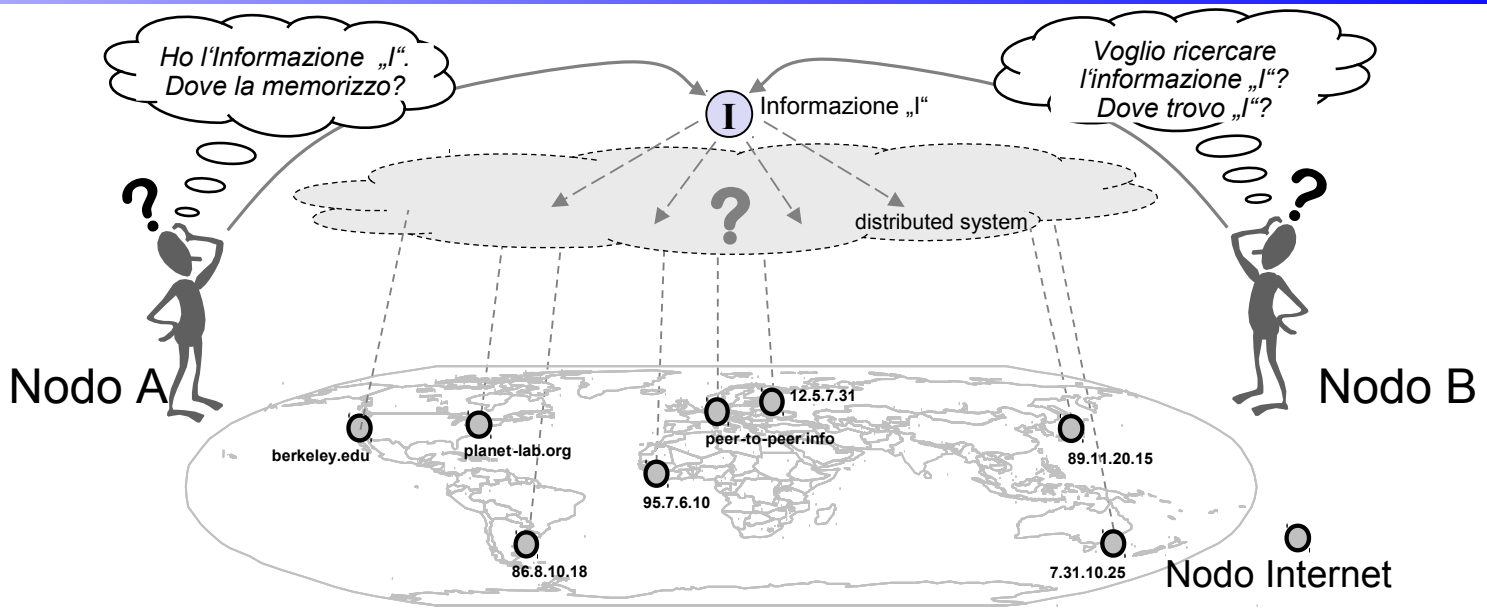
Lezione n.5
DISTRIBUTED HASH TABLES: INTRODUZIONE

Laura Ricci
6/3/2013

INDICE DELLE PROSSIME LEZIONI

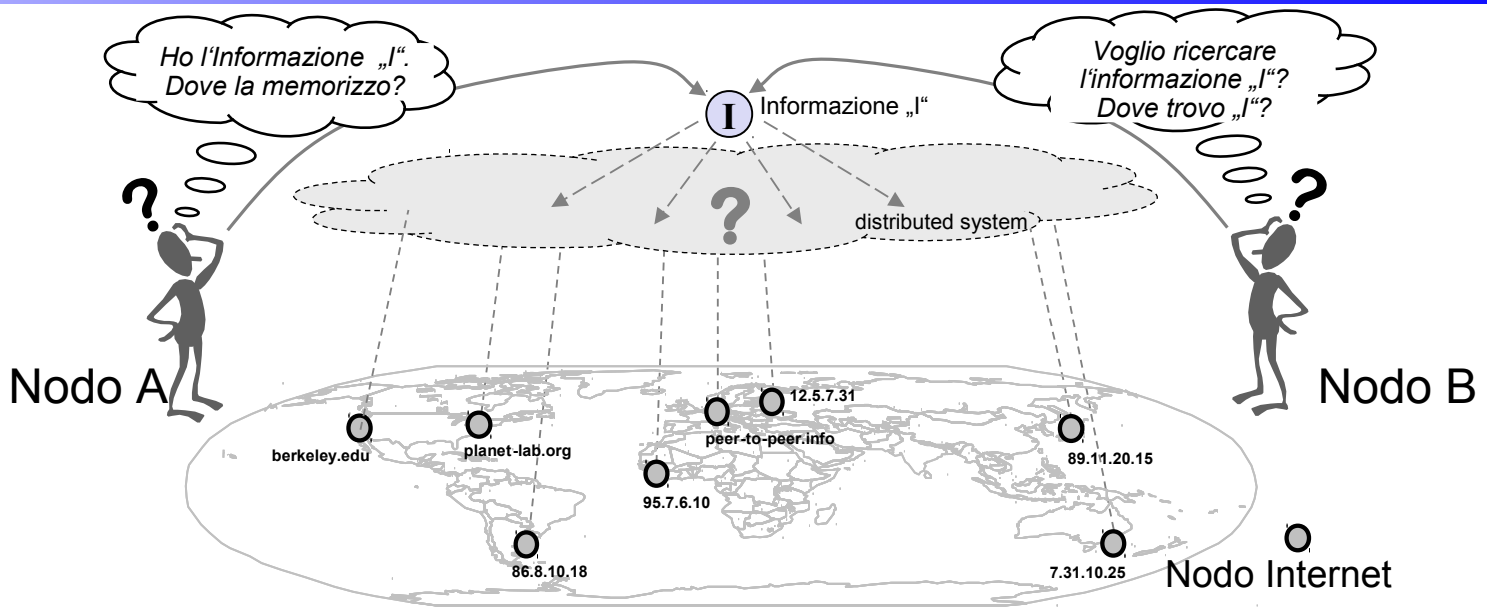
- Ricerca ed Indirizzamento
  - reti strutturate e non strutturate
- Distributed Hash Tables (DHT)
  - cosa sono le DHT?
  - come funzionano?
    - Indirizzamento dei dati
    - Routing
    - Inserzione/cancellazione dati
  - quale è il loro utilizzo ?
  - esempi: Chord, CAN, Pastry, Kademlia/KAD
  - aspetti formali: modellazione routing mediante Catene di Markov

RICERCA IN SISTEMI P2P



- Problema principale di un sistema P2P: individuazione di un dato nella rete
- Nei sistemi Gnutella analizzati le risorse sono memorizzate dal peer che le condivide
- Il problema principale, dovuto alla mancanza di strutturazione della rete è quello della ricerca. Dove si trova l'informazione con le caratteristiche desiderate?

SISTEMI P2P: IL PROBLEMA DELLA RICERCA



- A memorizza una informazione I all'interno del sistema distribuito, B vuole reperire I, ma non conoscere a priori l'effettiva locazione di I
- Come organizzare il sistema distribuito? In particolar modo, quali sono i meccanismi utilizzati per decidere dove memorizzare l'informazione e come reperirla?
- Qualsiasi soluzione deve tenere particolarmente in considerazione
  - Scalabilità del Sistema. Occorre controllare l'overhead di comunicazione e la memoria utilizzata da ogni nodo, in funzione del numero dei nodi del sistema
  - Robustezza ed adattabilità in caso di faults e frequenti cambiamenti

RICERCA ED INDIRIZZAMENTO

- Due strategie per l'individuazione di oggetti/entità in una rete
  - ricerca di oggetti guidata dal valore di alcune chiavi (attributi dell'oggetto)
  - indirizzamento di oggetti mediante identificatori unici
- Scelta ricerca/indirizzamento impatta su:
  - il modo con cui la rete viene costruita
  - il modo con cui gli oggetti vengono associati ai nodi della rete
  - efficienza nell'individuazione di oggetti
- Nei sistemi P2P orientati al file sharing all'inizio prevale la prima soluzione (flooding, TTL enhanced flooding,...), successivamente si introducono meccanismi di indirizzamento

RICERCA VS. INDIRIZZAMENTO

- Indirizzamento: ricerca di un oggetto mediante l'identificatore unico associato
  - URL di un oggetto sul web
- Vantaggi:
  - ogni oggetto identificato univocamente
  - individuazione efficiente di oggetti (routing logaritmico con numero logaritmico di vicini)
- Svantaggi
  - calcolo di ID
  - mantenimento della struttura per l'indirizzamento

- Ricerca: individuare oggetti specificando il valore di un insieme di chiavi di ricerca
  - Google
- Vantaggi
  - "user friendly": non richiesto calcolo di ID
  - non richiede strutture ausiliarie
- Svantaggi
  - inefficienza nella ricerca
  - inefficienza nel confronto di oggetti diversi: occorre confrontare interi oggetti

RICERCA ED INDIRIZZAMENTO

Reti non strutturate

- non esiste meccanismo di indirizzamento: TTL enhanced flooding
- non esiste una regola definita per la scelta dei vicini: ogni peer può scegliere in modo arbitrario i suoi vicini
- non esiste una regola regola che indica su quali peer allocare un oggetto
- tutto questo non implica però una mancanza completa di struttura... la rete può assumere, a posteriori, una colta creata, una certa struttura
  - scale free, power law, small world,....

Reti strutturate

- definiscono un meccanismo di indirizzamento
- la struttura della rete determina dove sono posizionati i peer e su quali peer devono essere memorizzati gli oggetti
- routing deterministico
- Distributed Hash Tables(DHT)

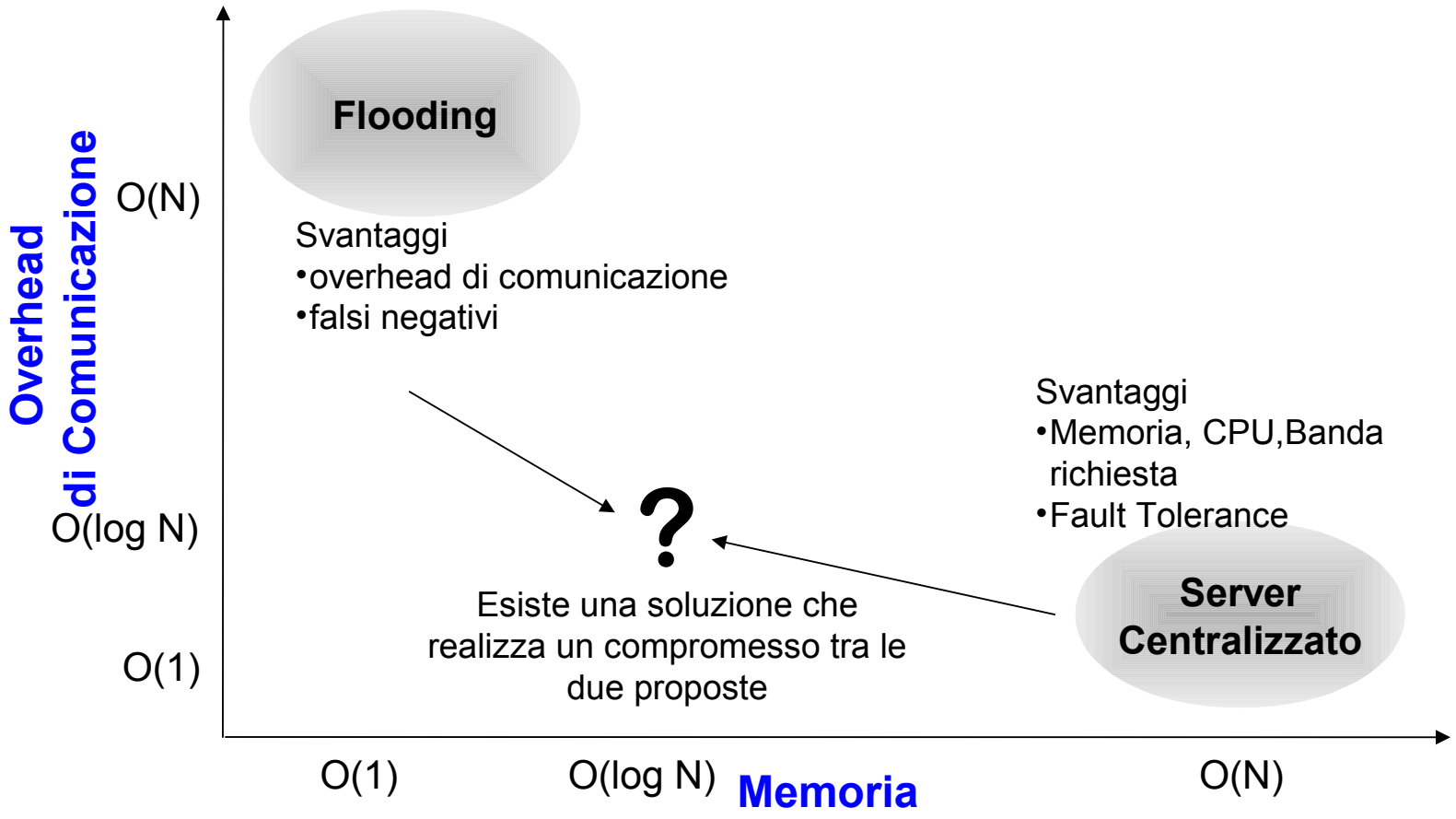
DHT: MOTIVAZIONI

- Approccio centralizzato: un server per l'indicizzazione dei dati
  - Ricerca: O(1) – "memorizzo l'informazione su un server centralizzato"
  - Quantità di Memoria Richiesta sul Server: O(N) (N = numero di informazioni disponibili nel sistema)
  - Banda richiesta (connessione server/rete): O(N)
  - Possibilità di sottomettere al sistema queries complesse

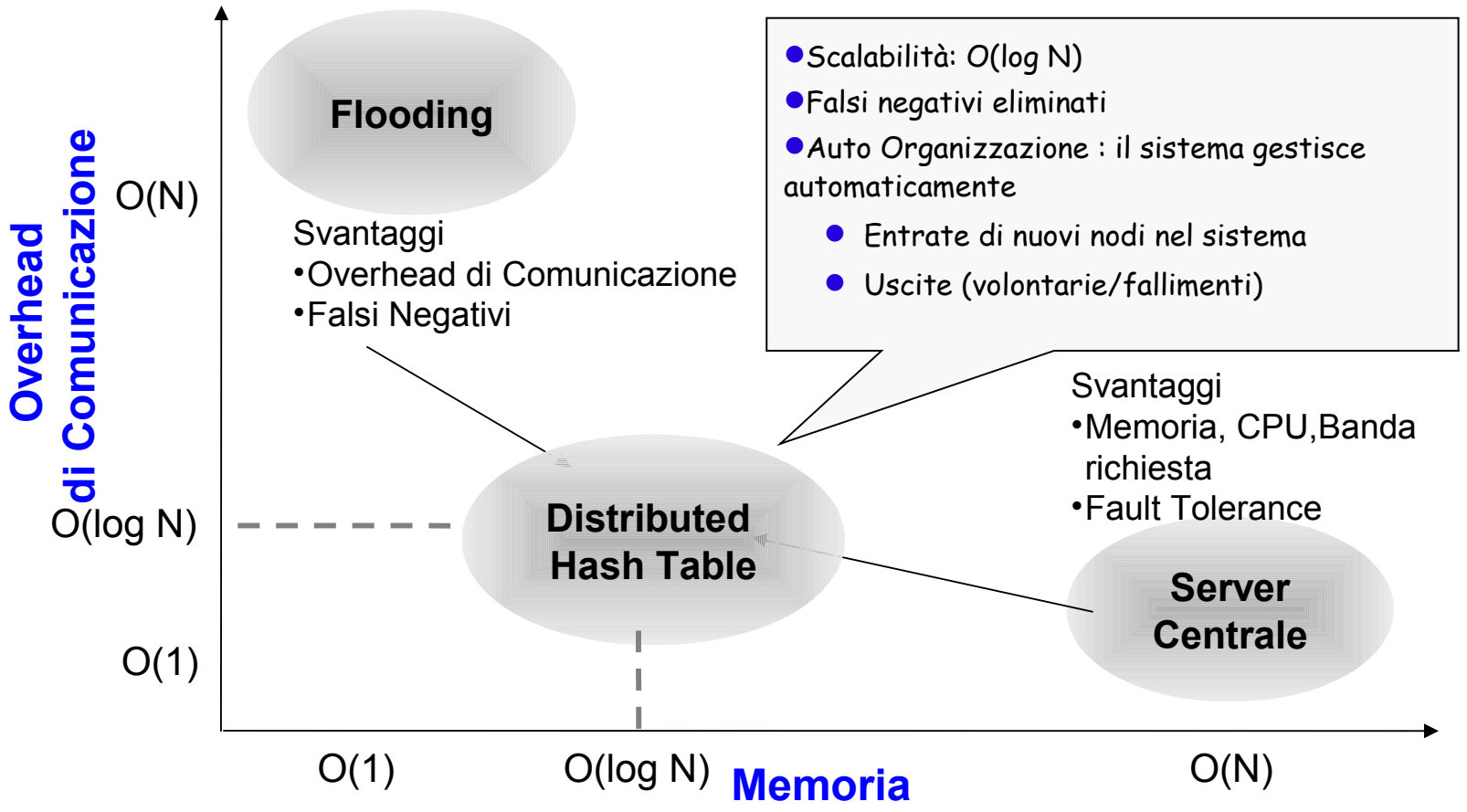
- Approccio Completamente Distribuito: rete non strutturata
  - Ricerca: caso pessimo $O(N^2)$ - "ogni nodo chiede a tutti i vicini ". Possibili ottimizzazioni (TTL, identificatori per evitare cammini ciclici)
  - Quantità di memoria richiesta : O(1)
    - Informazione condivisa: non dipende dal numero di nodi del sistema
    - Non si utilizzano strutture dati per ottimizzare il routing della query (flooding)

DISTRIBUTED HASH TABLES:MOTIVAZIONI

Analisi dei sistemi Esistenti



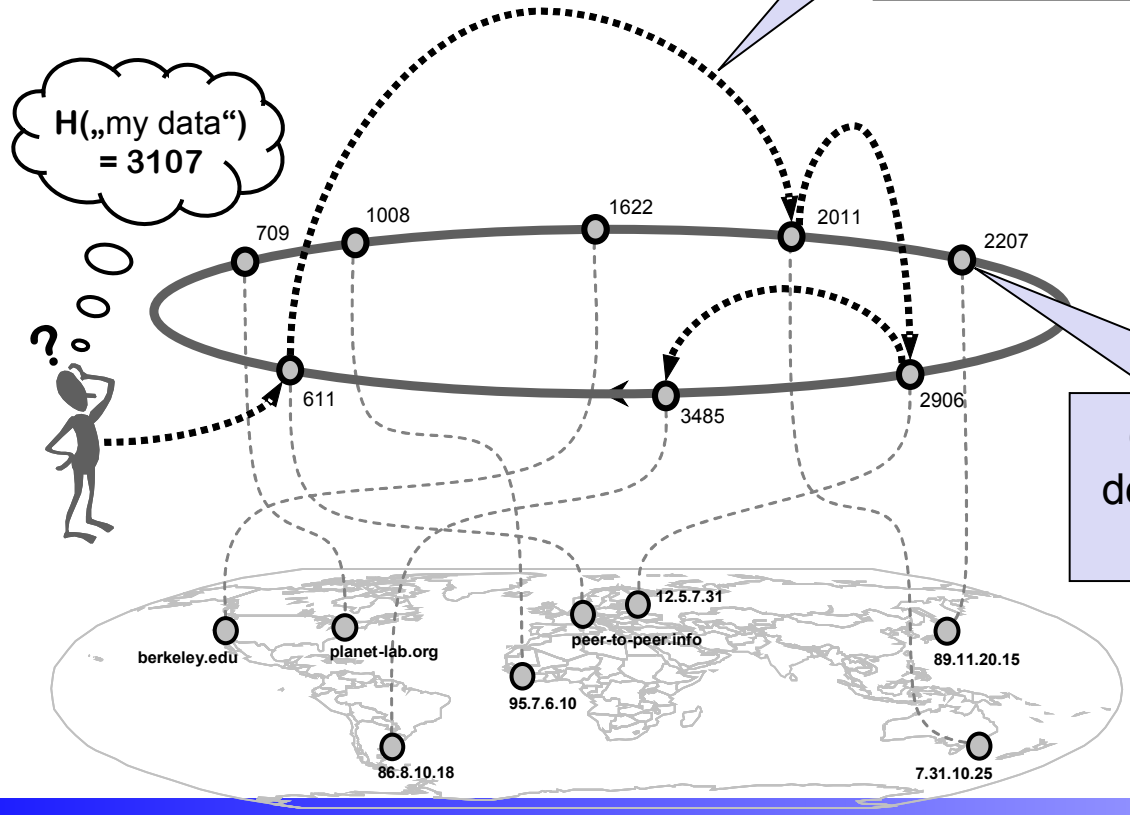
DISTRIBUTED HASH TABLES: MOTIVAZIONI



DISTRIBUTED HASH TABLES:OBIETTIVO

- Obiettivo principale è la scalabilità
  - O(log(N)) hops per la ricerca di un'informazione
  - O(log(N)) entrate nella tabella di routing

Il routing richiede O(*log(N)*) passi per raggiungere il nodo che memorizza l'informazione



O(*log(N)*) dimensione della tabella di routing di ogni nodo

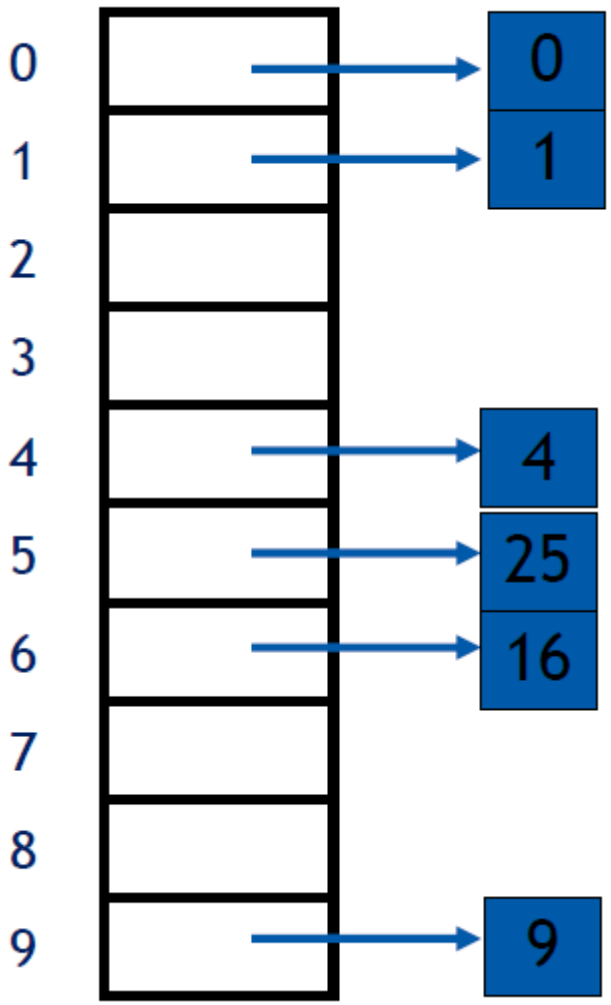
DISTRIBUTED HASH TABLES: OBIETTIVI

- DHT: Obiettivi
  - Scalabilità
  - Flessibilità
  - Affidabilità

- Adattabilità a fallimenti, inserimento ed eliminazione di nodi
  - Assegnamento di informazioni ai nuovi nodi
  - Re-assegnamento e re-distribuzione delle informazioni in caso di fallimento o disconnessione volontaria dei nodi dalla rete

- Bilanciamento delle informazioni tra i nodi
  - Fondamentale per l'efficienza della ricerca

RICHIAMI: HASH TABLES



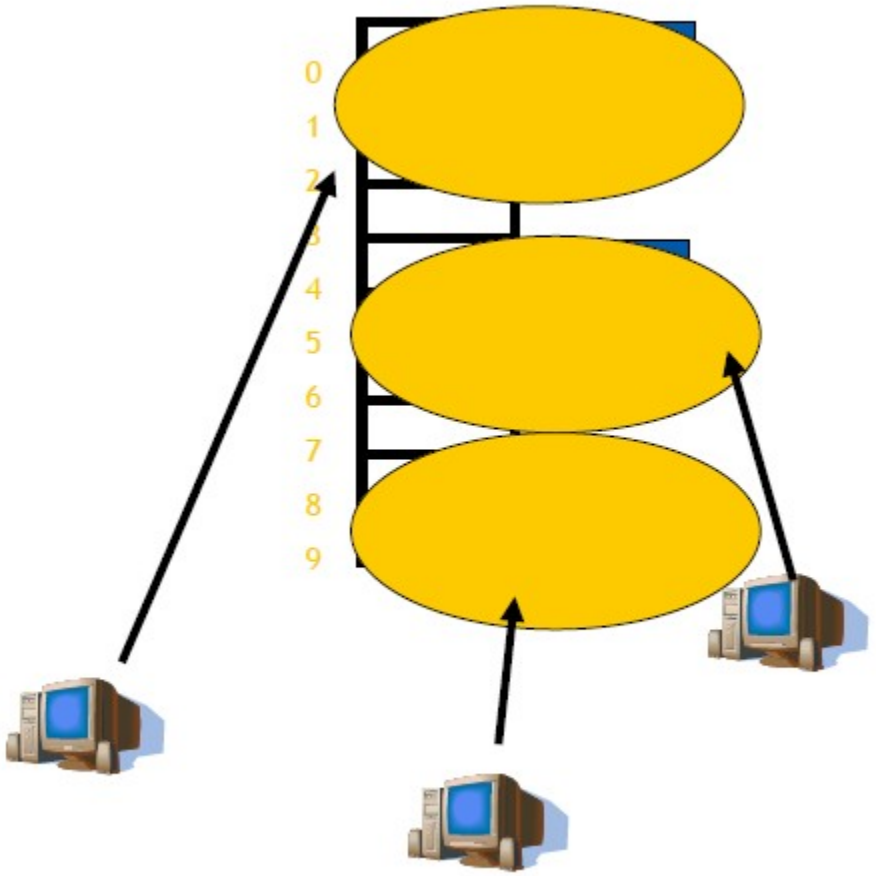
- Inserimento valori 0,1,4,9,16,25
- Funzione hash
  hash(x) = x mod 10
- mapping dominio input di grande dimensioni su un dominio di output di dimensione più piccola
- dominio delle chiavi troppo ampio per utilizzare direttamente la chiave come indice del vettore
- numero contenuto di collisioni
- ricerca di valori O(1)

RICHIAMI: HASH TABLES

- Inserzione, eliminazione, look-up in O(1)
- Hash Table: Array di dimensione fissa
  - Elementi= hash buckets
- Funzione hash: mappa chiavi in elementi del vettore
- Proprietà di una buona funzione hash
  - semplice da calcolare
  - buona distribuzione delle chiavi nella tabella hash
  - esempio: SHA1 Secure Hash Algorithm

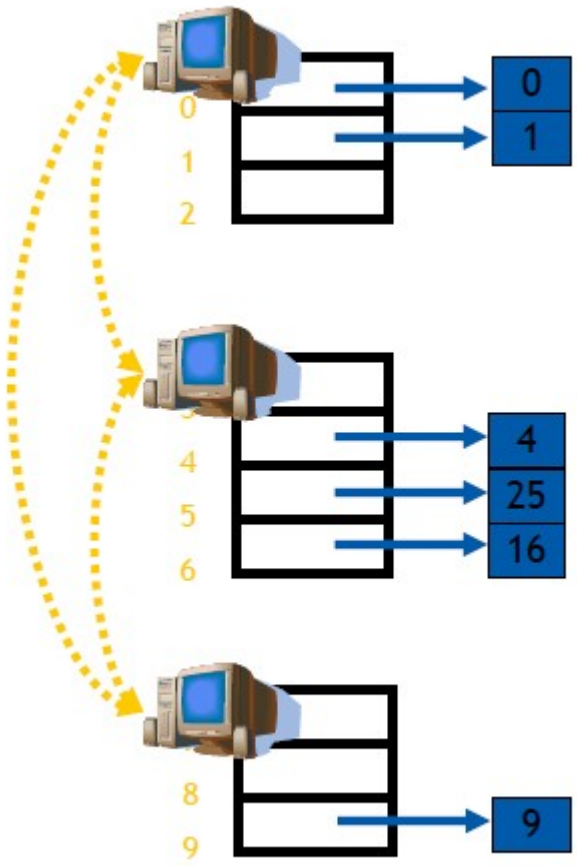
DISTRIBUTED HASH TABLES: IDEA GENERALE

- Distribuire i buckets ai vari peer
- Risultato: Distributed Hash Tables
- Richiedono:
  - meccanismo per individuare quale peer è responsabile di un bucket
  - meccanismo i di routing per raggiungere in modo efficiente il peer che gestisce un bucket



DISTRIBUTED HASH TABLES: IDEA GENERALE

- in una DHT, ogni nodo è responsabile della gestione di uno o più bucket
  - quando un nodo entra od esce dalla rete la responsabilità viene ceduta ad un altro nodo
- i nodi comunicano tra di loro per individuare il nodo responsabile di un bucket
  - definizione di meccanismi di comunicazione scalabili ed efficienti
- supportate tutte le operazioni di una DHT classica



DISTRIBUTED HASH TABLES: IDEA GENERALE

- Il meccanismo utulizzato per l'individuazione del peer che possiede un bucket caratterizza il tipo di DHT

- Comportamento tipico
  - Un nodo conosce l'ID dell'oggetto che vuole ricercare
  - Routing verso il nodo responsabile del bucket che contiene ID
  - Il nodo responsabile risponde inviando direttamente l'oggetto ricercato o un puntatore a quell'oggetto

- Astrazione definita da una DHT
  - memorizza coppie chiave-valore
  - data una chiave, la DHT ritrova il valore corrispondente
  - nessuna semantica associata alla coppia chiave/valore

DHT: GESTIONE DISTRIBUITA DEI DATI

Mapping dei nodi e dei dati nello stesso spazio di indirizzamento

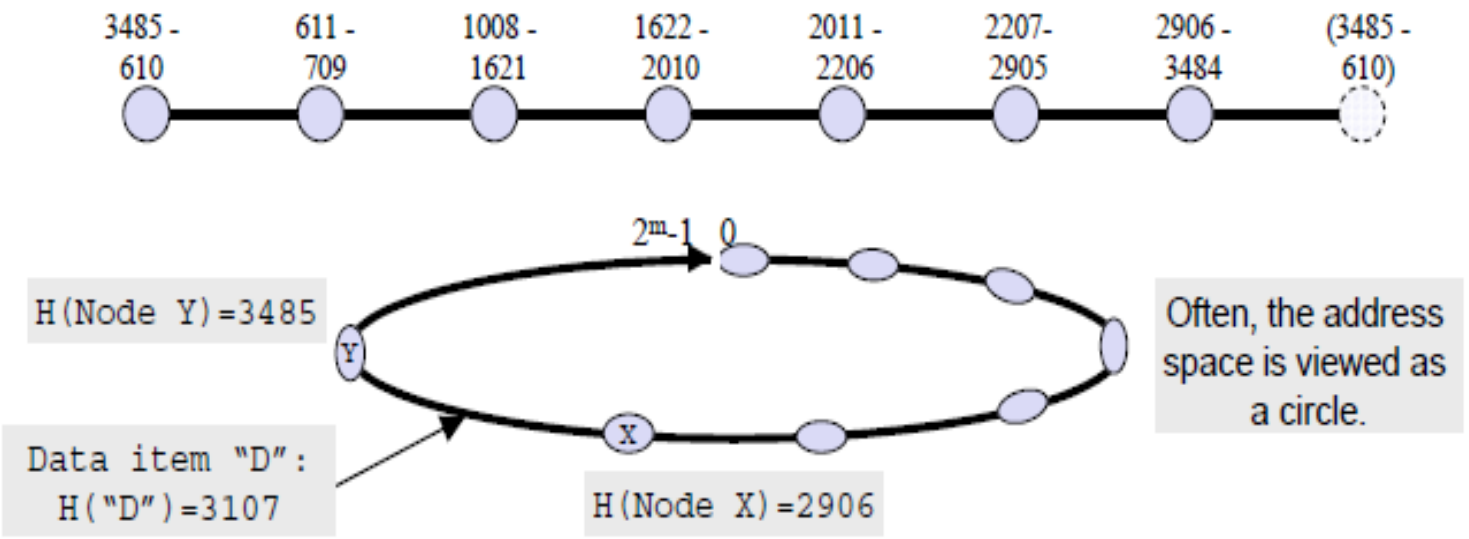
- Ai peers sono associati degli identificatori unici (ID), che li individuano univocamante all'interno del sistema
- Anche ai dati sono associati degli identificatori unici che gli identificano univocamente nel sistema
- Esiste uno spazio logico comune degli indirizzi per i dati e per i peer.
- I nodi sono responsabili della gestione di una porzione dello spazio logico degli indirizzi (uno o più buckets)
- La corrispondenza tra i dati ed i nodi può variare per l'inserimento/cancellazione di nodi

DHT: GESTIONE DISTRIBUITA DEI DATI

Memorizzazione/ Ricerca dei dati

- Ricerca di un dato = routing verso il nodo responsabile
- Ogni nodo mantiene una tabella di routing, che fornisce al nodo una visibilità parziale del sistema
- Key based Routing:Il routing è guidato dalla conoscenza dell'ID del dato ricercato
- Falsi negativi eliminati

PASSO 1: INDIRIZZAMENTO IN UNA DHT



Mapping nodi ed oggetti in uno spazio lineare di indirizzi 0, …, 2^m-1

Lo spazio lineare degli indirizzi logici è >> del numero di oggetti da memorizzare (es m=160),

Sullo spazio è definito un ordinamento totale (operazioni in modulo)

Esempio: Spazio degli indirizzi strutturato secondo un anello logico. S

Associazione nodi-indirizzi logici avviene mediante la funzione hash

Hash(String) mod 2^m, ad esempio Hash(''mydata'')=2313

PASSO 2: ASSOCIAZIONE INDIRIZZI/NODI

- Ogni nodo è responsabile di una porzione di indirizzi (alcuni buckets)

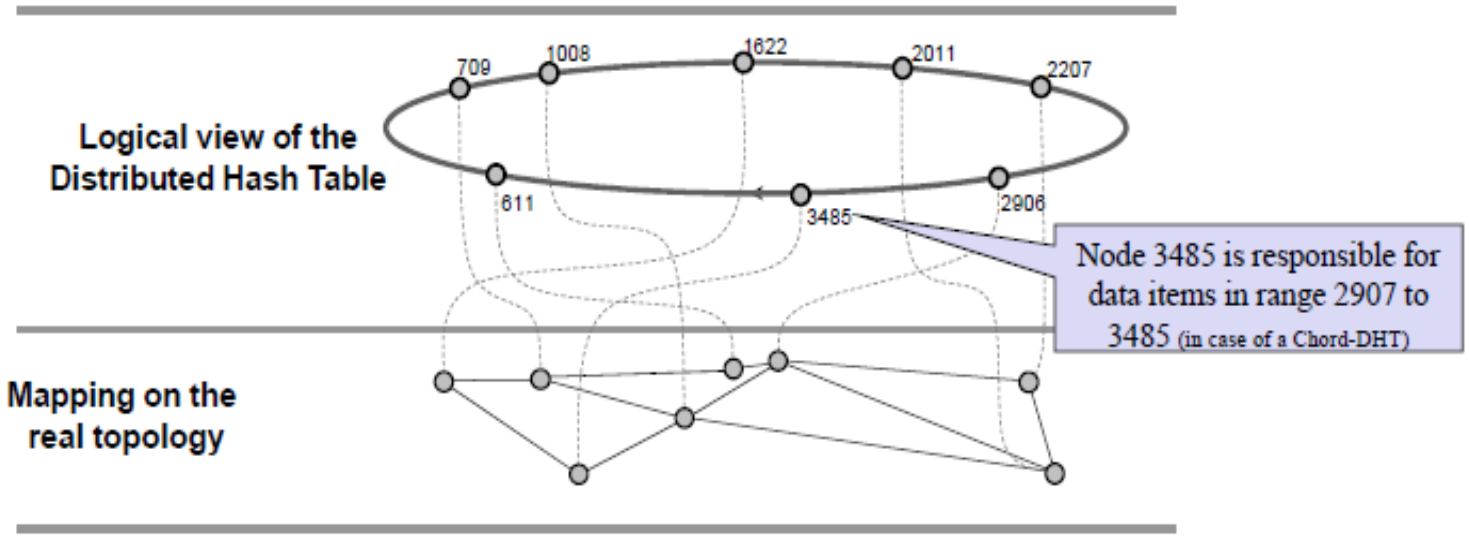
  In generale ad ogni nodo viene assegnata una porzione contigua dello spazio degli indirizzi.

- I dati vengono mappati nello stesso spazio degli indirizzi dei nodi, mediante la funzione hash
  - E.g., Hash(*String*): H("'LucidiLezione060313 ") → 2313
  - Esempi: hashing del nome del file o del suo intero contenuto

- Ogni nodo memorizza informazioni relative ai dati mappati sulla propria porzione di indirizzi

- Spesso si introduce una certa ridondanza (overlapping)

PASSO 2: ASSOCIAZIONE INDIRIZZI/NODI



Ogni nodo è responsabile di un intervallo di identificatori

Può esserci ridondanza (overlapping di intervalli)

Adattamento continuo

Topologia sottostante (underlay) e overlay logico non correlato

DHT: BILANCIAMENTO DEL CARICO

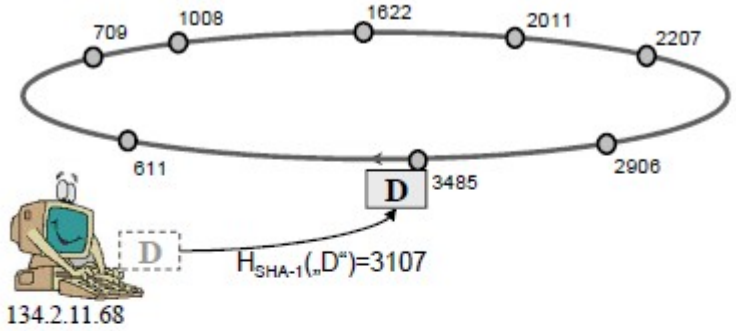
- Distribuzione degli intevalli ai nodi, cause di sbilanciamento del carico:
  - un nodo deve gestire una grossa porzione dello spazio degli indirizzi
    - soluzione: uniformità della funzione hash
  - gli spazi degli indirizzi sono distribuiti in modo uniforme tra i nodi, ma gli indirizzi gestiti da un nodo corrispondono a molti dati
  - un nodo deve gestire diverse queries, perché i dati corrispondenti agli indirizzi gestiti sono molto richiesti

- Sbilanciamento del carico comporta
  - minor robustezza del sistema
  - minor scalabilità
  - O(log N) non garantito

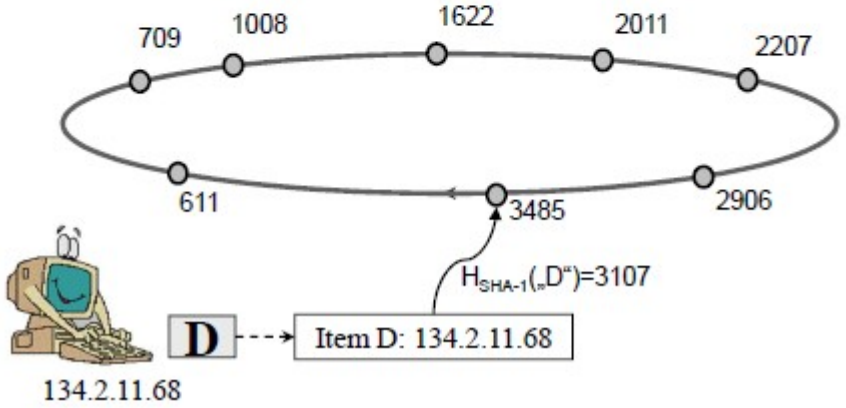
- Soluzioni
  - definizione di hash uniforme
  - definizione di algoritmi di bilanciamento del carico

PASSO 3: STORAGE DEI DATI

Direct Storage



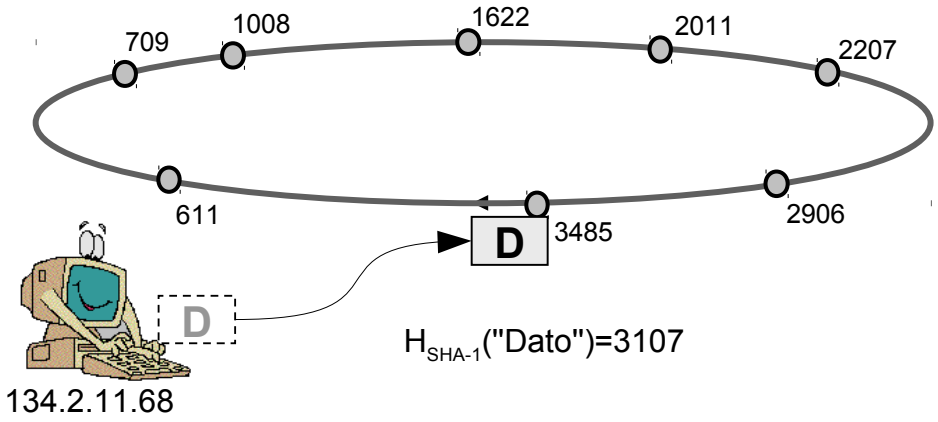
Indirect Storage



PASSO 3: DIRECT STORAGE

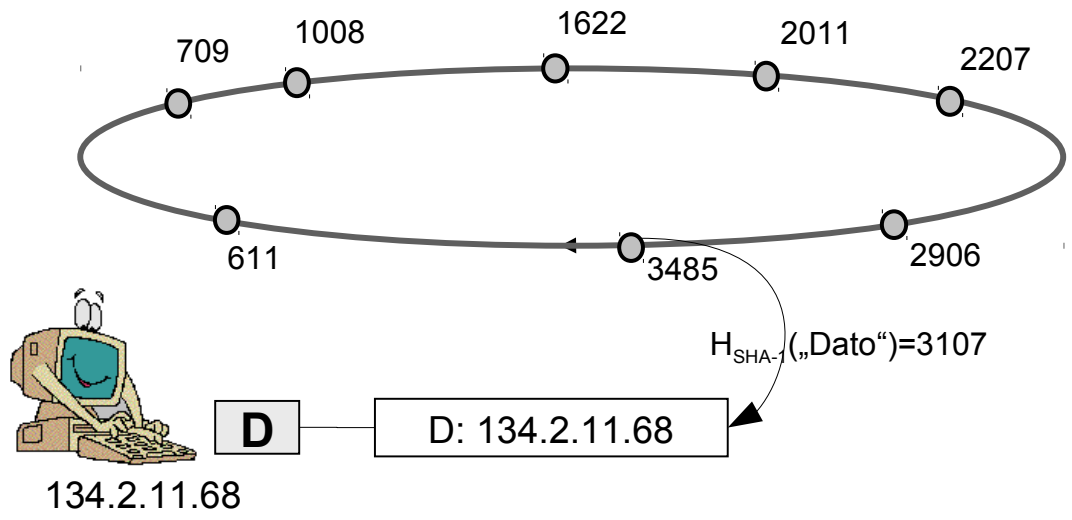
- La DHT memorizza coppie del tipo (key,valore)
- Valore = valore del dato ricercato
- Il dato viene copiato, al momento del suo insermento nella DHT, nel nodo che ne è responsabile.
  - tale nodo non è in generale il nodo che ha inserito il dato nella DHT.

Esempio: key = H("Dato") = 3107 . Il dato viene memorizzato sul nodo responsabile dell'indirizzo 3107.

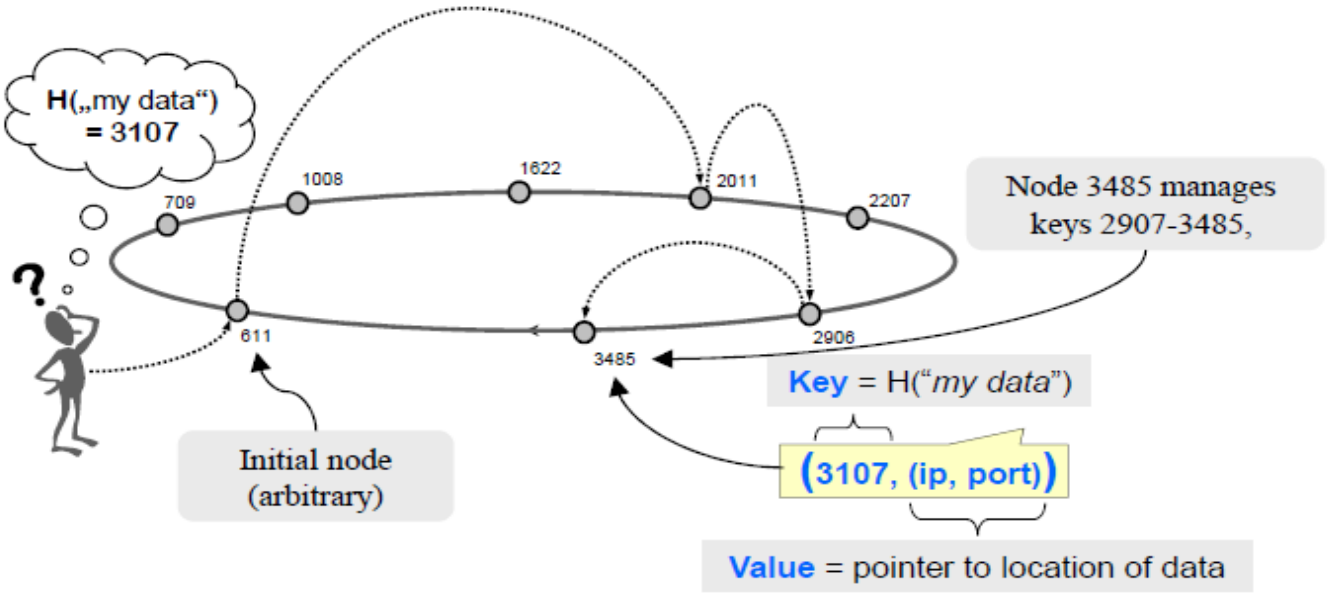


PASSO 3: INDIRECR STORAGE

- Valore = può essere un riferimento al dato ricercato (es: indirizzo fisico del nodo che memorizza il contenuto)
- Il nodo che memorizza il dato può essere quello che lo ha inserito nel sistema
- Più flessibile, richiede un passo in più per l'accesso al dato

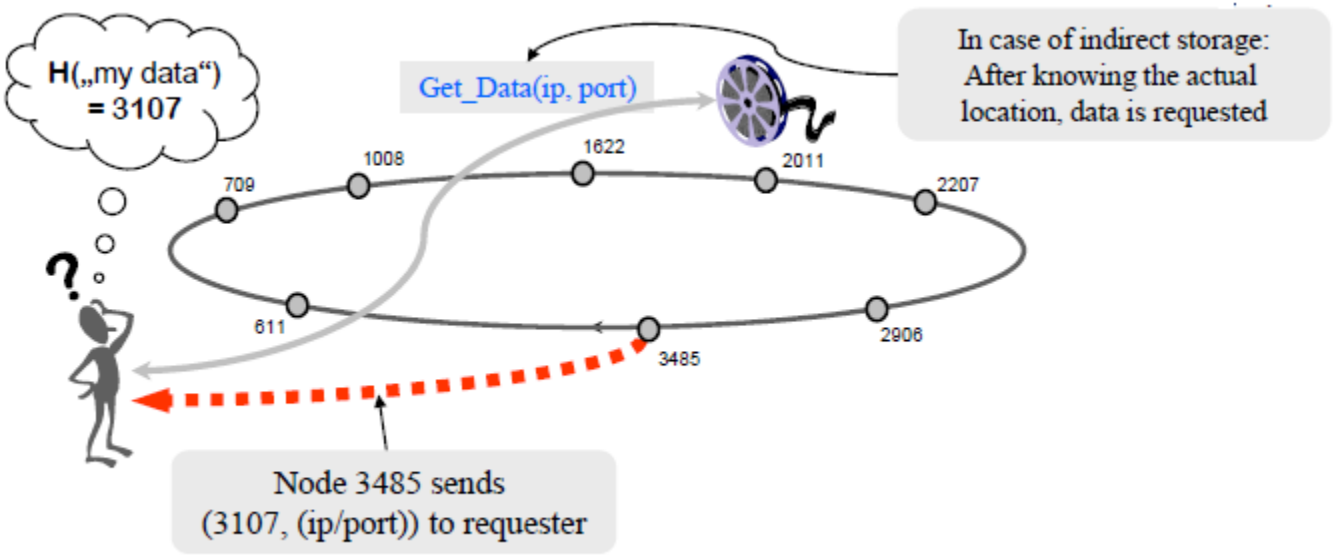


PASSO 4: ROUTING



- La ricerca di D inizia in un nodo arbitrario della DHT ed è guidata da Hash(D)
- Ogni nodo ha in genere una visione limitata degli altri nodi
- Next hop: dipende dall'algoritmo di routing.
  - Esempio: può essere basato sulla "vicinanza" tra l'ID del dato e l'ID del nodo (routing content based), tra i nodi visibili nella tabella di routing
  - Valore associato alla chiave: indirizzo IP+porta del peer che memorizza D.

PASSO 5: DATA RETRIEVAL



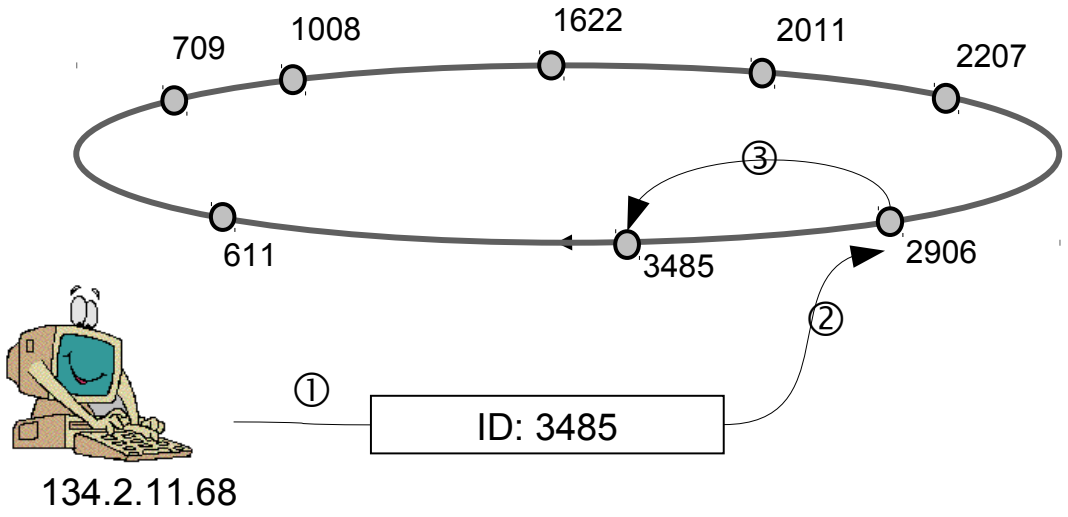
Content Download

Si spedisce indirizzo IP e porta al richiedente

Il richiedente effettua il download dei dati da un altro peer (caso di indirizzamento indiretto.

DHT:INSERZIONE DI NUOVI NODI

- Calcolo dell' identificatore ID del nodo
- Il nuovo nodo contatta un nodo arbitrario della DHT (bootstrap)
- Individua il punto della DHT in cui inserirsi (nodo predecessore o successore)
- Assegnamento di una porzione dello spazio degli indirizzi ad ID
- Copia delle coppie K/V assegnate (in genere si utilizza ridondanza)
- Inserzione nella DHT (collegamento con nodi vicini)



DHT:USCITA/FALLIMENTO DI NODI

- Ritiro Volontario di un nodo
  - Partizionamento della propria porzione degli indirizzi sui nodi vicini
  - Copia delle coppi chiave/valore sui nodi corrispondenti
  - Eliminazione del nodo dalle tabelle di routing

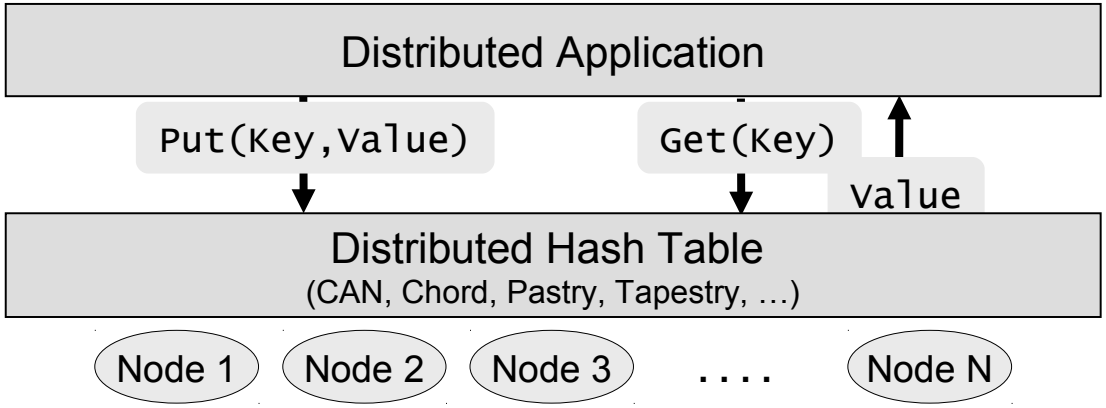
- Fallimento di un Nodo
  - Se un nodo si disconnette in modo inatteso, tutti i dati memorizzati vengono persi a meno che non siano memorizzati su altri nodi
    - Memorizzazione di informazioni ridondanti (replicazione)
    - Perdita delle informazioni; refreshing periodico delle informazioni
  - Utilizzo di percorsi di routing alternativi/ridondanti
    - Probing periodico dei nodi vicini per verificarne la operatività. In caso di fault, aggiornamento delle routing tables

CONFRONTI TRA I DIVERSI APPROCCI

Approccio	Memoria per Nodo	Overhead di Comunicazione	Queries Complesse	Falsi Negativi	Robustezzza
Server Centrale	O(N)	O(1)	✓	✓	✗
P2P puro (flooding)	O(1)	$O(N^2)$	✓	✗	✓
DHT	O(log N)	O(log N)	✗	✓	✓

DHT: API

- Interfaccia (API) per l'accesso alla DHT
  - Inserimento di Informazione Condivisa
    - PUT(key,value)
  - Richiesta di Informazione (content serach)
    - GET(key)
  - Risposte
    - Value
- L'interfaccia è comune a molti sistemi basati su DHT



DHT: APPLICAZIONI

- Le DHT offrono un servizio generico distribuito per la memorizzazione e l'indicizzazione di informazioni

- Il valore memorizzato in corrispondenza di una chiave può essere
  - Un file
  - Un indirizzo IP
  - O qualsiasi altro dato……

- Esempi di applicazioni che possono utilizzare le DHT
  - Realizzazione di DNS
    - Chiave: hostname, valore: lista di indirizzi IP corrispondenti
  - P2P storage systems: es. Freenet

……

CONCLUSIONI

Proprietà delle DHT

- Il routing è basato sul contenuto della query
- Le chiavi sono equamente distribuite tra i nodi della DHT
  - Si evitano i colli di bottiglia
  - Supportano l'inserzione incrementale di chiavi nel sistema
  - Tolleranti ai guasti
- Sistemi auto-organizzanti
- Realizzazione semplice ed efficiente
- Supportano un ampio spettro di applicazioni
  - i valori associati alle chiavi dipendono dalla applicazione

DHT: SISTEMI ESISTENTI

- Chord

  UC Berkeley, MIT
- Pastry

  Microsoft Research, Rice University
- Tapestry

  UC Berkeley
- CAN

  UC Berkeley, ICSI
- P-Grid

  EPFL Lausanne

- Kademlia , rete KAD di EMule…

- Symphony, Viceroy, …