Multi-dimensional range queries for sensor networks

Alessandro Ghidotti Piovan

a.a. 2009/2010

GRUPPO 20

MOTIVAZIONI

- Nelle reti di sensori dati o eventi identificati da attributi aventi valori (solitamente) scalari.
  - *Raccolta dati da un ambiente:* $E=<A_1, A_2, \ldots, A_m>$
- Range query multi-dimensionali modo naturale per ricercare o correlare eventi di interesse
  - *Ricerche specifiche da parte di un utente*
    - SCENARIO: "Elencare tutti gli eventi per i quali la temperatura è nel range 25° e 30° ed il livello di luce è tra 10 e 15"
  - *Correlazione automatica di eventi*
    - SCENARIO: "Si può capire se un uccello si posa su un nido da un certo range di valori termici e acustici. Si può rilevare con precisione l'arrivo di uno stormo di uccelli ed attivare un sistema di telecamere"

SUPPORTO ATTUALE

- **Indici *k*-dimensionali pre-computati centralizzati**
  - Progettati per funzionare su un singolo nodo
  - *Bassa efficienza energetica, banda limitata*
  - Sensori hanno scarse risorse computazionali
  - *Costo di inserimento può essere dominante*
- **Storage esterno**
  - Nodo esterno alla rete adibito a storage
  - *Possibile collo di bottiglia, nodi più sollecitati di altri (hotspot)*
- **Altri indici distribuiti** (e.g. GHT, DIFS, etc.)
  - *GHT (Geographic Hash Tables)*
    - *Non supporta le range query multidimensionali*
  - *DIFS (Distributed Index for Features in Sensor networks)*
    - *Supporto parziale e limitato alle range query multidimensionali*
- **Flooding**
  - Invio query in broadcast a tutti i nodi, solo qualcuno risponde
  - *Scarse scalabilità ed efficienza (anche energetica)*

OBIETTIVI

- Indice distribuito
- Supporto range query multi-dimensionali
- Scalabilità
- Efficienza energetica
- Robustezza a guasti e perdita messaggi
- Performance inserimento eventi e query

DIM

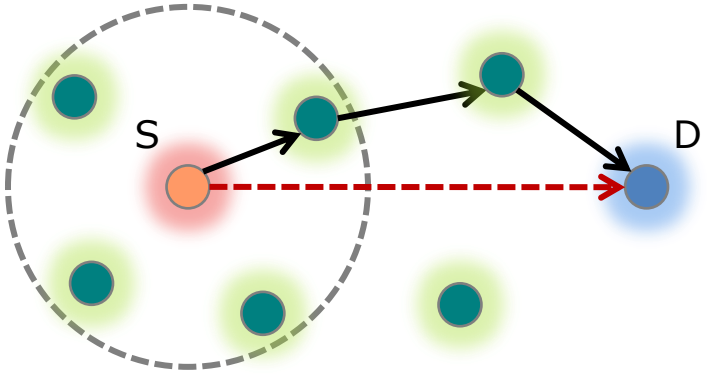
(***D**istributed **I**ndex for **M**ultidimensional data*)

IL DESIGN DEL DIM

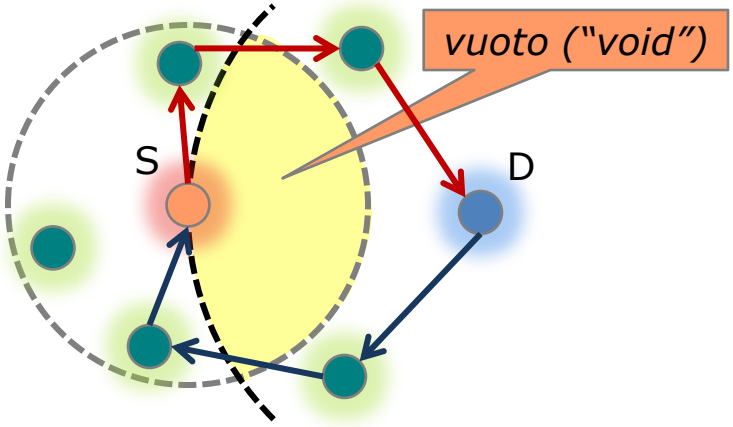
- Si basa su **due elementi fondamentali**
  - *Funzione di hash geografico che preserva la località*
    - Ogni nodo associato ad una parte dello spazio degli attributi
    - Eventi con valori vicini sono memorizzati su nodi vicini
  - *Algoritmo di routing geografico (GPSR)*
    - Un nodo può inviare un pacchetto ad un altro nodo avente una posizione geografica specifica
    - Utilizza messaggi di segnalazione e la posizione geografica dei nodi
    - Si raggiunge una destinazione inoltrando i messaggi a nodi vicini
- Vengono fatte **due assunzioni di base**
  - *Tutti i nodi conoscono l'estensione della zona coperta dalla rete di sensori*
  - *Ogni nodo conosce la propria posizione geografica*
    - Sistema di riferimento 2-dimensionale

GPSR – DUE PAROLE...

GREEDY FORWARDING



PERIMETER FORWARDING



***IDEA:* basandosi sulla posizione geografica dei nodi, è in grado di inoltrare messaggi utilizzando eventualmente nodi intermediari.**

GPSR (Greedy Perimeter Stateless Routing for Wireless Networks)

- *Algoritmo per effettuare il routing efficiente di pacchetti da una sorgente ad una destinazione in una rete di sensori wireless*
- ***Stateless**: i nodi non devono mantenere informazioni di stato durante il routing* (nessuna occupazione risorse nodi)
- ***Greedy forwarding**: si inoltra sempre al nodo geograficamente più vicino alla destinazione*

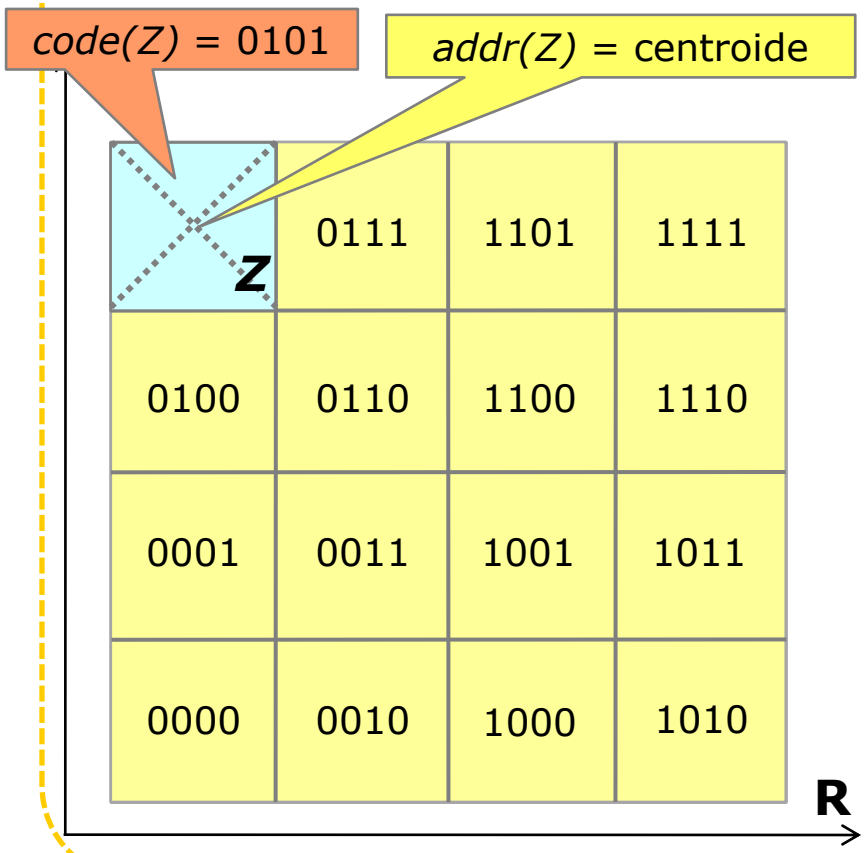
ASSUNTO: i nodi conoscono la propria posizione geografica e quella dei vicini (quest'ultima rilevata dinamicamente)

Modalità di funzionamento "***Perimeter forwarding***"

- *Se tra i nodi interni alla portata radio della sorgente non vi sono nodi in direzione della destinazione, si ha un "**vuoto**".*
- *A partire dal primo nodo vicino entro il range della sorgente si dirotta il pacchetto circumnavigando il perimetro del vuoto fino a raggiungere la destinazione.*
- *Ad ogni hop si applica la* regola della mano destra *per selezionare il prossimo nodo a cui inoltrare il pacchetto (senso antiorario, indice puntato sulla destinazione).*

COSTRUZIONE DELLE ZONE

***IDEA*: metodo per suddividere la regione in zone assegnando loro un codice univoco.**



Fissato un sistema di riferimento *x-y* il rettangolo *R* rappresenta l'area geografica di copertura della rete di sensori.

Una zona è un sottoinsieme dell'area di *R* ottenuto mediante il seguente procedimento:

- Si fanno *k* suddivisioni della regione *R*
  - Se *k* dispari, suddivisione verticale
  - Se *k* pari, suddivisione orizzontale
- N.B.: ad ogni passo si dividono a metà tutte le sottoregioni

i.e.: alla i-esima divisione, si hanno 2^i rettangoli di eguale dimensione

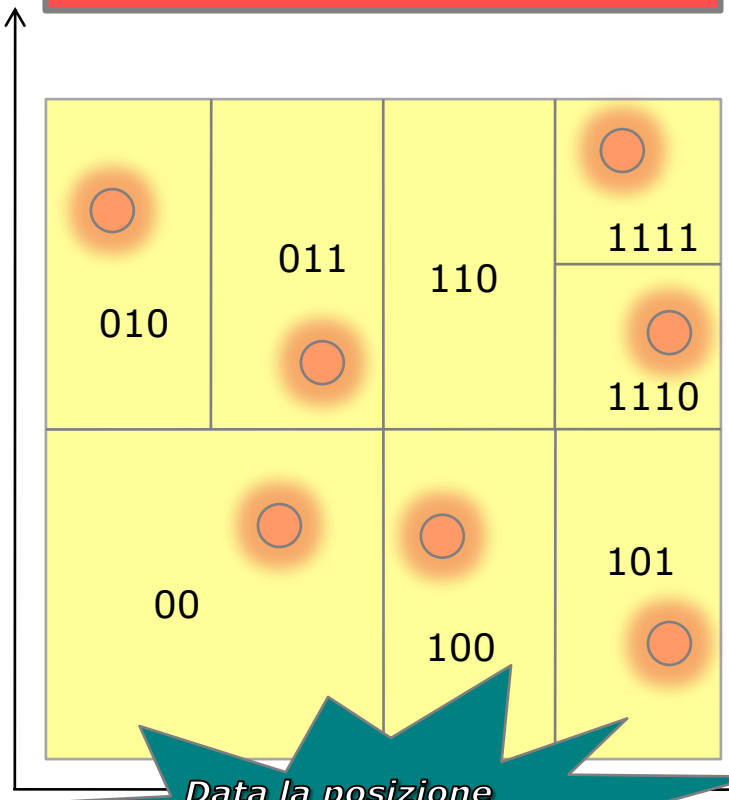
- *k* si definisce livello della zona *Z* – ***level(Z)***

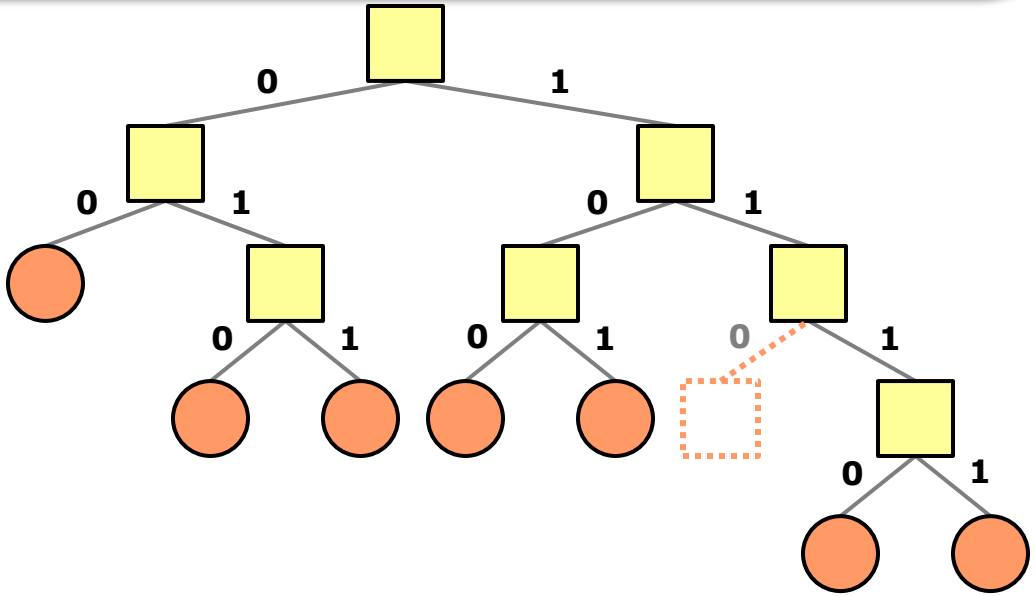
Una zona Z può essere identificata sia da un codice – ***code(Z)*** – sia da un indirizzo – ***addr(Z)***.

- *code(Z)* = stringa di bit di lunghezza *level(Z)* ottenuta come segue:
  - Se *Z* è nella metà di sinistra di *R*, il primo bit è 0, altrimenti 1.
  - Se *Z* è nella metà bassa di *R*, il secondo bit è 0, altrimenti 1.
- Il procedimento è applicato ricorsivamente per ciascun quadrante di *R* e così via.
- *addr(Z)* = coordinate del centroide del rettangolo definito da *Z*

ZONE TREE

***IDEA*: definire zone con un solo nodo al loro interno.**





Distribuzione fisica dei nodi in genere non regolare

- *Per un certo k alcune zone potrebbero essere vuote ed altre avere più di un nodo al loro interno.*

***DIM consente di associare i nodi a zone di dimensioni differenti*.**

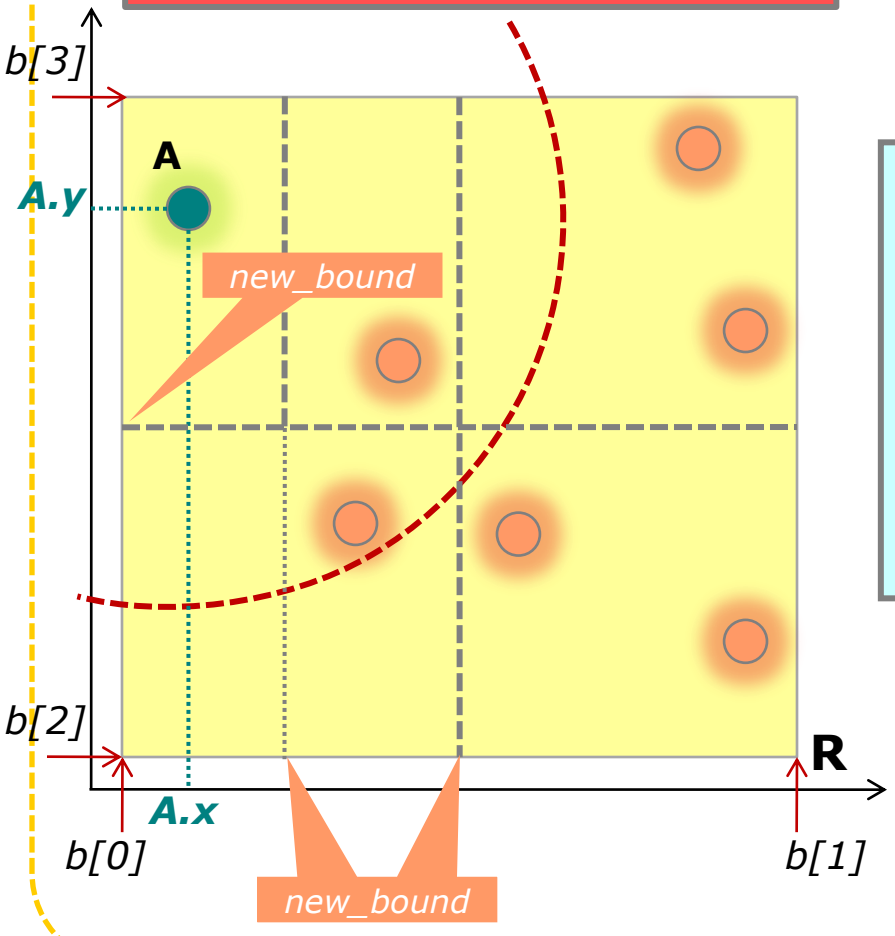
Concetto di ***zone ownership***: più larga zona al cui interno è presente uno ed un solo nodo.

- *Nodo A unico presente nella zona Z_A: A si dice **owner** della zona Z_A.*

Data la posizione di un nodo, come determinare quale zona possiede?

RISOLUZIONE DELLE ZONE

***IDEA*: metodo affinché un nodo risolva i propri confini di zona.**



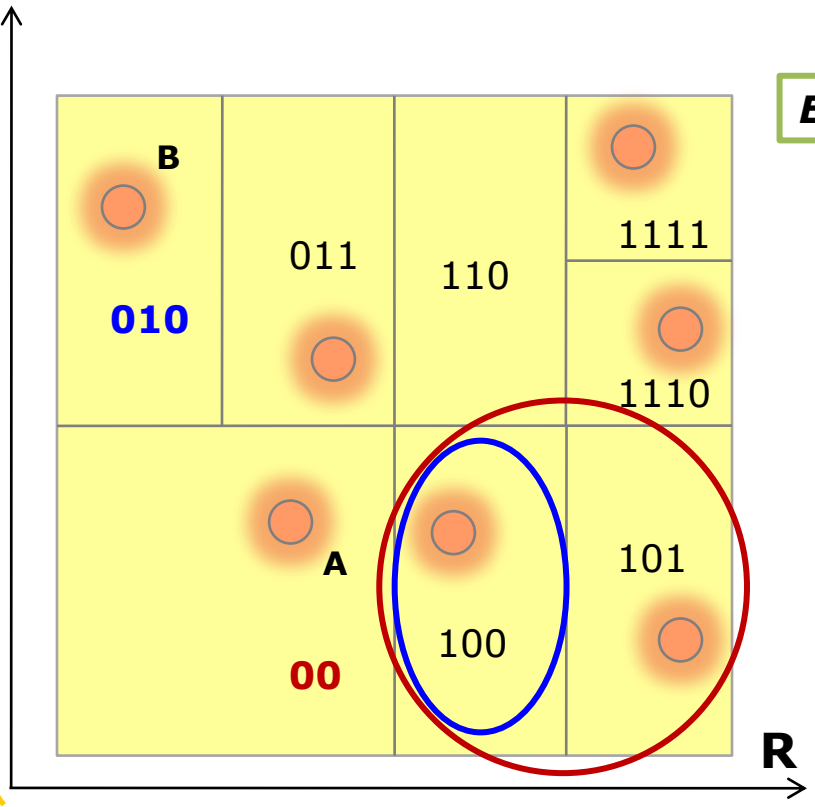
TRUE se il nodo "a" è contenuto entro i confini della zona correntemente associata ad "A"

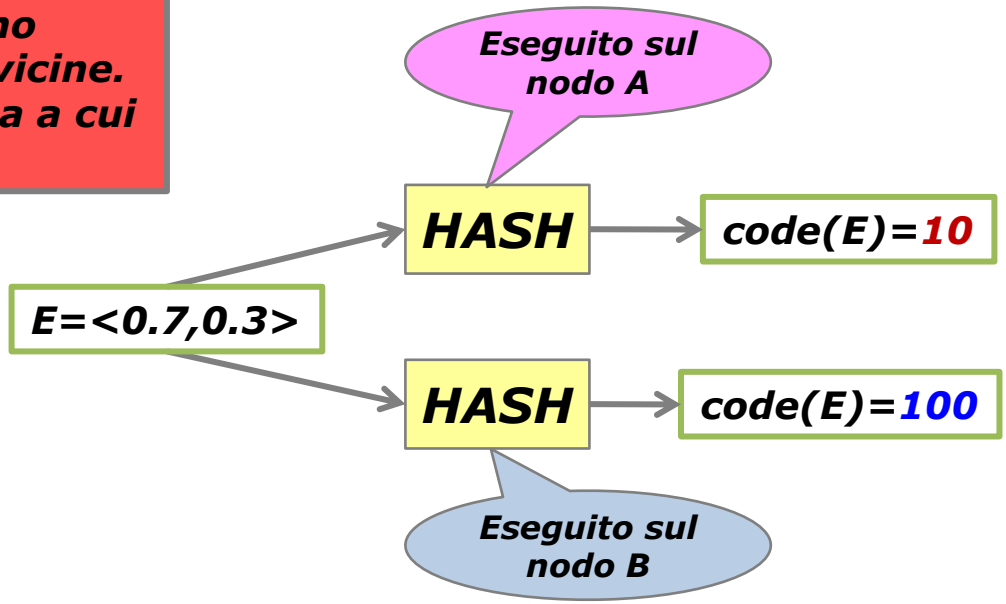
```
BUILD-ZONE(a)
 1  while CONTAIN(Z_A, a)
 2  do if length(code(Z_A)) mod 2 = 0
 3        then new_bound ← (bound[0] + bound[1])/2
 4             if A.x < new_bound
 5                then bound[1] ← new_bound
 6                else bound[0] ← new_bound
 7        else new_bound ← (bound[2] + bound[3])/2
 8             if A.y < new_bound
 9                then bound[3] ← new_bound
10                else bound[2] ← new_bound
11       Update zone code code(Z_A)
```

INIZIO → $code(Z_A)$ = NULL → $lenght(code(Z_A))=0$
Step1 → $code(Z_A) = 0$ → $lenght(code(Z_A))=1$
Step2 → $code(Z_A) = 01$ → $lenght(code(Z_A))=2$
FINE → $code(Z_A) = 010$ → $lenght(code(Z_A))=3$

HASH GEOGRAFICO

***IDEA**: attributi con valori vicini sono mappati in zone geograficamente vicine. Il codice calcolato identifica la zona a cui viene associato l'evento.*





Dato un evento (o una query), la funzione di hash calcola un codice di zona.

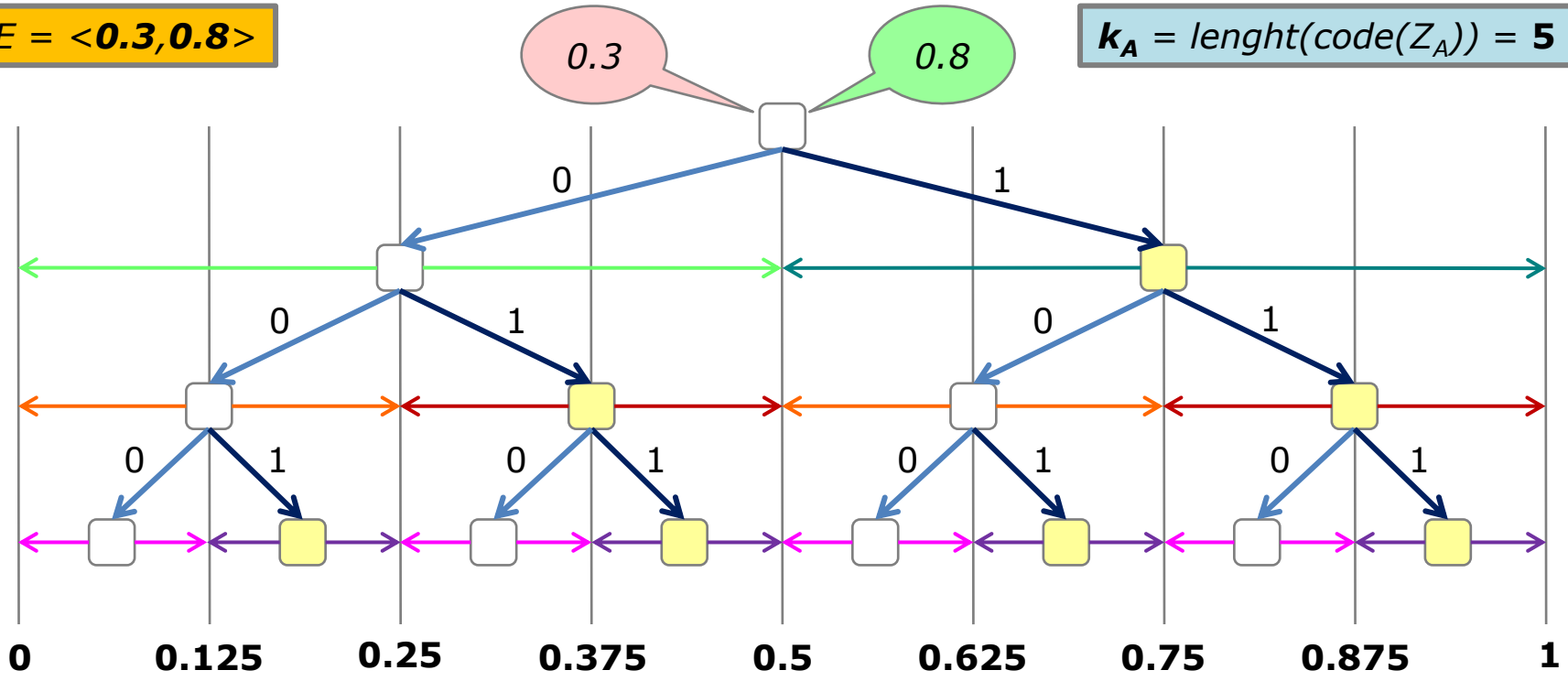
La "precisione" del codice calcolato dipende dal livello della zona posseduta dal nodo che calcola l'hash.

La lunghezza della stringa di bit risultante dall'hash è uguale alla lunghezza del codice di zona del nodo che lo calcola.

ESEMPIO: HASH EVENTO

$E = \langle 0.3, 0.8 \rangle$

$k_A = lenght(code(Z_A)) = 5$



$code(E) =$ **01110**

Lo schema di hash usa i valori degli attributi in modo ***round-robin*** su un albero binario (i.e. lo zone tree) per mappare gli eventi alle zone.

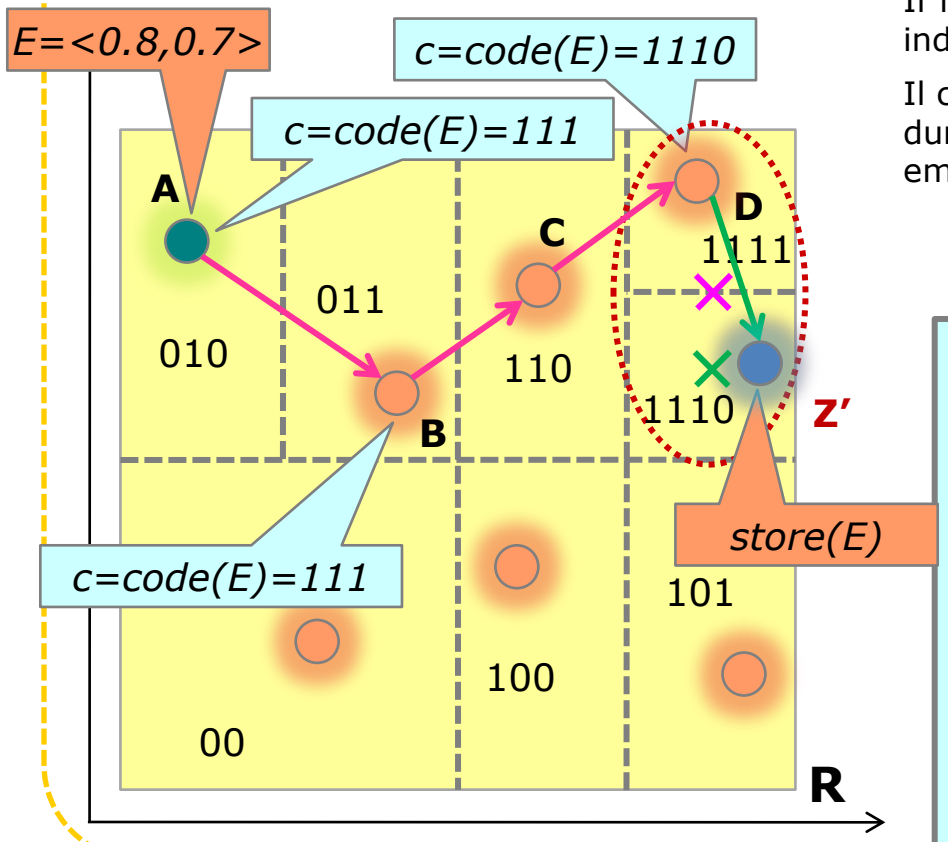
For i between 1 and m, **if** $A_i < 0.5$, the i-th bit of the zone code is assigned 0, **else** 1.

For i between m+1 and 2m, **if** $A_{i-m} < 0.25$ or A_{i-m} in [0.5,0.75), the i-th bit of the zone is assigned 0, **else** 1.

Repeat the procedure until all k_A bits have been assigned.

ROUTING DI UN EVENTO

***IDEA*: utilizzo di GPSR per inoltrare un evento al nodo che lo dovrà memorizzare.**



Hash di un evento calcolato per individuare la zona il cui nodo proprietario è designato a memorizzare l'evento. Il nodo è determinato in base all'identificativo dell'evento:

STORAGE DATA-CENTRIC

Il nodo individuato si dice *proprietario* dell'evento *E* e si indica come ***owner(E)***.

Il codice della zona di destinazione viene via via raffinato durante il routing, non pre-computato dal nodo che emette l'evento:

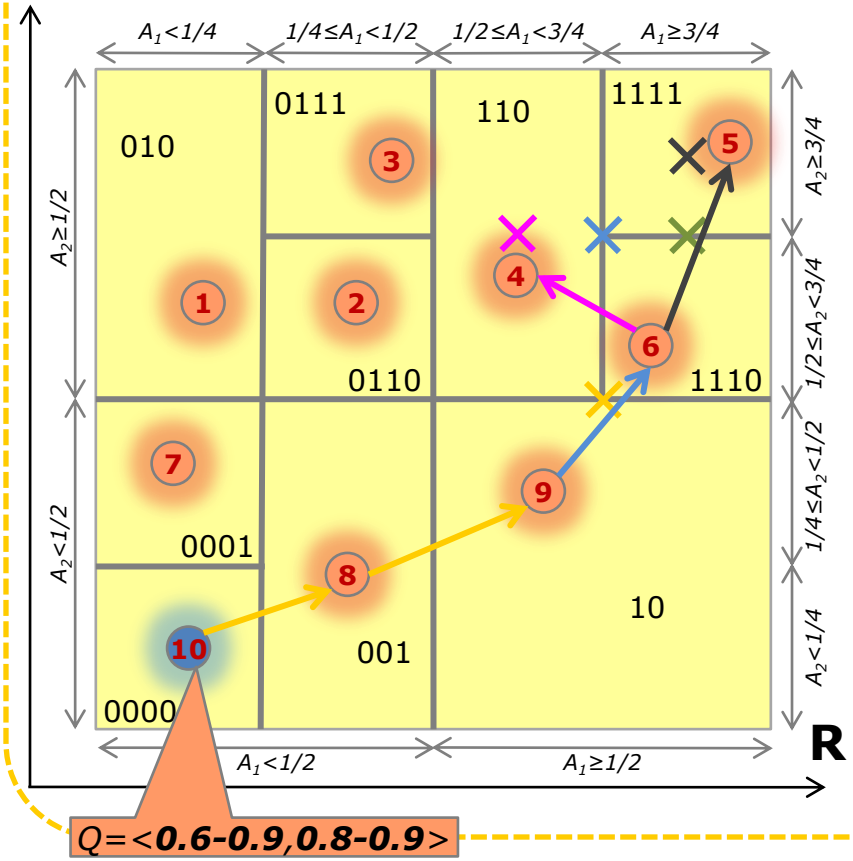
LAZY EVALUATION

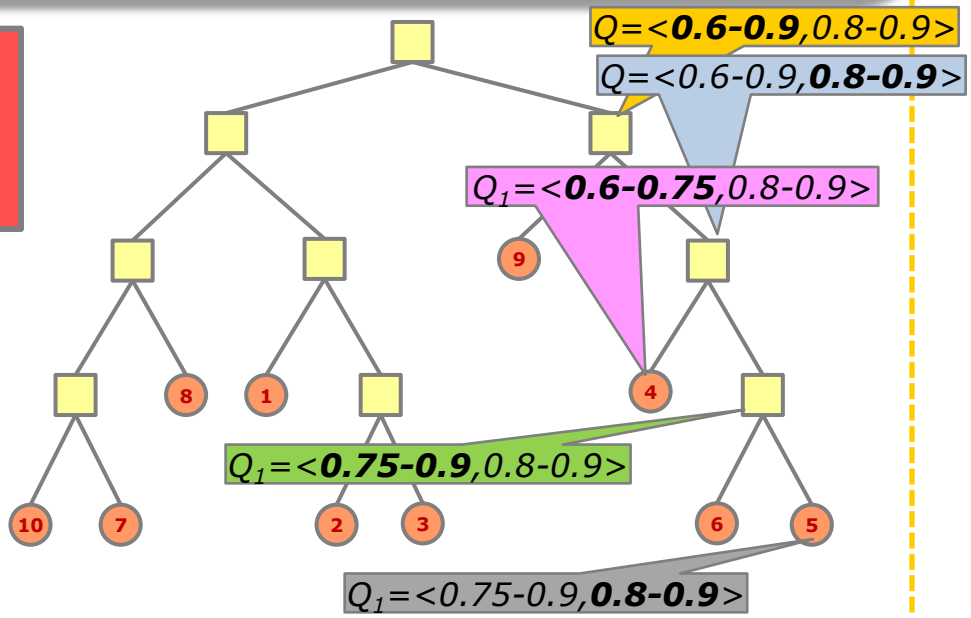
```
INSERT-EVENT(e)
1  c ← ENCODE(e)
2  if CONTAIN(Z_A, c) = true and IS_INTERNAL() = true
3     then Store e and exit
4  SEND-MESSAGE(c, e)

SEND-MESSAGE(c, m)
1  if ∃ neighbor Y, CLOSER(Y, owner(m), m) = true
2     then addr(m) ← addr(Y)
3     else if length(c) > length(code(m))
4             then Update code(m) and addr(m)
5                  source(m) ← caller
6          if IS_OWNER(msg) = true
7             then owner(m) ← caller's code
8  SEND(m)
```

ROUTING RANGE QUERIES

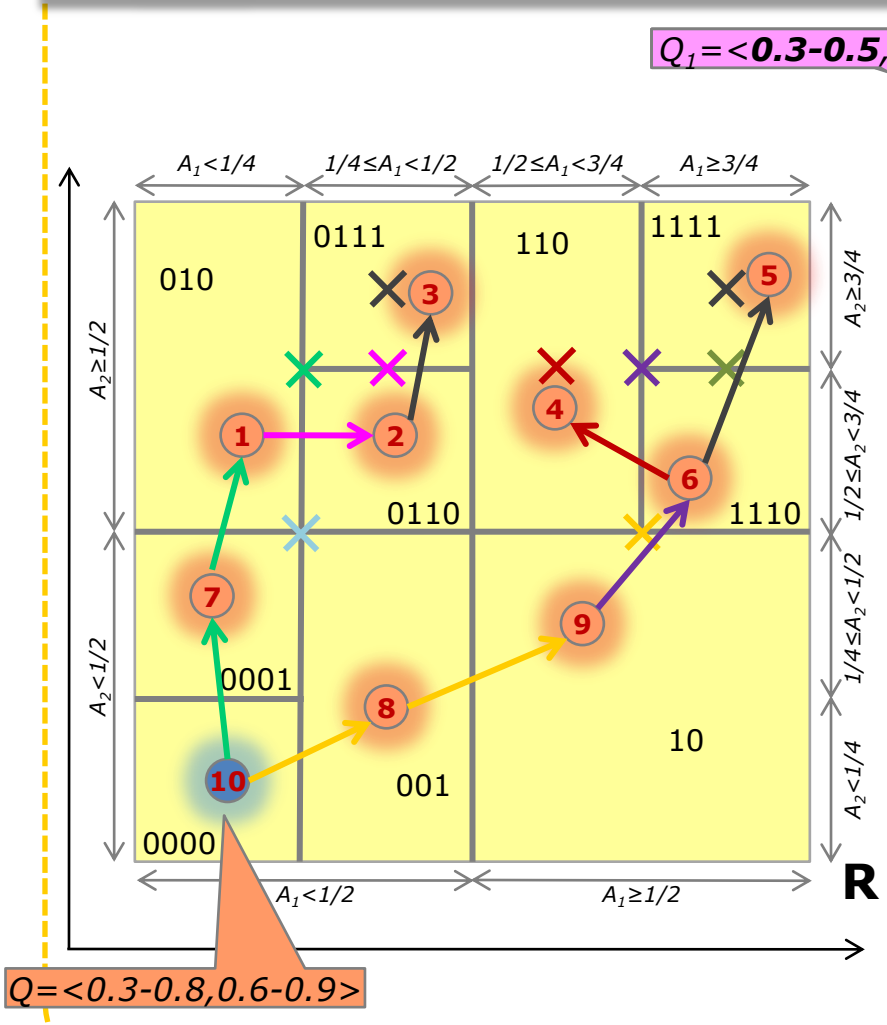
***IDEA**: inoltrare la query alla zona che mappa l'intero range e poi suddividerla in query con range più piccoli da inviare ai nodi competenti per quei range.*

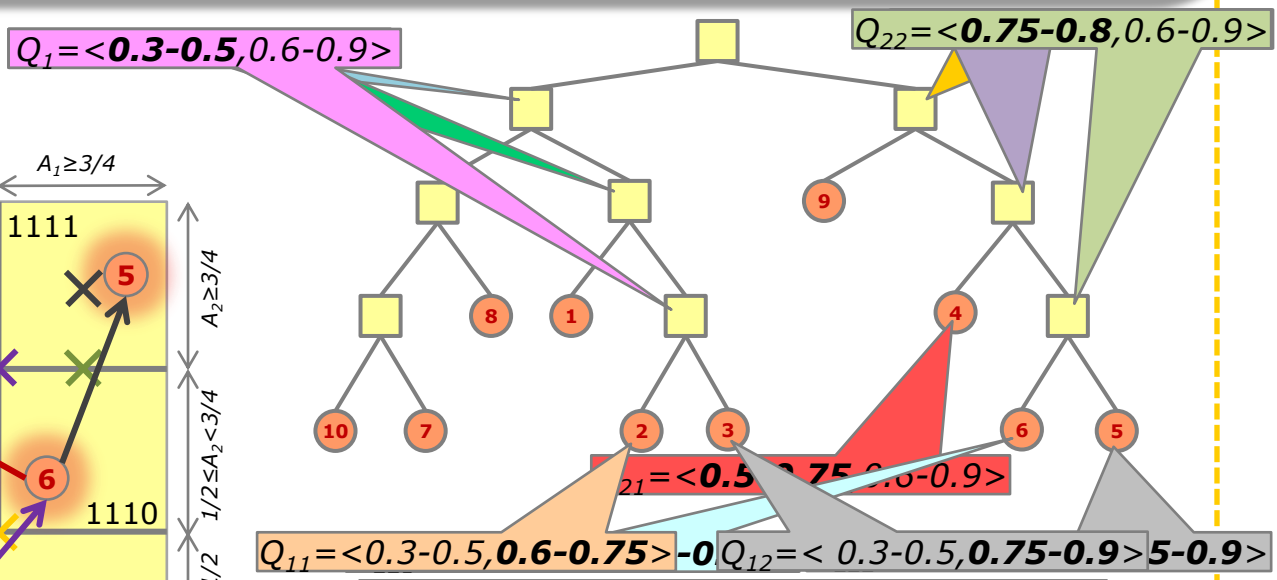




- Si mappa la query ad un prefisso di zona (macro-zona contenente tutto il range) definito con *code(Q).*
- Se il nodo che riceve la query è owner di una zona che si sovrappone con quella di codice *code(Q)*, divide la query in due sottoquery di range inferiore.
- Se il *code(Q)* di una query non ha sovrapposizione con la zona del nodo che la riceve, il nodo la inoltra in direzione del centroide della zona di codice *code(Q)*.

ESEMPIO PIU COMPLESSO





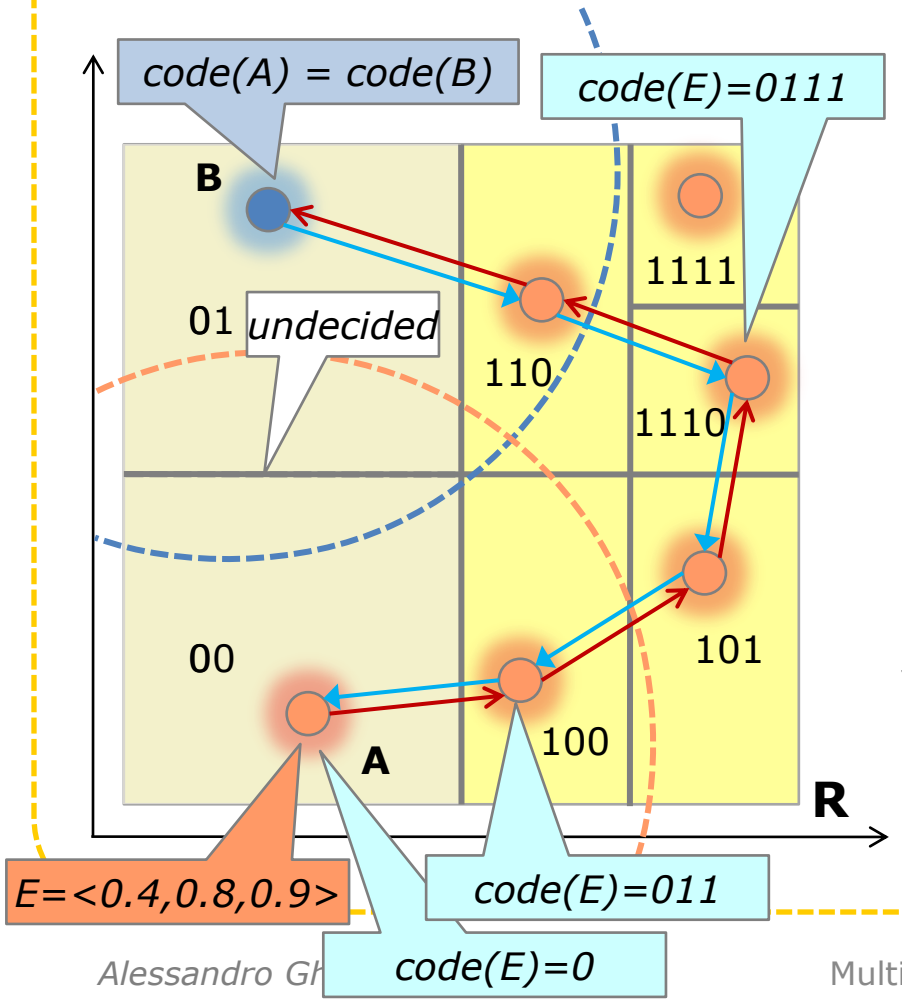

```
RESOLVE-RANGE-QUERY(Q)
 1  Q_sub ← NIL
 2  q_0, Q_sub ← SPLIT-QUERY(Q)
 3  if q_0 = NIL
 4     then c ← ENCODE(Q)
 5          if CONTAIN(c, code(A)) = true
 6             then  go to step 12
 7             else  SEND-MESSAGE(c, q_0)
 8     else  RESOLVE(q_0)
 9          if IS_INTERNAL() = true
10             then  Absorb (q_0)
11             else  Append q_0 to Q_sub
12          if Q_sub ≠ NIL
13             then for  each subquery q ∈ Q_sub
14                  do c ← ENCODE(q)
15                     SEND-MESSAGE(c, q)
```

CONFINI "UNDECIDED"

- Può succedere che a causa di ostacoli o di ridotta portata radio non sia rilevato nessun nodo in una certa direzione.
  - *Viene costruita un'idea "provvisoria" della zona considerando "**undecided**" il lato in direzione del quale non risulta esserci nessun altro nodo.*
- La risoluzione completa di una zona con confini "*undecided*" viene differita all'atto di inserimento di un evento o di avvio di una query.

RIDIMENSIONAMENTO CONFINI ZONE "UNDECIDED"

***IDEA*: differire la risoluzione dei confini "undecided" durante il routing di un evento/query.**



La risoluzione dei confini "undecided" delle zone viene **differita** all'atto dell'inserimento di un evento o di avvio di una query. Tre possibilità:

1) Il nodo *A* che emette l'evento e lo inoltra, riceve quello stesso messaggio senza che nessun altro nodo intermedio abbia modificato le informazioni in esso contenute.

- *A capisce di essere il proprietario dell'intera zona compresa la parte "undecided" e memorizza l'evento in locale.*

2) Il nodo *A* che emette l'evento lo inoltra ad un vicino *D* che calcola un nuovo hash più lungo per l'evento prima di inoltrarlo nella direzione del nuovo codice di zona. Il messaggio tuttavia ritorna a *D* ma l'owner dell'evento è rimasto *A*.

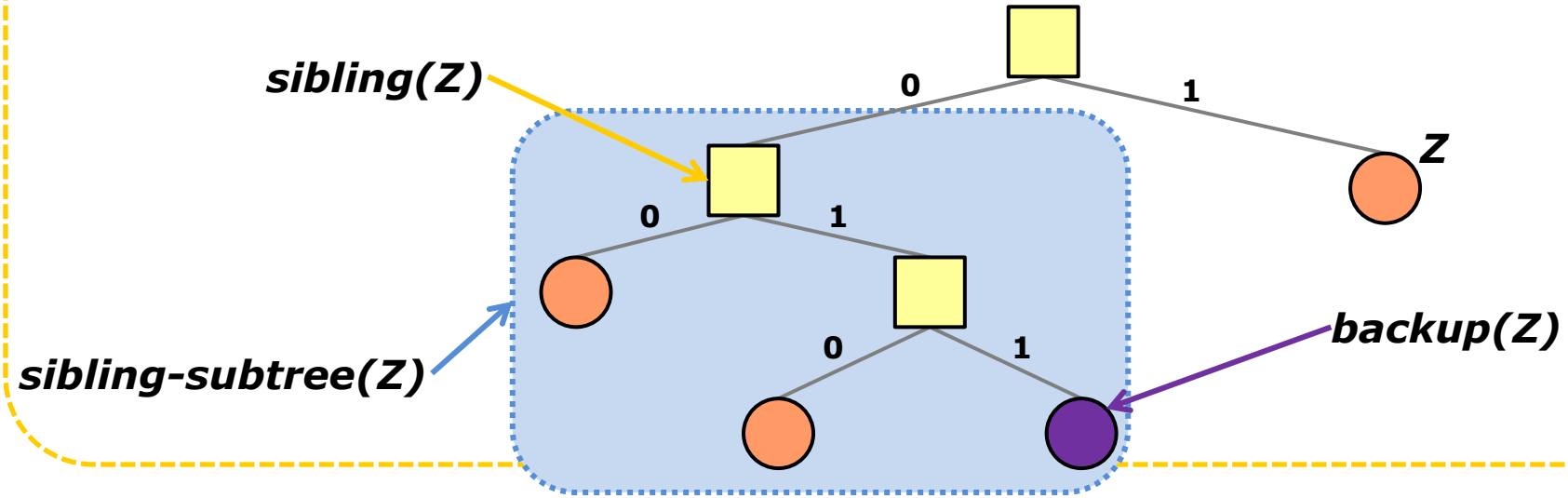
- *D capisce che A era il vero proprietario dell'evento e quindi lo incapsula in un messaggio e lo rispedisce ad A che a quel punto capisce di essere il proprietario dell'intera zona e memorizza quindi l'evento.*

3) Il nodo A che emette l'evento lo inoltra circumnavigando il perimetro del vuoto dato dal lato "undecided". Un altro nodo intermedio *B* riceve il messaggio e nota che il suo codice di zona coincide con quello del nodo *A* mittente (*v. esempio a lato*).

- *B ridimensiona la propria zona e poi invia una richiesta di ridimensionamento ad A che a sua volta elimina il lato "undecided" ed aggiorna il proprio codice di zona.*

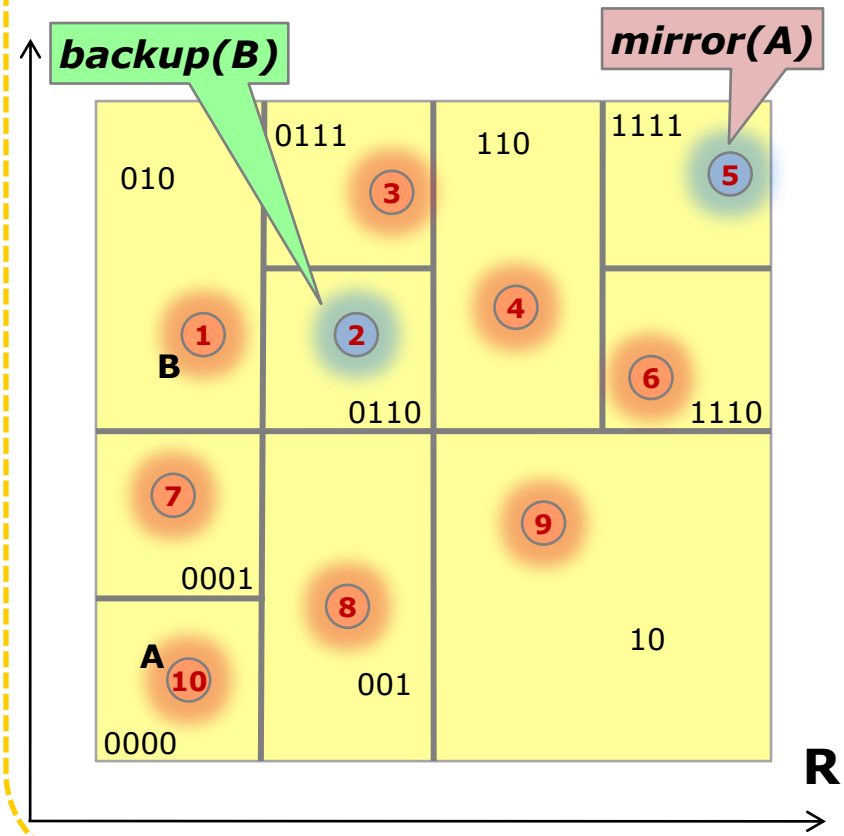
ALCUNE DEFINIZIONI

- Due zone si dicono ***sibling*** se il loro codice di zona differisce solo per l'ultimo bit.
- Il ***sibling subtree*** di *Z* è il sottoalbero che ha per radice la zona sibling della zona *Z*.
- Data una zona *Z* e detto *p* l'insieme dei *k*-1 bit del codice di *Z:*
  - *Se code(Z)=p*1*, allora il codice della backup-zone di Z sarà code(backup(Z))=p*01* che finisce con il maggior numero di 1.
  - *Se code(Z)=p*0*, allora il codice della backup-zone di Z sarà code(backup(Z))=p*10* che finisce con il maggior numero di 0.
  - N.B.: *X** sta ad indicare 0 o più ripetizioni di *X*.



ROBUSTEZZA

***IDEA*: utilizzare la replicazione per rendere il sistema robusto a guasti o alla perdita di dati.**



- Se si verifica un guasto si ha una potenziale perdita di dati
  - ***Mirror replication**: un evento di codice* code(E) *viene inoltrato alla zona con tale codice e nella zona con codice negato (i.e.: 0110 → 1001)*
    - Il mirror di un nodo è dalla parte diametralmente opposta della regione coperta dalla rete di sensori.
    - Soluzione costosa: coinvolgimento molti nodi per inviare le repliche
  - ***Local replication:** si usa il nodo di backup. Essendo in genere vicino alla zona di codice* code(E) *il costo delle repliche non domina il traffico di rete.*
    - Soluzione più economica: coinvolti nell'invio di repliche solo pochi nodi vicini alla zona di destinazione (in reti dense tipicamente ***1-2 hop*** per raggiungere il nodo di backup).

ROBUSTEZZA (DETTAGLI)

- DIM supporta aggiunta/rimozione di nodi
  - *Il meccanismo di costruzione delle zone è dinamico e le zone saranno ridimensionate se si hanno dei cambiamenti nella topologia della rete.*
  - *Prima che un nodo venga spento, tutti i dati in esso memorizzati vengono inviati al suo **nodo di backup**.*
  - *Espansione di zona permessa solo tra **nodi sibling**.*
- DIM è robusto a guasti dei nodi
  - *Se non arrivano segnali GPSR da una zona vicina, il nodo viene considerato guasto.*
  - *Tutti i dati che erano stati inoltrati al nodo guasto, vengono presi in carico dal nodo che ne ha rilevato la zona oppure dal relativo nodo di backup.*
- DIM è robusto alla perdita di pacchetti
  - *Si fanno più tentativi richiedendo la stessa query. Se non si ottiene comunque una risposta per l'intero range richiesto, si restituisce solo la porzione di dati ricevuta.*

DIM: ANALISI PERFORMANCE

- Parametri di valutazione e confronto:
  - ***Costo medio di inserimento di un evento***
    - Numero medio di messaggi necessari per inserire un evento
  - ***Costo medio di risoluzione di una query***
    - Numero medio di messaggi per inoltrare una query a tutti i nodi rilevanti
- In una rete con *N* nodi, il costo medio di inserimento è proporzionale a ***sqrt(N)***.
- Il costo di risoluzione delle query dipende dall'ampiezza del range
  - *L'algoritmo esegue lo split solo quando la query arriva nella zona rilevante*

DIM: ANALISI PERFORMANCE

- Supponiamo che le ampiezze delle query seguano una funzione di densità di probabilità *f(x)*.
  - *Il costo medio di risoluzione è dato da*

$$\int_{1}^{N} x f(x) dx$$

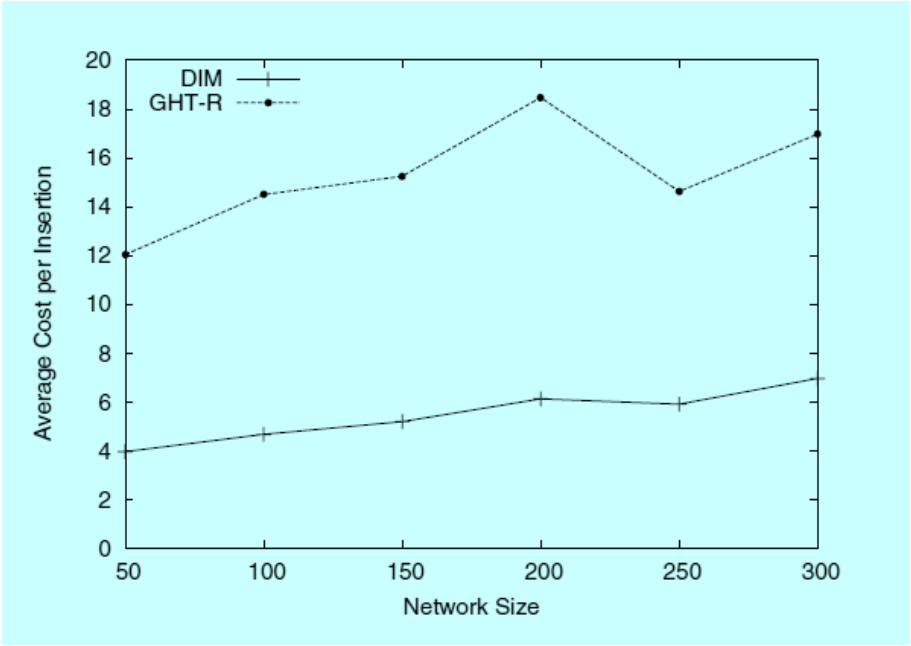
dove x è la dimensione della query e f(x) la densità di probabilità per quel range.

Tipo distribuzione	Costo medio range query
Uniforme	*O(N)*
Uniforme con limite B	*O(B)*
Algebrica → $f(x) = e^{-cx}$	*O(sqrt(N))*
Esponenziale → $f(x) = x^{-k}$	*O(sqrt(N))*

SIMULAZIONI

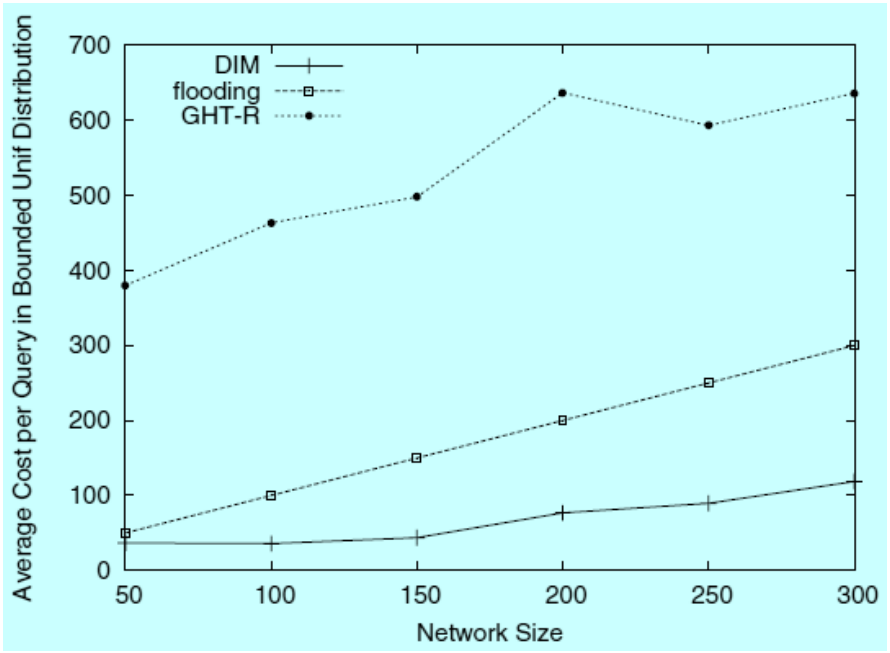
- Utilizzo del sistema *ns-2* modificato: aggiunto supporto a GPSR, DIM e GHT-R.
- Dimensione rete varia da 50 a 300 nodi
- Ogni nodo ha in media 20 nodi all'interno della propria copertura radio

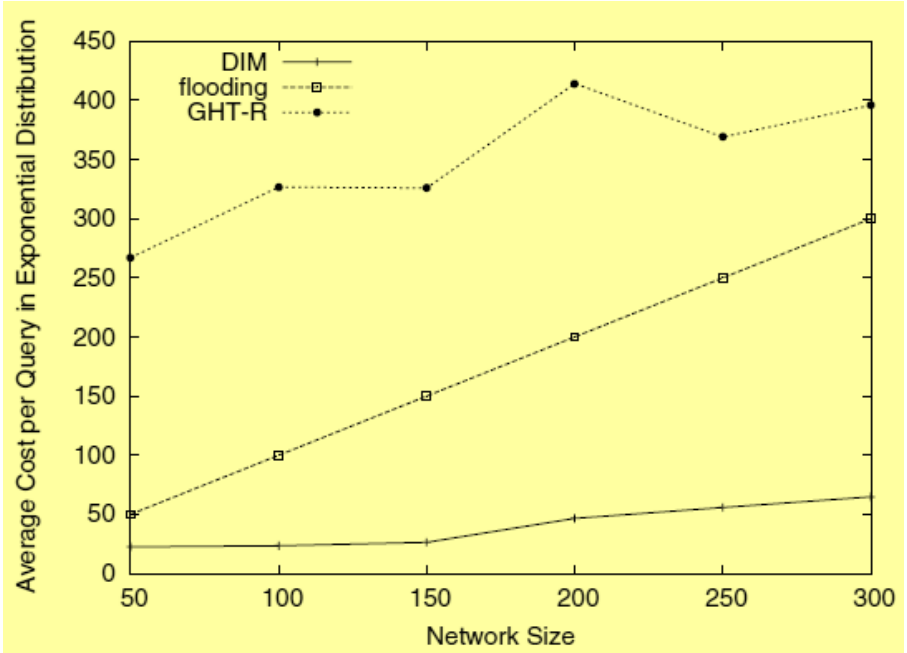
RISULTATI INSERIMENTO



- Il DIM ha performance migliori al GHT-R per quanto riguarda l'inserimento degli eventi (il costo in sistemi basati su flooding è nullo - storage locale al nodo)
- Questo è dovuto al fatto che le circumnavigazioni dei vuoti si hanno solo quando si hanno boundary undecided, mentre con il GHT-R vengono fatte quasi per ogni inserimento.

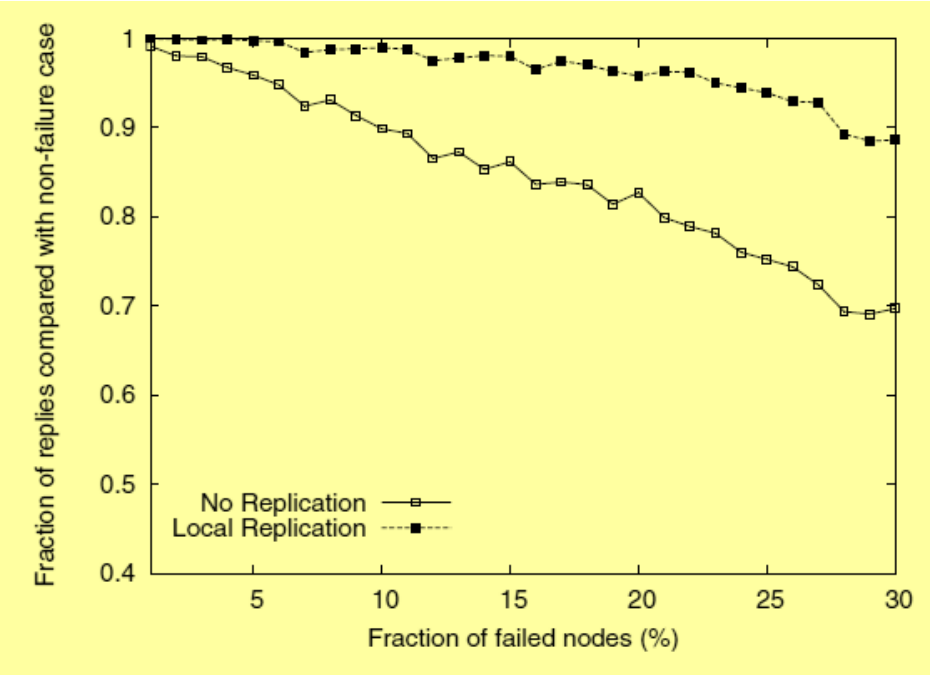
RISULTATI QUERY





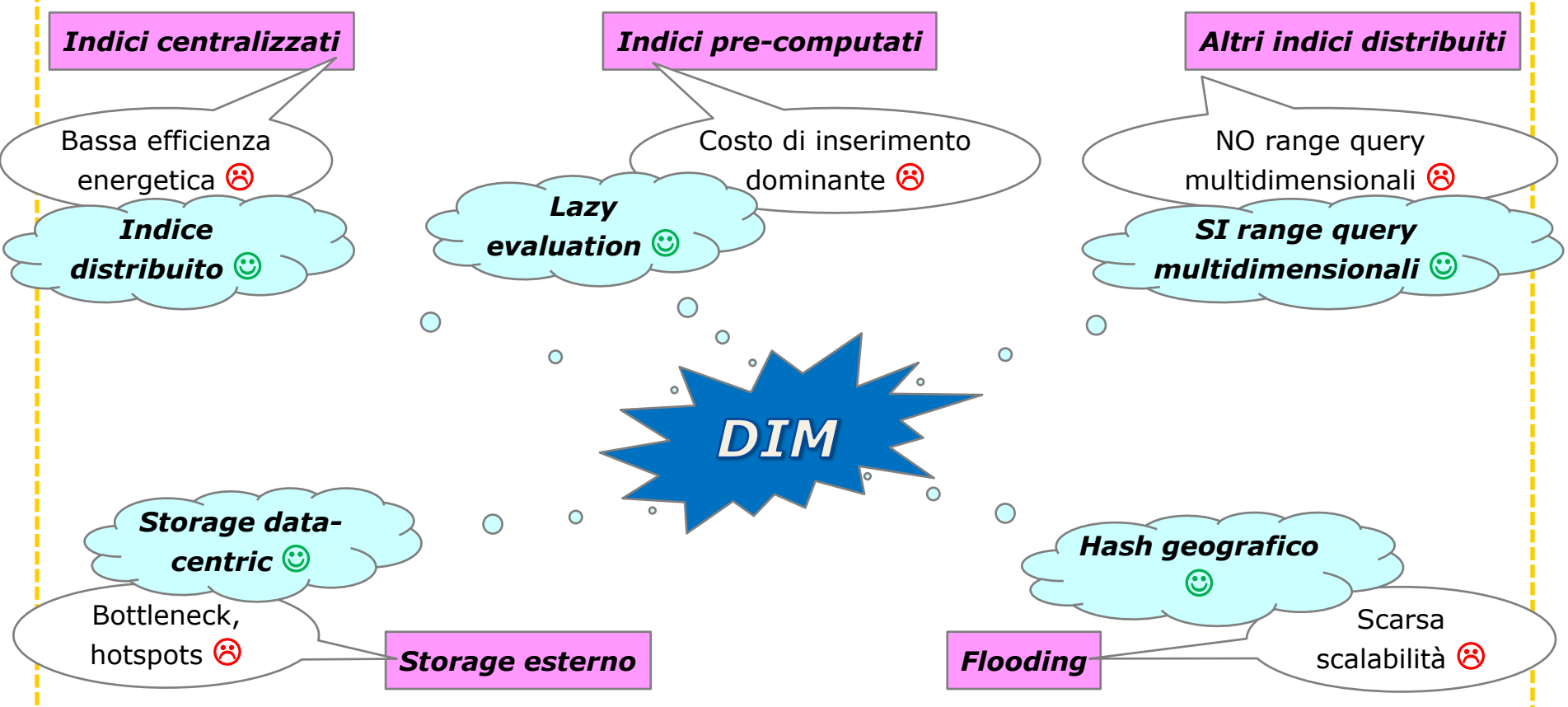
- Il DIM ha performance migliori al GHT-R e al flooding sia con distribuzione di range uniforme limitata sia con distribuzione esponenziale.
- Tipicamente, i range delle query richieste sono comunque limitati (è raro vengano effettuate query su range molto ampi)
  - *E' ragionevole affermare che il costo medio di query tende a* ***sqrt(N)***

RISULTATI REPLICAZIONE



- In una rete con 100 nodi sono stati inseriti eventi in modo uniformemente distribuito su tutti i nodi e poi è stata emessa una range query che coprisse l'intera rete, registrando i risultati.
- Conoscendo i risultati, si è simulato il guasto *simultaneo* di una certa percentuale di nodi in modo casuale e si è riemessa la query.
- Considerando un 30% di nodi spenti, con la replicazione locale si ottiene circa il 90% delle risposte, viceversa solo il 70%.

CONCLUSIONI



Alessandro Ghidotti Piovan

GRAZIE PER L'ATTENZIONE

GRUPPO 20