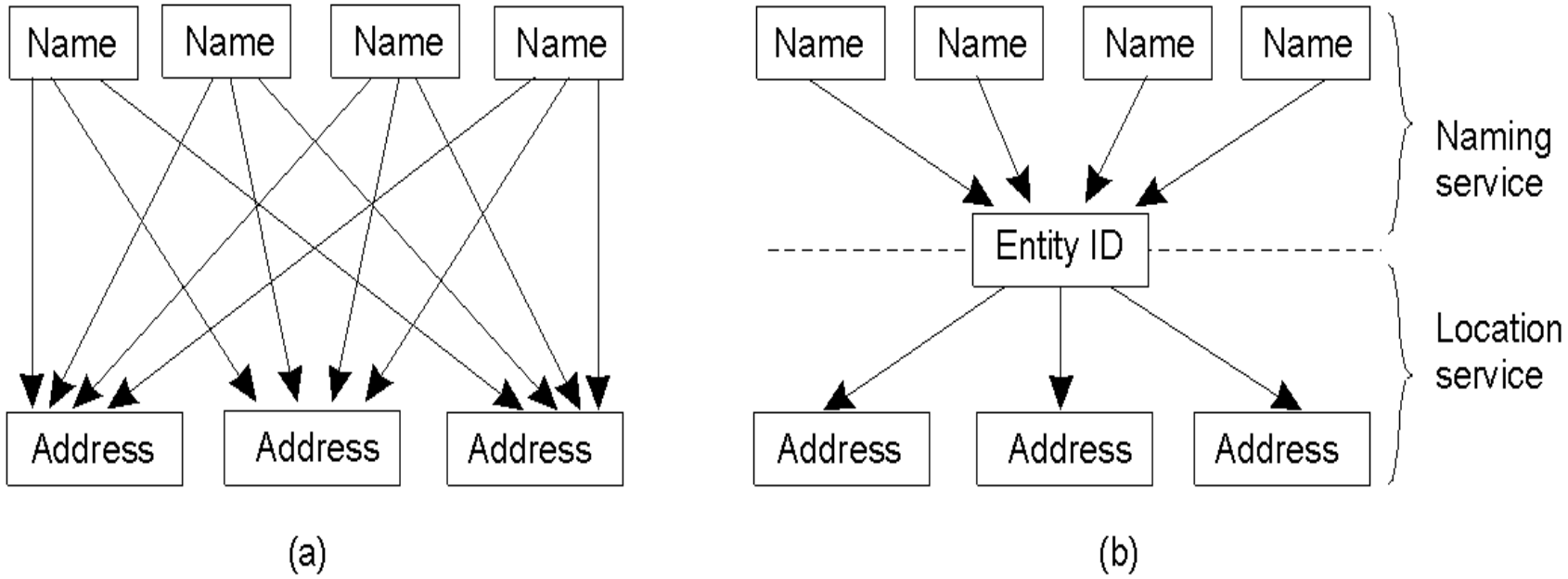
- Come gestire lo spostamento dei server in domini differenti?

  a) Memorizzare l'indirizzo della nuova macchina nel DNS entry della vecchia macchina.

  b) Memorizzare il nome della nuova macchina nel DNS entry della vecchia macchina.

- Un look up a più passi (multi-step) è necessario.

Naming e Localizzazione di Entità



a) Mapping diretto, singolo livello tra nomi e indirizzi
   - Non adatto a risorse mobili

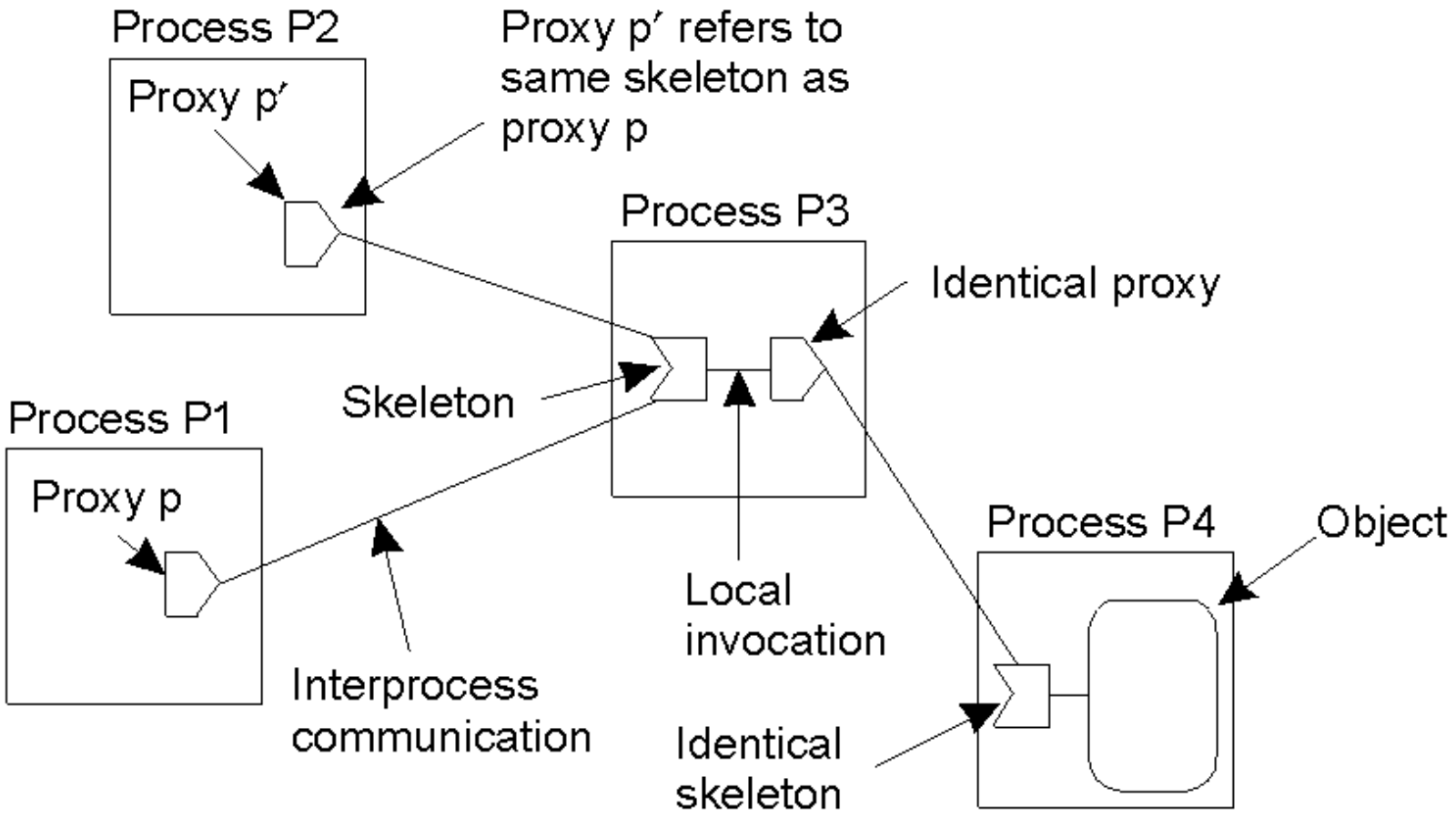
b) Mapping a due livelli usando identità
   - Flessibile e adatto a gestire risorse mobili

Localizzazione di entità: Broadcasting e Multicasting

- In una LAN con pochi nodi può essere usato il meccanismo di broadcasting.
  - Un identificatore di entità è inviato ad ogni macchina chiedendo il controllo del proprietario dell'entità.
  - Esempio: in ***ARP*** (***address resolution protocol***) si usa una broadcast di un IP per cercare un indirizzo data link di un computer.

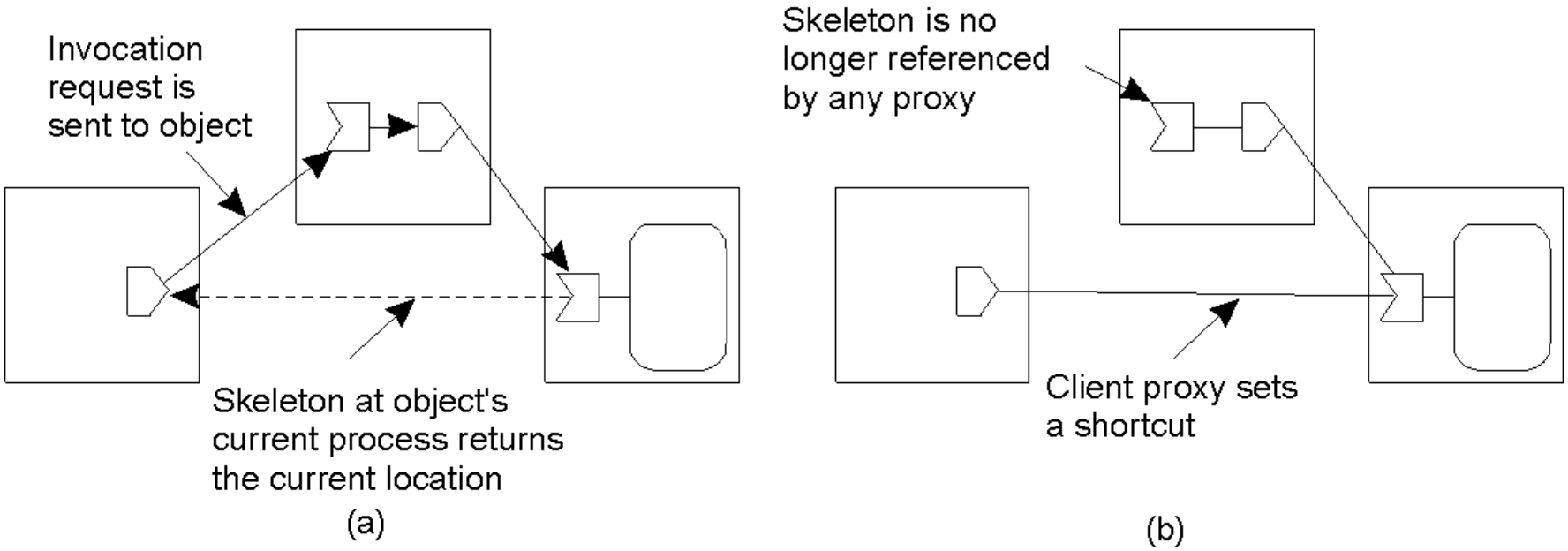
- Quando il numero dei nodi è elevato può essere usato il meccanismo di multicasting.
  - Si definiscono dei gruppi di nodi e si inviano le richieste a tutti i nodi di un gruppo.

Localizzazione di entità: Forwarding Pointers (1)



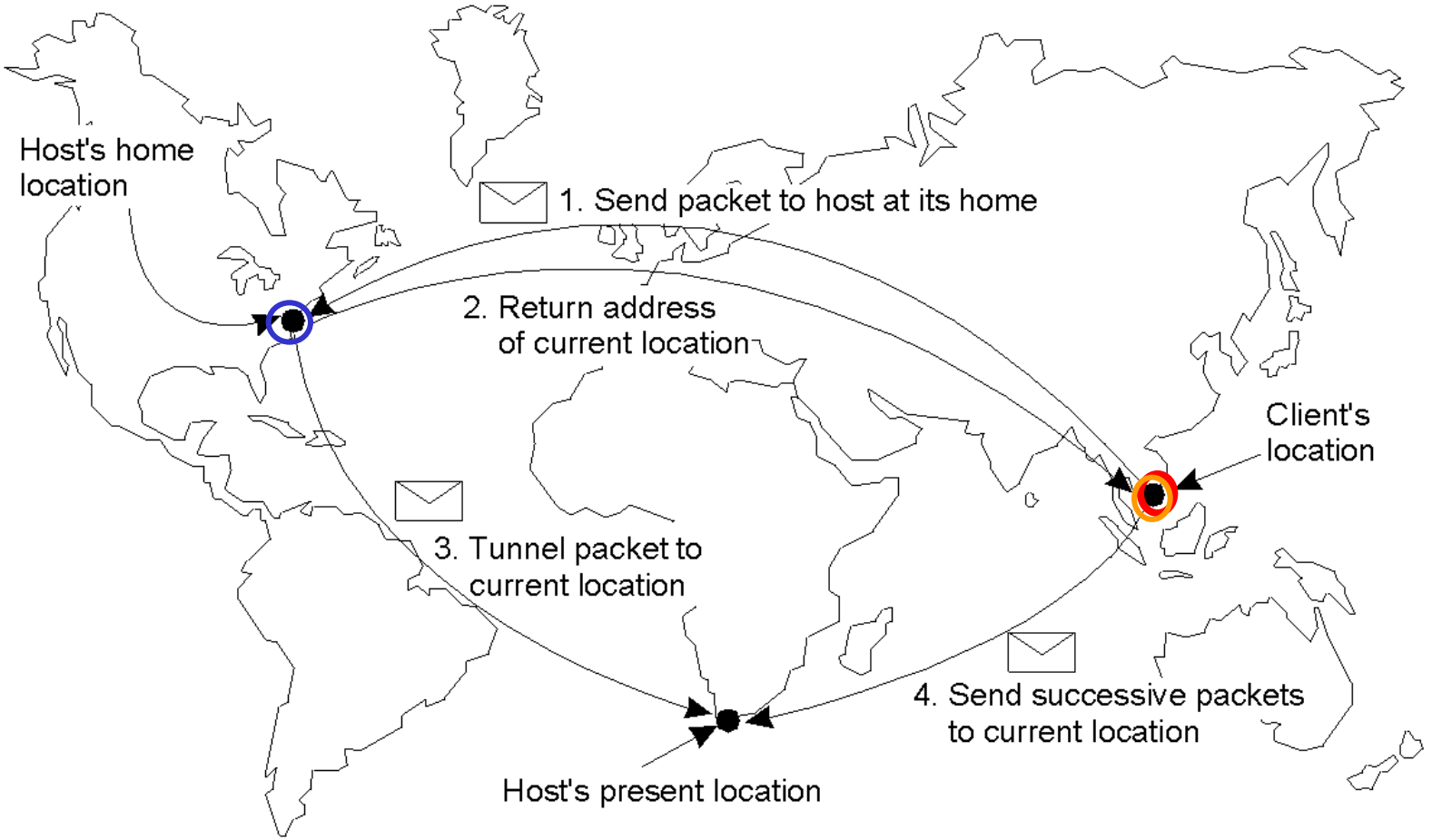
Il modello dei forwarding pointers usando le coppie (*proxy, skeleton*) o (*client stub, server stub*) per raggiungere una entità.

Localizzazione di entità: Forwarding Pointers (2)



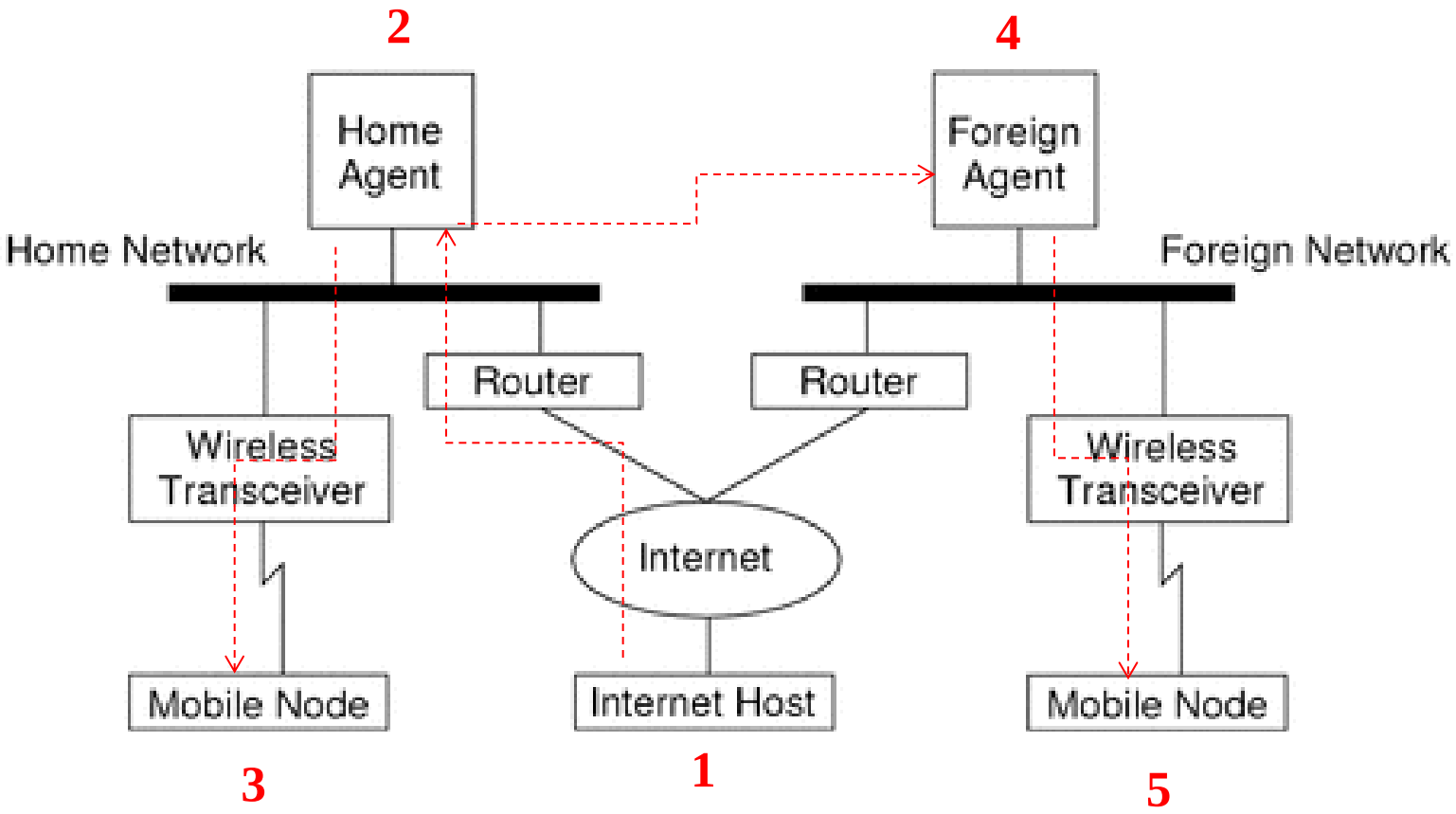
Redirezione di un forwarding pointer tramite la memorizzazione di un cammino in un proxy per ridurre le comunicazioni.

Localizzazione di entità: Approcci Home-Based



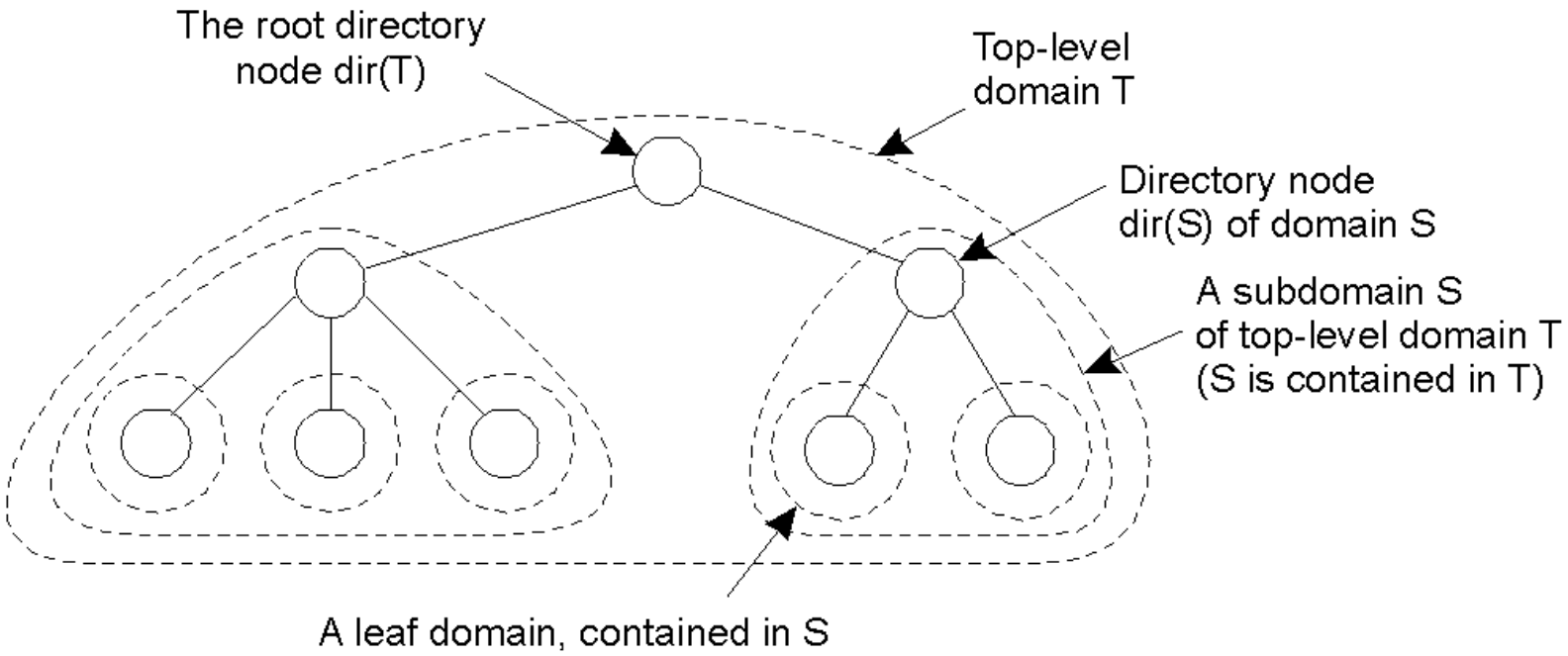
Il principio del Mobile IP (home location e current location).

Localizzazione di entità: Approcci Home-Based



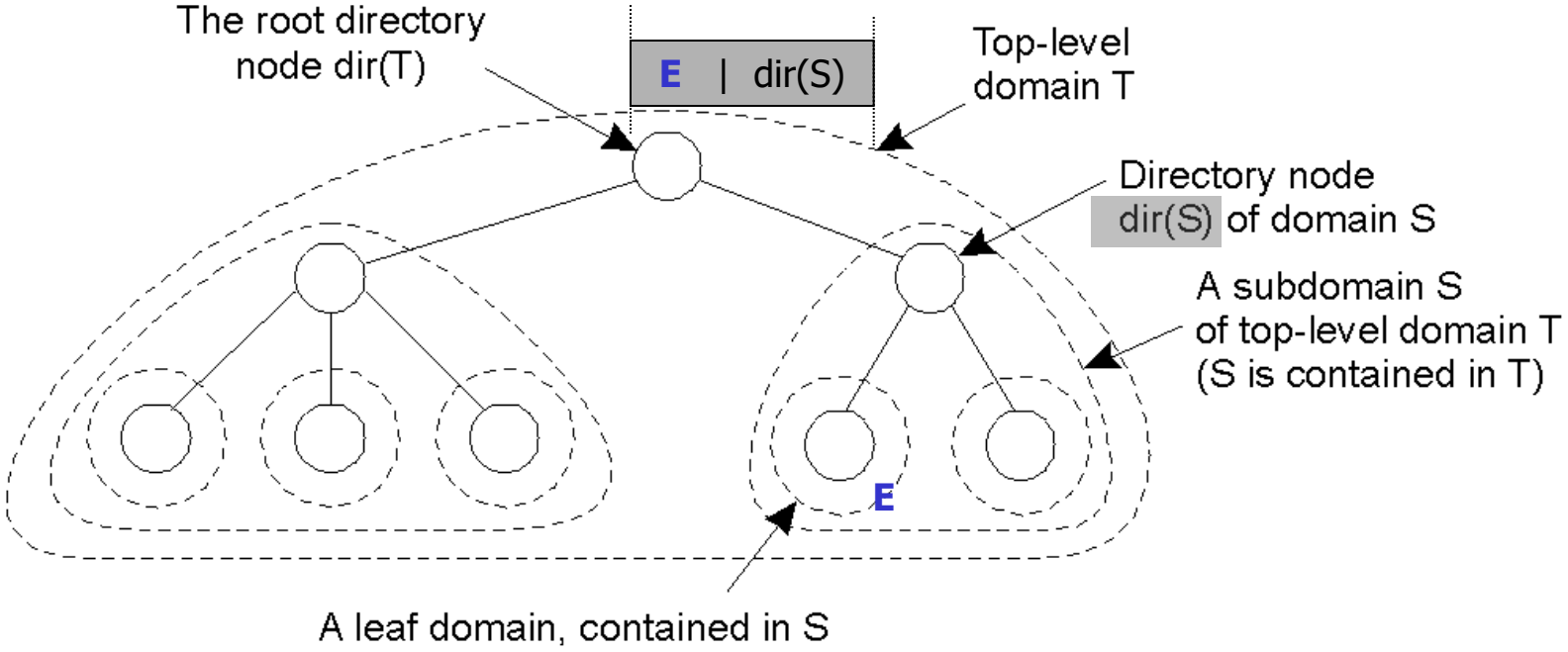
Topologia e componenti del Mobile IP.

Approcci Gerarchici (1)



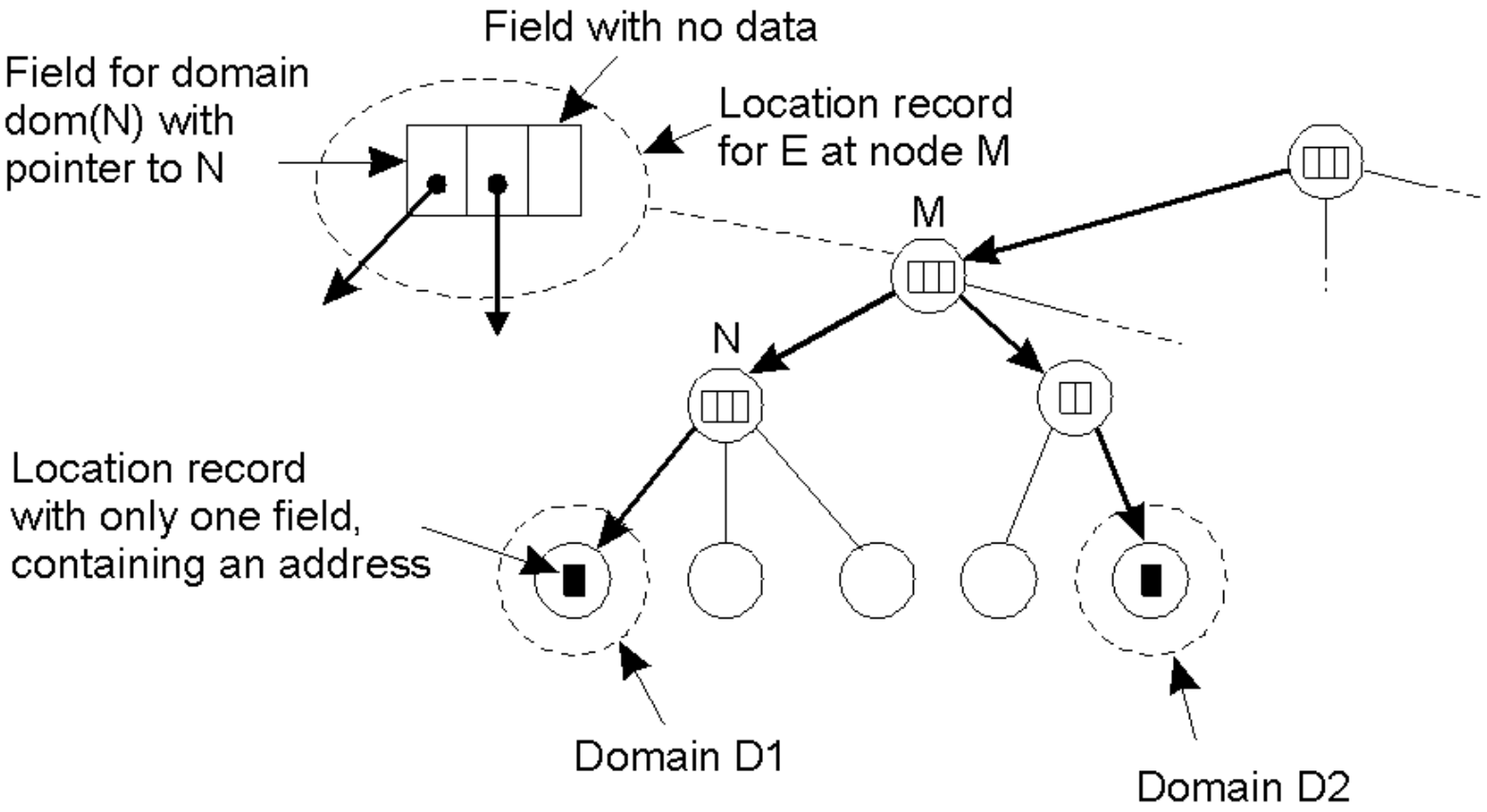
- Organizzazione gerarchica di un location service in domini, ognuno avente un directory node associato.
- Una entità in **D** è identificata da un location record in **dir(D)**.

Approcci Gerarchici (2)



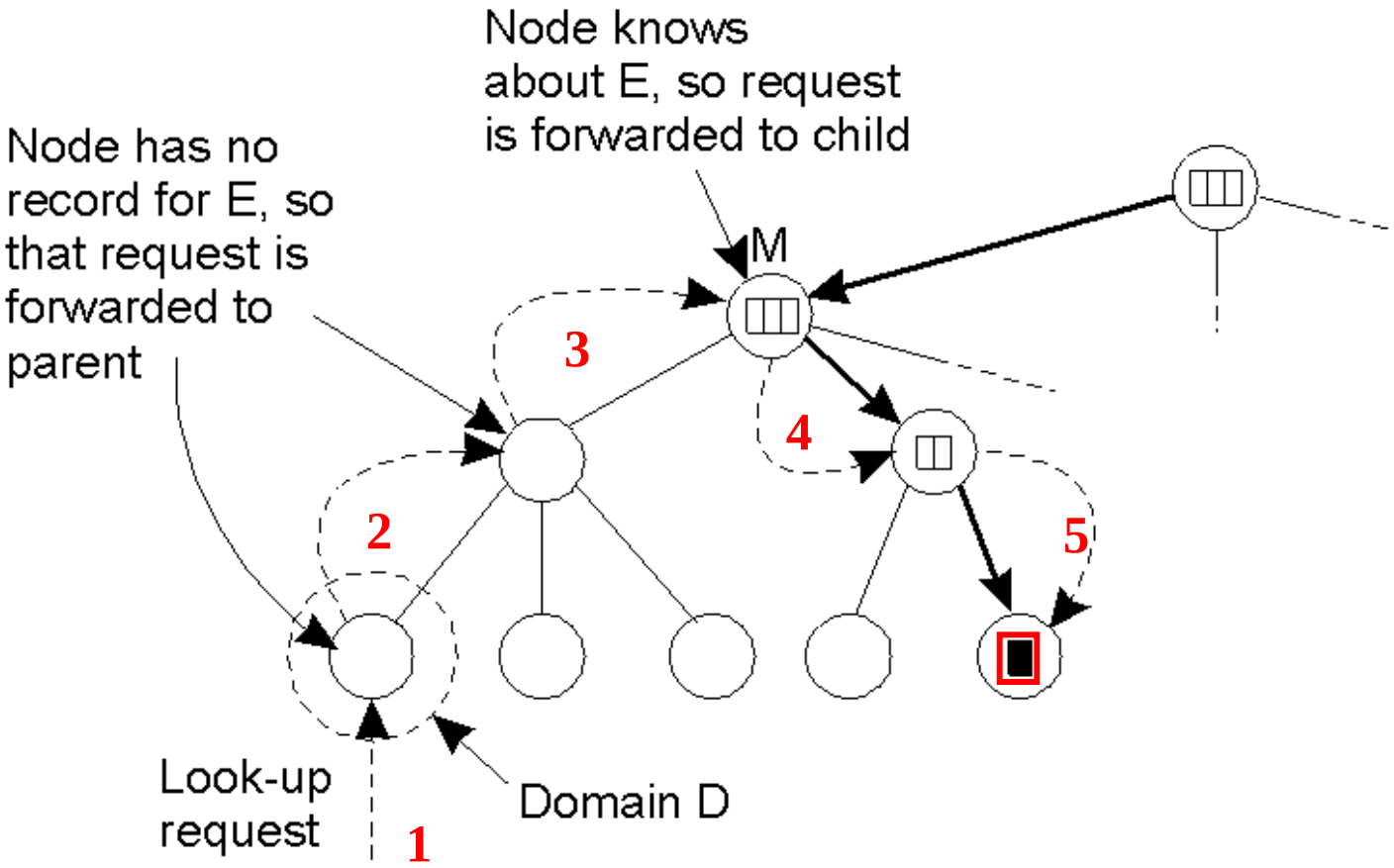
- Un nodo radice di un sotto-albero contiene una entry per ogni entità **E**.
- Il location record contiene un puntatore al directory node del successivo sotto-dominio del livello più basso che contiene l'entità **E.**

Approcci Gerarchici (3): Replicazione di entità



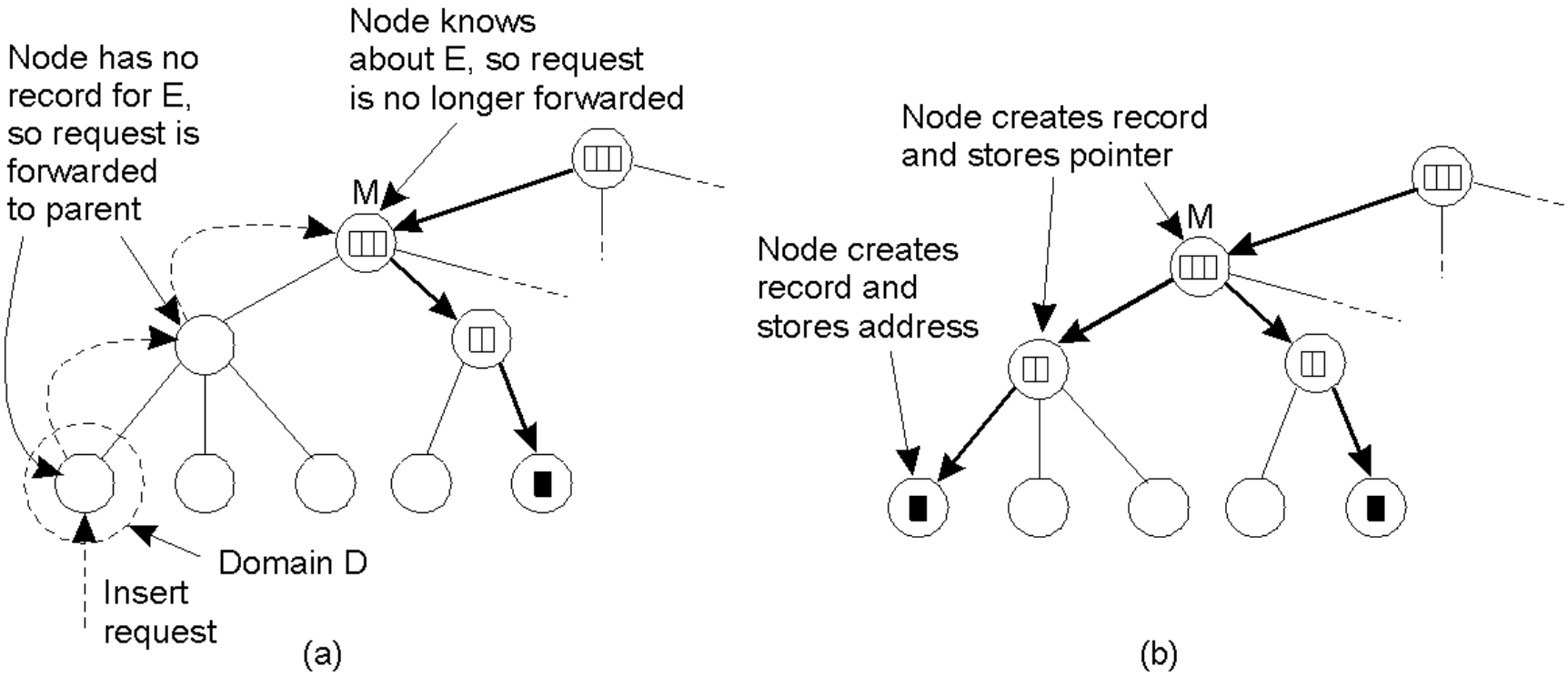
Un esempio di memorizzazione di informazione di una entità replicata che ha due indirizzi in differenti domini foglie.

Approcci Gerarchici (4): richiesta di accesso



Accesso ad una locazione in un location service organizzato gerarchicamente.

Approcci Gerarchici (5): richiesta di inserimento



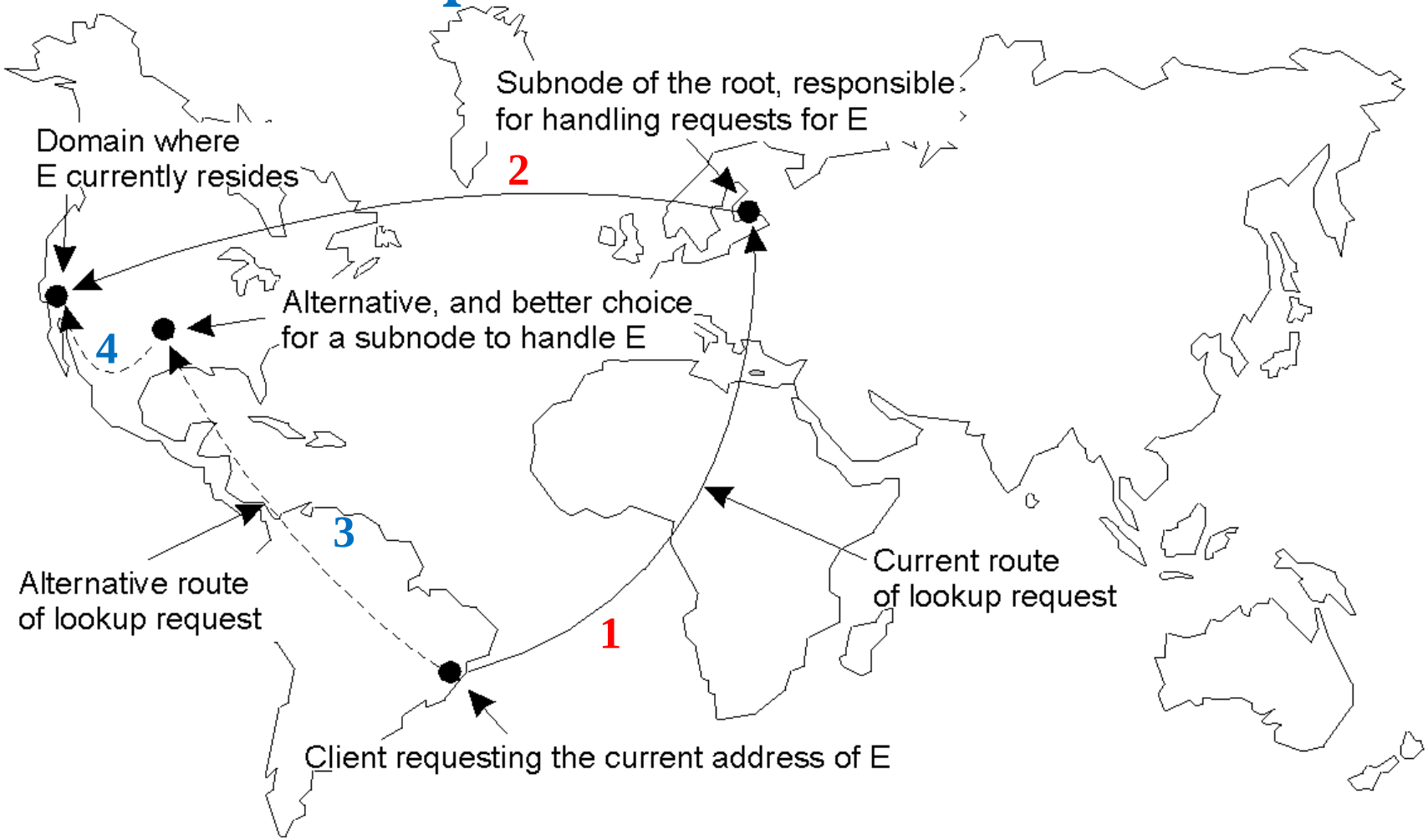
a) Una ***insert*** *request* è inviata al primo nodo che conosce l'entità *E*.

b) Viene creata una catena di forwarding pointers fino al nodo foglia.

Aspetti di Scalabilità

- In un servizio di locazione gerarchico il nodo radice deve memorizzare le ***entry per tutte le entità***.
- Il nodo radice ***può diventare il collo di bottiglia*** del sistema.
- Può essere partizionato in un insieme di nodi che gestiscono un sottoinsieme di entità (può essere replicato).
- Trovare il modo migliore per localizzare i nodi è molto complesso.

Aspetti di Scalabilità



Problemi di scalabilità relativi alla distribuzione uniforme di sotto-nodi di un nodo radice partizionato: dove fare il lookup?

Sistemi di Naming P2P

- In una rete peer-to-peer i nodi sono organizzati in una ***overlay network*** che realizza *una rete logica su una rete fisica*.

- I nodi logicamente in connessione diretta tra loro (nella rete logica) non lo sono necessariamente a livello fisico (nella rete fisica).

- I messaggi scambiati tra due nodi possono attraversare altri nodi.

- Una rete P2P può essere ***strutturata*** o ***non-strutturata***. Nel primo caso i messaggi vengono instradati secondo la struttura logica della rete.

Sistemi di Naming P2P

- Mentre nelle reti P2P non-strutturate si usa la tecnica del ***flooding*** per raggiungere un(a) dato/risorsa, in quelle strutturate l'overlay è costruita secondo regole deterministiche e le ***distributed hash table*** (***DHT***) sono il modello più usato per organizzare la rete.

- Sia ai nodi sia ai dati viene assegnata una ***chiave*** (***identificatore***) per raggiungerli.

- La chiave di ogni dato permette di raggiungere il nodo che lo memorizza e quindi fornendo la chiave la funzione hash instrada la richiesta verso il nodo che possiede il dato cercato.

Distributed Hash Table (DHT)

- Le **tabelle di hash distribuite** (**distributed hash tables**, **DHTs**) sono una classe di strutture distribuite che partizionano l'appartenenza di un set di *chiavi* tra i nodi partecipanti.

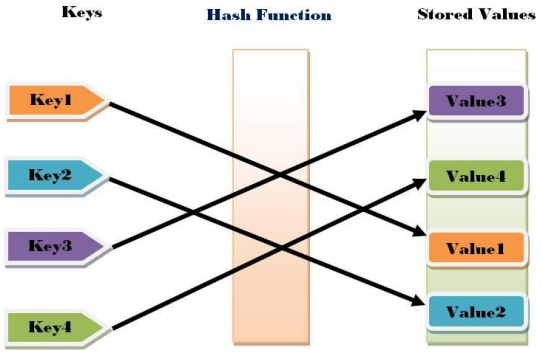
- Le DHT permettono di ricercare/inviare in maniera efficiente i dati/messaggi/entità raggiungendo il proprietario di una determinata chiave. Ciascun nodo contiene un array slot in una hash table (che quindi è distribuita).

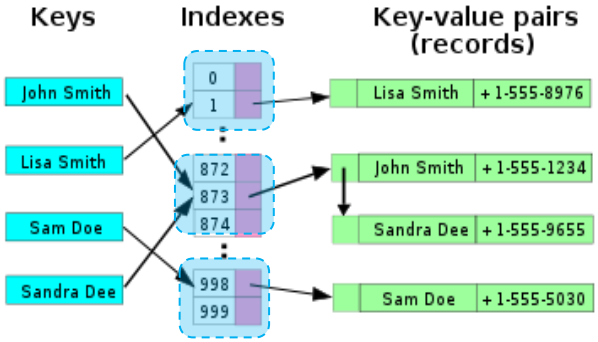
- Le DHT sono progettate per gestire un vasto numero di nodi, anche nei casi in cui ci siano continui ingressi o improvvisi guasti di alcuni di essi (sistemi P2P, Grid, Web caching).

Distributed Hash Table (DHT)

- Sistemi P2P basati su DHT sono ad esempio: CAN, **Chord**, Pastry, BitTorrent e Tapestry.

- Le DHT hanno le seguenti proprietà:
  - **Decentralizzazione**: i nodi formano il sistema senza alcun coordinamento centrale.
  - **Scalabilità**: il sistema è predisposto per un funzionamento efficiente anche con un numero elevatissimo di nodi.
  - **Tolleranza ai guasti**: il sistema risulta affidabile anche in presenza di nodi che entrano ed escono dalla rete (*churn*) o sono soggetti a malfunzionamenti.





DHT in Chord

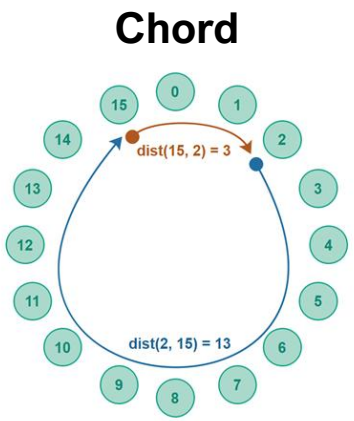
- Molte DHT usano una funzione

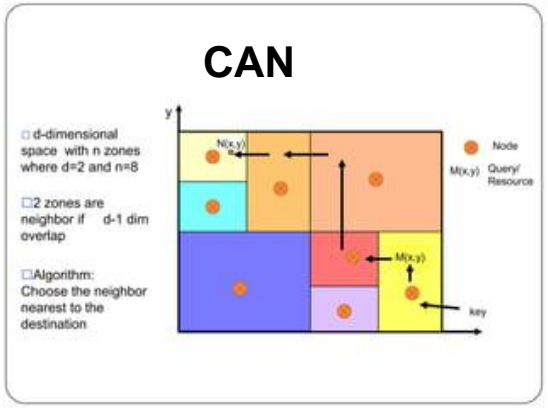
  $$\delta(k_1, k_2)$$

  che definisce la nozione astratta di *distanza* tra le chiavi k_1 e k_2.

- A ciascun nodo è assegnata una chiave che è detta ***identificatore*** (***id***).

- Ad un nodo con ***id*** uguale ad ***i*** appartengono tutte le chiavi per cui ***i*** è l'***id*** più vicino, misurando in base alla distanza **δ**.





DNS vs Chord

DNS

- Fornisce un mapping tra il nome di una macchina e il suo indirizzo IP
- Utilizza un certo numero di root server
- I nomi riflettono dei domini amministrativi
- È specializzato per ritrovare host e/o servizi a partire dai loro nomi

Chord

- Offre lo stesso servizio: Nome = *chiave*, IP = *valore*
- Non richiede server con ruoli particolari
- Non impone una struttura di naming
- Può essere usato per trovare dati e risorse che non sono legati a certe macchine

DHT in Chord

- L'algoritmo **Chord** considera le chiavi come punti su una circonferenza, e $\delta(k_1,k_2)$ è la distanza percorsa (in senso orario) sul cerchio tra k_1 e k_2.
- Perciò, lo spazio delle chiavi (**keyspace**) è circolare ed è diviso in segmenti contigui i cui punti terminali sono gli identificativi di nodo.
- Se i_1 e i_2 sono due **id** adiacenti, allora il nodo con id i_2 è proprietario di tutte le chiavi che cadono tra i_1 e i_2 **(Tutti i k tali che: $i_1 < k \leq i_2$)**.
- Ad esempio, se $i_1 =$ 7 e $i_2 =$ 12, le chiavi 8, 9, 10, 11 e 12 sono associate al nodo i_2 che è il loro proprietario.

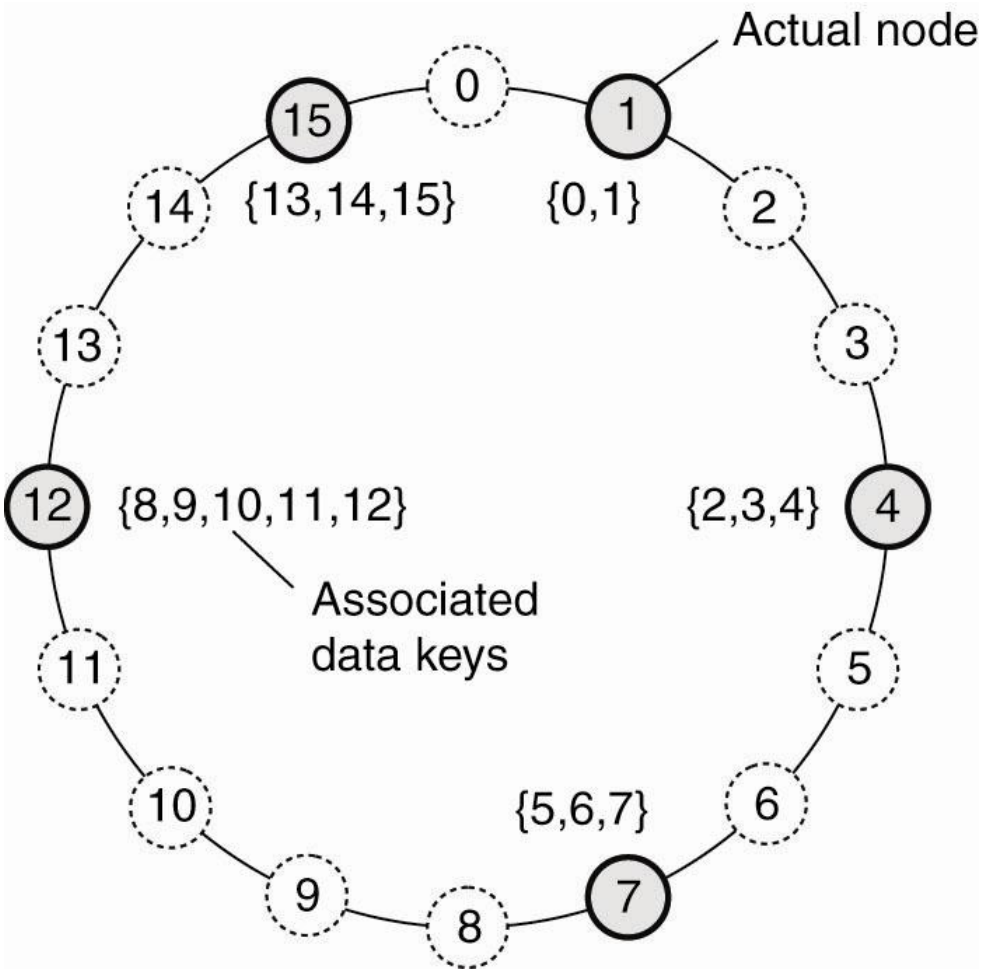
DHT in Chord

- Mapping di dati in nodi di una rete Chord.

Esempio di rete con 5 nodi e 11 dati/risorse.

Se i_1 e i_2 sono due id adiacenti, allora il nodo con id i_2 è proprietario di tutte le chiavi che cadono tra i_1 e i_2.

(Tutte le chiavi k tali che: $i_1 < k \leq i_2$).
i_2 è detto il successore di k: ***succ(k)***



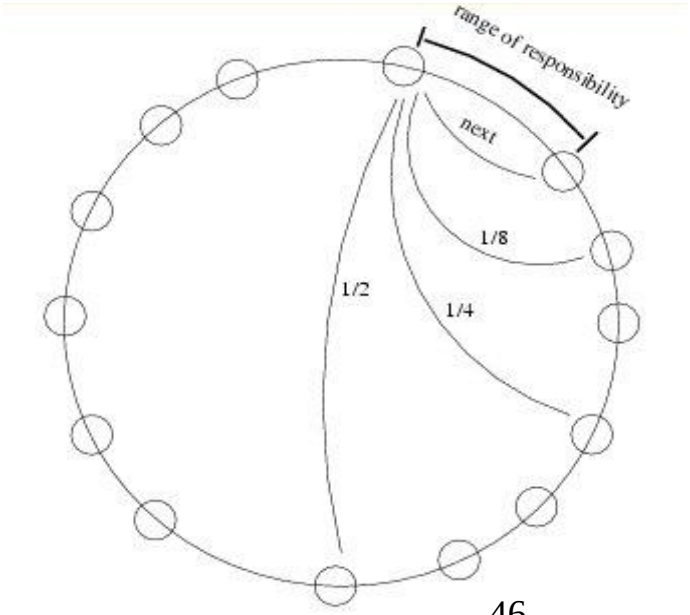
DHT in Chord

- ***N*** nodi con $N = 2^m$, ogni nodo è identificato con un *id* a ***m*** bit (Es: *N= 32, m=5*, oppure: *N= 64, m= 6,* oppure*: N=65536, m=16*).

- Ogni nodo ***p*** ha una **finger table *FT*** di ***m*** elementi che contiene solo gli indirizzi dei $p+2^{i-1}$ nodi; con ***i = 1, .., m***. (Quindi: ogni nodo punta ad altri ***m*** nodi).

- Ogni nodo memorizza informazioni su un numero limitato di altri nodi.

- La finger table di un nodo generalmente non contiene abbastanza informazioni per determinare **direttamente** il successore di una chiave arbitraria **k.**



DHT in Chord

- ***N*** nodi con $N = 2^m$, ogni nodo è identificato con un *id* a ***m*** bit (Es: *N= 32, m=5*, oppure: *N= 64, m= 6,* oppure*: N=65536, m=16*).

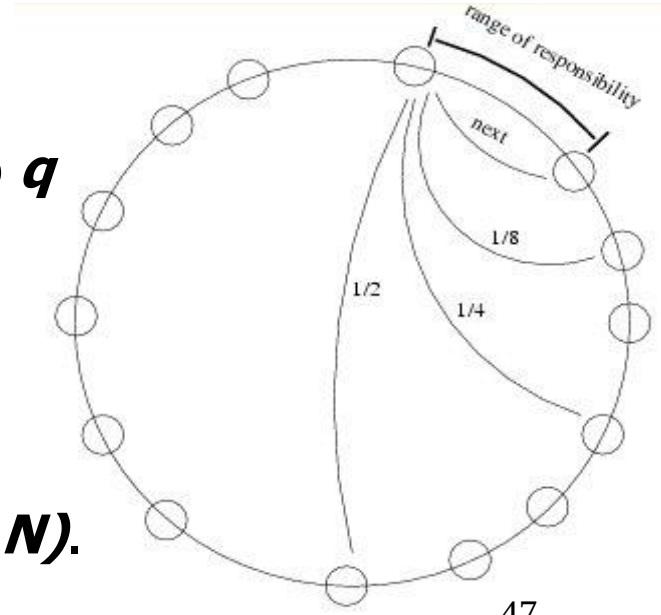
- Ogni nodo ***p*** ha una **finger table** ***FT*** di ***m*** elementi che contiene solo gli indirizzi dei $p+2^{i-1}$ nodi; con $i = 1, .., m$. (Ogni nodo punta ad altri ***m*** nodi)

$$FT_p[i]=succ(p+2^{i-1})$$

- Un nodo ***p*** invia una richiesta di ***k*** ad un nodo ***q*** con indice ***j*** nella finger table di ***p***

  dove: $q = FT_p[j] \leq k < FT_p[j+1]$

- ***N-1*** messaggi sono inviati in ***m*** passi: ***O(log N)***.



DHT in Chord

Risoluzione della chiave *k=26* dal nodo *p=1*

⟶

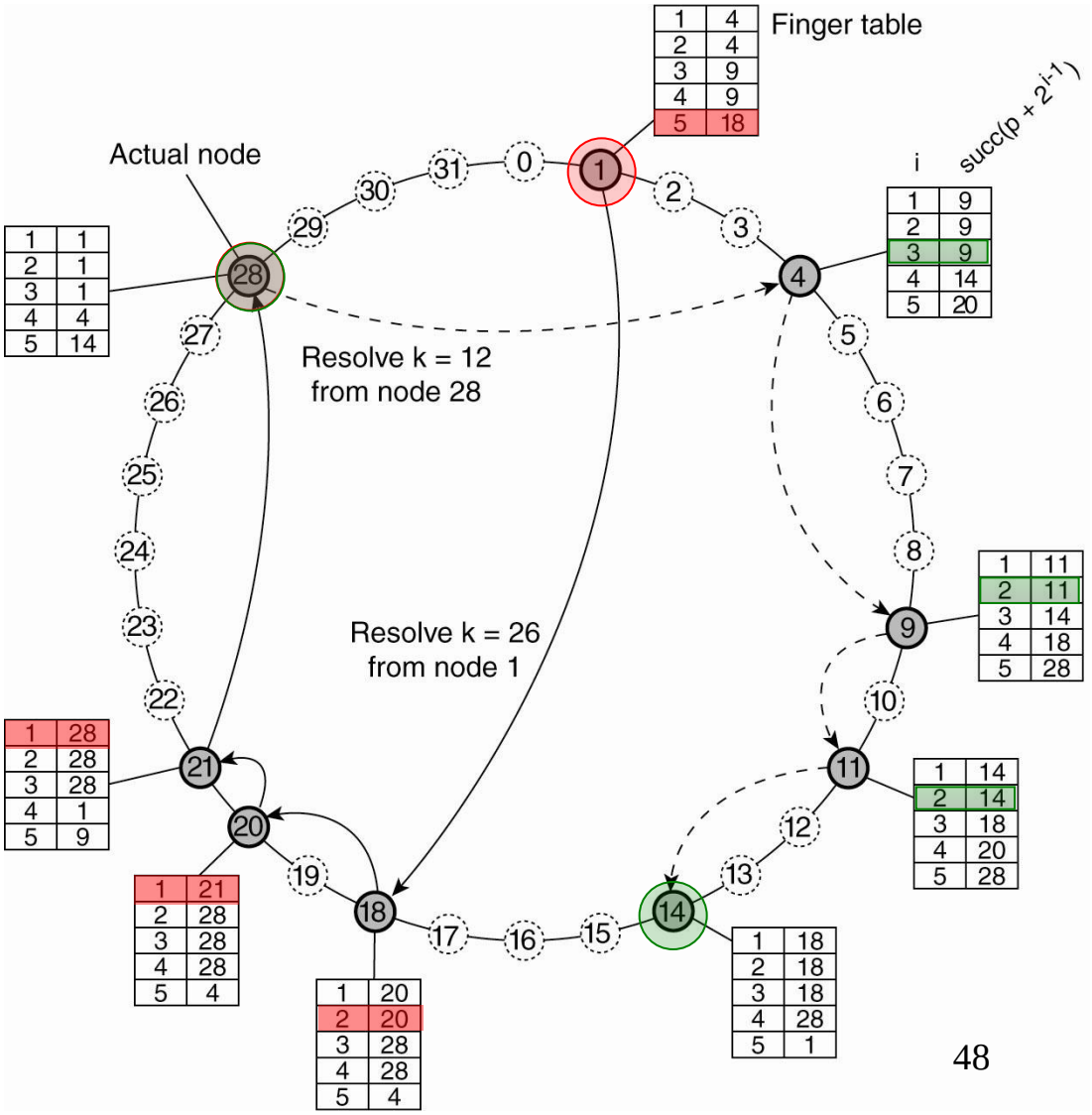
$q = FT_p[j] \leq 26$

e

della chiave *k=12* dal nodo *p=28*

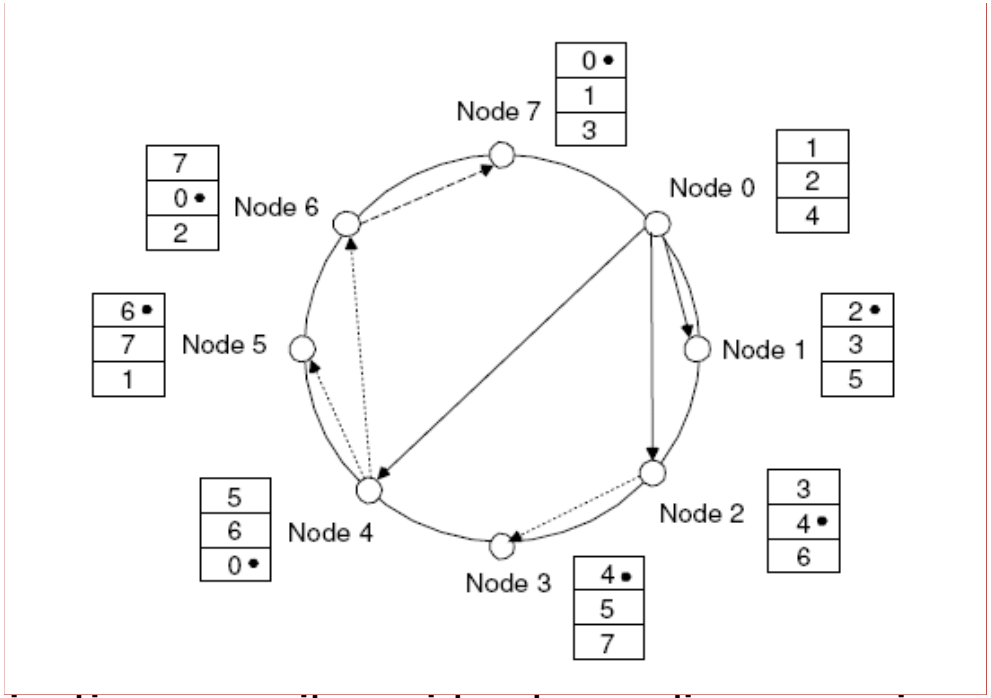
⇢

$q = FT_p[j] \leq 12$



Broadcast con DHT

- ***Comunicazione broadcast su una DHT***: ogni nodo contatta un certo numero di nodi tra quelli contenuti nella propria DHT.



- Informazione aggiuntiva per evitare ridondanze di messaggi: valore **limite** inviato nel messaggio.
- Il limite inviato nel messaggio per il nodo puntato dal finger *i* è il finger *i* + 1 (del mittente).