Lezione n.8
KADEMLIA

http://xlattice.sourceforge.net/components/protocol/kademlia/specs.html

Laura Ricci
25/10/2013

PLAXTON ROUTING/MESH

- Plaxton Routing: meccanismo per la diffusione efficiente degli oggetti su una rete, pubblicato nel 1997, prima dei primi sistemi P2P...

- Idea Base:
  - Un oggetto A viene memorizzato sul nodo con identificatore ID che presenta il prefisso più lungo a comune con A

  - prefix matching routing: basato sul confronto
    - prefisso del nodo
    - prefisso del dato
  - generalizzazione di algoritmi di routing definiti su ipercubi

- utilizzato come base del routing per una famiglia di DHT
  - Pastry
  - Tapestry
  - Kademlia

PLAXTON ROUTING/MESH

- Plaxton Routing: meccanismo per la diffusione efficiente degli oggetti su una rete, pubblicato nel 1997, prima dei primi sistemi P2P...

- utilizzato come base del routing per una famiglia di DHT
  - Pastry
  - Tapestry
  - Kademlia

- idee base:
  - mappare nodi ed oggetti su numeri in base b utilizzando m digit
  - assegnare un oggetto A al nodo il cui identificatore condivide con il prefisso più lungo con quello di A

b = 4 and m = 6:

PLAXTON MESH

- Prefix Matching DHT: lo spazio degli identificatori viene rappresentato mediante un albero
  - profondità dell'albero = m, dove m è la lunghezza degli identificatori
  - le foglie corrispondono ai possibili identificatori
  - i nodi interni corrispondono a prefissi di identificatori

- Kademlia considera un albero binario
  - ogni nodo dell'albero corrisponde ad una stringa binaria
  - Nodi interni= prefissi binari di identificatori

PLAXTON MESH

Tabelle di routing:

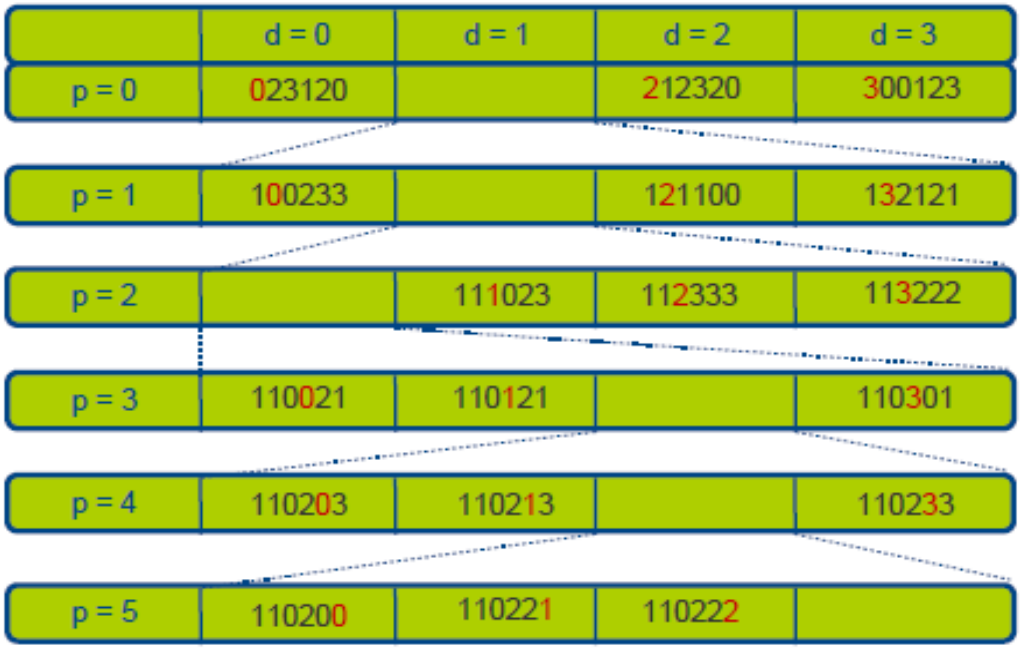
- m righe
- Ogni riga corrisponde ad un prefisso dell'identificatore del nodo
- ogni riga contiene un insieme di k puntatori (k-buckets) a
  - nodi caratterizzati da quel prefisso

b = 4, m = 6
nodeID = 110223

	d = 0	d = 1	d = 2	d = 3
p = 0	0xxxxx	110223	2xxxxx	3xxxxx
p = 1	10xxxx	110223	12xxxx	13xxxx
p = 2	110223	111xxx	112xxx	113xxx
p = 3	1100xx	1101xx	110223	1103xx
p = 4	11020x	11021x	110223	11023x
p = 5	110200	110221	110222	110223

PLAXTON MESH

b = 4, m = 6
nodeID = 110223



PLAXTON MESH

Prefix Matching Routing Lookup: N_1 ricerca la chiave K

- N_1 ricerca nella sua routing table
  - un nodo N_2 con un id la cui configurazione binaria coincide almeno nei b bits più significativi con quella di K
  - inoltra K verso N_2

- Il Routing
  - Ricerca ad ogni passo un 'match' di lunghezza sempre maggiore tra gli id della chiave e quello del nodo.

  - termina non appena non è possibile individuare all'interno della tabella di routing un match di lunghezza maggiore
    - il nodo più vicino alla chiave è stato raggiunto

PLAXTON MESH

b = 4, m = 6, nodeID = 110223 → lookup(322010)

Node 110223 routing table

	d = 0	d = 1	d = 2	d = 3
p = 0				300123
p = 1				
p = 2				
p = 3				
p = 4				
p = 5				

PLAXTON MESH

b = 4, m = 6, nodeID = 110223 → lookup(322010)

Node 300123 routing table

	d = 0	d = 1	d = 2	d = 3
p = 0				
p = 1			323130	
p = 2				
p = 3				
p = 4				
p = 5				

PLAXTON MESH

b = 4, m = 6, nodeID = 110223 → lookup(322010)

Node 323130 routing table

	d = 0	d = 1	d = 2	d = 3
p = 0				
p = 1				
p = 2			322321	
p = 3				
p = 4				
p = 5				

PLAXTON MESH

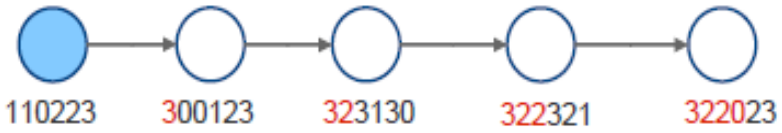
b = 4, m = 6, nodeID = 110223 → lookup(322010)

Node 322321 routing table

	d = 0	d = 1	d = 2	d = 3
p = 0				
p = 1				
p = 2				
p = 3	322023			
p = 4				
p = 5				

PLAXTON MESH

b = 4, m = 6, nodeID = 110223 → lookup(322010)

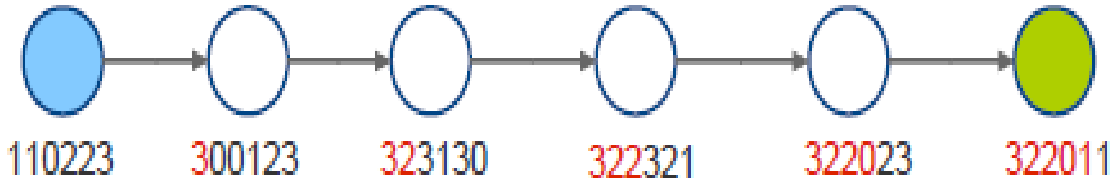


Node 322023 routing table

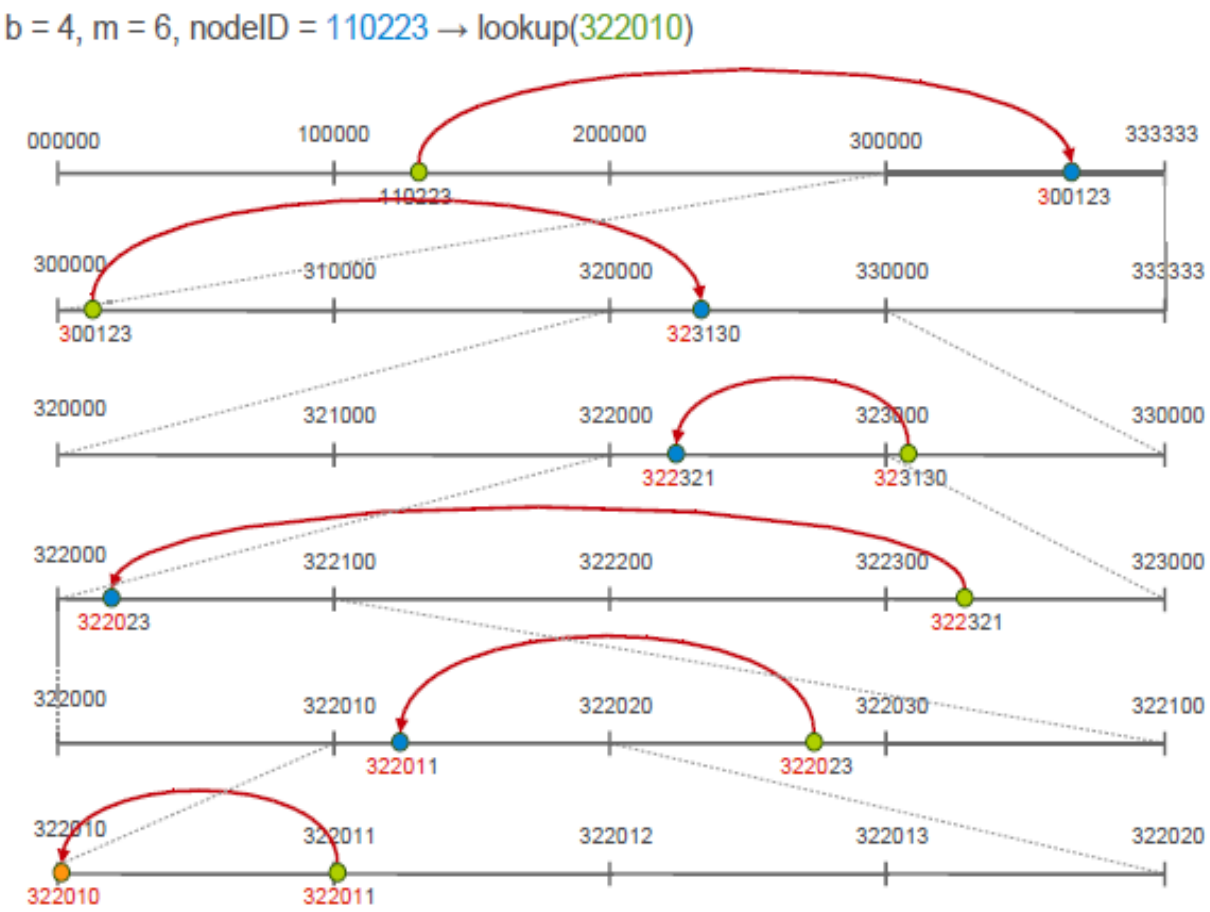
	d = 0	d = 1	d = 2	d = 3
p = 0				
p = 1				
p = 2				
p = 3				
p = 4		322011		
p = 5				

PLAXTON MESH

b = 4, m = 6, nodeID = 110223 → lookup(322010)



PLAXTON MESH



PLAXTON MESH

- b = number of bits corrected at each step = symbol size
- Kademlia exploits b=1, at each "llok up" step, it corrects at least a bit.
- Pastry usually exploits b=4, at each step increases the match between the key and the node identifier by 4 bits
- Routing: $O(\log_b (n))$
- Routing table size look-up hops number depends on b
- Observation: some IP protocols may be considered prefix-matching protocols, where the matching is computed on the IP address.

PLAXTON MESH

- K- buckets: lista di riferimenti a nodi (contatti) memorizzati in ogni elemento della tabella di routing

- ad ogni passo di lookup ogni nodo ha la possibilità di scegliere tra K diversi contatti
  - K = 1 in Pastry
  - K ≅ 20 in Kademlia

- un valore di K > 1 garantisce
  - una maggiore robustezza e tolleranza ai guasti del routing
  - la possibilità di scegliere percorsi di routing alternativi
  - la possibilità di effettuare la ricerca della stessa chiave in parallelo su più percorsi

- Chord definisce un solo contatto (finger) per ogni riga della tabella di routing

PLAXTON MESH

- Parallel Routing: una chiave K ricevuta da un nodo viene inoltrata in parallelo ad α nodi prelevati da un k-bucket
- Strategie di routing
  - routing iterativo:
    - il nodo che invia una richiesta di look up coordina l'intero processo di ricerca
    - ad ogni passo, un nodo invia una richiesta di look up ed attende una risposta
    - la risposta ricevuta indica quale è il successivo passo di routing
  - routing ricorsivo: la richiseta di look up viene inoltrata automaticamente da un peer al successivo
- Kademlia utilizza un routing iterativo

KADEMLIA: CONCETTI GENERALI

- proposta da P. Maymounkov e D. Mazières (University of New York)
- presenta un insieme di caratteristiche non offerte simultaneamente da nessun sistema esistente (Chord, Pastry,..)
- distanza tra identificatore definita mediante una metrica basata su or esclusivo

$$X = 0\,1\,0\,1\,1\,0$$
$$Y = 0\,1\,1\,0\,1\,1$$
$$d(x,y) = x \oplus y = 0\,0\,1\,1\,0\,1$$
$$d(x,y) = 13$$

- una chiave viene assegnata al nodo il cui identificatore è più vicino alla chiave, secondo la metrica definita (non è detto che sia quello numericamente più vicino)

KADEMLIA: APPLICAZIONI

KADEMLIA: CARATTERISTICHE GENERALI

- Prefix Matching Routing
- Routing
  - parallelo
  - iterativo
- Nessuna DHT riunisce tutte queste caratteristiche
  - Prefix Match Routing caratteristico anche di Pastry, Tapestry ed altre DHT

KADEMLIA:CONCETTI GENERALI

- In Kademlia un identificatore di 160 bits per ogni nodo ed ogni chiave
- Distanza tra due nodi (non geografica, ma sull'overlay) definita mediante l' OR esclusivo calcolato bit a bit degli identificatori.
- Alcune proprietà dell'OR esclusivo utilizzate in Kademlia:

$d(x, y) > 0$ se $x \neq y$

Simmetria $\forall x, y : d(x, y) = d(y, x)$

Disuguaglianza Triangolare $d(x, y) + d(y, z) \geq d(x, z)$

Unidirezionalità: dato un x ed una distanza Δ, esiste un solo y tale che $d(x, y) = \Delta$

KADEMLIA: LA METRICA

- maggiore è la lunghezza del prefisso che caratterizza due nodi, minore è la loro distanza calcolata mediante l' or esclusivo (⊕)

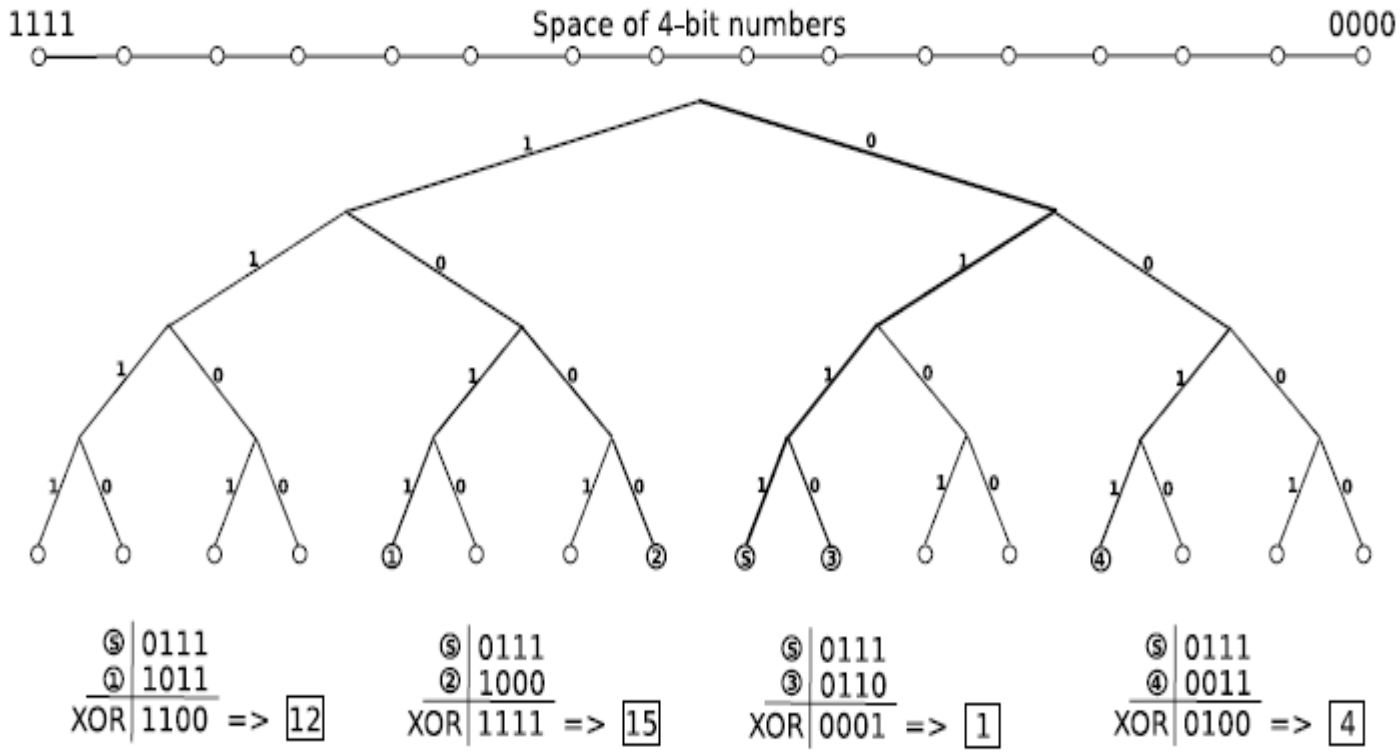
  000100 ⊕ 000110 = 000010 prefisso comune 0001 , distanza 2

  010000 ⊕ 000001 = 010001 prefisso comune 0 , distanza 17

- nodi "vicini" sono caratterizzati da un lungo prefisso comune

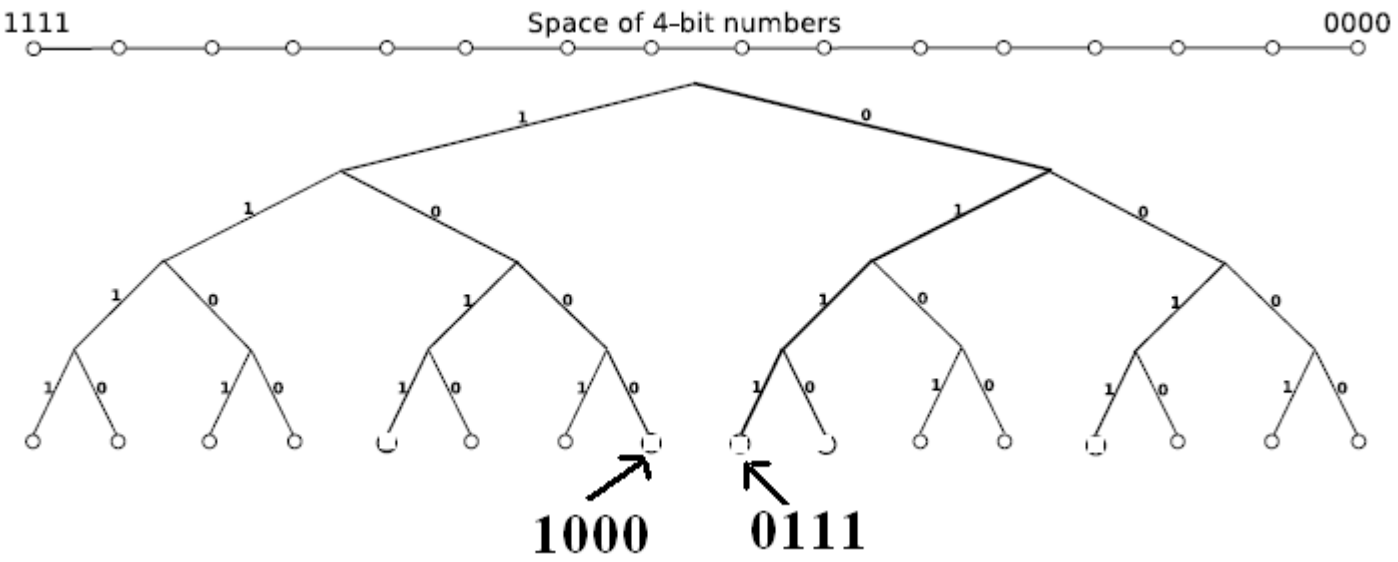
- spazio degli identificatori rappresentato mediante un albero binario, in cui le foglie corripondono agli identificatori
  - l'albero mette in evidenza i nodi vicini secondo la metrica basata su ⊕

LO SPAZIO DEGLI IDENTIFICATORI



- ogni nodo n dell'albero è identificato da ID(n) = stringa di bit ottenuta concatenando le cifre binarie presenti sul cammino dalla radice ad n
- solo un sottoinsieme degli identificatori è, in ogni istante, assegnato ai peer

LO SPAZIO DEGLI IDENTIFICATORI



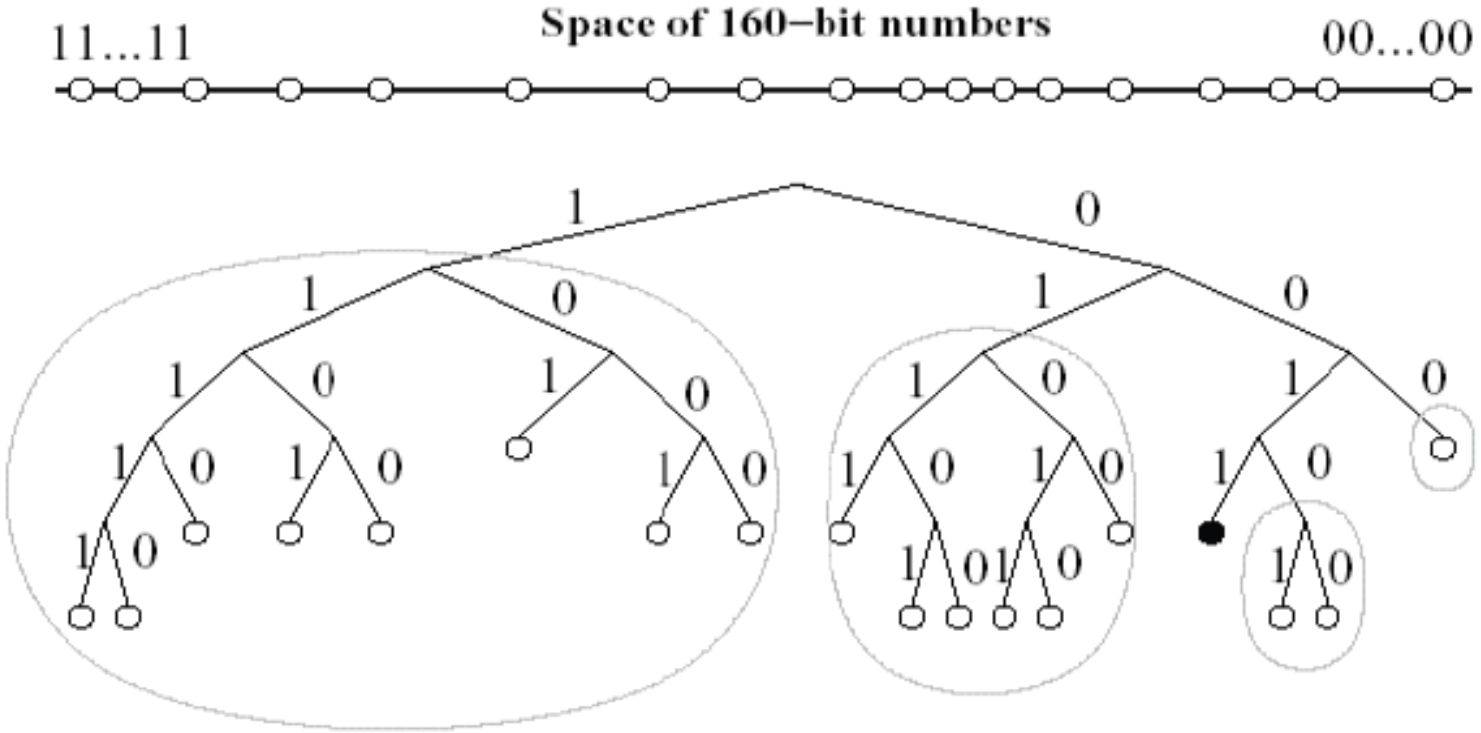
due foglie possono avere identificatori numericamente vicini, ma distanti secondo la metrica $\oplus$ se caratterizzate da un piccolo prefisso comune

$1000 \oplus 0111 = 1111$, $d(1000,0111) = 15$, differenza numerica= 1

FUNZIONE DISTANZA: USO

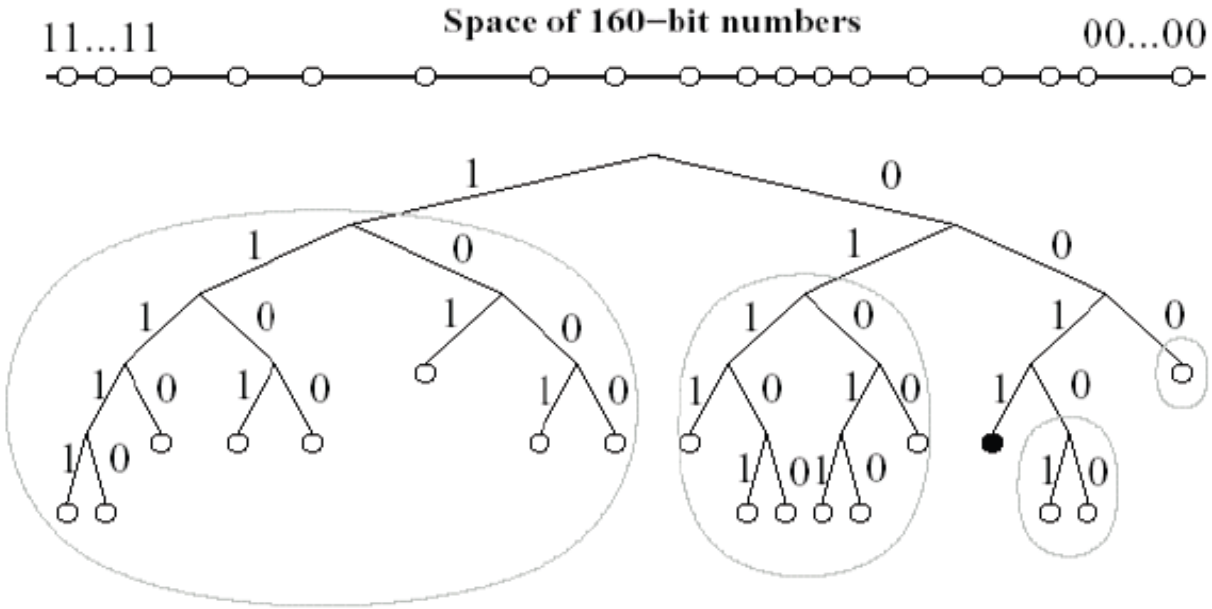
- La coppia <chiave, valore> è memorizzata nei k nodi più vicini alla chiave, la distanza è calcolata con $\oplus$
- i nodi più vicini alla chiave non hanno necessariamente hanno identificatori numericamente vicini alla chiave
- Considerando l'albero degli identificatori, i nodi vicini alla chiave si trovano in sottoalberi vicini a quello in cui si trova la chiave

KADEMLIA: L'ALBERO DEI PEER



- Rappresentazione dei peer in una rete Kademlia
  albero dei peer: albero binario non bilanciato che metta in evidenza solo gli identificatori assegnati a peer presenti sulla rete
  ogni foglia dell'albero corrisponde ad un peer presente sulla rete
  l'identificatore della foglia è un prefisso dell'identificatore del peer

KADEMLIA: L'ALBERO DEI PEER



Il prefisso dell'identificatore corrispondente alla foglia identifica univocamente il peer all'interno dell'overlay

il prefisso 0011 identifica in modo univoco il peer nero

in questo caso non è importante rappresentare la rimanente parte dell'identificatore

DISTANZA TRA IDENTIFICATORI

- consideriamo identificatori di m bits
- sia X un identificatore ed I l'insieme degli identificatori che condividono con X i primi p bits, ma differiscono negli altri i=m-p bits
- la distanza tra X e gli identificatori di I soddisfa la seguente condizione

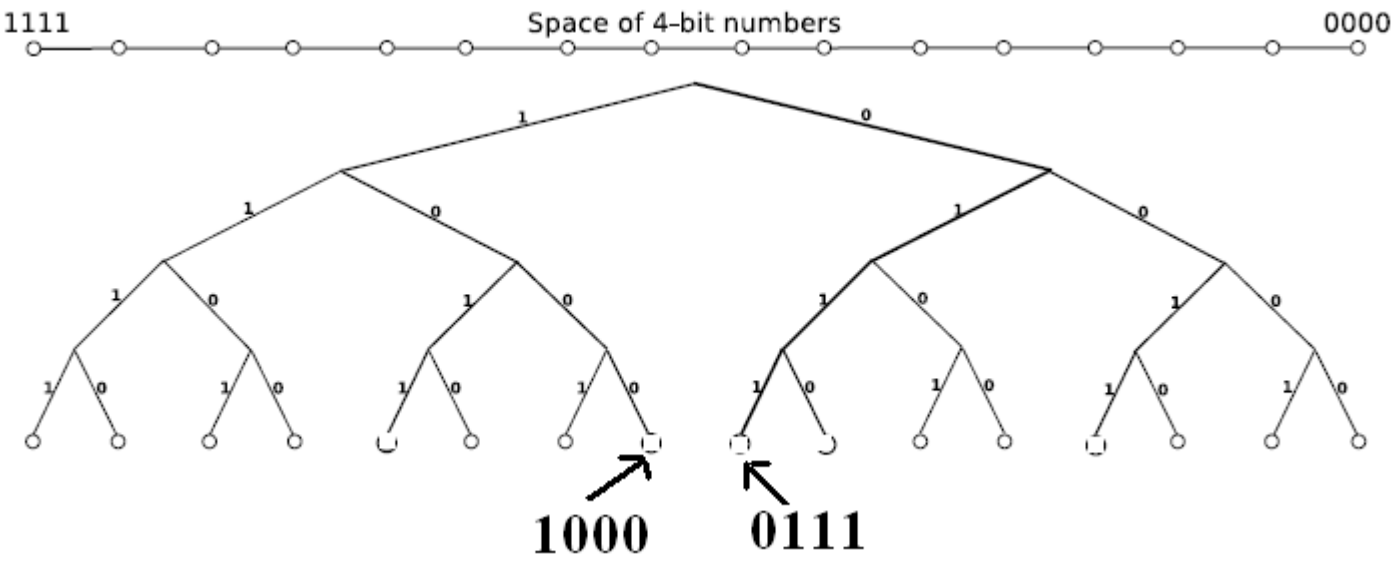
$$2^{i-1} \leq d(x,y) < 2^{i}$$

X = 0 1 0 1 1 0
Y = 0 1 1 1 1 0
X ⊕ Y= 0 0 1 0 0 0, d(x,y) = 2^3=8 (distanza minima)

X = 0 1 0 1 1 0
Y = 0 1 1 0 0 1
X ⊕ Y = 0 0 1 1 1 1, d(x,y) = 15 (distanza massima)

I= altezza dell'albero più piccolo che contiene entrambi gli identificatori

LO SPAZIO DEGLI IDENTIFICATORI

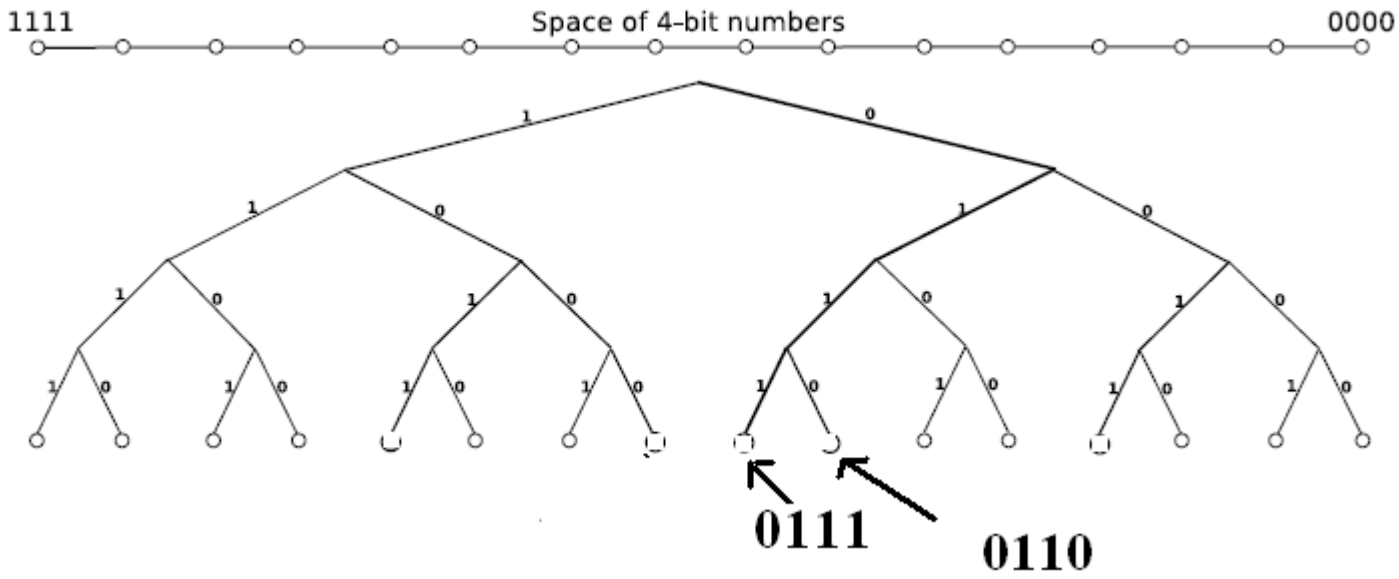


distanza tra n = 0111 ed una qualsiasi foglia appartente al sottoalbero sinistro della radice, di altezza 3

$$0111 \oplus 1000 = 1111 = 15$$
$$0111 \oplus 1111 = 1000 = 8$$

$$2^3 \leq d < 2^4$$

LO SPAZIO DEGLI IDENTIFICATORI

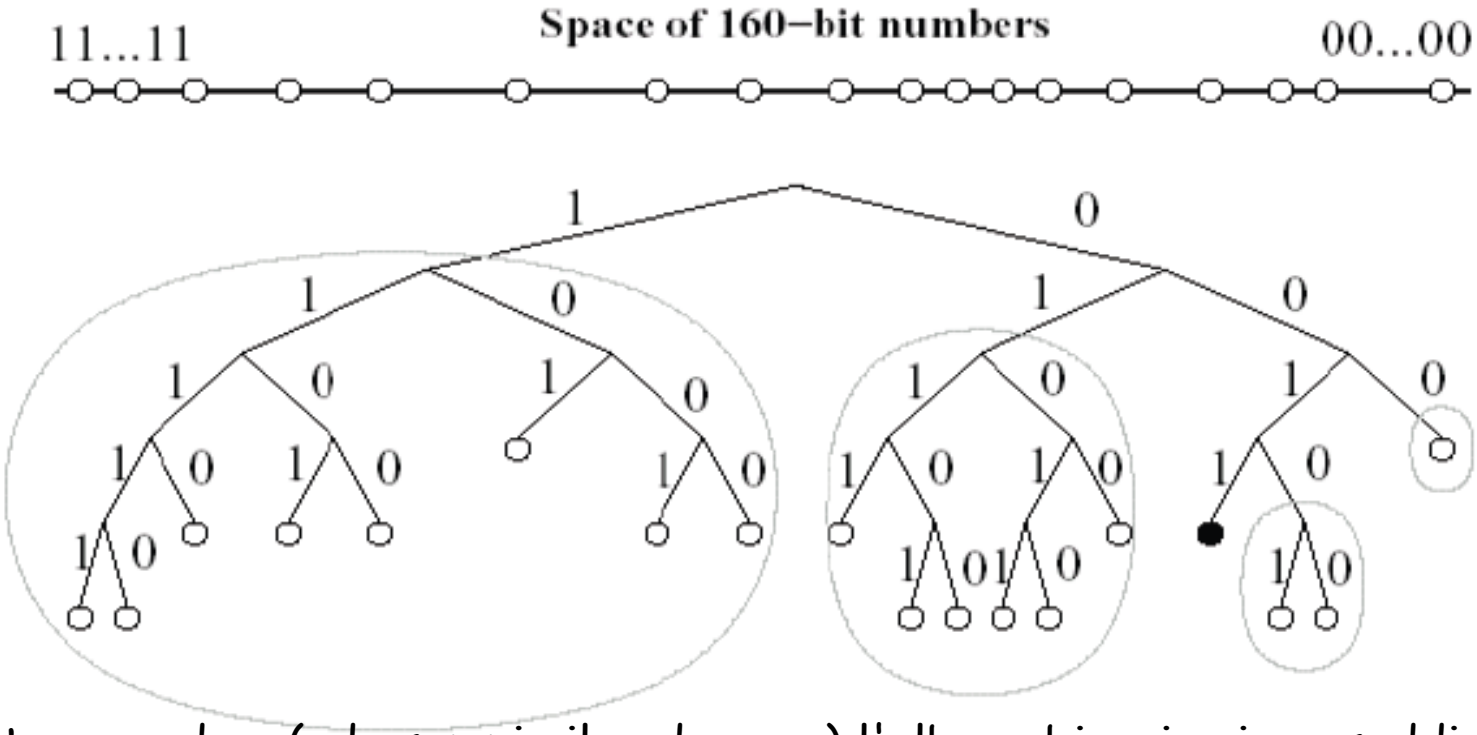


distanza tra 0111 e 0110, appartenente ad un sottoalbero di altezza 1

$$0110 \oplus 0111 = 0001 = 1$$

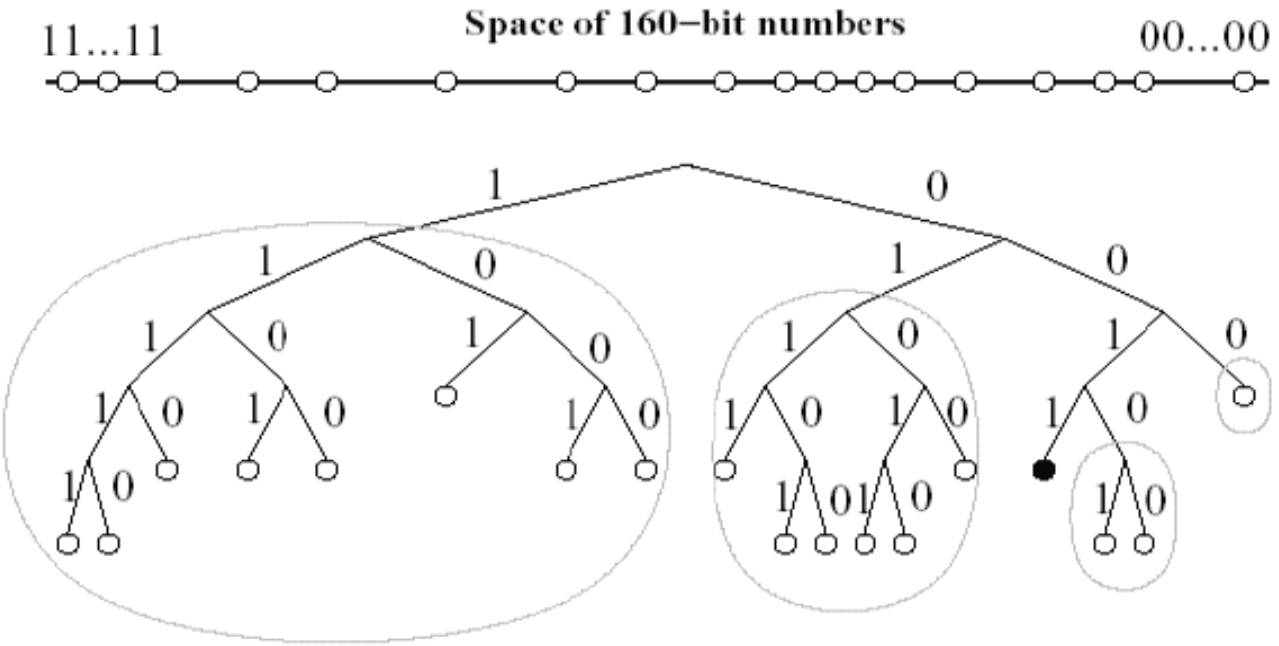
$$2^0 \leq d < 2^1$$

KADEMLIA: LE TABELLE DI ROUTING



- dato un nodo n (ad esempio il nodo nero),l'albero binario viene suddiviso, a partire dalla radice, in una successione di sottoalberi sempre più piccoli che non contengono n
- il primo sottoalbero contiene metà dell'albero binario che non contiene n
- il secondo sottoalbero contiene la metà della rimanente parte dell'albero che non contiene n e così via

KADEMLIA: LE TABELLE DI ROUTING



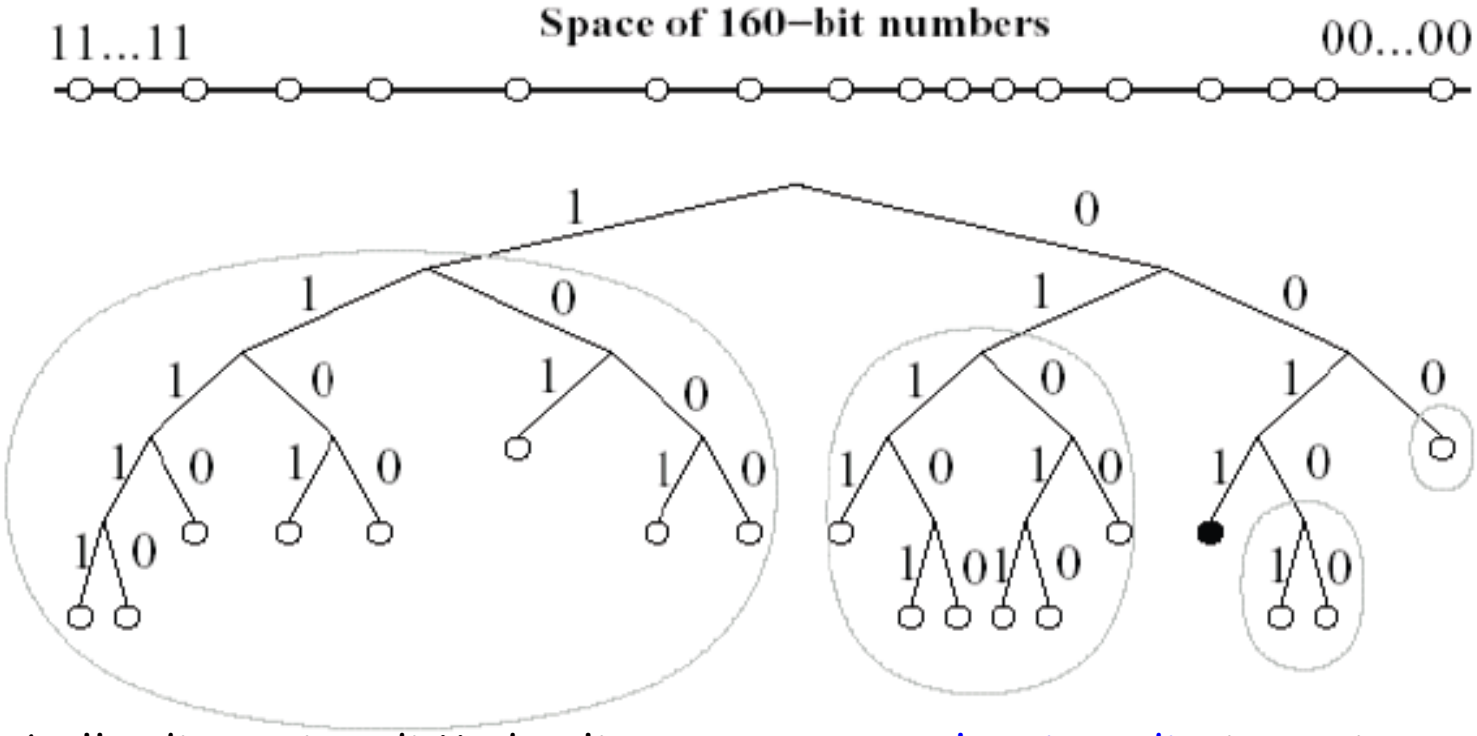
- consideriamo il peer nero n, ogni peer nel sottoalbero sinistro della radice ha un id che differisce da quello di n nella cifra più significativa
- la distanza d tra n ed un qualsiasi peer nel sottoalbero sinistro è tale che

$$2^{159} \leq d < 2^{160}$$

- in generale, vale la formula vista in precedenza

$$2^{i-1} \leq d < 2^{i}$$

KADEMLIA: LE TABELLE DI ROUTING



- Le tabelle di routing di Kademlia contengono alcuni nodi in ogni sottoalbero così individuato
- Ogni query viene inviata ad uno dei nodi conosciuti in un sottoalbero
- Il nodo contattato propaga la query verso sottoalberi via via più vicini al target

KADEMLIA: TABELLE DI ROUTING

- Tabelle di routing basate sulla costruzione di k-Buckets.

- K-Bucket: Lista contenente (al massimo) K (K in genere = 20) contatti. Ogni contatto è una tripla del tipo

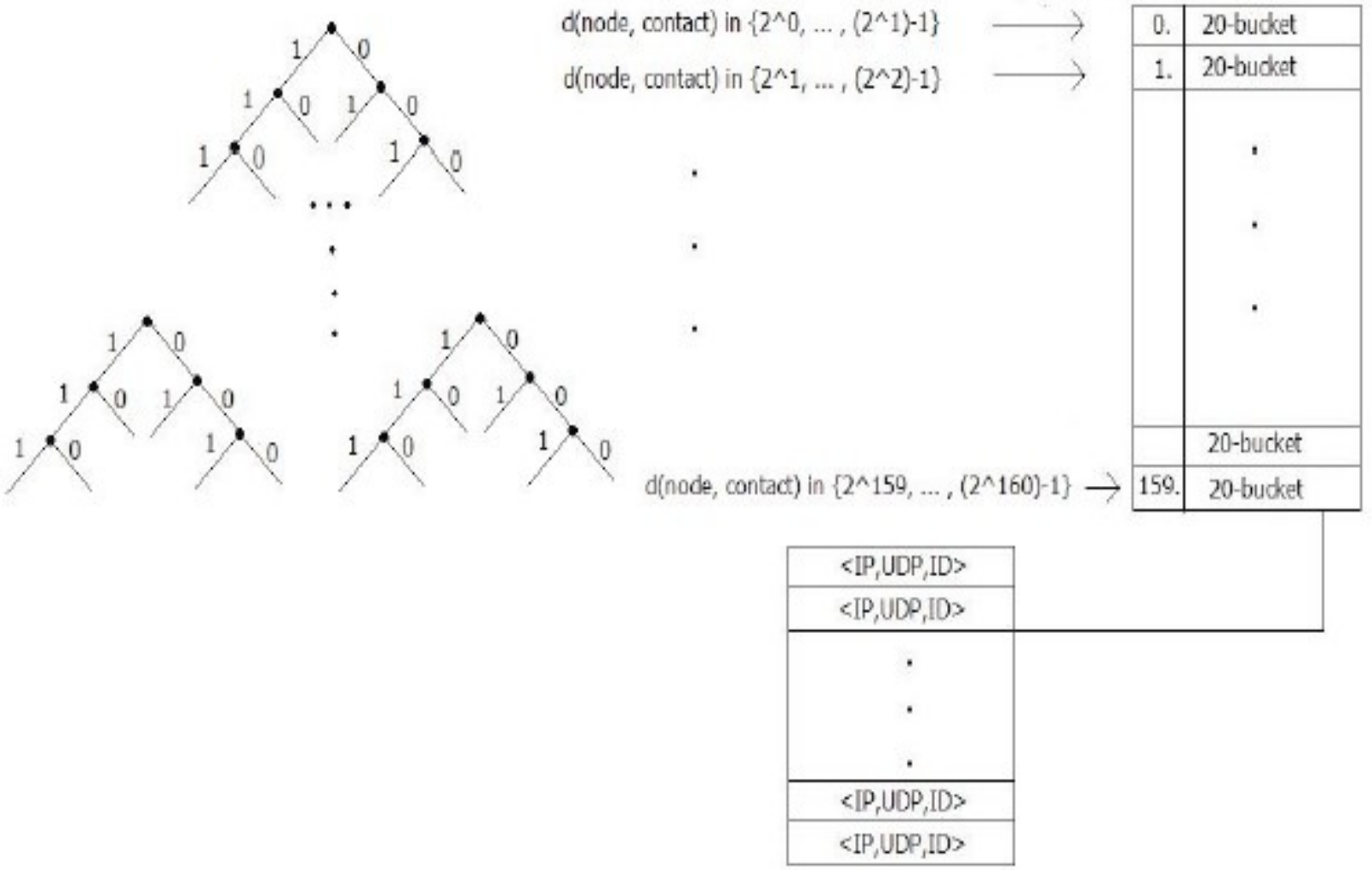
  <IP address, UDP port, Node ID>

- La tabella di routing contiene un numero di k-buckets pari a B, con B =160 (lunghezza degli ids).

- Ogni k-bucket contiene quindi riferimenti ad altri nodi che si trovano in un sottoalbero che condivide con il nodo un prefisso di lunghezza I.

- Il k-bucket di indice i del nodo n, per ogni $0 \leq i < 160$, contiene riferimenti a nodi che si trovano ad una distanza compresa tra 2^{i-1} e 2^i da n.

KADEMLIA: LE TABELLE DI ROUTING



KADEMLIA: LA TABELLA DI ROUTING

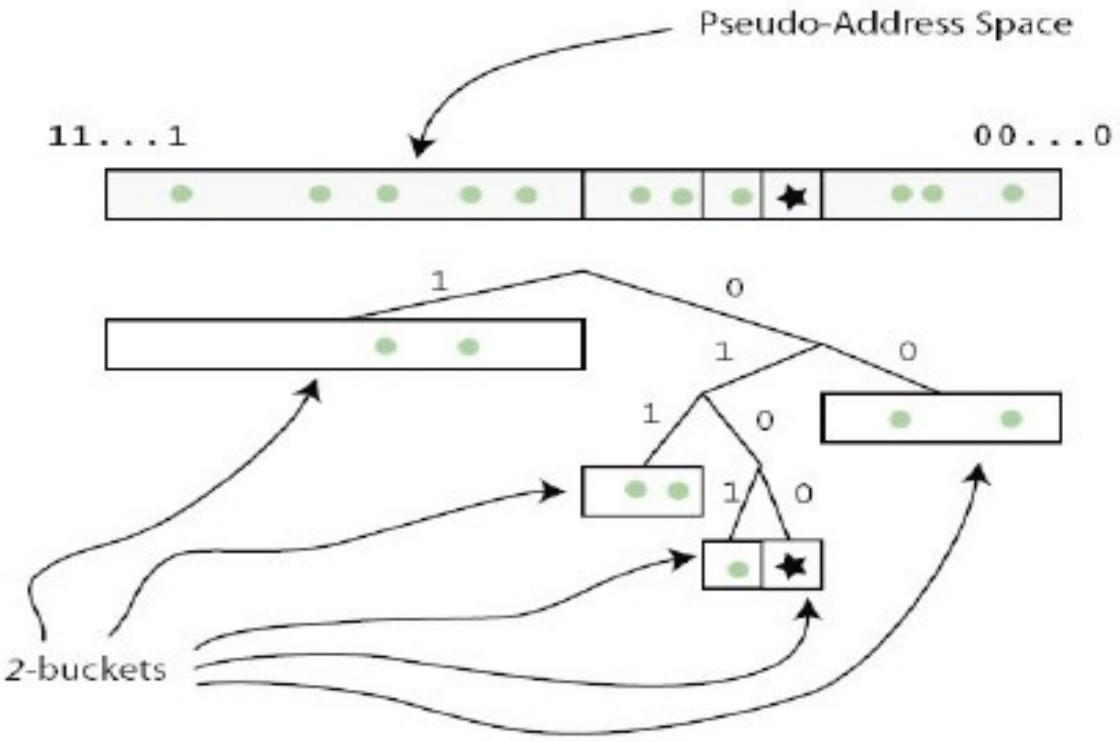
- ogni K bucket corrisponde ad un prefisso nell'albero degli identificatori e contiene nodi che presentano quel prefisso a comune con n
  - alcune entrate della tabella di routing contengono poche entrate
  - altri buckets contengono un maggior numero di contatti, ma mai più di K

- ogni K bucket copre un sottospazio dello spazio degli identificatori e l'insieme tutti i k-buckets copre l'intero spazio degli identificatori

- il valore di K viene stabilito in modo che la probabilità che si verifichi un crash di più di K nodi del sistema sia un evento altamente improbabile

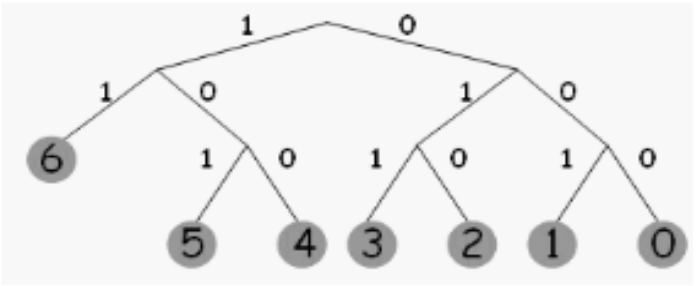
KADEMLIA: LE TABELLE DI ROUTING

Relazione tra tabella di routing ed albero degli identificatori



KADEMLIA: LA TABELLA DI ROUTING

Esempio: tabella di routing continente un insieme di 2-buckets per ogni riga



m=3 #bit chiave

Tabella nodo 0

0	[1,2)
1	[2,4)
2	[4,0)

nodo1

nodo2 → nodo3 } K-bucket

nodo4 → nodo5

K = numero massimo di elementi in un k-bucket:
parametro di sistema fisso (qui k=2)

KADEMLIA: GESTIONE DEI K-BUCKETS

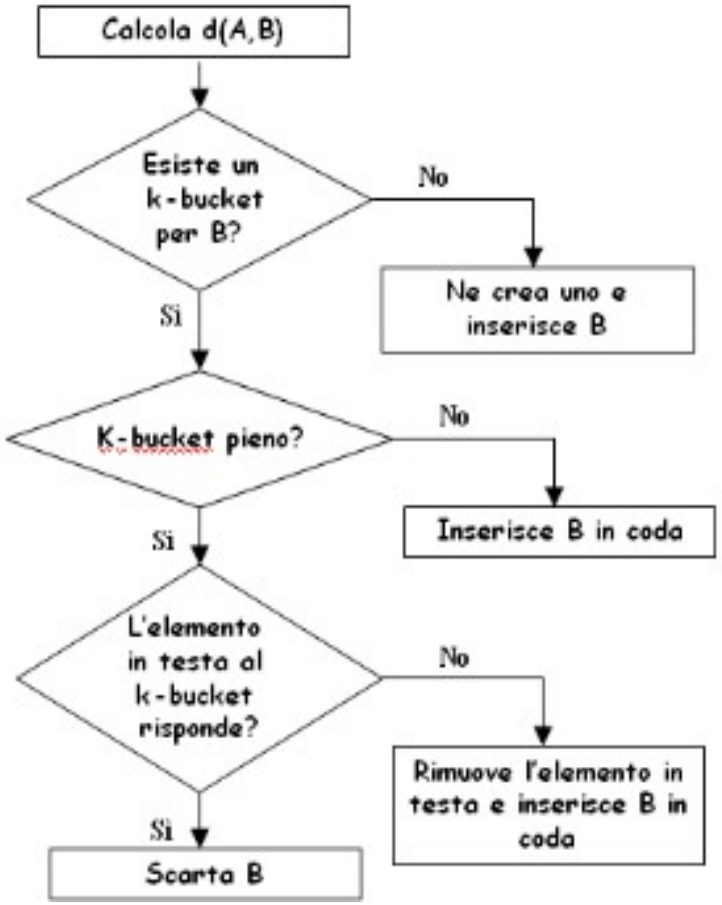
- Quando un nodo di Kademlia riceve un messaggio da un nodo M, può inserire il contatto M nel bucket B corrispondente

- Si possono verificare i seguenti casi :
  - B non contiene M ed B non è pieno, M viene aggiunto in coda a B
  - B non contiene M e B è pieno, N invia un ping al nodo P con cui ha avuto contatti meno recentemente (quello in testa alla lista)
    - se P non risponde entro un certo intervallo di tempo, N elimina dal bucket P e inserisce M in coda
    - se P risponde, M viene scartato e P viene spostato in fondo alla lista
    - nodi contattati meno di recente in testa alla lista

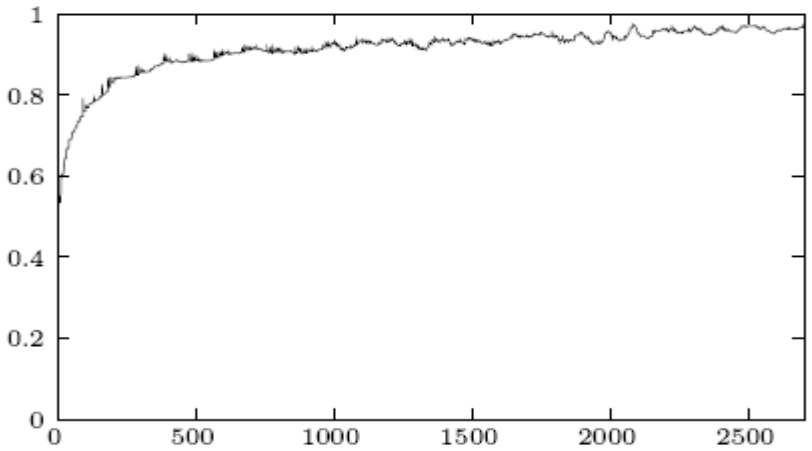
- Politica per la gestione dei k-buckets: least recently seen evinction
  - i nodi eliminati sono quelli in testa alla lista del k-bucket, cioè quelli contattati meno di recente
  - un nodo che fa ancora parte della rete non viene mai rimosso

KADEMLIA: GESTIONE DEI K-BUCKETS

Supponiamo che il nodo A riceva un messaggio da un nodo B

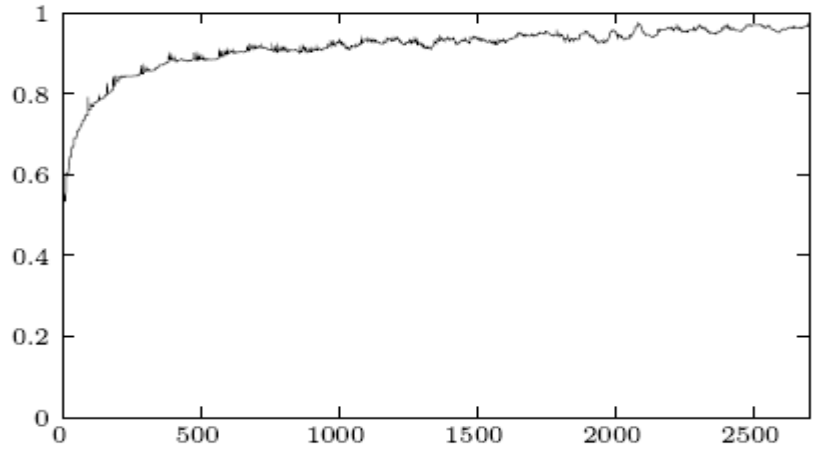


KADEMLIA: GESTIONE DEI K BUCKETS



- da una analisi della rete Gnutella
- asse x: minuti
- asse y: percentuale di nodi che, essendo stati on-line per x minuti, lo sono stati anche per x+60 minuti
    - percentuale di nodi di una rete Gnutella che rimangono online nell'ora successiva in funzione del loro uptime
    - più un nodo rimane online più è alta la probabilità che vi rimangno ancora per un lungo intervallo di tempo

KADEMLIA: GESTIONE DEI K BUCKETS



- l'analisi del grafico precedente è alla base della politica di Kademlia per la gestione dei contatti nei k-buckets

- preferire sempre i contatti che sono presenti sulla rete da una maggior quantità di tempo, perchè, con alta probabilità, rimarranno sulla rete anche successivamente

KADEMLIA: GESTIONE DEI K-BUCKETS

- **Esempio:**

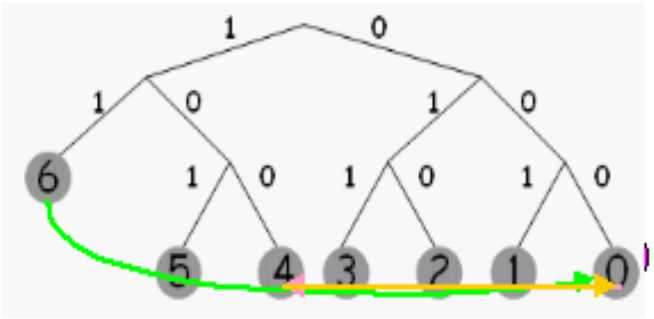


Tabella nodo 0

0	[1,2) → nodo1
1	[2,4) → nodo2 → nodo3
2	[4,0) → nodo4 → nodo5 → nodo4

m=3; k=2

1. 6 manda un messaggio a 0, ma il k-bucket è pieno (k=2)
1. 0 interroga 4
1. 4 risponde e viene spostato al fondo (6 è fuori)

KADEMLIA: GESTIONE DEI K-BUCKETS

- Esempio:

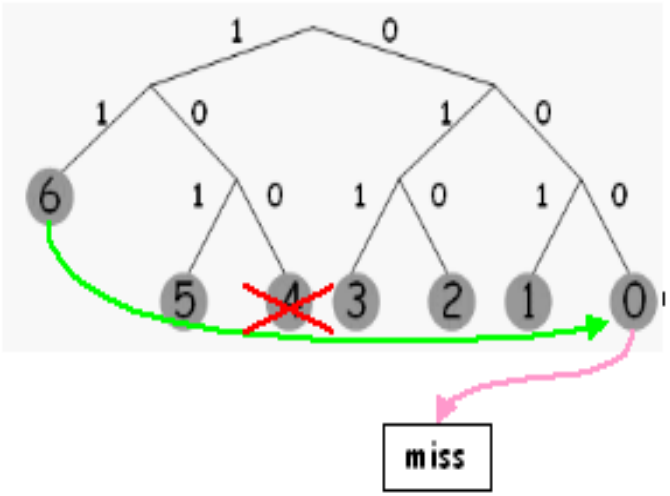


Tabella nodo 0

0	[1,2)
1	[2,4)
2	[4,0)

nodo1
nodo2
nodo3
nodo4
nodo5
nodo6

m=3; k=2

1. 6 manda un messaggio a 0, ma il k-bucket è pieno (k=2)
1. 0 interroga 4 (nodo in testa)
1. 4 non risponde: 6 è messo in fondo al k-bucket

KADEMLIA: REFRESH PERIODICO DEI K-BUCKETS

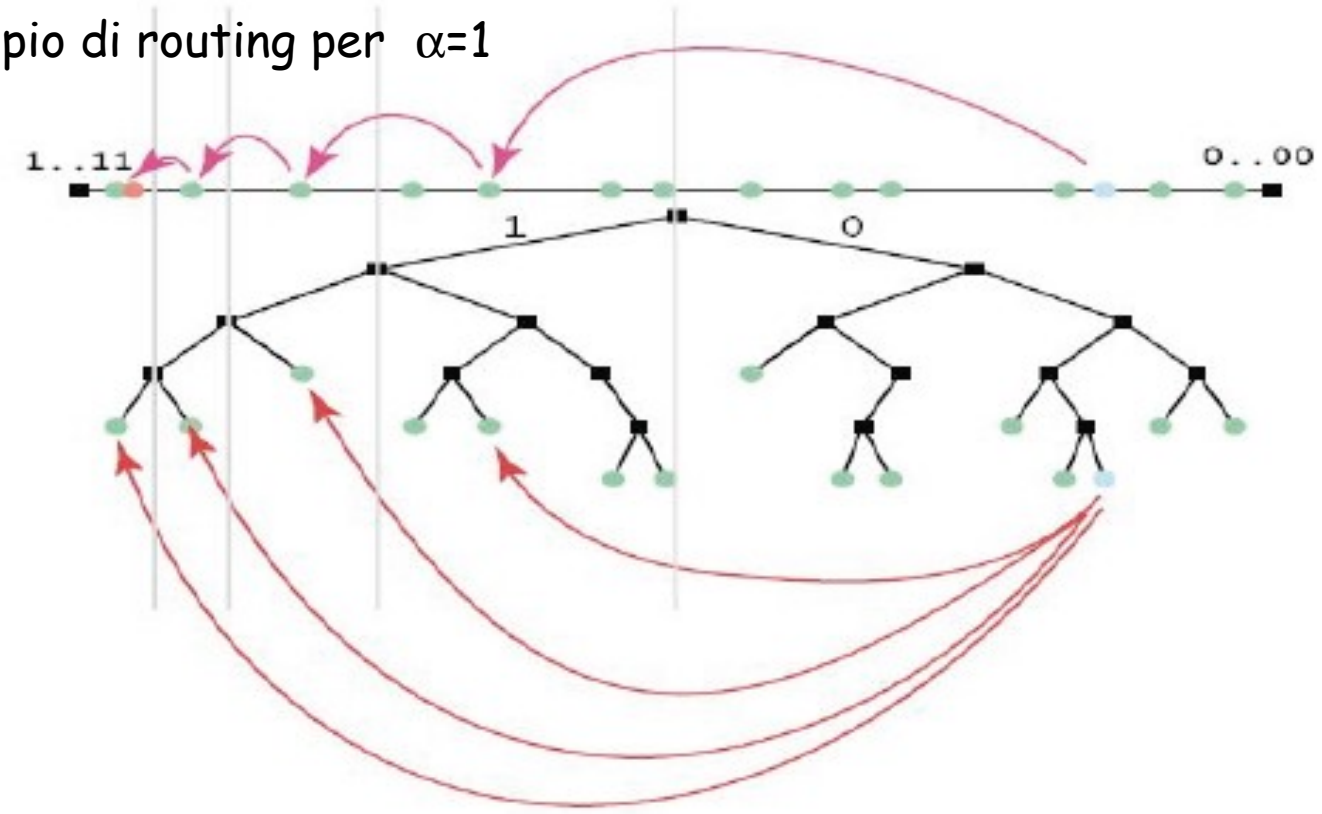
- i k-buckets sono rinfrescati quando viene gestita una query
  - anche se qualche nodo abbandona la rete, le nuove informazioni ricevute dai nodi a cui inoltro le query 'rinfrescano' la k-bucket list
- può tuttavia accadere che un k-bucket non sia rinfrescato per un certo periodo a causa della mancanza di richieste sui nodi del range coperto dal k-bucket
- Per questa ragione un refresh viene comunque effettuato periodicamente (una volta ogni ora)
  - si sceglie casualmente un identificatore all'interno del range coperto dal bucket e si effettua una ricerca per quell'identificatore

KADEMLIA VS. CHORD

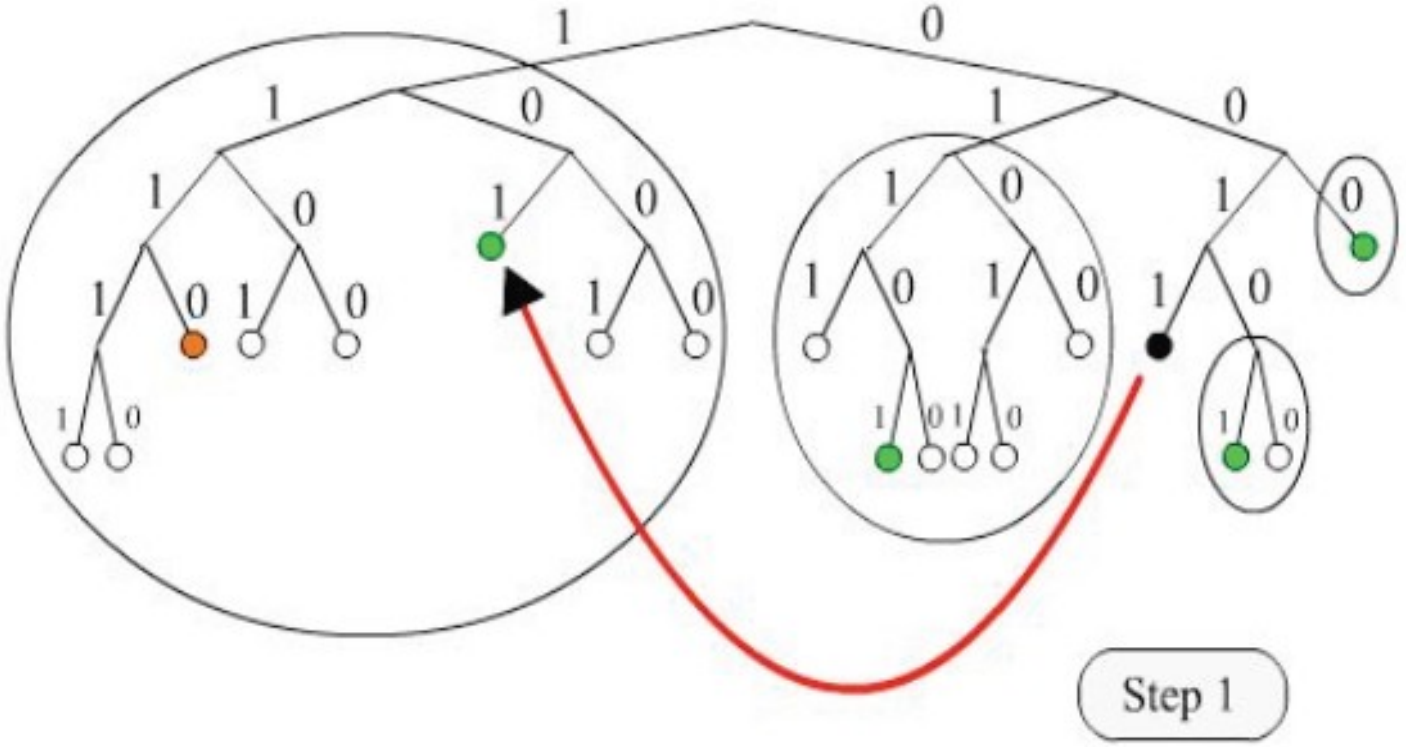
- La metrica simmetrica implica la possibiltà di sfruttare la conoscenza di tutti gli identificatori di nodi da cui si ricevono le query
  - Ogni nodo può arricchire la propria routing table mediante le informazioni contenute nelle query
- Al contrario, in Chord:
  - se un nodo x riceve una query dal nodo y, y possiede nella propria finger table un riferimento ad x, ma non necessariamente y è un finger da x
  - l'informazione contenuta nella query non può, in generale, essere utilizzata da a per arricchire la propria tabella di routing, perchè è possibile che nessun finger di a punti a b

KADEMLIA: IL ROUTING

- α= numero di nodi a cui viene propagata la query, ad ogni passo
- Routing iterativo
- Esempio di routing per α=1



KADEMLIA: IL ROUTING (α=1)

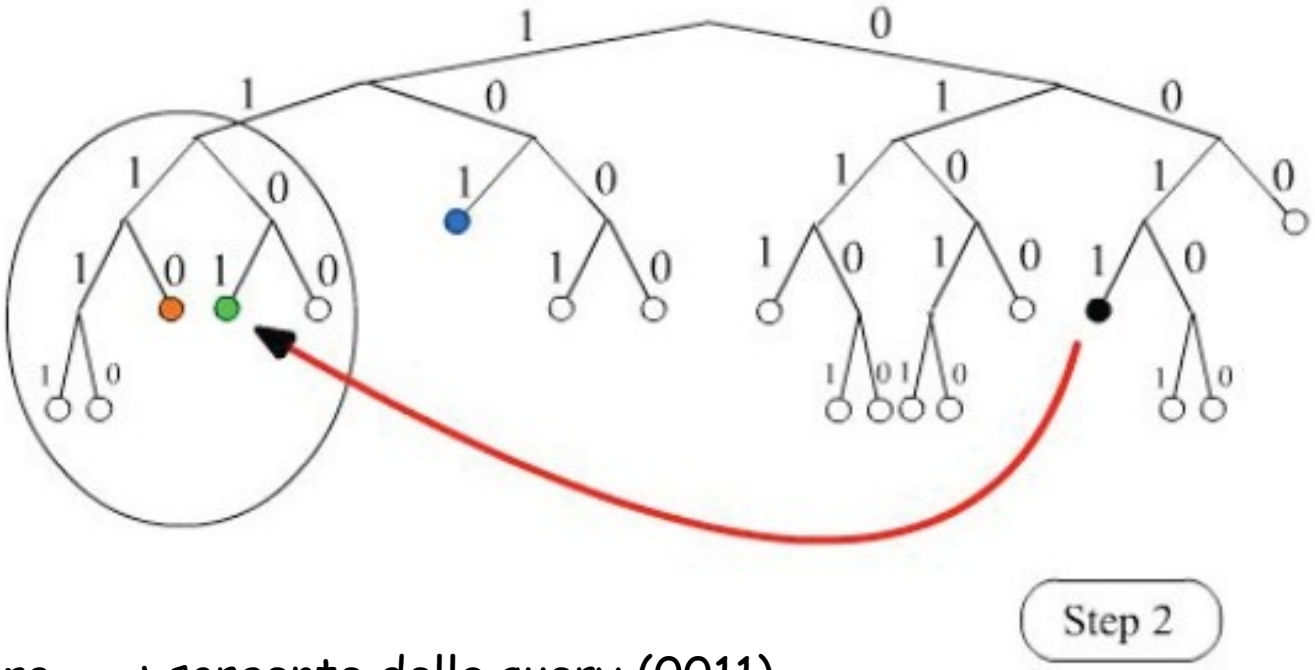


Nodo nero :sorgente della query (0011)

Nodo arancio : target della query (1110)

Nodi verde : nodi conosciuti dalla sorgente negli altri sottoalberi

KADMELIA: IL ROUTING(α=1)



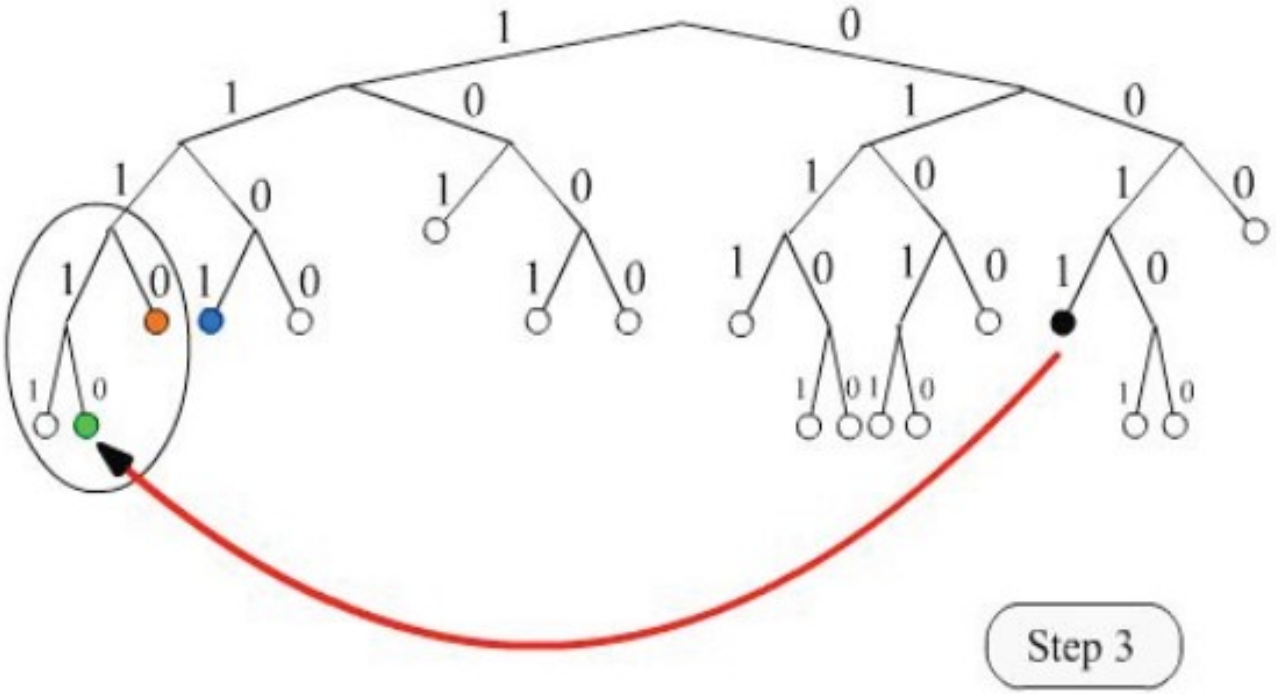
Nodo nero : sorgente della query (0011)

Nodo arancio : target della query (1110)

Nodi verde : nodi conosciuti dalla sorgente negli altri sottoalberi

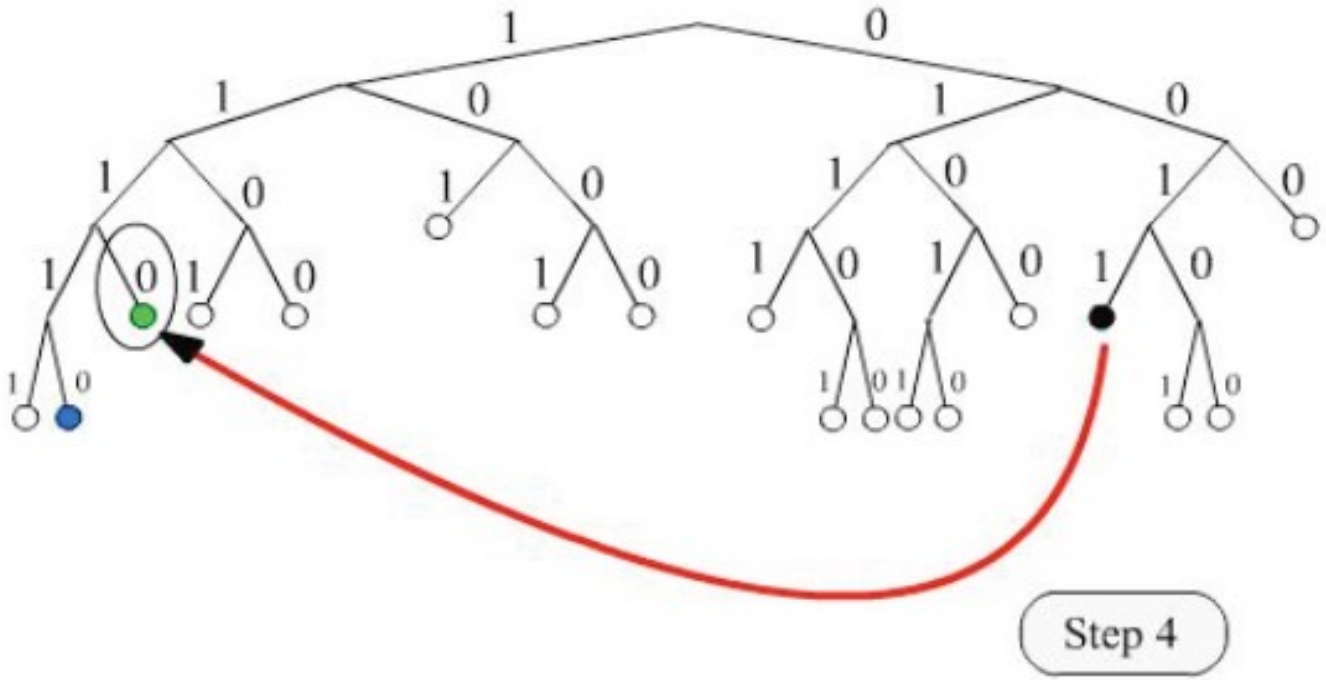
Nodo Blu : restituisce alla sorgente un ulteriore nodo da contattare (iterative routing)

KADMELIA: IL ROUTING(α=1)



Nodo nero : sorgente della query (0011)
Nodo arancio : target della query(1110)
Nodi verde : nodi conosciuti dalla sorgente negli altri sottoalberi
Nodo Blu : restituisce alla sorgente un ulteriore nodo da contattare

KADMELIA: IL ROUTING(α=1)



Nodo nero : sorgente della query (0011)

Nodo arancio : target della query (1110)

Nodi verde : nodi rappresentanti negli altri sottoalberi

Nodo Blu : restituisce alla sorgente un ulteriore nodo da contattare

Prefix match routing: ad ogni passo aumenta il prefisso a comune con la chiave

KADEMLIA: IL ROUTING

- Consideriamo un nodo x e altri due nodi y e z, con

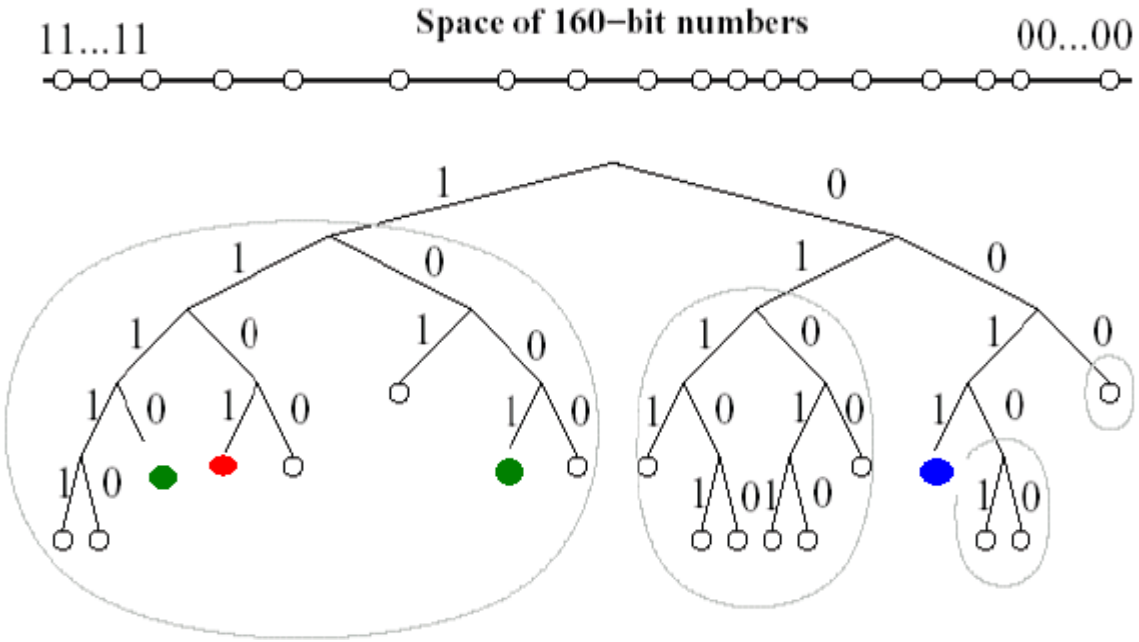
  dist(y,x)< dist(z,x).

  è possibile che z conosca x, y non conosca x. Questo è dovuto alla strategia con cui vengono creati i k-buckets, che prevede di costruire incrementalmente i bucket con l'informazione di cui si viene dinamicamente a conoscenza

- Non sempre l'invio della query al nodo più vicino al target porta al cammino più breve verso il target

- Routing: si invia la query agli $\alpha>1$ nodi più vicini al target. La proprietà della unidirezionalità garantisce che tutti i cammini convergano verso il target

IL ROUTING PARALLELO



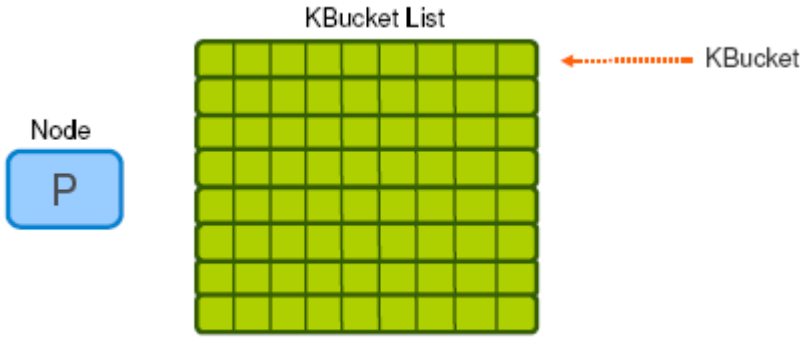
Supponiamo che il nodo blu 0011 ricerchi il nodo rosso 1101.
Possiede un riferimento ai nodi verdi 1001 e 1110, per cui vale

dist (1101, 1001) = 4

dist (1101, 1110) = 3

è possibile che il nodo più distante dal target, 1001, possieda un riferimento al target, mentre il nodo più vicino 1110 non lo possiede
Routing parallelo: il nodo blu invia la richiesta ad entrambe i nodi

KADEMLIA: IL ROUTING

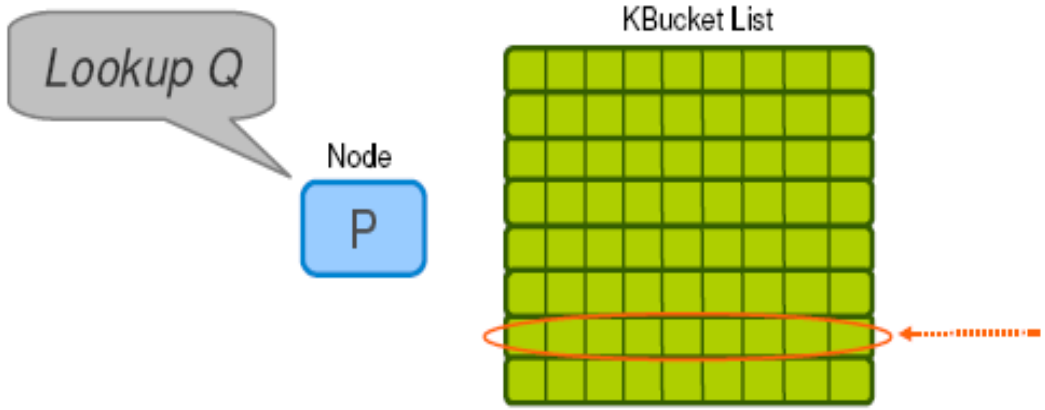


Consideriamo il caso generale $\alpha>1$

La query viene inviata in parallelo a α contatti

Il nodo si mantiene una lista di contatti da interrogare

KADEMLIA: IL ROUTING

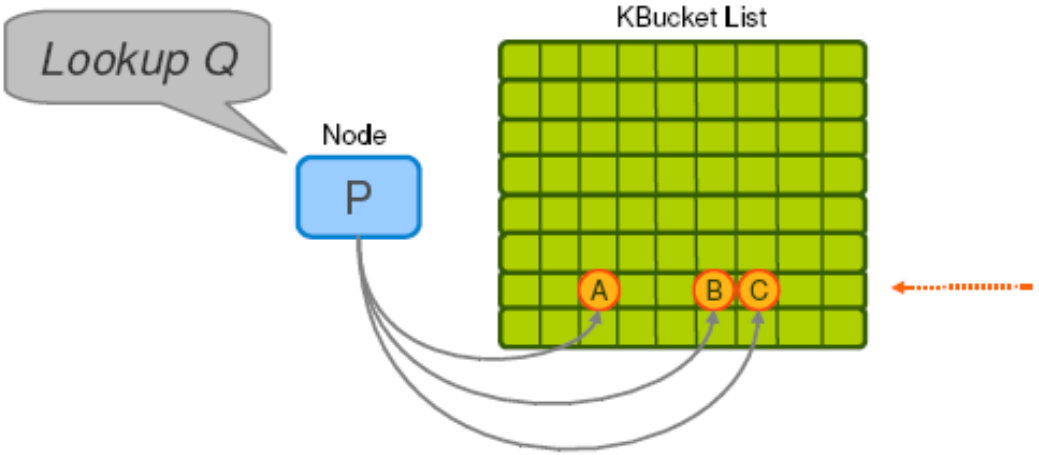


Memorizza i nodi più vicini a Q

- P ricerca la chiave Q (l'identificatore di un nodo o un dato)

- ricerca nella bucket-list gli α nodi più vicini a Q
- ricerca nel k-bucket non vuoto più vicino alla chiave. Se contiene meno di α nodi, ricerca nei bucket vicini
- i contatti selezionati possono appartenere anche a k-buckets diversi

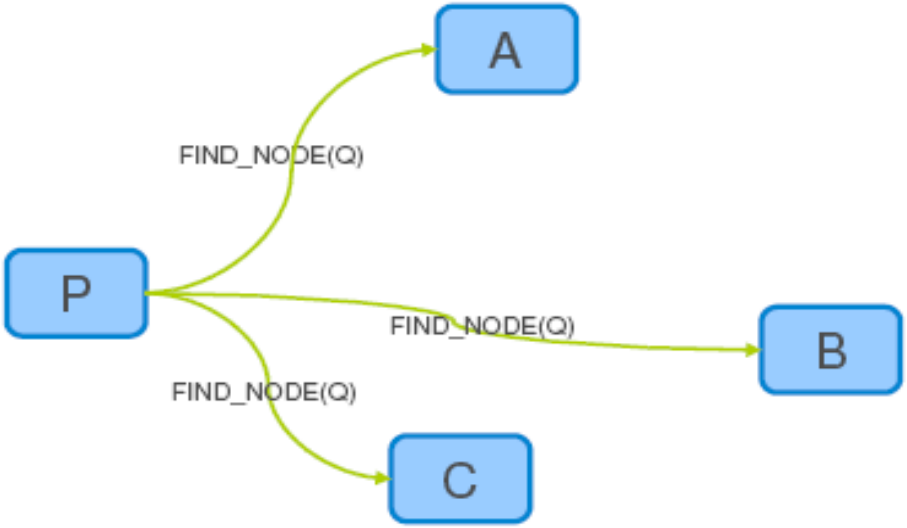
KADEMLIA: IL ROUTING



Memorizza i nodi più vicini a Q

- P seleziona α nodi dal bucket individuato

KADEMLIA:IL ROUTING

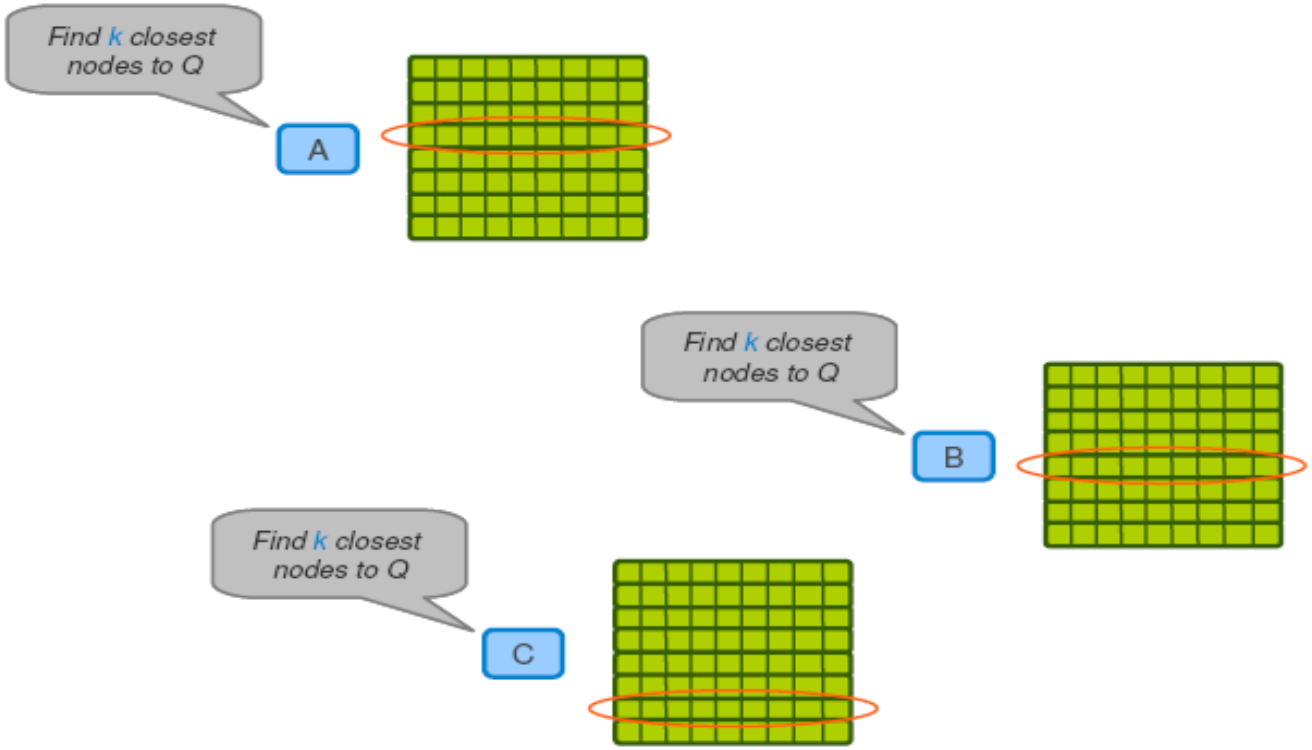


- P inoltra la query in parallelo a tutti i nodi individuati
- Si ricorda in una lista i contatti individuati

a differenza di Chord, la query è inviata in parallelo a più nodi

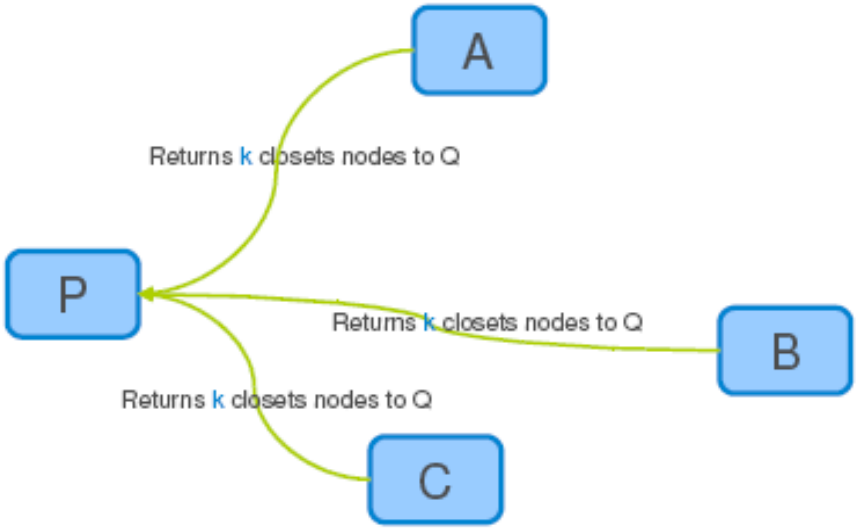
Messaggio FIND_NODE(Q)

KADEMLIA: IL ROUTING



- Ogni nodo contattato,individua a sua volta k nodi più vicini alla chiave
- Ogni nodo può sfruttare una diversa riga nella bucket list

KADEMLIA: IL ROUTING



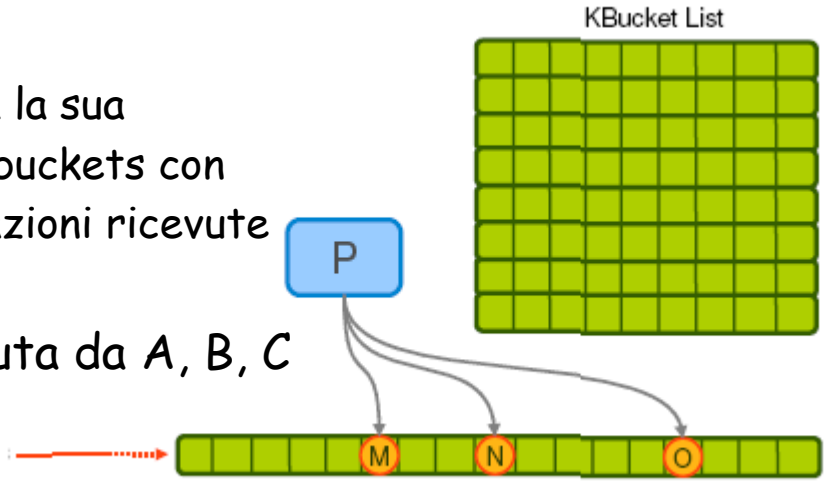
Routing Iterativo:

- ogni nodo restituisce i risultati individuati a P
- i risultati vengono inseriti in una lista che viene ordinata in base alla distanza tra i nodi restituiti e Q
- P continua il processo di routing con i risultati ricevuti

KADEMLIA: IL ROUTING

P aggiorna la sua
lista di k-buckets con
le informazioni ricevute

Informazione ricevuta da A, B, C

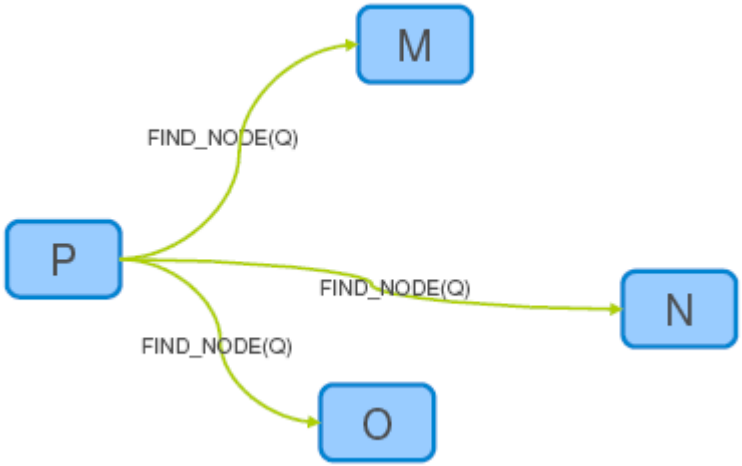


P seleziona nuovamente α nodi dalla
informazione ricevuta

se si sono ottenuti nuovi nodi più vicini al target
rispetto ai precedenti, si interrogano quei nodi

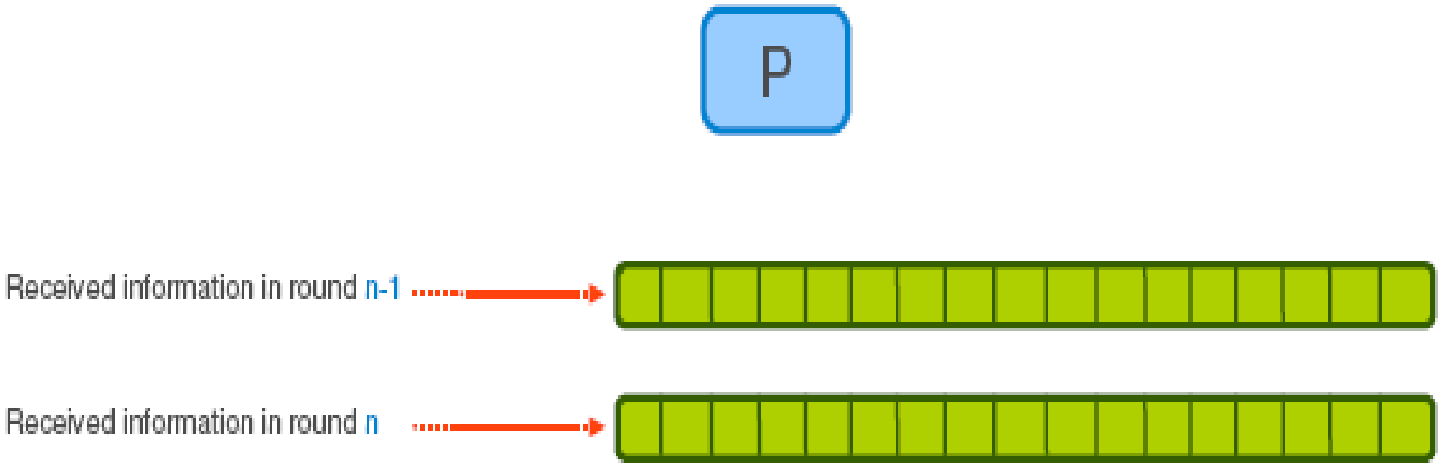
Altrimenti si scelgono globalmente
altri a nodi tra quelli non ancora interrogati

KADEMLIA:IL ROUTING



- P inoltra la richiesta ai nuovi nodi individuati

KADEMLIA:IL ROUTING



La sequenza di ricerche parallele continua fino a che

nessun nodo restituito è più vicino del nodo più vicino già contattato

sono stati contatti k nodi

JOIN, LEAVE

- Per effettuare un join ad un'overlay Kademlia, un nodo n deve conoscere qualche nodo w appartenente all'overlay. Si utilizzano nodi di bootstrap

- Procedura di join:
  - n inserisce w nel bucket relativo
  - effettua un look up di se stesso (con il suo ID) attraverso w
    - in questo modo trova alcuni nodi vicini a se stesso e li inserisce nei k-buckets. Si riempiono così i k-buckets di indice basso
    - il nodo si 'fa conoscere' dagli altri nodi della rete
    - successivamente effettua look-up per nodi scelti casualmente nei k-buckets di indice più alto
    - i k-buckets vengono poi arrichiti attraverso l'informazione contenuta nelle query che passano attraverso il nodo

- L'uscita di un nodo non richiede altre operazioni
  - se un nodo non risponde, esso viene eliminato dai k-bucket dei peer che gli inviano un ping

KADEMLIA: IL PROTOCOLLO

Chiavi replicate su k nodi per evitare la perdita di informazioni

- PING: utilizzato per verificare la presenza di un nodo
- FIND_NODE (ID, k): dato un identificatore logico ID appartenente allo spazio logico di Kademlia, restituisce k triple (Indirizzo IP, porta UDP, ID logico) corrispondenti ai k nodi più vicini al nodo
- FIND_VALUE (ID): si comporta come la FIND_NODE e restituisce il valore associato alla chiave
- STORE (KEY,VALUE): individua i k nodi più vicini alla chiave e memorizza su di essi la coppia chiave, valore

TASK PERIODICI

- Per assicurare la persistenza dei dati inseriti nell'overlay, ogni nodo ripubblica periodicamente le coppie <chiave,valore>

- Il meccanismo di pubblicazione periodica dei dati è introdotto per
  - evitare la perdita di dati in conseguenza dell'uscita volontaria o dovuta al guasto di un nodo dall'overlay

- Definite alcune ottimizzazione per diminuire il numero di messaggi scambiati
  - Se un nodo riceve una STORE può supporre che la STORE sia stata inviata ad altri k-1 nodi vicini e non ripubblica la chiave nell'ora successiva

CONCLUSIONI

Confronto Chord vs. Kademlia

- Tabelle di Routing
  - In Chord la routing table è rigida e richiede costose operazioni nel caso di fallimento di un nodo ed in caso di uscita volontaria di un nodo
  - Kademlia offre una routing table flessibile:
    - Metrica simmetrica per la distanza
    - Esistono percorsi alternativi verso un nodo
    - Le ricerche sono parallelizzate
    - La manutenzione della routing table ha un costo inferiore
    - È possibile memorizzare il round-trip-time assieme ad ogni contatto per scegliere i nodi raggiungibili con minore latenza

- Routing
  - Kademlia $O(\log(N))$
  - Chord $O(\log(N))$