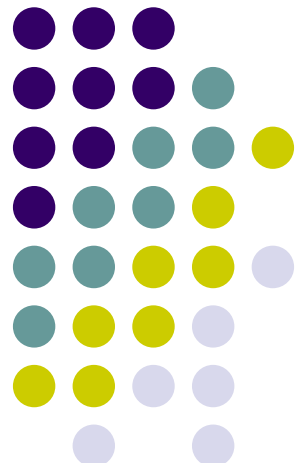
Efficient and Scalable Query Routing for Unstructured Peer-to-Peer Networks

Presented by 曾胤燁

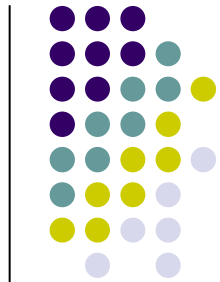
IEEE INFOCOM2005



Outline

- Introduction
- Design
- Experiment
- Conclusion

Introduction



- Design an efficient query routing mechanism for unstructured peer-to-peer networks.

- To build routing tables at nodes.
  - Use a novel data structure EDBF

- Scalable Query Routing (SQR) mechanism
  - Maximum probability

EDBF

- Exponential Decay Bloom Filter
- EDBF is an extension of the traditional Bloom filter to encode the probabilistic forwarding table.

Bloom filter

- Given a query, EDBF simply return indicator θ (x), the number of 1's in the filter.

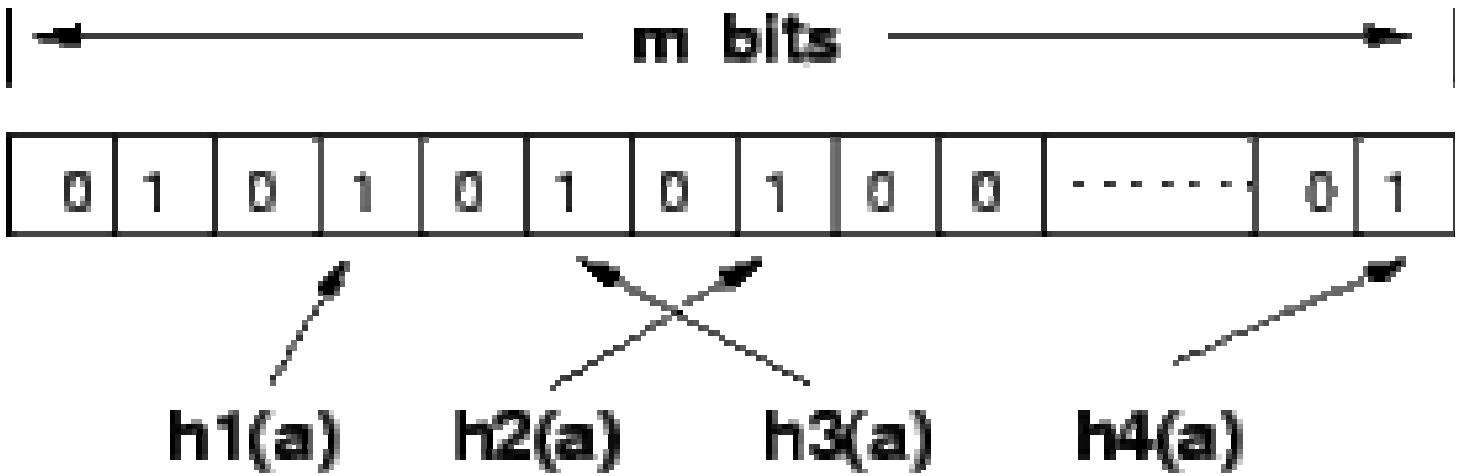
  θ (x) / k -> probability

EDBF

Bloom filter

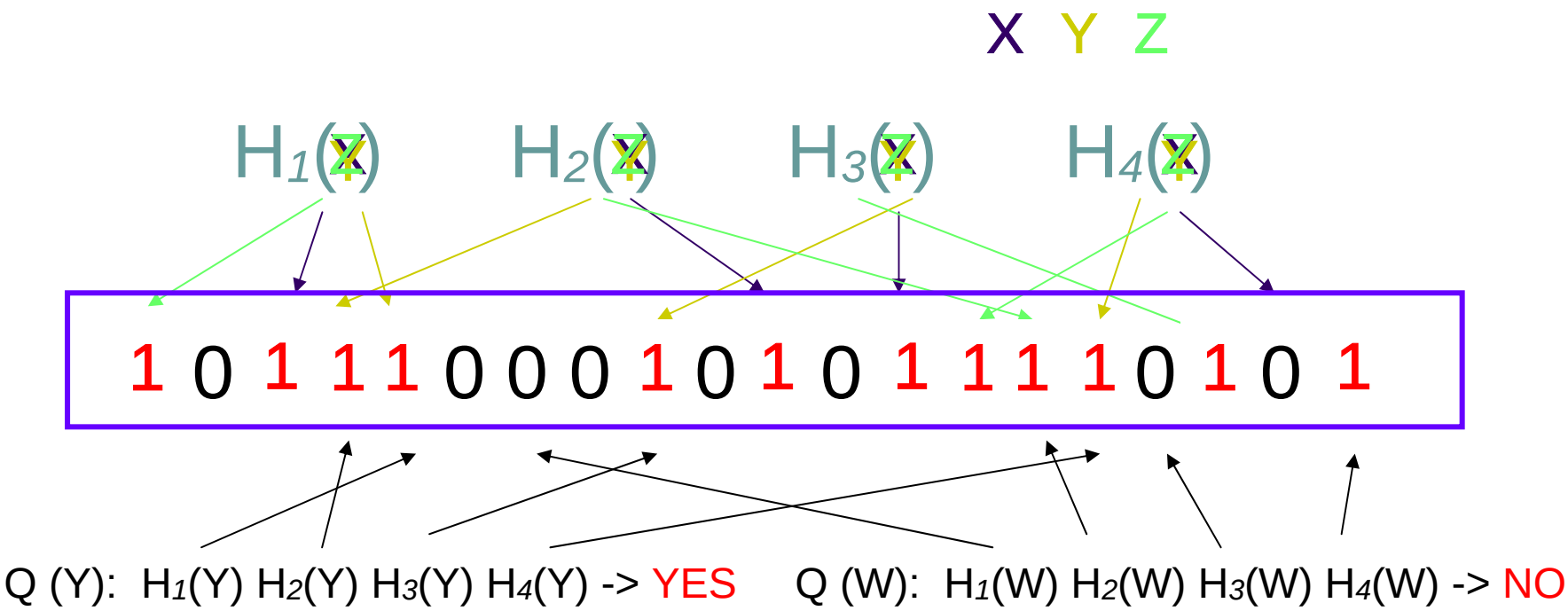
- A data structure for approximately answering set membership questions.
- An array of bits A initialized to all 0.
- Fixed number (k) of hash functions

- When a inserts, $h_i(a)$, i =1,2,..k are set to 1.



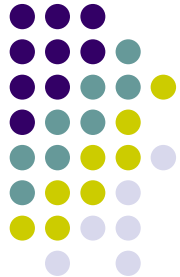
- A query for y returns a yes if and only if all $h_i(y)$, i=1,2,..k are 1.

EXAMPLE


X Y Z
H1(X) H2(X) H3(X) H4(X)
1 0 1 1 1 0 0 0 1 0 1 0 1 1 1 1 0 1 0 1
Q (Y): H1(Y) H2(Y) H3(Y) H4(Y) -> YES
Q (W): H1(W) H2(W) H3(W) H4(W) -> NO

優點

- 可將多筆資料紀錄於同一array
- 利用多個hash function可減少碰撞的機會



EDBF

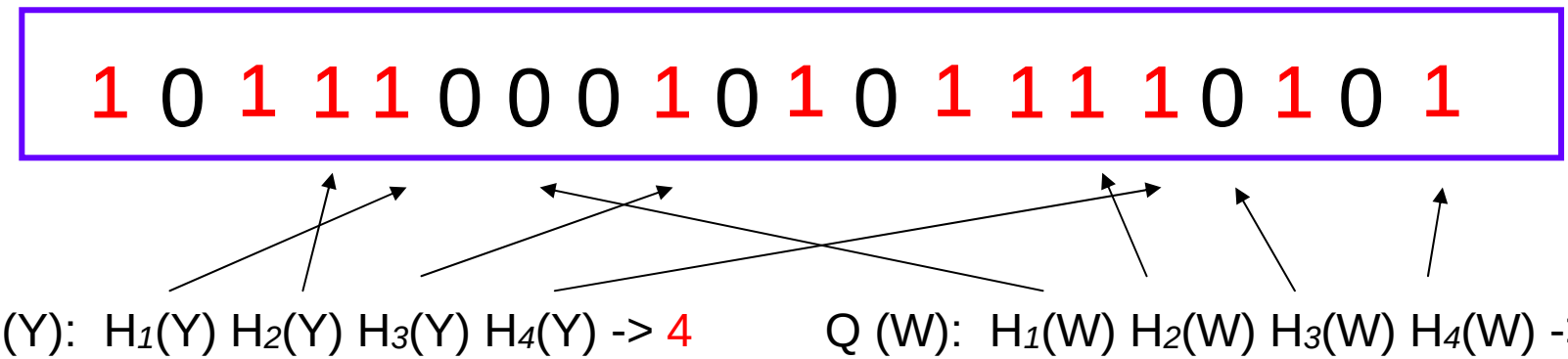
- Exponential Decay Bloom Filter
- EDBF is an extension of the traditional Bloom filter to encode the probabilistic forwarding table.

Bloom filter

- Given a query, EDBF simply return indicator θ (x), the number of 1's in the filter.

  θ (x) / k -> probability

EDBF


1 0 1 1 1 0 0 0 1 0 1 0 1 1 1 1 0 1 0 1
Q (Y): H1(Y) H2(Y) H3(Y) H4(Y) -> 4
Q (W): H1(W) H2(W) H3(W) H4(W) -> 2

EDBF

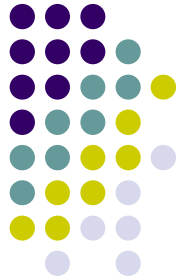
- Exponential Decay Bloom Filter
- EDBF is an extension of the traditional Bloom filter to encode the probabilistic forwarding table.

Bloom filter

- Given a query, EDBF simply return indicator θ(x), the number of 1's in the filter.

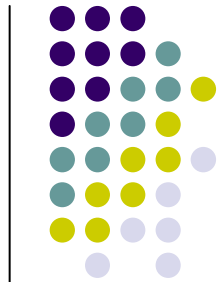
  θ (x) / k -> probability

EDBF

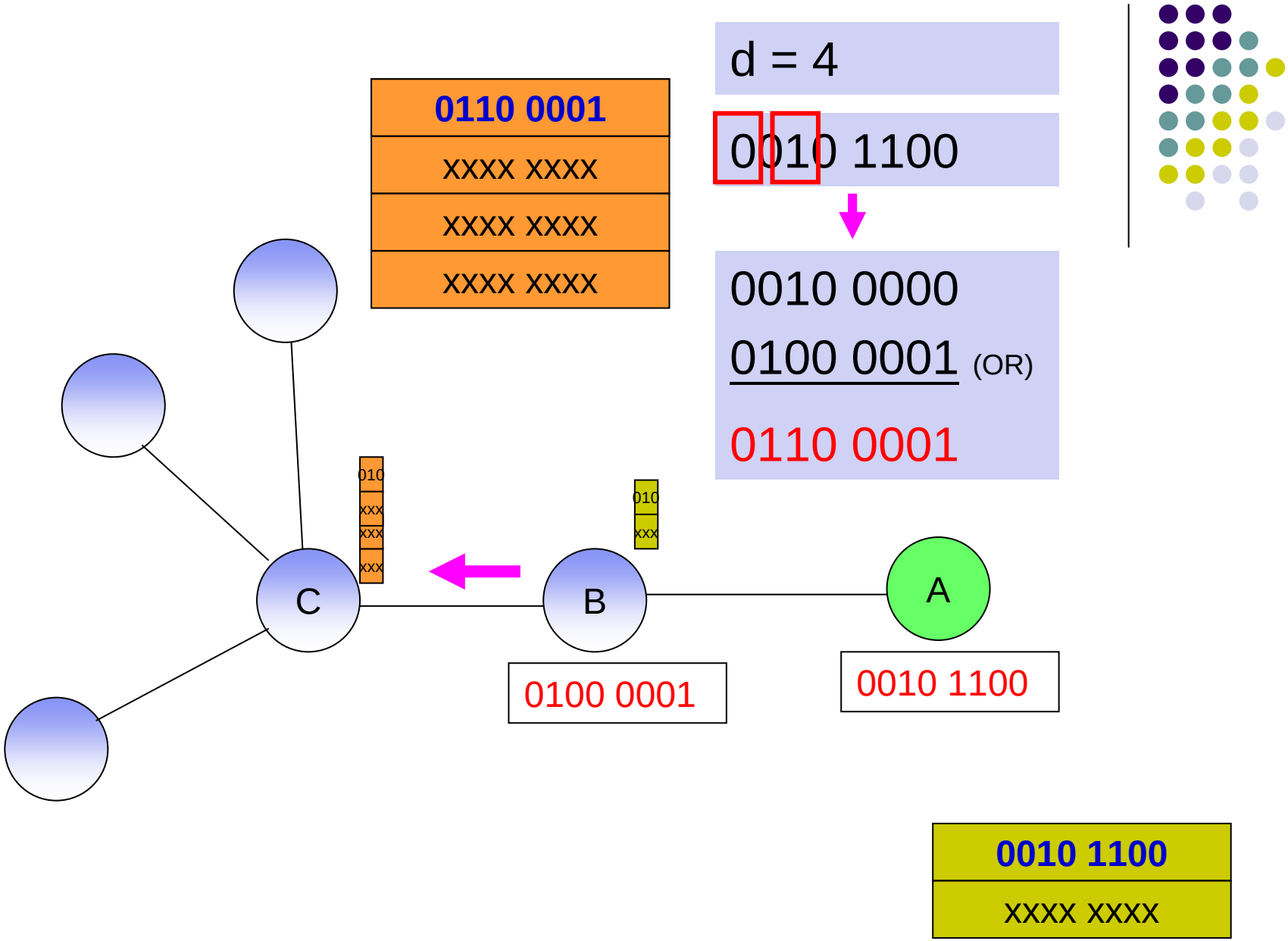


- The number of bits decays exponentially with its distance form the node where the object x is stored.→EDBF

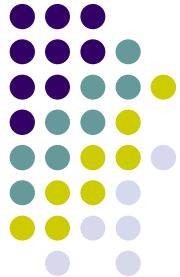
Maintenance of routing tables



- A node's (with degree j) routing table consists of j EDBF arrays.
- Maintain consistency by periodic update
- Information regarding content x will decay with distance


d = 4
0010 1100
0010 0000
0100 0001 (OR)
0110 0001
0110 0001
XXXX XXXX
XXXX XXXX
XXXX XXXX
010
XXX
XXX
XXX
010
XXX
C
B
A
0100 0001
0010 1100
0010 1100
XXXX XXXX

Algorithms for creating updates



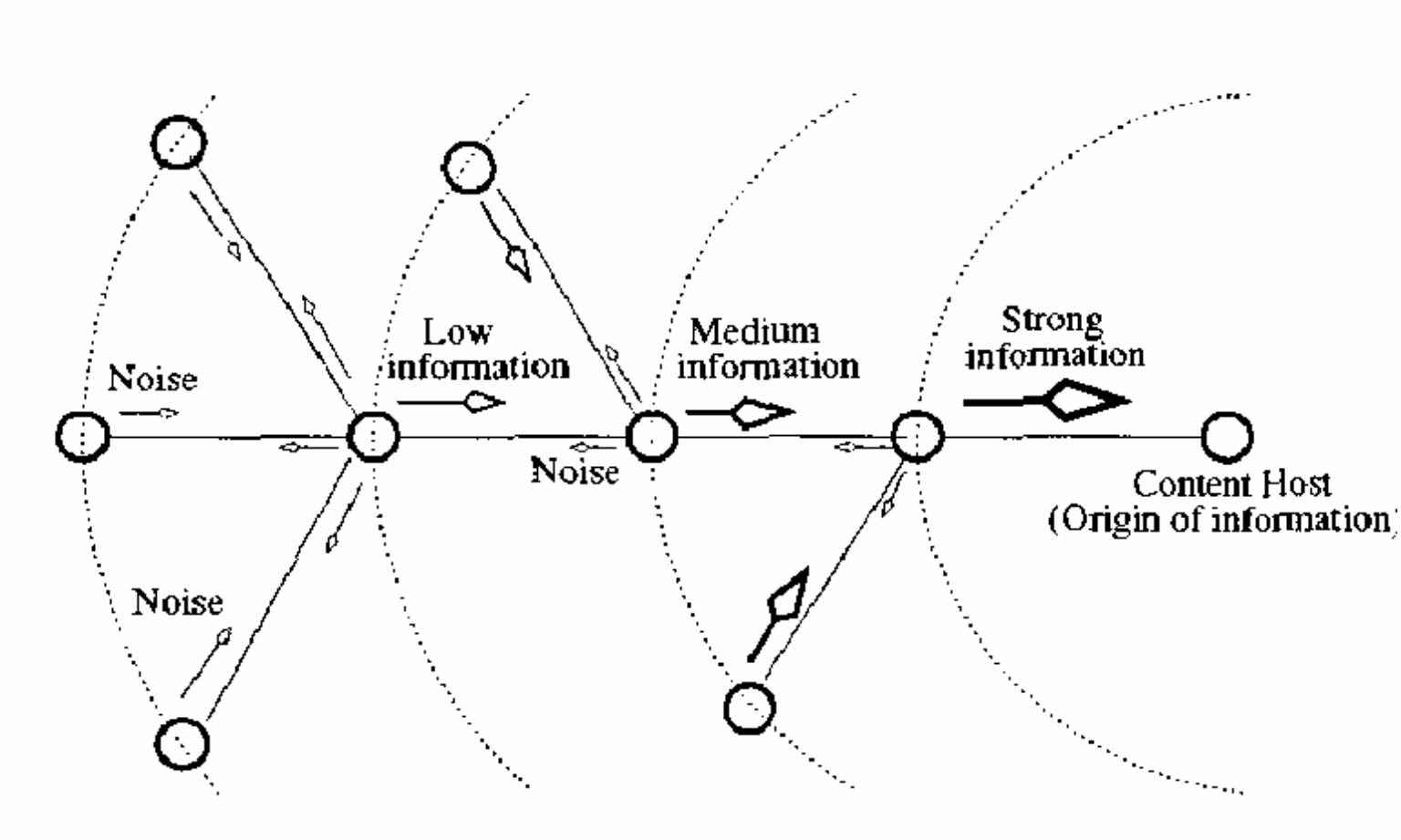
```
Create Local EDBF (given local content X):
      // Populate local EDBF A.
1.  ∀x ∈ X
2.        Set bits A[h_1(x)], ..., A[h_k(x)] to 1;

Create Update (for neighbor j):
    // Copy all the bits from the local EDBF A into
    // the update U_j.
1.  U_j ← A;
    // Decay the information received from all neighbors
    // other than j by a factor of d, and add the
    // surviving bits to U_j.
2.  ∀i ∈ neighbor_list, i ≠ j
3.        ∀r ∈ {1, ..., m}
4.              if(A_i[r] == 1)
5.                    with probability 1/d, U_j[r] ← 1;
6.  Return U_j;
```

Query Forwarding

-Scalable Query Routing (SQR) mechanism

- The query is forwarded to the neighbor with the highest indicator value θ (x).

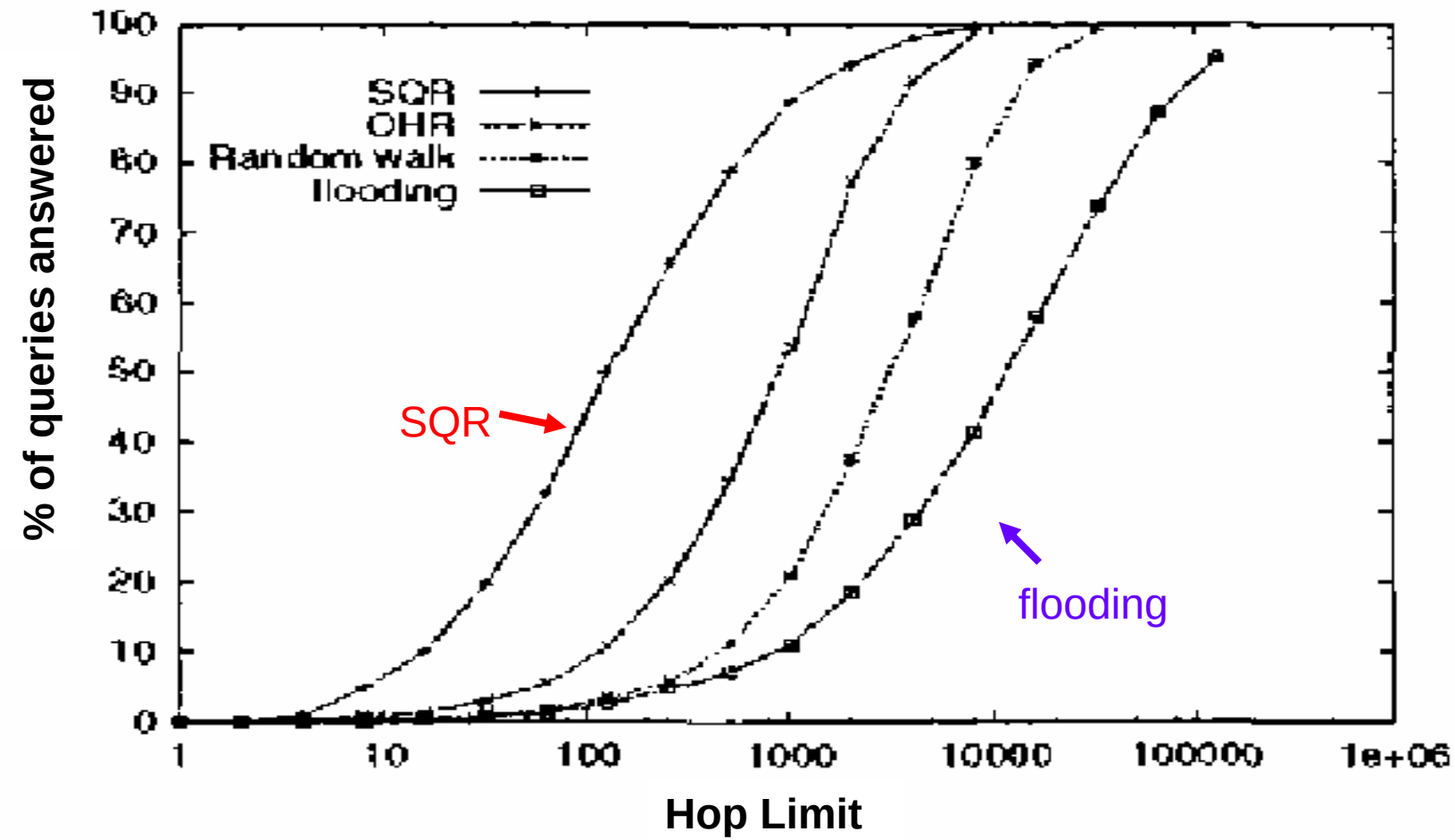
```
Forward Query (given query Y):
        // Forward previously seen queries to neighbor i, .
        // chosen randomly from neighbor_list.
1. if ( Seen Query(Y))
2.      Deliver Query(Y, i);
3.  else
    //Forward previously unseen queries to the neighbor
    // with the maximum information about this query.
4.      Θ = Lookup (Y);
5.      Pick i such that θ_i = max(Θ);
6.      Deliver Query(Y, i);

Lookup (given query Y):
1. ∀i ∈ neighbor_list
2.      ∀q ∈ {1, ···, k}
3.          θ_i += A_i[h_q(y)];
4. Return Θ;            /*Θ = {θ_i}*/
```

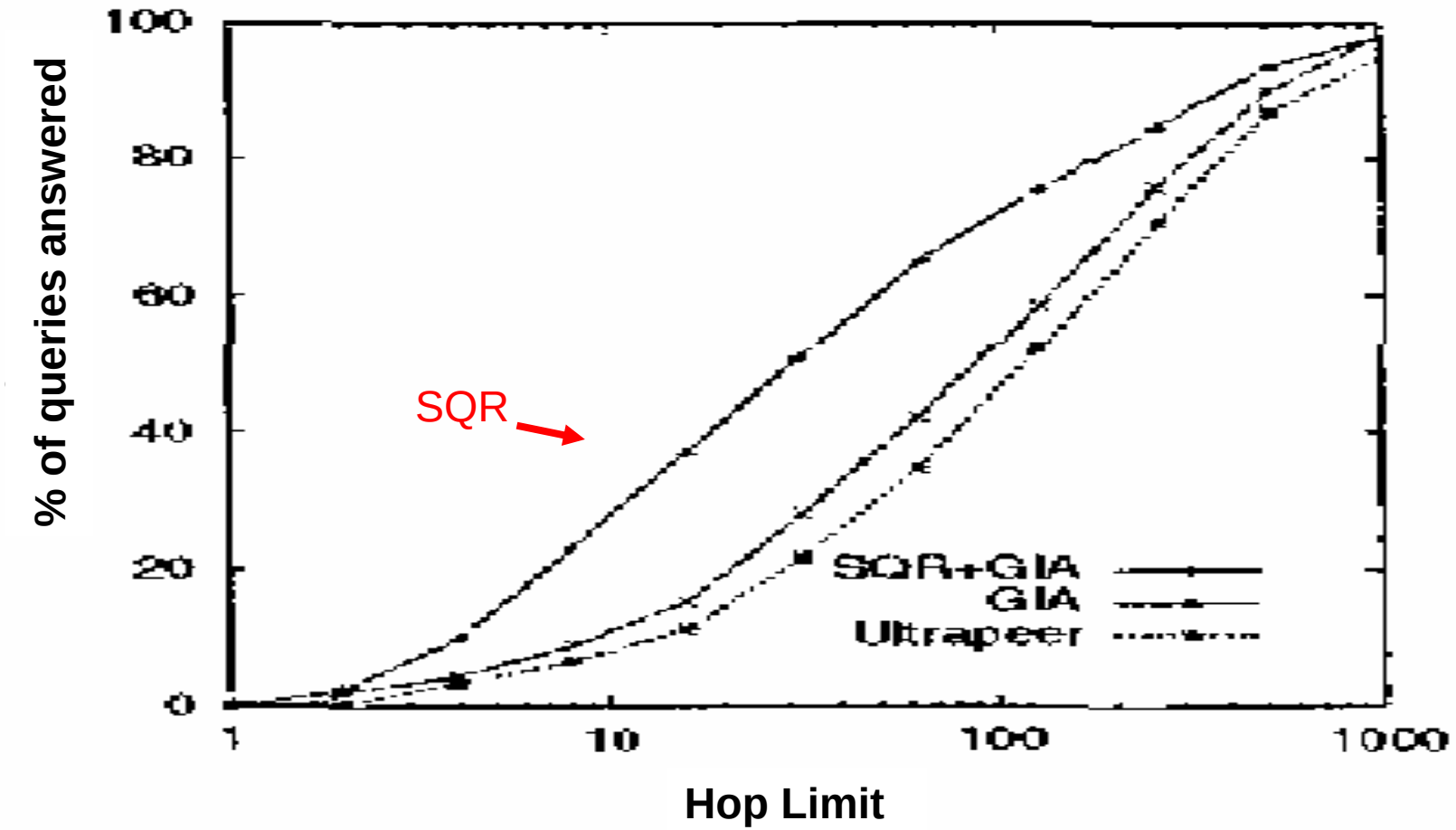

Noise
Low
information
Medium
information
Strong
information
Noise
Content Host
(Origin of information)
Noise

優點

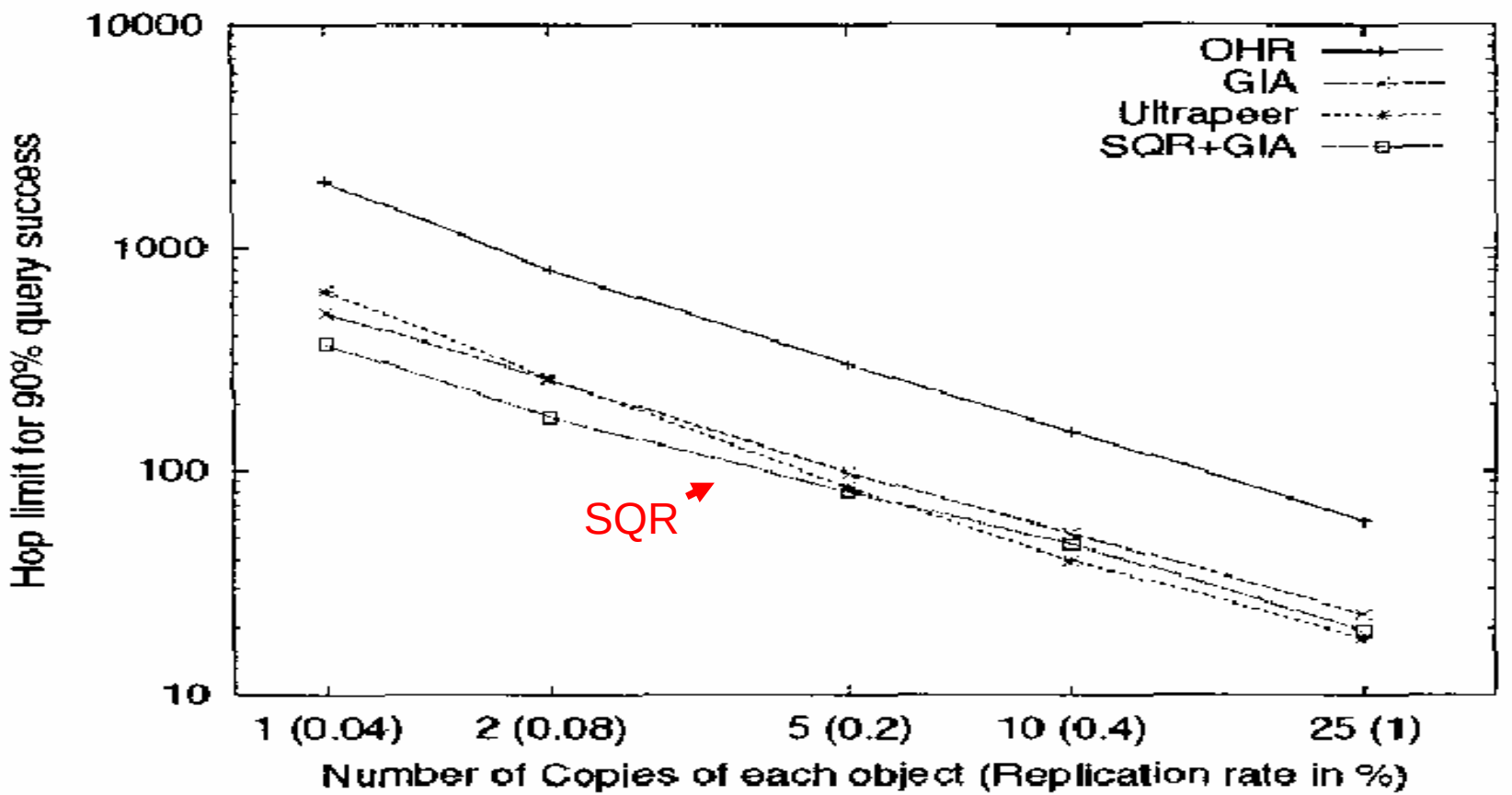
- 使用BF，可使每一node所儲存的資料量不至於太大
- 利用information decay，可使在每一peer中的資料重要性被區分出來，距離較遠的資訊比較不會干擾到距離較近較重要的資訊-避免Noise

Query response rate



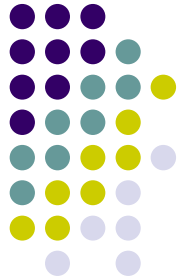

% of queries answered
100
80
60
40
20
0
1
10
100
1000
Hop Limit
SQR
SQR+GIA
GIA
Ultrapeer

Impact of Replication



Conclusions

- In this paper, we have explored the approach of spreading probabilistic information about the location of hosted content in its neighborhood.
- Simulation based comparison with various query routing mechanisms establish the performance advantages of SQR



Thank you☺