Lezione n.6
DISTRIBUTED HASH TABLES: CAN

Laura Ricci

13/03/2009

Bulford, capitolo 4

CONTENT ADDRESSABLE NETWORK

- Proposto da ricercatori dell'Università della California, Berkeley e dell'AT&T

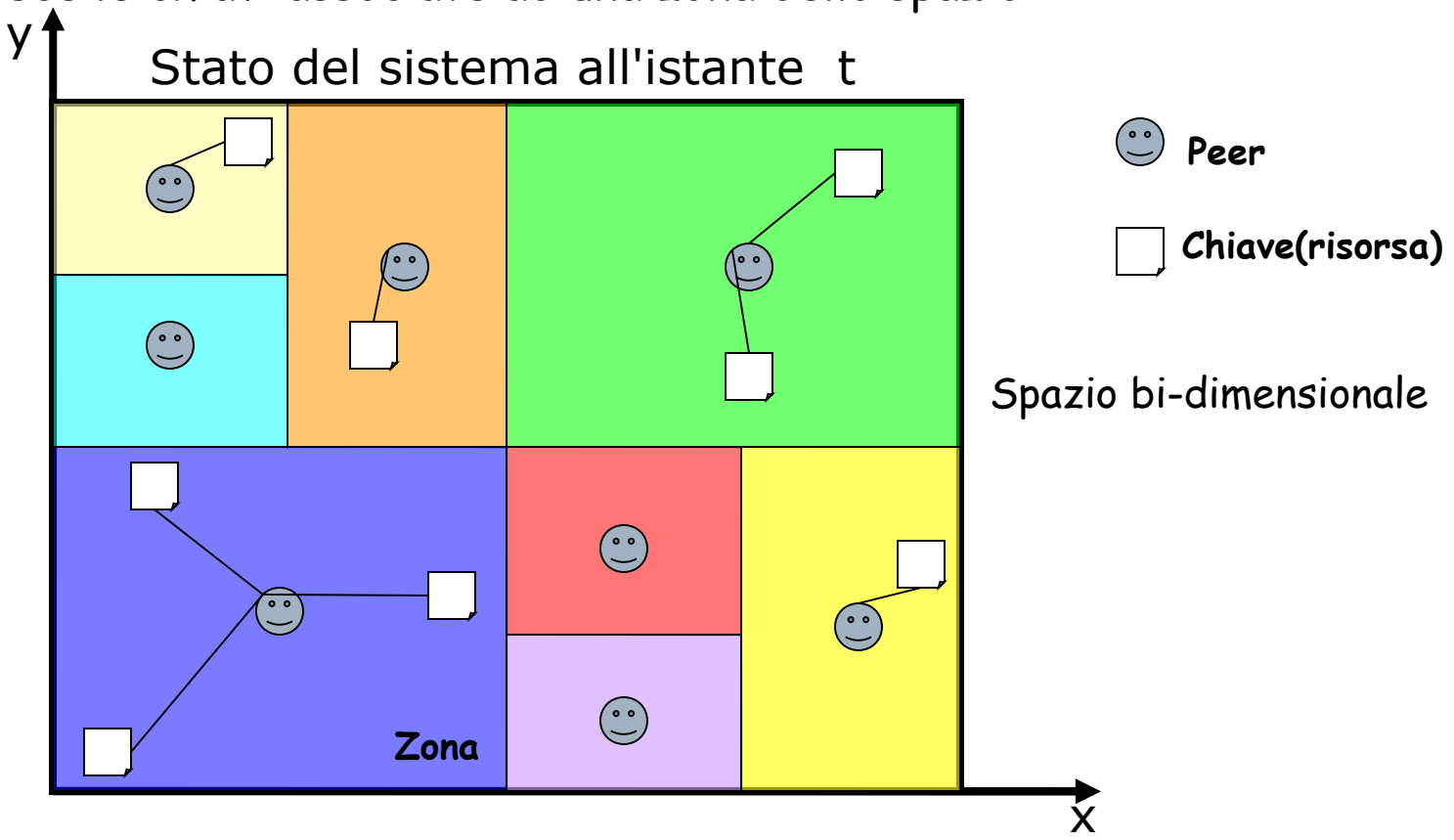
- Materiale: A Scalable Content-Addressable Network, Sylvia Ratnasamy, P.Francis, M.Handley, R. Karp, S. Shenker " proceedings ACM SIGCOM 2001 Applications, Technologies, Architectures, and Protocols for Computer Communication

- Proposto nel 2001

- Diverse implementazioni esistenti

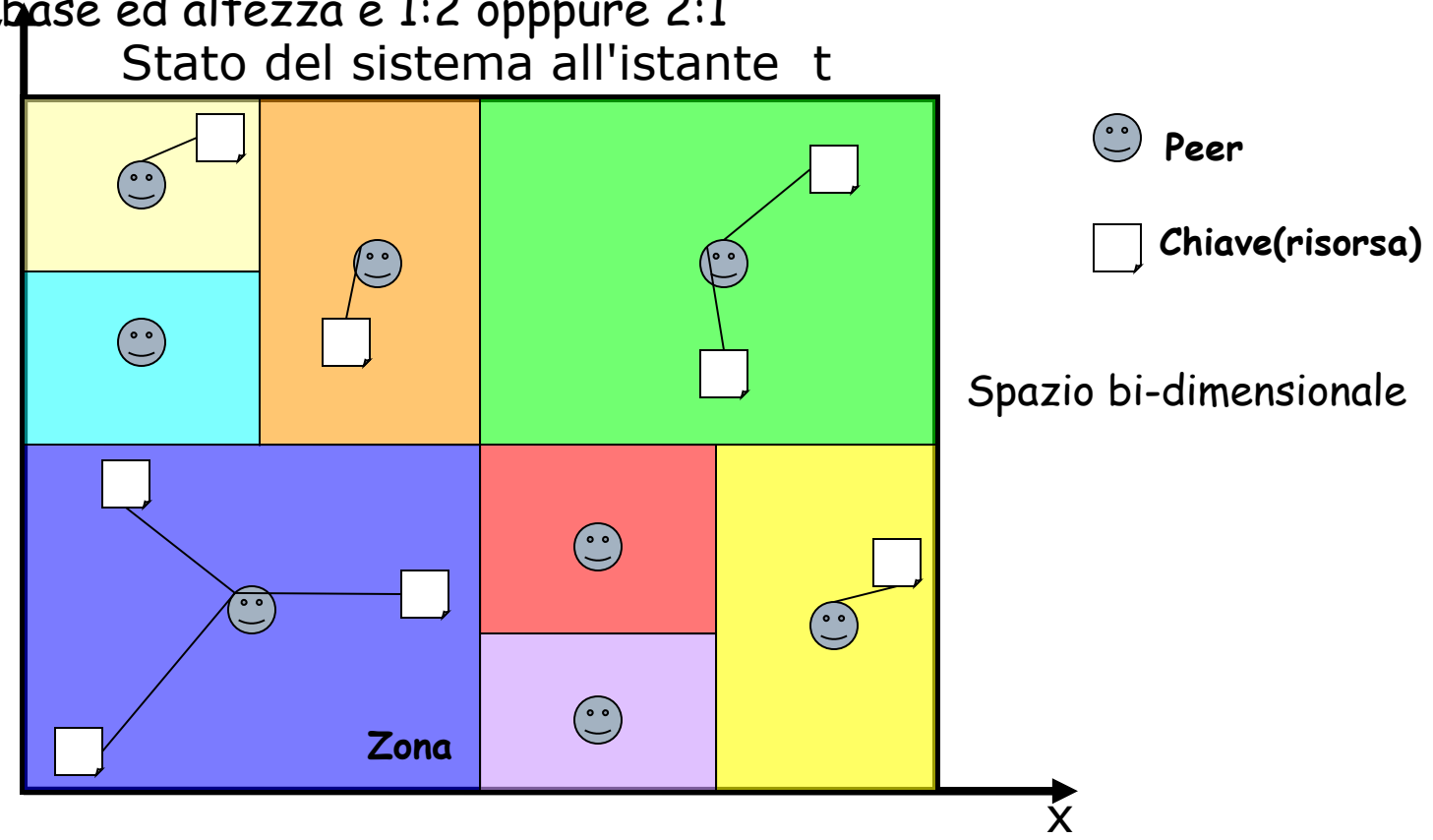
CAN: IDEE DI BASE

- associa ad ogni nodo e ad ogni informazione un punto in uno spazio cartesiano d-dimensionale, con richiusure toroidali
- ogni peer gestisce le chiavi associate ad una zona dello spazio



CAN: IDEE DI BASE

- Lo spazio è diviso tra i nodi e viene interamente coperto dai peer
- Ogni nodo copre un'area quadrata oppure un'area rettangolare in cui il rapporto tra base ed altezza è 1:2 opppure 2:1



CONTENT ADDRESSABLE NETWORK

- Obiettivi
  - Scalabilità a centinaia di migliaia di nodi
  - Gestione efficiente di inserimento e caduta di nodi

- Caratteristiche per una rete a d dimensioni
  - dimensione della Tabella di Routing $O(d)$
  - numero medio di passi necessari per individuare un'informazione (routing hops) $d * n^{1/d}$, dove n è il numero totale di nodi

CAN: CONTENT ADDRESSABLE NETWORK

- Per ricavare le coordinate di un nodo o di un dato nello spazio d-dimensionale
  - Applicazione di una funzione hash all'identificatore del nodo o al dato
  - La funzione hash mappa il nodo/dato in una sequenza di bits
  - La sequenza di bits ottenuta è partizionata in d sottosequanze. Ogni sottosequenza rappresenta una coordinata nello spazio a d dimensioni

- Le coordinate in ogni dimensione variano in un intervallo di valori

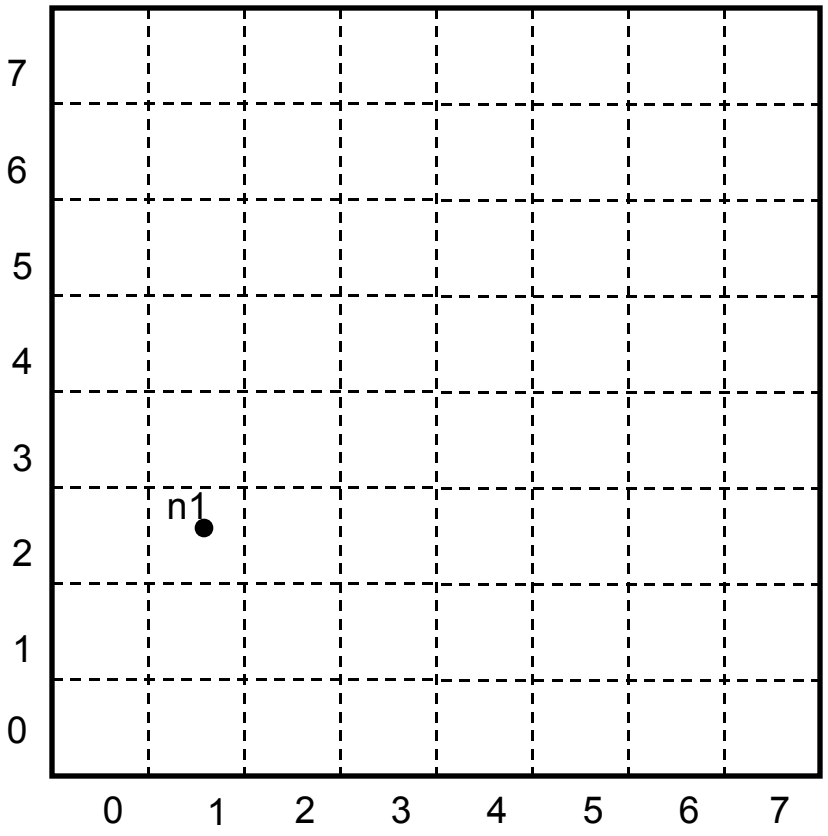
- Tutte le operazioni sono calcolate in modulo rispetto alla valore più grande della coordinata in una dimensione

- Lo spazio degli indirizzi CAN può essere rappresentato come un toro a d dimensioni

CAN: PARTIZIONAMENTO DELLO SPAZIO

- Il primo nodo gestisce l'intero spazio delle chiavi
- Esempio: Il nodo n1 mappato nel punto

(1, 2) è il primo nodo che entra nel sistema e ad esso viene associato l'intero spazio bi-dimensionale

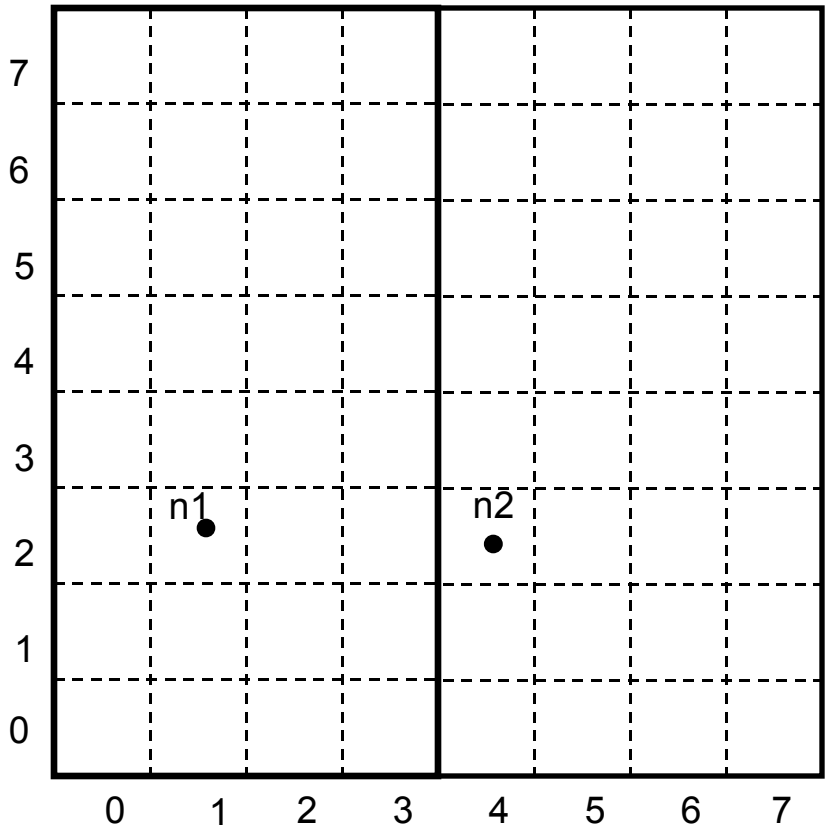


CAN: PARTIZIONAMENTO DELLO SPAZIO

Il nodo n2:(4, 2) si unisce alla rete CAN

$\Rightarrow$

lo spazio viene partizionato tra n1 ed n2

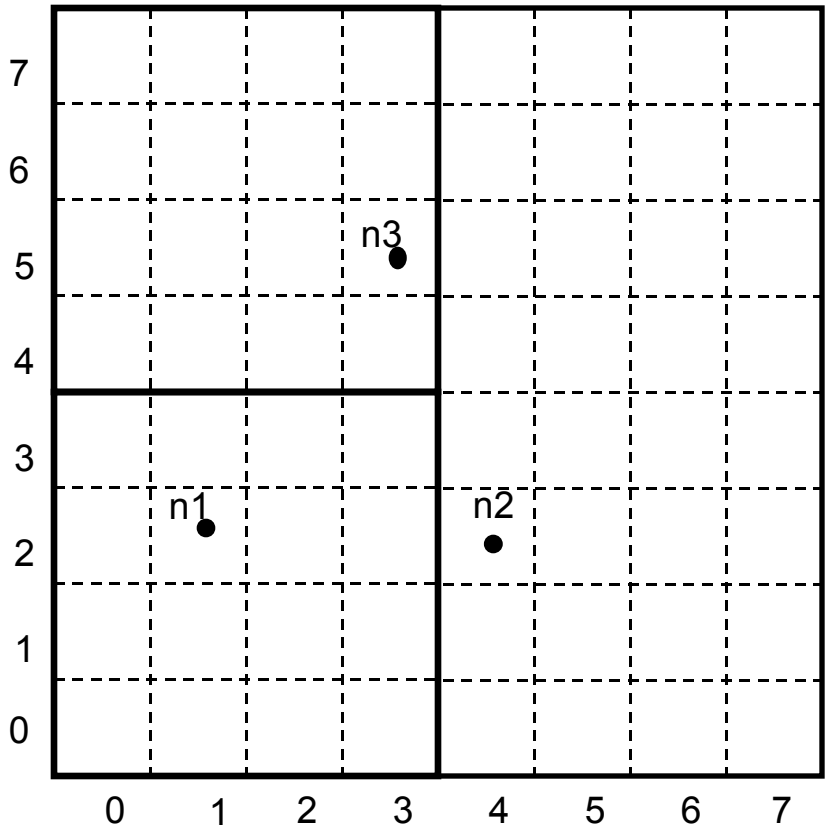


CAN: PARTIZIONAMENTO DELLO SPAZIO

il nodo n3:(3, 5) si unisce alla rete

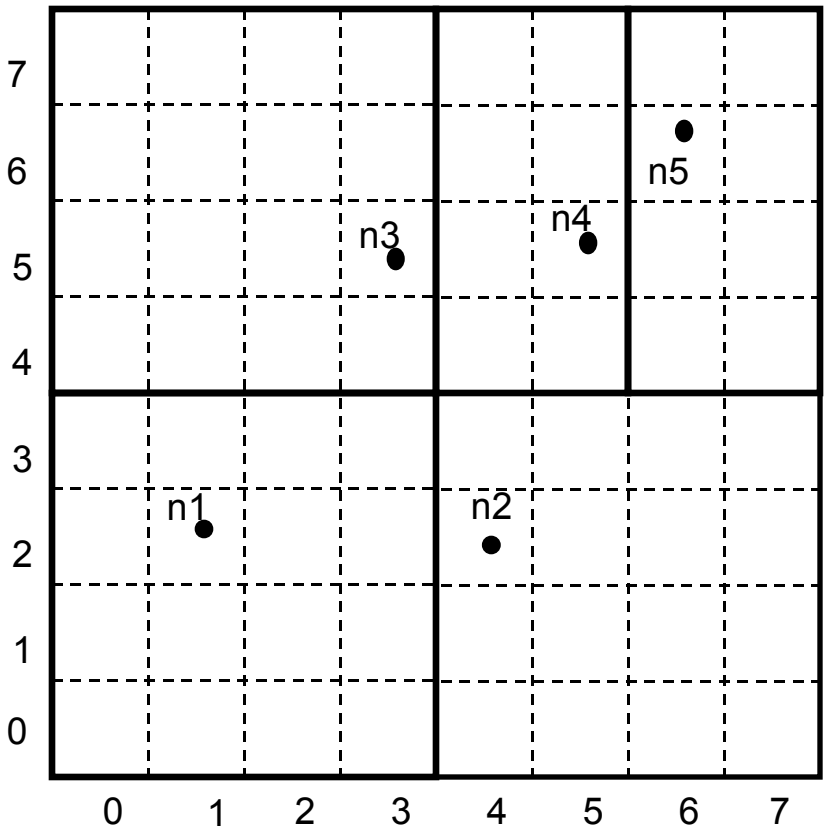
$\Rightarrow$

lo spazio viene diviso tra n1 ed n3



CAN: PARTIZIONAMENTO DELLO SPAZIO

n4:(5, 5) and n5:(6,6) si uniscono alla rete CAN



CAN: ASSOCIAZIONE DELLE CHIAVI AI NODI

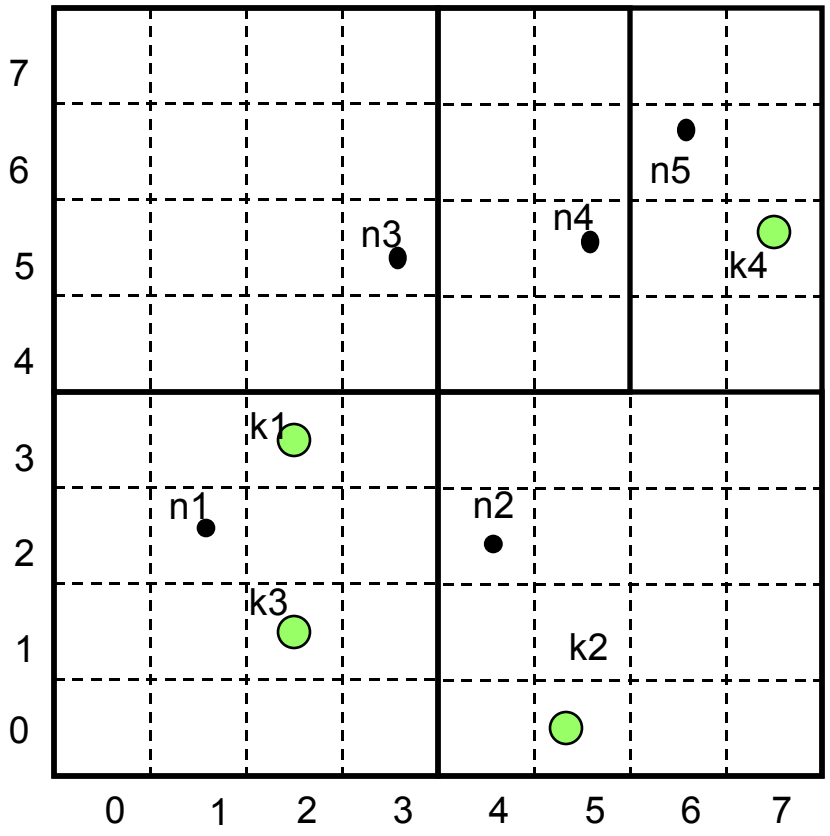
Ogni chiave viene memorizzato dal nodo che gestisce la zona in cui è stato mappata la chiave stessa

Nodi:

n1:(1, 2); n2:(4,2); n3:(3, 5); n4:(5,5);n5:(6,6)

Chiavi

k1:(2,3); k2:(5,1); k3:(2,1); k4:(7,5);



CAN: ASSOCIAZIONE DELLE CHIAVI AI NODI

Ogni chiave viene memorizzato dal nodo che gestisce la zona in cui è stato mappata la chiave stessa

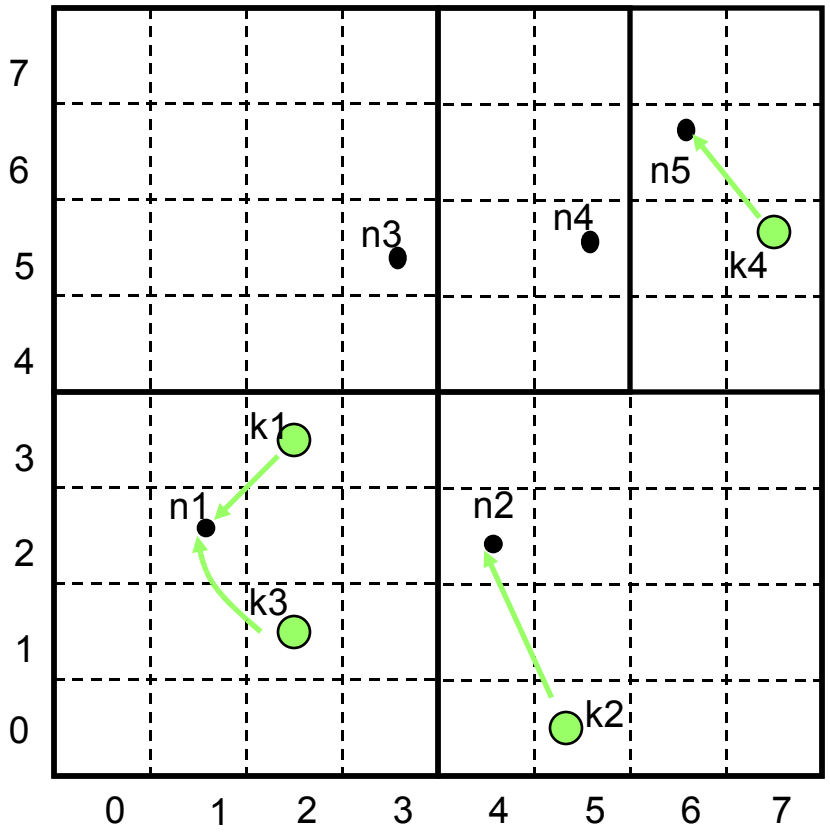
Nodi:

n1:(1, 2); n2:(4,2); n3:(3, 5);

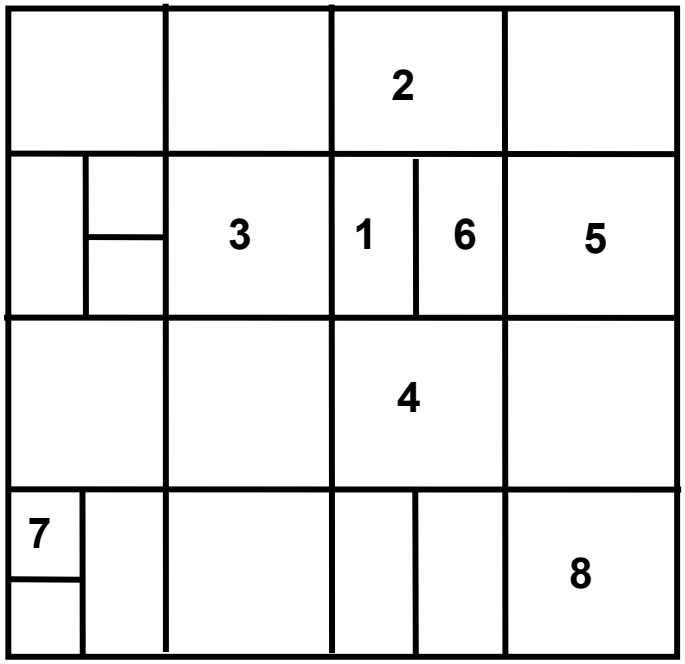
n4:(5,5);n5:(6,6)

Chiavi

k1:(2,3); k2:(5,1); k3:(2,1); k4:(7,5);



CAN: DEFINIZIONE DEI VICINI

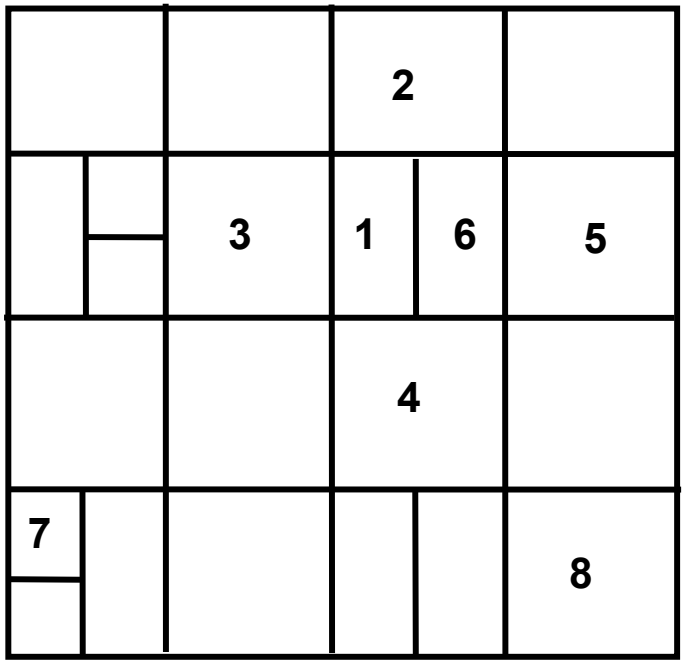


- Due nodi sono considerati vicini se e solo se le rispettive zone "sono in contatto" in più di un punto
  - due zone in uno spazio a D dimensioni sono in contatto se i loro lati sono adiacenti rispetto ad una dimensione e si sovrappongono rispetto alle rimanenti d-1 dimensioni

•Ad esempio il nodo 5 è adiacente al nodo 6: i lati delle rispettive zone sono adiacenti rispetto all'asse x e si sovrappongono lungo l'asse y

•i nodi 4 e 5 non sono vicini: i loro lati sono adiacenti in entrambe le dimensioni, ma non si sovrappongono

CAN: TABELLE DI ROUTING



- D=2
- vicini di 1: 2,3,4,6
- vicini di 6: 1,2,4,5
- considerare le richiusure:
  7 ed 8 sono considerati vicini

Tabelle di Routing contiene un elemento per ogni vicino P e memorizza

- l'indirizzo IP e la porta di P
- le coordinate della zona che P gestisce

CAN: TABELLA DI ROUTING

- Il valore restituito dalla funzione hash è considerato un punto in uno spazio cartesiano a D dimensioni
- Ogni nodo è responsabile di una zona "cubo a D dimensioni"
- Lo spazio a D dimensioni è uno spazio logico e non presenta alcuna relazione con lo struttura della rete fisica sottostante
- Dimensione della tabella di routing: se si considera uno spazio a D dimensioni diviso in n zone della stessa ampiezza, ogni nodo deve mantenere 2D vicini
- Approssimazione: distribuzione uniforme dei punti nello spazio. In questo caso, al crescere del numero di nodi del sistema, l'informazione richiesta per il routing è O(D).

CAN: IL ROUTING

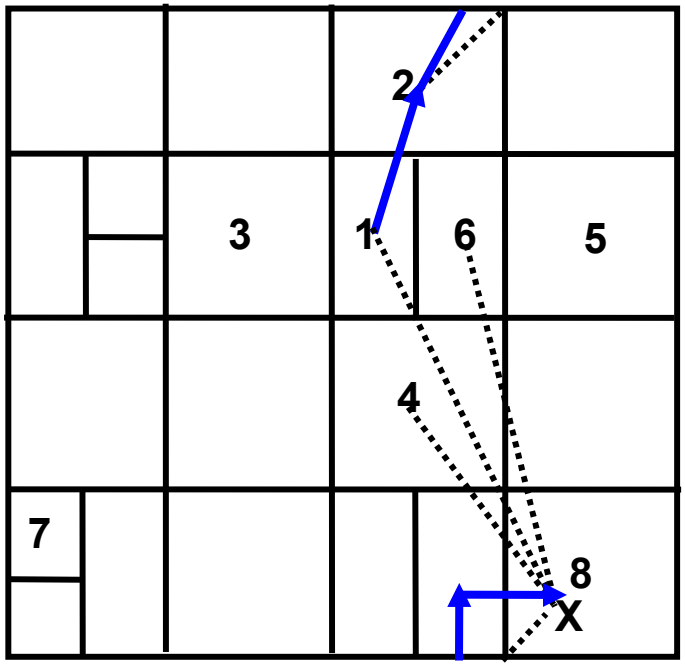
- Ogni messaggio CAN contiene le coordinate di un punto P destinazione ad esempio le coordinate associate dalla funzione hash ad una chiave.

- Ogni nodo instrada il messaggio verso il nodo che gestisce la zona che contiene P.

- Routing: Approssimazione di "seguire il segmento che collega sorgente e destinazione"

- Algoritmo di routing (strategia greedy):
  - Il nodo n inoltra un messaggio diretto verso il punto P di coordinate (c1,c2,…..,cD) verso il vicino che gestisce la zona più vicina a P.
  - La distanza tra una zona ed il punto P viene misurata rispetto al punto centrale della zona.

CAN: IL ROUTING



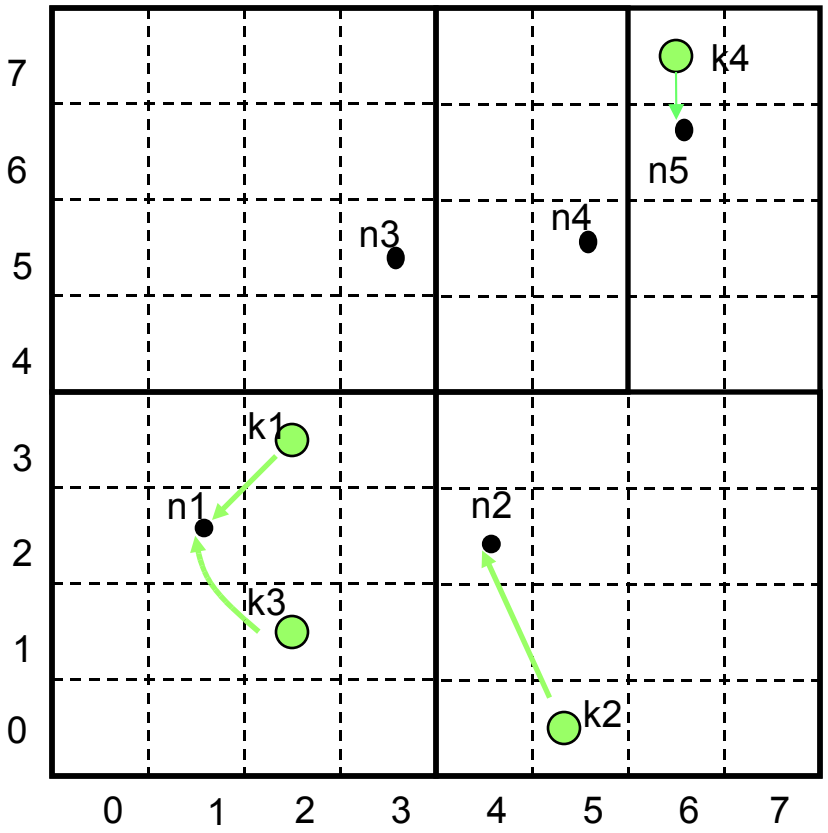
- Esempio: il nodo che gestisce la regione 1 deve ricercare la chiave X
  - controlla le distanze da X di tutte le zone associate ai vicini
  - inoltra il messaggio al nodo 2 (considerare le richiusure toroidali)

- la distanza cartesiana dal punto destinazione decresce ad ogni hop

CAN: IL ROUTING

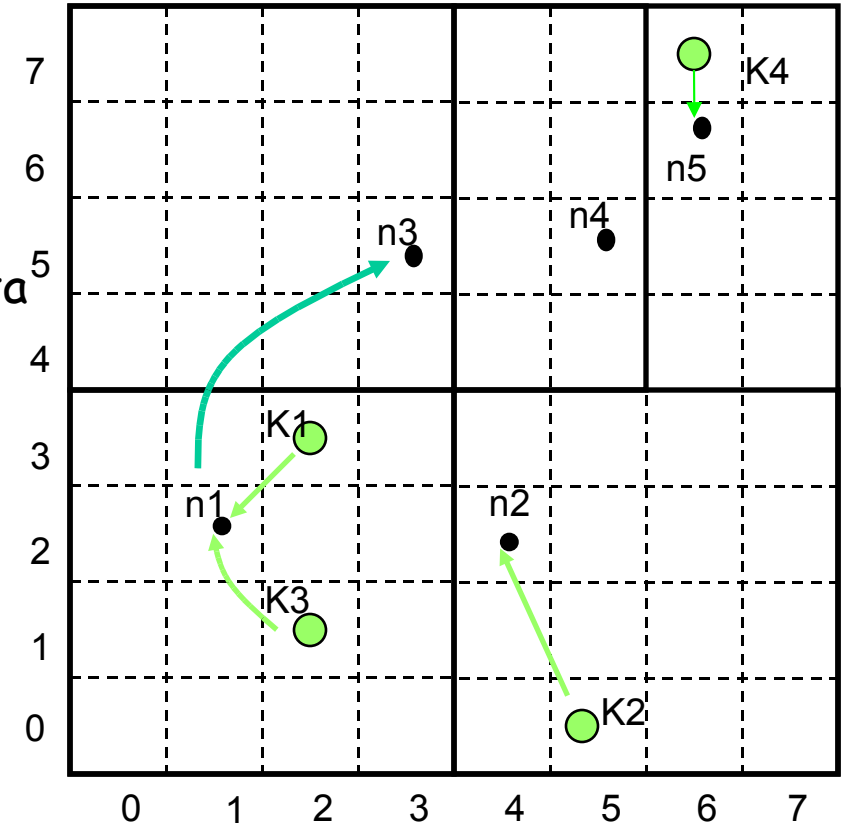
- Ogni nodo conosce i suoi vicini nello spazio a d dimensioni
- La query viene inoltrata al vicino che possiede la zona le cui coordinate risultano più vicine a quelle del dato ricercato
- Esempio:

  n1 ricerca k4



CAN: IL ROUTING

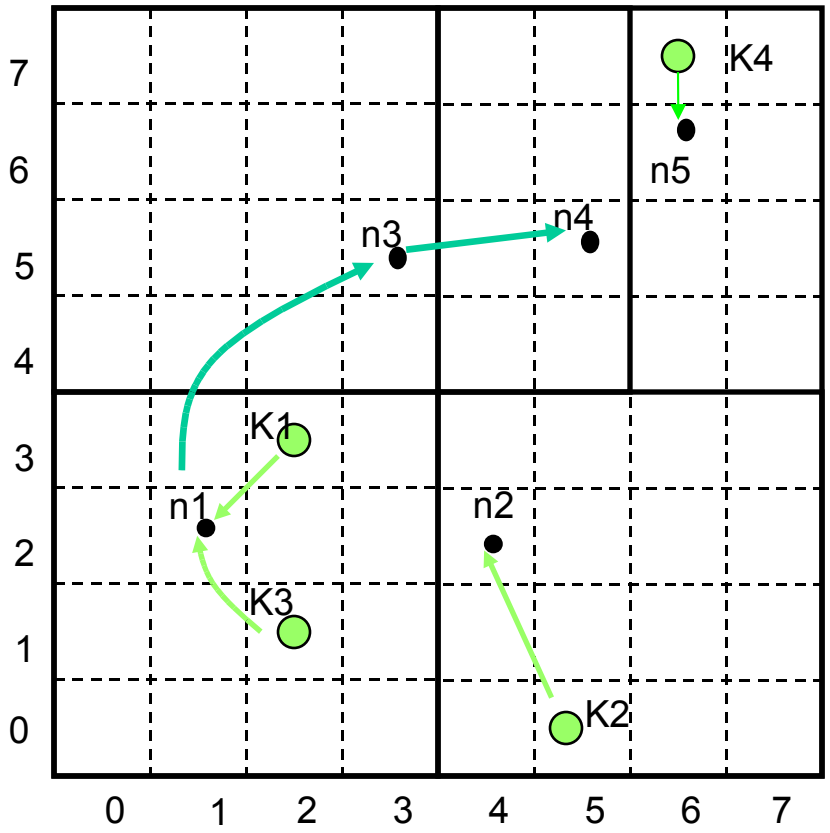
- Ogni nodo è a conoscenza dei suoi vicini nello spazio a d dimensioni
- La query viene inoltrata al nodo che risulta più vicino all'id della query
- Esempio:
  - n1 ricerca k4



CAN: IL ROTUTING

- Ogni nodo è a conoscenza dei suoi vicini nello spazio a d dimensioni
- La query viene inoltrata al nodo che risulta più vicino all'id della query
- Esempio:

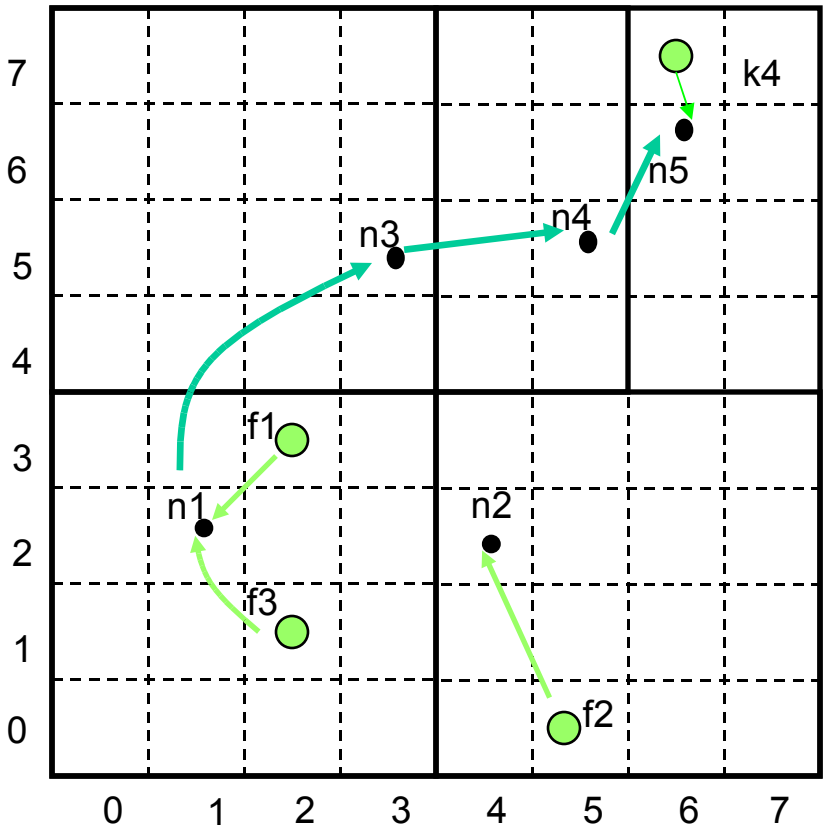
  n1 ricerca K4



CAN: IL ROUTING

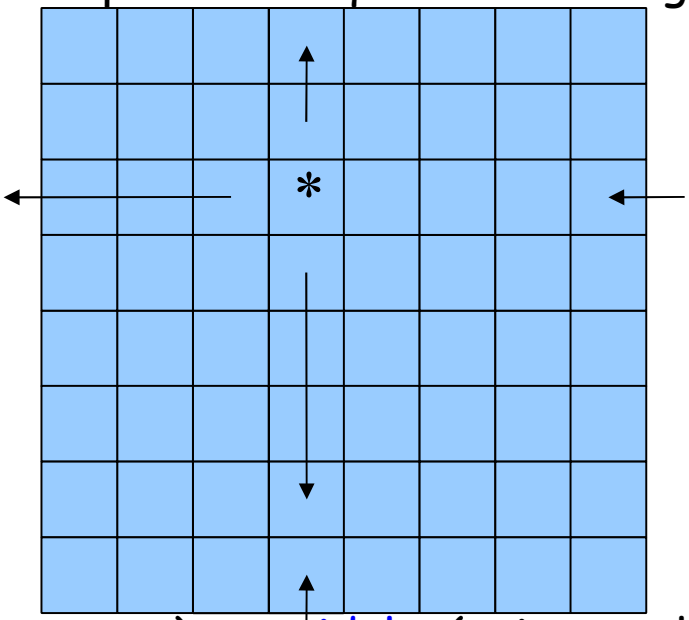
- Ogni nodo è a conoscenza dei suoi vicini nello spazio a d dimensioni
- La query viene inoltrata al nodo che risulta più vicino all'id della query
- Esempio:

  n1 ricera K4



CAN: COMPLESSITA' DEL IL ROUTING

- Complessità del routing $O(D*N^{1/D})/4$, se lo spazio è equamente diviso in n zone
- Consideriamo uno spazio bi-dimensionale decomposto in quadrati di uguale dimensione



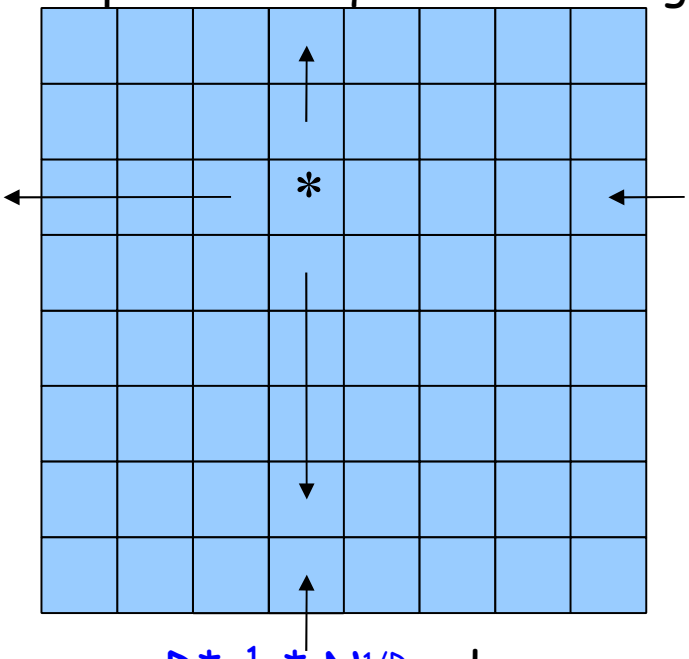
N = 64

- Ogni dimensione ha $N^{1/D}$ nodi ($16^{1/2}=4$)
- Consideriamo una dimensione: poichè la struttura è toroidale (esistono le richiusure) in $\frac{1}{2} * N^{1/D}$ partendo da una qualsiasi posizione, raggiungo una qualsiasi altra posizione in quella dimensione
- Nel caso pessimo percorro $\frac{1}{2} * N^{1/D}$ passi in ognuna delle dimensioni delle quindi $D * \frac{1}{2} * N^{1/D}$

CAN: COMPLESSITA' DEL IL ROUTING

- Complessità del routing $O(D*N^{1/D})/4$, se lo spazio è equamente diviso in n zone
- Consideriamo uno spazio bi-dimensionale decomposto in quadrati di uguale dimensione

N = 64



- Caso Medio: 0 passi nel caso ottimo, oppure 1 passo,.., $D* \frac{1}{2} * N^{1/D}$ nel caso pessimo
- In media occorrono $\sum i /k+1$, dove $k = D* \frac{1}{2} * N^{1/D}$,i=0..k
- $\sum i /k+1$, = k (k+1)/2*k+1 = k/2
- Poichè $k = D* \frac{1}{2} * N^{1/D}$, si ottiene il risultato $O(D*N^{1/D})/4$.

CAN: COMPLESSITA'

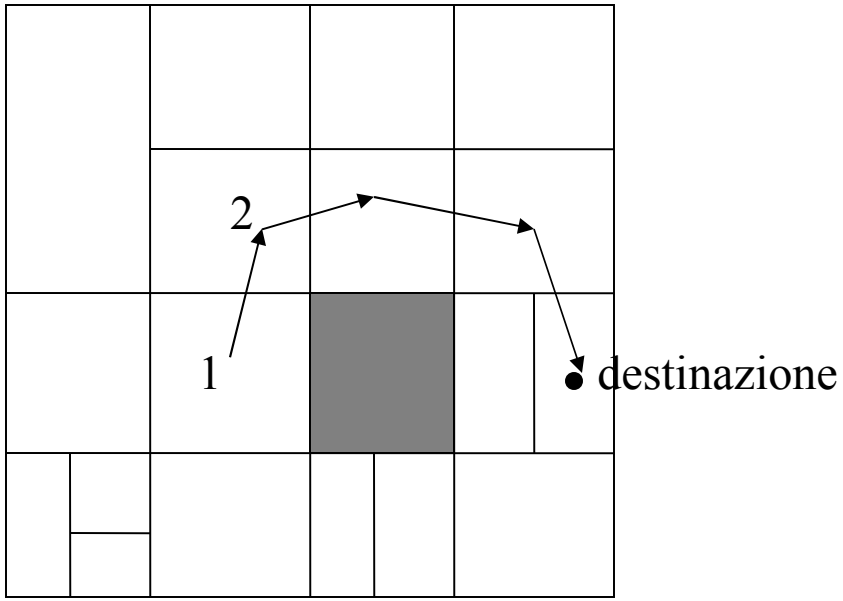
Risultati ottenuti per uno spazio d dimensionale partizionato in n zone di uguale dimensione

- Ogni nodo deve mantenere il riferimento a 2d vicini (due vicini in ogni dimensione, uno per 'avanzare', l'altro per 'retrocedere' in quella dimensione. I riferimenti ai vicini rappresentano la tabella di routing del nodo
- Routing $O(d*N^{1/d})/4$

Conclusione: Il numero di entrate della tabella di routing non dipende dal numero di zone (nodi). Il numero di passi di routing cresce come la radice di N di indice d

Aumentando il numero di dimensioni diminuisce il numero di passi per il routing

ROUTING: FALLIMENTO DI NODI



- L'esistenza di cammini alternativi tra due punti permette di ottenere un routing affidabile anche in presenza di nodi guasti
- 1-hop-route check: Se un nodo p (es: il nodo 1)verifica che un messaggio non può progredire verso il nodo destinazione a causa del guasto di un nodo vicino, p chiede agli altri vicini (es: il nodo 2) se essi possono far progredire il messaggio verso la destinazione
- Se l'1-hop-route-check fallisce, cono necessarie regole più complesse

CAN: INSERZIONE DI NUOVI NODI

Quando un nuovo nodo n si unisce ad una rete R di nodi CAN:

- ricerca un nodo di bootstrap B già esistente all'interno di R
- si associa un nome simbolico ad una rete CAN, il DNS restituisce l'IP di uno o più nodi CAN di bootstrap
- sceglie, in modo casuale, un punto P nello spazio CAN o applica una funzione hash al proprio indirizzo IP + porta
- invia un messaggio di JOIN, utilizzando P come chiave. Il messaggio viene recapitato al nodo C che gestisce la zona che contiene P
- il messaggio di JOIN viene spedito mediante il nodo B
- il messaggio raggiunge C mediante il meccanismo di routing di CAN

CAN: INSERZIONE DI NUOVI NODI

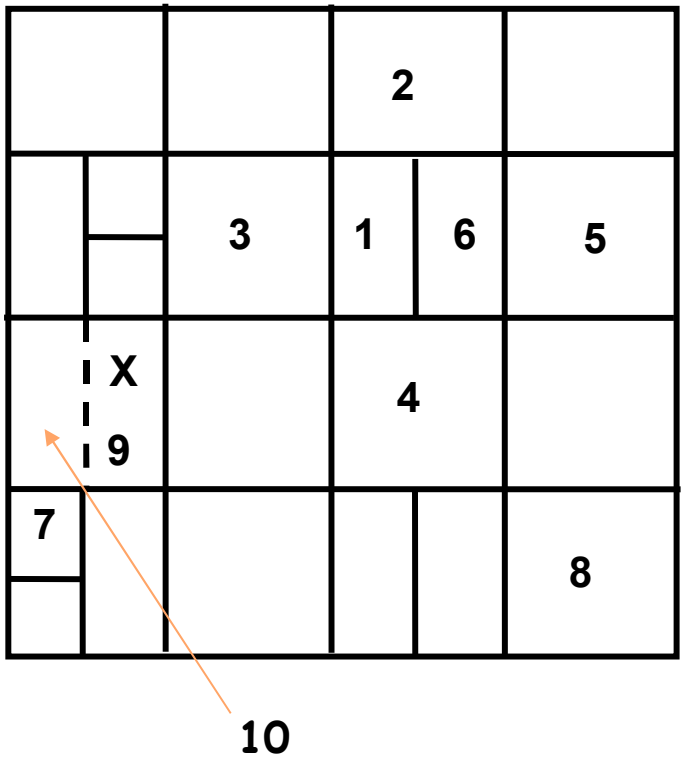
- Il nodo C che riceve un messaggio di JOIN dal nodo n divide la propria zona in due parti
  - il primo taglio viene effettuato lungo la prima dimensione
  - se l'ultimo taglio è effettuato lungo la direzione i<D, il successivo taglio avviene lungo la dimensione i+1-esima (modulo D)
  - Se D=2, si divide prima lungo l'asse delle x, poi lungo quello delle y

- C tiene metà dello spazio per se e cede la restante parte a n

- C cede le chiavi eventualmente mappate sullo spazio ceduto a n

- *Osservazione:* la probabilità che una zona venga scelta dipende dalla sua dimensione

CAN: INSERZIONE DI NUOVI NODI



- Identifichiamo i nodi con i numeri 1,..,10, I numeri sono introdotti solo per l'identificazione dei nodi

- Tenere presente che CAN non assegna un identificatore al nodo, ma una zona

- il nuovo nodo 10 sceglie, in modo casuale, un punto,ad esempio X

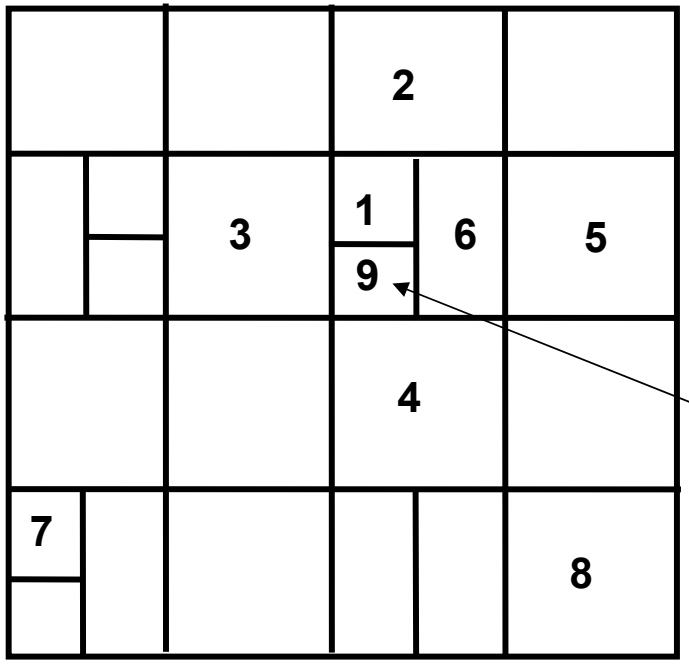
- X è contenuto nella zona gestita dal nodo 9

- 9 cede metà della propria zona a 10

- Se ci sono chiavi mappate su quella parte della zona assegnata al 9, queste vengono trasferite al 10

CAN: AGGIORNAMENTO VICINI

- Il nuovo nodo n riceve le informazioni relative ai suoi vicini dal nodo C che ha ceduto parte della propria zona
- i vicini di n sono un sottoinsieme dei vicini di C
- Il nodo che ha ceduto la zona aggiorna i propri vicini in base alle coordinate della zona ceduta

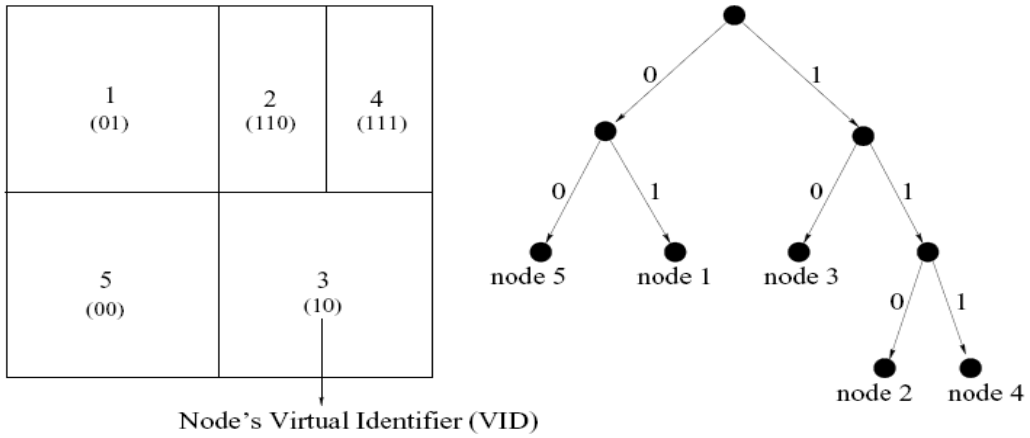


Dopo l'inserimento del nuovo nodo

- vicini di 9 ={1,3,6,4}
- vicini di 1 = {2,3, 4,9}

PARTITION TREES

- In ogni istante, la rete virtuale CAN può essere rappresentata mediante un albero binario che rappresenta la 'storia' del partizionamento
- Nodi interni dell'albero = zone che sono state divise, le foglie corrispondono alle zone
- Gli archi dell'albero sono etichettati con numeri binari: ad esempio: 0 per la zona a sinistra, 1 per qulla a detra,....
- Ad ogni nodo viene associato un identificatore unico che rappresenta il cammino dalla radice alla foglia che rappresenta la zona posseduta dal nodo



CAN: USCITA DI NODI DALLA RETE

- Quando un nodo n decide volontariamente di uscire dalla rete, la gestione della zona Z controllata da n deve essere passata ad un nodo vicino (takeover node)

- n consegna autonomamente le informazioni associate a Z ad un nodo n' che controlla una zona vicina Z'. Le informazioni trasmesse ad n' sono:
  - (key,value) mappate su Z
  - identificazione dei vicini di n (zone + indirizzi IP)

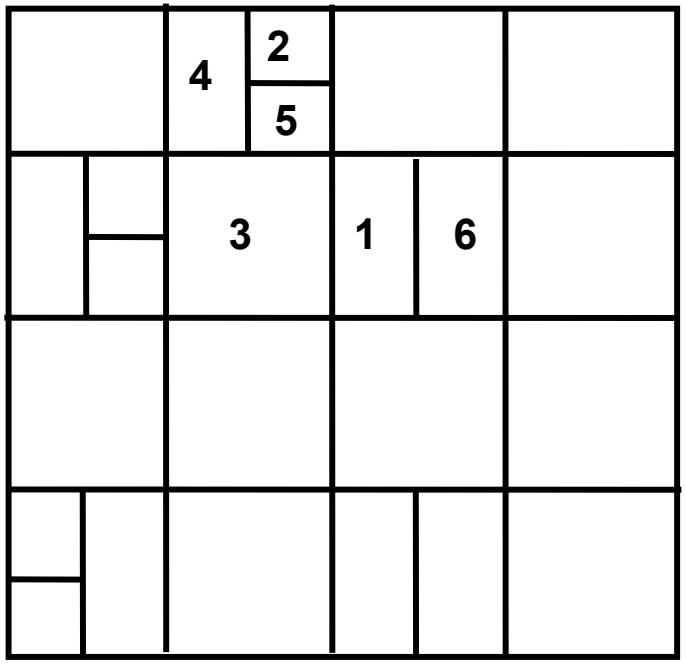
- Se possibile, le zone di n ed n' vengono fuse in modo da produrre una zona più ampia che mantenga la struttura CAN

- Altrimenti, n' prende in carico sia Z che Z', momentaneamente senza effettuare alcuna fusione.

CAN: USCITA DI NODI DALLA RETE

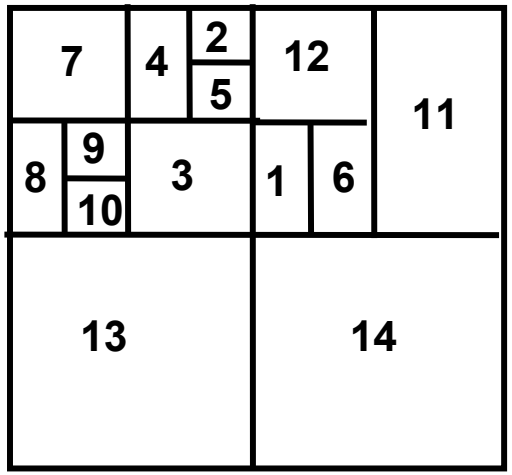
- La fusione delle zone deve mantenere la struttura topologica di CAN

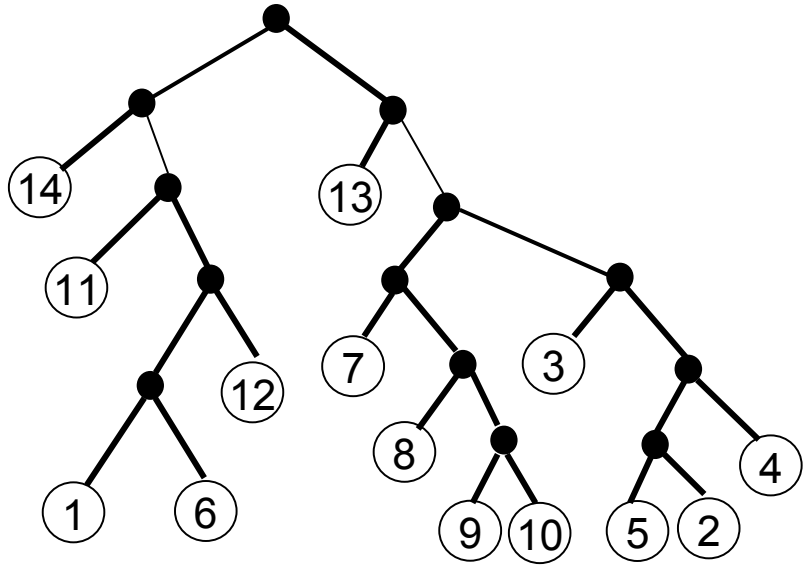
- Esempio: 6 lascia la rete. La sua zona può essere fusa con quella controllata da 1



- Esempio: 3 lascia la rete. La sua zona non può essere fusa con le zone adiacenti (4,5). 5 prende in consegna la zona 3.

CAN: PARTITION TREE

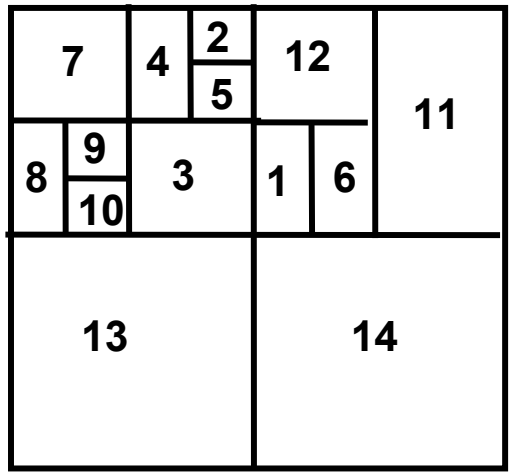


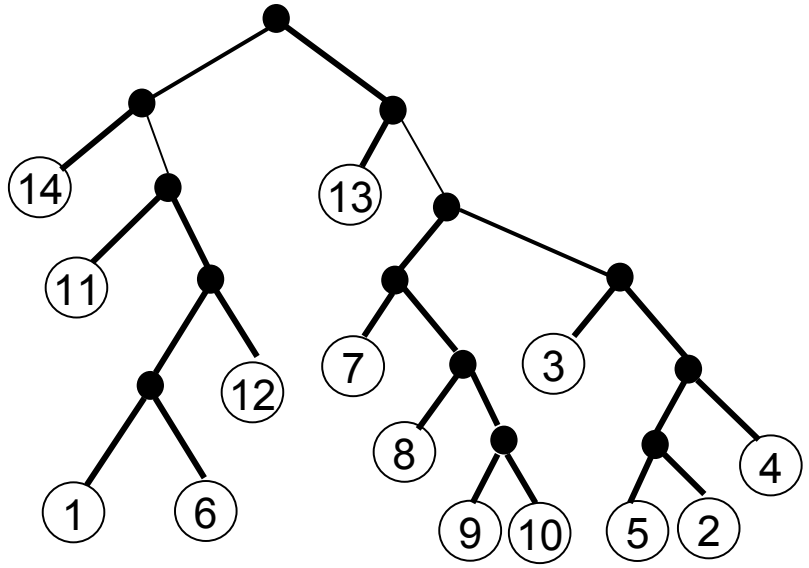


Il partizionamento dello spazio CAN puo' essere rappresentato mediante un albero binario in cui

- le foglie rappresentano le zone assegnate ai nodi
- i nodi interni rappresentano zone che non esistono più, in quanto decomposte
- i figli di un nodo rappresentano le due zone in cui è stata decomposta una zona

CAN: INDIVIDUAZIONE DEL TAKE OVER NODE

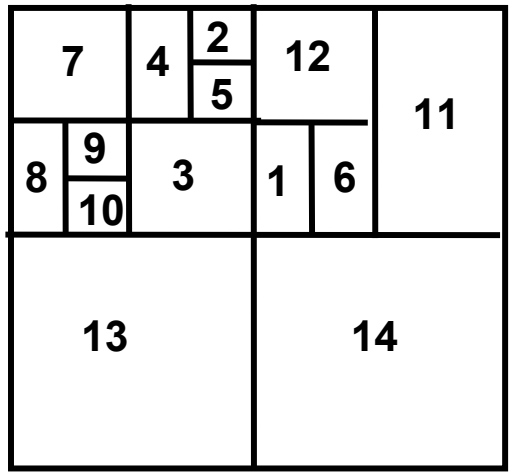


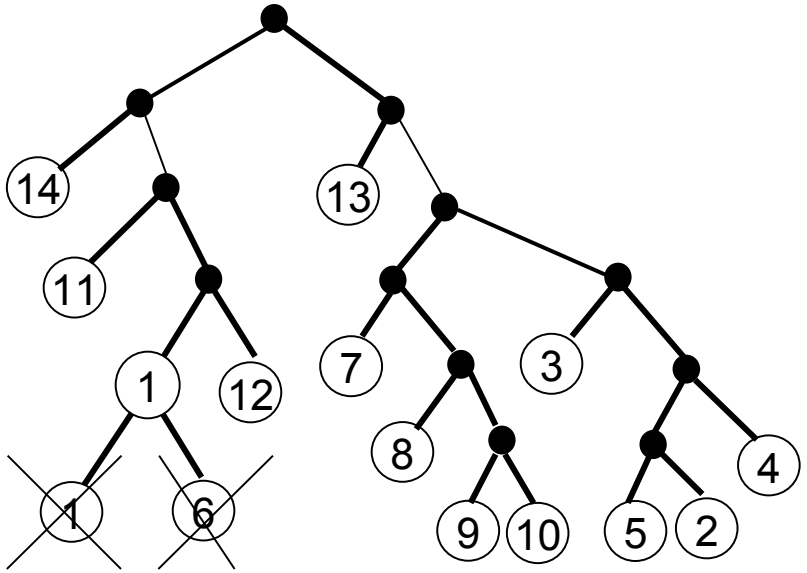


Caso 1: la foglia s viene rimossa, il fratello di s è una foglia t

- s e t vengono fuse
- il nodo padre p diventa un nodo foglia
- la zona corrispondente a p viene presa in carico dal nodo CAN che in precedenza gestiva t

CAN: INDIVIDUAZIONE DEL TAKE OVER NODE



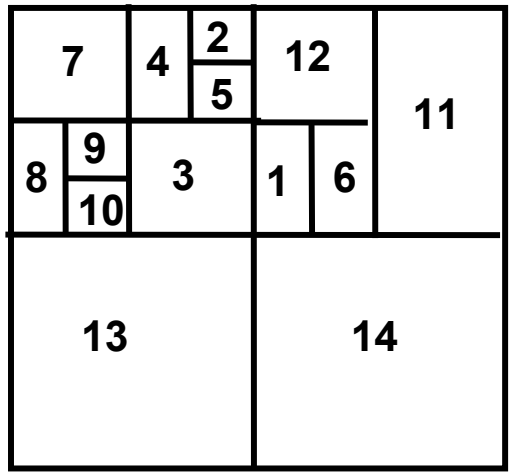


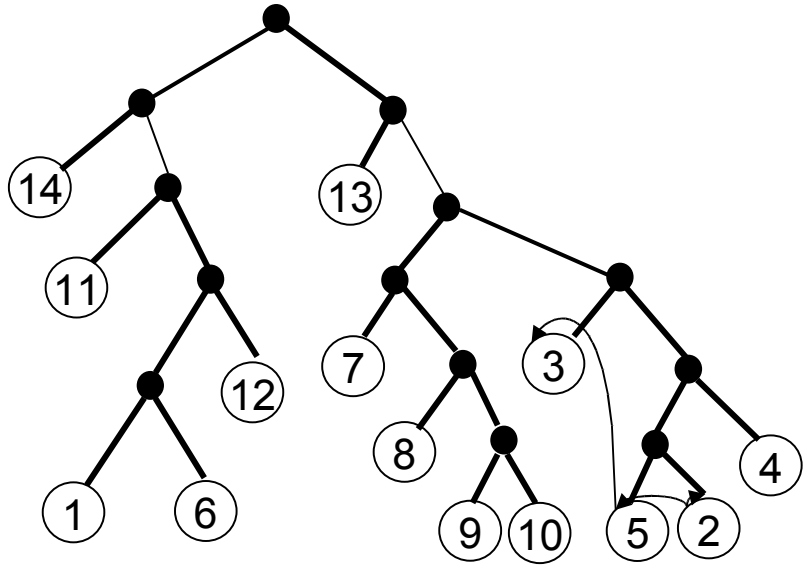
Caso 1: la foglia s viene rimossa, il fratello di s è una foglia t

Esempio: 6 lascia la reta CAN

- le zone 1 e 6 vengono fuse
- 1 prende in consegna la zona risultante

CAN: INDIVIDUAZIONE DEL TAKE OVER NODE

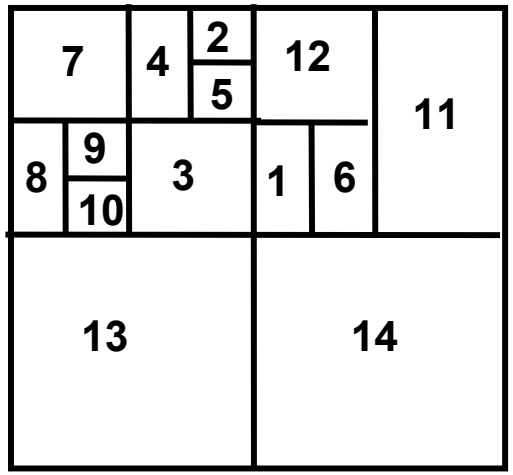


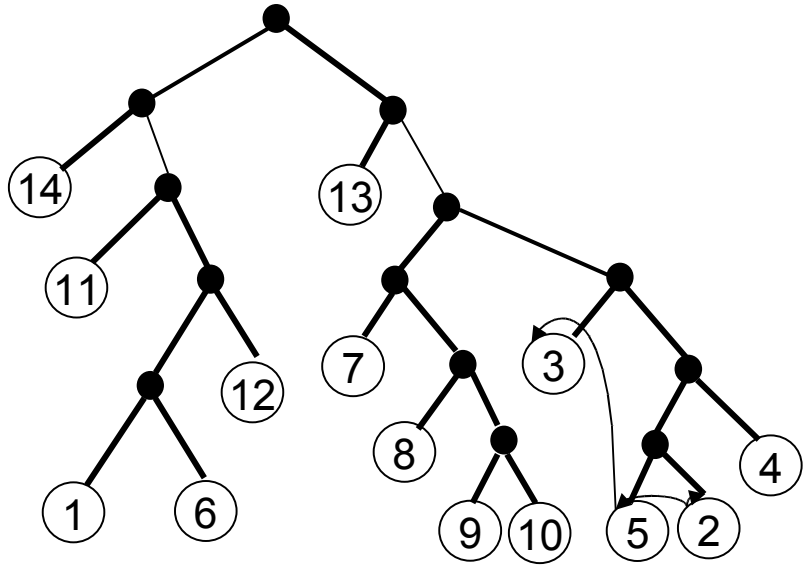


Caso 2: la foglia s viene rimossa, il fratello t di s non è una foglia

- si effettua una visita depth first (DFS) nel sottoalbero radicato in t fino ad individuare una foglia z che ha come fratello un nodo foglia
  - se s è il fratello sinistro di t la DFS visita per primo il sottoalbero sinistro e poi quello destro, viceversa se s è il fratello destro di t.
- Il nodo corrispondente a z prende in consegna la zona corrispondente ad s
- Il nodo corrispondente al fratello di z prende in consegna la zona gestita da z

CAN: INDIVIDUAZIONE DEL TAKE OVER NODE





Caso 2: la foglia s viene rimossa, il fratello t di s non è una foglia

Esempio: 3 lascia la rete CAN

- Si effettua una visita DFS a partire dal fratello di 3 nell'albero, visitando per primo il sottoalbero sinistro, fino a trovare la foglia 5
- Il nodo corrispondente a 5 prende in consegna la zona 3
- La zona 5 viene presa in consegna dal nodo che gestisce la zona 2

CAN: FALLIMENTO DI NODI

- CAN non prevede meccanismi di replicazione. Le informazioni memorizzate sul nodo fallito sono perse definitivamente

- Si suppone che l'applicazione che utilizza CAN "rinfreshi" periodicamente l'informazione memorizzata all'interno della rete

- individuazione del fallimento di un nodo
  - ogni nodo invia periodicamente ai vicini messaggi di update contenenti le coordinate della sua zona ed una lista dei vicini con le proprie zone
  - l'assenza prolungata di messaggi di update da parte di un vicino viene interpretata come il fallimento del nodo

- se un nodo fallisce, la zona di cui esso è responsabile viene presa in carico da un nodo take over node ,un altro nodo attivo sulla rete CAN.

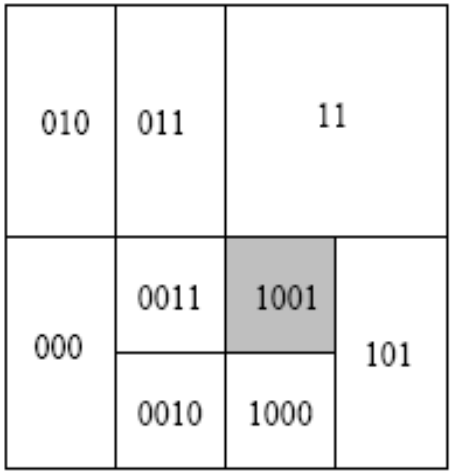
CAN: INDIVIDUAZIONE DEL TAKE-OVER NODE

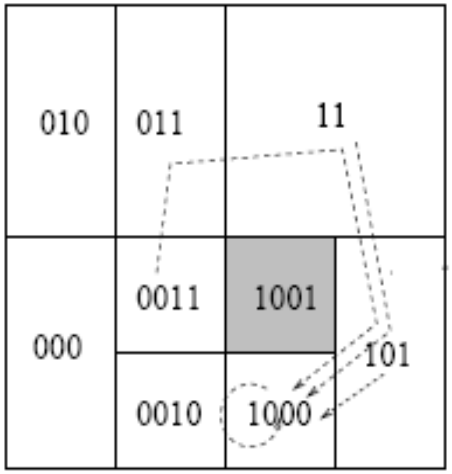
- La procedura precedente utilizza una conoscenza globale dell'albero bianrio che rappresenta il partizionamento dello spazio
- E' necessario definire un algoritmo distribuito che utilizzi solamente l'informazione locale di un nodo
- I nodi non possono memorizzare esplicitamente la struttura dell'albero
- Prima soluzione: individuare il take over node mediante il VID del nodo
- Ogni nodo
  - memorizza il proprio VID che identifica univocamente la sua locazione all'interno dell'albero
  - utilizza il proprio VID per individuare il takeover node
  - Il takeover node di un nodo N è il nodo M che ha il VID numericamente piu' vicino ad N
  - Problema:definizione di un algoritmo distribuito per l'individazione del take-over node

CAN: ALGORITMO DI RECOVERY

- Ogni nodo scopre quindi autonomamente di essere il nodo takeover ed occupa la zona del nodo fallito
- I vicini del nodo fallito devono scoprire l'identità del take over node
- Algoritmo distribuito per l'aggiornamento delle tabelle di routing dei vicini del nodo fallito
  - si cancella il nodo fallito dalla lista dei nodi vicini
  - invio di un recovery message al nodo più vicino al nodo fallito, la distanza tra due nodi è calcolata mediante i VID dei nodi coinvolti
  - in questo modo, il messaggio arriva a nodi con VID sempre più vicino al nodo fallito, fino ad arrivare al takeover node
  - Il takeover node
    - non invia messaggi ai vicini
    - acquisisce conoscenza dei nuovi vicini
    - informa i vicini circa la propria identità

CAN: ALGORITMO DI RECOVERY





- Il nodo 1001 fallisce
- Il nodo 0011 si accorge del fallimento ed invia un recovery message al nodo 11 (il più vicino numericamente al nodo fallito)

INDIVIDUAZIONE DEL TAKE-OVER NODE

- Soluzione alternativa: quando un nodo n individua il fallimento di un nodo vicino
  - inizializza un timer con un valore proporzionale alla dimensione della sua zona
  - quando il timeout scade, invia un messaggio di takeover ai vicini del nodo caduto
- I nodi che gestiscono zone più piccole inviano i loro messaggi per primi
- Quando un nodo riceve un messaggio di takeover, cancella il proprio timer, se la sua zona ha dimensione maggiore a quella notificata nel messaggio
- In questo modo si sceglie come nodo di takeover il nodo che gestisce la zona più piccola
- Il nodo di takeover prende in consegna la zona del nodo fallito e la fonde con la sua, se possibile, altrimenti gestisce temporaneamente entrambe le zone
- Possibile fragmentazione della rete:necessario algoritmo distribuito per la ricostruzione della rete

ROUTING: OTTIMIZZAZIONI

- L'effettiva efficienza del routing deve essere misurata in termini di
  - numero di CAN hops
  - latenza media di ogni hop
- Importante: tra due nodi adiacenti nello spazio virtuale di CAN possono esistere diversi hop IP
- Se un cammino c1 include un numero maggiore di hops CAN rispetto ad un cammino c2, la latenza di c1 può in realtà essere inferiore a quella di c2
- Ottimizzazioni:
  - Diminuire il numero di CAN hops
  - Ridurre la latenza del singolo hops. Richiede la conoscenza della dislocazione degli hosts all'inteno della rete fisica
- CAN definisce diverse euristiche per l'ottimizzazione del routing. Ogni euristica in genere aumenta la quantità di informazione che ogni nodo deve memorizzare nella propria tabella di routing

PROXIMITY ROUTING

- Osservazione: il numero di routing hops può essere ridotto aumentando il numero di dimensioni di CAN
- Se si vuole intervenire sulla latenza di ogni hop, è necessario definire qualche forma di proximity routing
- Soluzione 1: Modificare la metrica utilizzata per scegliere il vicino ad ogni routing hop
  - Per scegliere il vicino verso cui inoltrare il messaggio si considera
    - il progresso del messaggio, in termini di distanza cartesiana verso la destinazione
    - la latenza del collegamento con i nodi vicini
  - Ogni nodo misura periodicamente il RTT (Round Trip Time) con ognuno dei suoi vicini
  - Il messaggio viene inoltrato al vicino che presenta il miglior rapporto progresso/RTT

PROXIMITY NEIGHBOUR SELECTION

- Quando un nodo sceglie i propri vicini, considera la latenza dei collegamenti con essi
- Zone Overloading: Ad ogni zona viene associato un insieme di peer, di cardinalità max =MAXPEER
- Quando un nodo N si unisce ad una rete CAN, individua la zona Z in cui si posiziona.
  - se Z è gestita da un numero di nodi inferiore a k (k soglia) la zona non viene divisa ed N si unisce ai nodi che gestiscono quella zona.
  - se Z è piena, allora viene divisa
- Intuizione:
  - se un nodo N è associato ad una zona Z, N può individuare, per ogni zona confinante con Z, un insieme di vicini
  - è possibile selezionare in questo insieme il vicino caratterizzato da una minore latenza
  - Inoltre la lunghezza dei cammini in termini di hops CAN diminuisce perché l'effetto dell'overloading è una diminuzione delle zone del sistema

PROXIMITY NEIGHBOUR SELECTION

- Ogni nodo mantiene riferimenti a
  - tutti gli altri nodi associati alla stessa zona
  - un nodo per ogni zona confinante con la sua

- Ogni nodo
  - chiede periodicamente ad ogni nodo V che gestisce una zona vicina Z la lista L dei nodi che gestiscono la stessa zona di V
  - valuta il RTT per ogni nodo in L
  - Sceglie come vicino associato a Z il nodo caratterizzato dal valore minore del RTT

- La dimensione della tabella di routing di un nodo rimane O(d)

- Ogni nodo deve però mantenere un riferimento a tutti i peer che gestiscono la sua stessa zona

PROXIMITY NEIGHBOUR SELECTION

- Le chiavi mappate su una zona possono essere
  - replicate su ogni peer che gestisce la zona. Questa soluzione
    - incrementa la robustezza del sistema, ma aumenta lo spazio di memorizzazione richiesto da ogni nodo
    - Richiede la definizione di meccanismi per garantire la consistenza dell'informazione replicata
  - partizionate tra i peer che gestiscono la zona

GEOGRAPHICAL LAYOUT

- Nella soluzione di base, la scelta di una zona è completamente casuale
- Alternativa: considerare la vicinanza fisica con i nodi vicini nella fase di scelta della zona iniziale con l'obiettivo di evitare zig-zag routing
- Approccio:
  - viene scelto un insieme di nodi ben noti (esempio: i root server del DNS) che considerati 'punti di riferimento' sulla rete fisica
  - ogni nodo CAN misura il suo RTT con ognuno di questi nodi e li ordina in ordine crescente di RTT
  - ogni nodo CAN è caratterizzato da un diverso ordinamento
  - per m punti di riferimento m! possibili ordinamenti
  - lo spazio CAN viene partizionato in m! zone
  - ogni nodo CAN sceglie un punto casuale nella porzione di spazio CAN associato al suo ordinamento
  - in questo modo nodi 'fisicamente vicini' vengono mappati nella stessa porzione dello spazio
  - problema: distribuzione non uniforme dei nodi,necessari algoritmi di bilanciamento del carico

MULTIPLE HASH FUNCTIONS

- Si definiscono k diverse funzioni hash per mappare una chiave su k punti dello spazio virtuale CAN
- Una chiave viene replicata su k nodi diversi
- Una query per una chiave k viene inviata in parallelo a tutti i nodi che memorizzano la chiave
- In questo modo si riduce la latenza
- Tecnica di replicazione utilizzata anche per garantire la disponibilità delle chiavi in caso di fallimento o partenza volontaria dei nodi