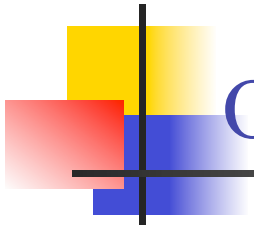
Supporting Multi-dimensional Range Queries in Peer-to-Peer Systems

Yanfeng Shu, Beng Chin Ooi, Kian-Lee Tan

School of Computing, National University of Singapore

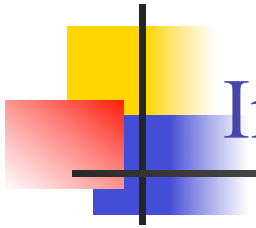
Aoying Zhou

Department of Computer Science, Fudan University, China



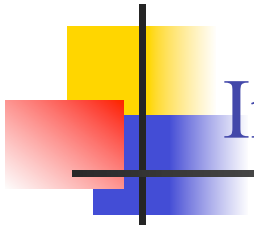
Outline

- Introduction
- ZNet
- Experimental Results
- Conclusion



Introduction

- Today's P2P systems are unable to cope well with range queries on multi-dimensional data
  - *Unstructured systems (e.g., Gnutella)*
    - mainly depend on flooding
    - no performance guarantee
    - cannot ensure data availability
  - *Structured systems (e.g., Chord)*
    - Have guarantee on search efficiency and data availability
    - mainly designed for ***exact*** key lookup

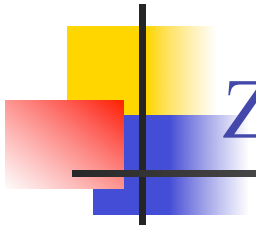


Introduction

cont …

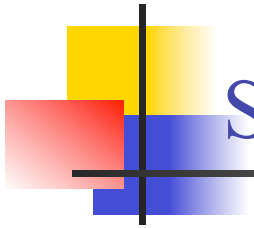
To extend existing P2P systems and thus support range queries on multi-dimensional data *efficiently* and *effectively*:

- *Space Partitioning and Mapping*
  --- how is a data space partitioned and mapped onto nodes?
- *Query Processing*
  --- what overlay is based on and what strategy is employed for query processing?
- *Load Balancing*



ZNet

- The whole data space is dynamically partitioned and mapped
  - *-- space is partitioned into different granularities*
  - *-- Space Filling Curves(SFCs) at different orders are used to preserve data locality*
- Range queries are efficiently supported
  - *Skip graphs are extended for query rouing, with each node maintaining only O(logN) states;*
  - *An efficient range query resolution strategy is proposed, which evaluates queries in a specific way to avoid unnecessary node visits.*
- Both Static and Dynamic load balancing are addressed

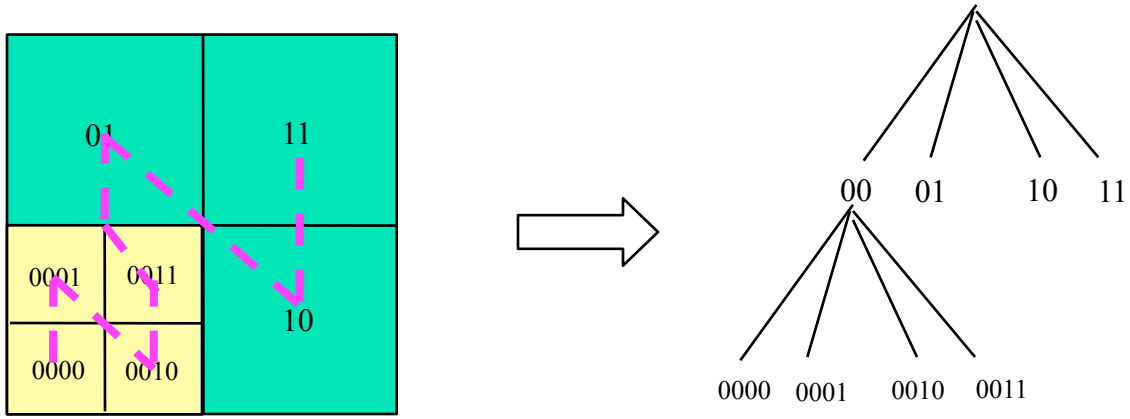


Space Partitioning and Mapping

- When?
  - *Node join, leave/failure*
  - *Load balancing*
- How?
  - *halved in all dimensions*
  - *each subspace is called a zone, which is numbered according to its position* w.r.t. *the split dimensions*
- The whole partitioning process can be modelled as a tree, *partition tree*

Space Partitioning and Mapping

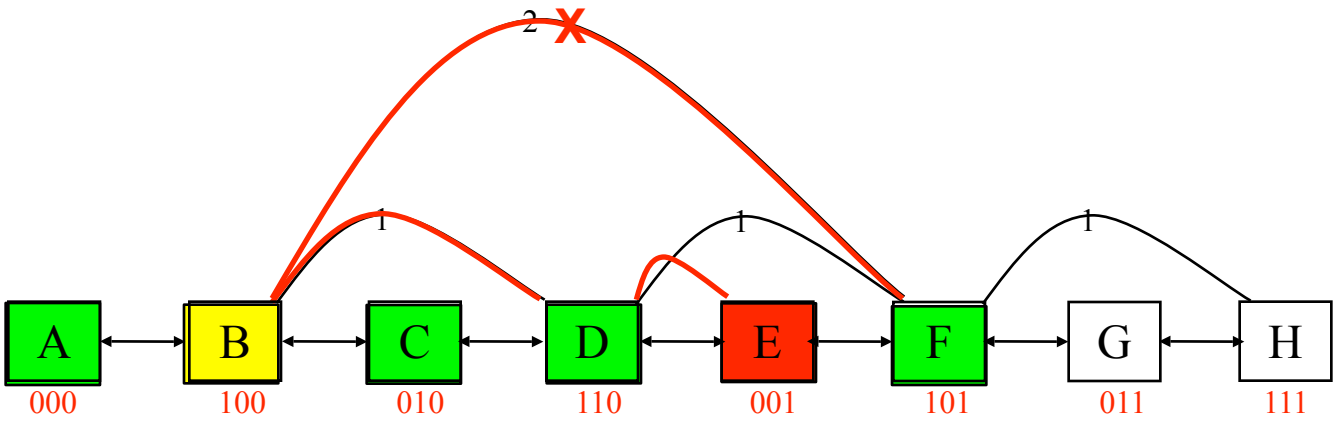
cont …



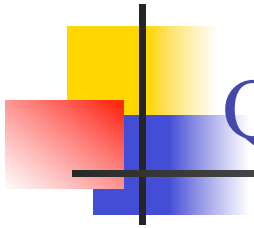
- This equals to filling zones (from one partitioning) with a first order z-curve. The whole space is filled with z-curves at different orders, and a zone' code is just its Z-address
- Continuous zones whose Z-addresses are continuous are mapped to the same node or nearby nodes

Query Processing

Skip Graphs: multiple-level linked lists with each node's links (neighbors) are selected at random



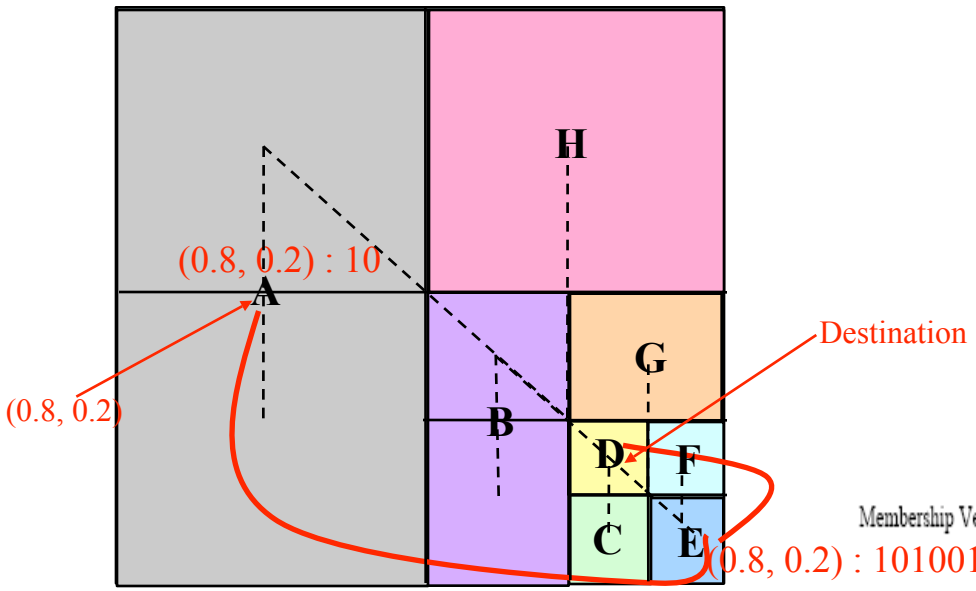
Query processing *cont …*

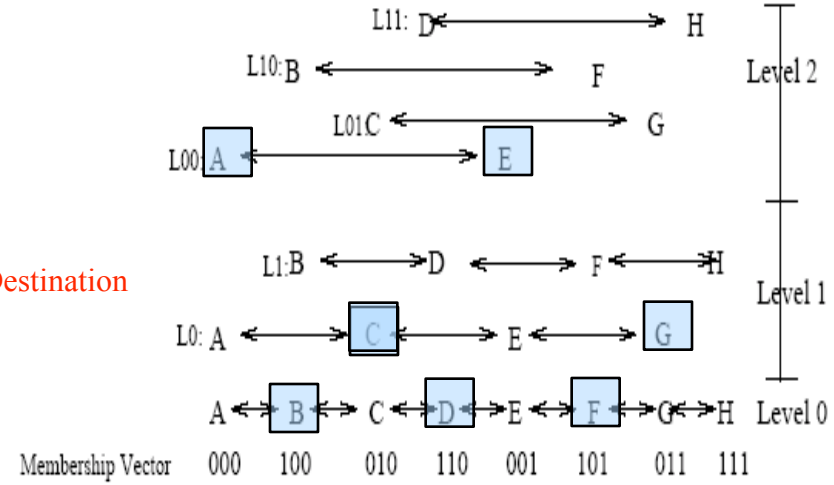


- ZNet extends Skip Graphs by assigning continuous zones to nodes
- Each node only needs to maintain O(log N) states, where N is the number of nodes
- When given a search key (a point), a node will first transform the point to a z-address, then the search request is routed according to the z-address by following the routing process of Skip Graphs
- Complexities rise in that each node has only incomplete knowledge about space partitioning

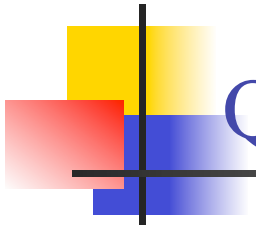
Query processing

cont …





Query Processing *cont …*



- Point /Range queries
- One method: convert the query range QR to a set of continuous Z-address ranges covered by QR, then send each request to each node which contains the minimum Z-address of each range
  - *Inefficient*
  - *Cannot deal with the dynamic case*
- Our method: convert QR to a superset of zones covered by QR. As the query is routed, the superset is refined
  - *The query is routed along two opposite directions to avoid unnecessary node revisits*

Query Processing

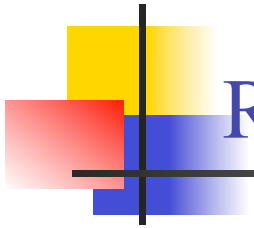
cont …

An Example:

Given QR [(0.8,0.1),(0.9,0.2)], which overlaps with C, D, E, and F, A first computes a superset of zones [10,10], then routes it to E, where the superset is refined to [101000, 101011]. Since E's space is overlaps with QR, the query will be partitioned into two parts:[101000,101001] and [101011,101011], and so on….

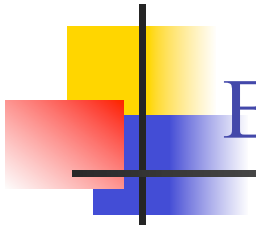
Load Balancing

- For new nodes, appropriate joining destinations are chosen
  --- *try several possible joining destinations, and the one which has the heaviest load will be chosen as the joining destination.*
- At run-time, heavily loaded nodes can migrate some of their load to lightly loaded nodes
  - *A node is heavily loaded if its load is larger than* $\overline{L} + \delta\overline{L}$*; and lightly loaded if its load is less than* $\overline{L} - \delta\overline{L}$
  - $\overline{L}$ *is average load, which is approximated by sampling loads of a node's neighbors*
  - *A node periodically changes load information with its neighbors, and heuristics are adopted to find lightly loaded nodes.*



Recent Proposals for comparison

- Squid and SCRAP
  - *Both use space-filling curves to map multi-dimensional space to one dimensional space as ZNet*
  - *They partition the space **statically** – the partitioning level needs to be decided beforehand*
  - *Squid's performance may be affected by data skewness, as it is based on Chord. SCRAP has no efficient query processing scheme*
- MURK and SkipIndex
  - *Both use K-d trees for space partitioning and mapping*
  - *They cannot cope well with dynamic load balancing*



Experimental Setup

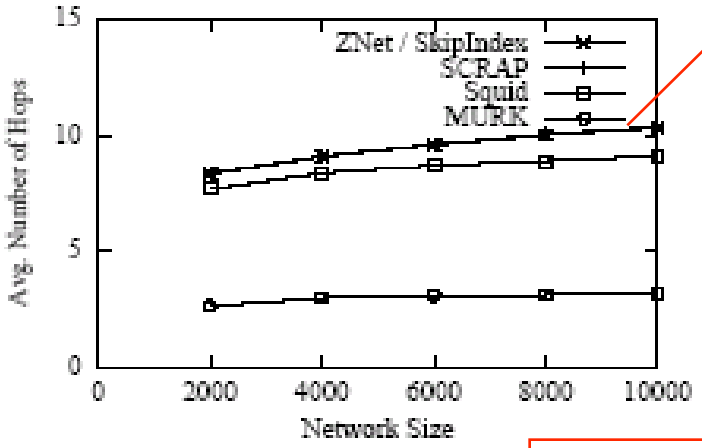
- Experiments are done via simulation on two kinds of synthetic datasets: one is skewed datasets based on normal distribution; the other is uniform datasets
- By default, we use data sets with skewed 8-dimensional 300,000 data, and 6,000 nodes in the network
- The dimensionality is varied from 2 to 20, and the number of nodes is varied from 2,000 to 10,000

Routing Cost

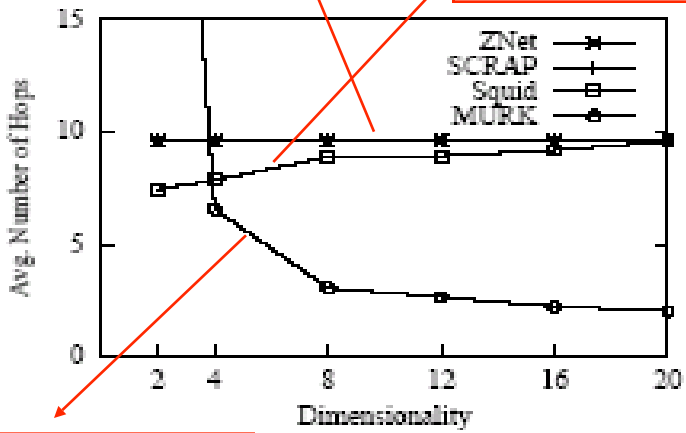
- Assume the maximum partition level of ZNet is globally known in SCRAP and Squid

The add-on dynamicity of Znet does not affect its routing performance

Squid is affected by data skewness



(a) Dimensionality = 8



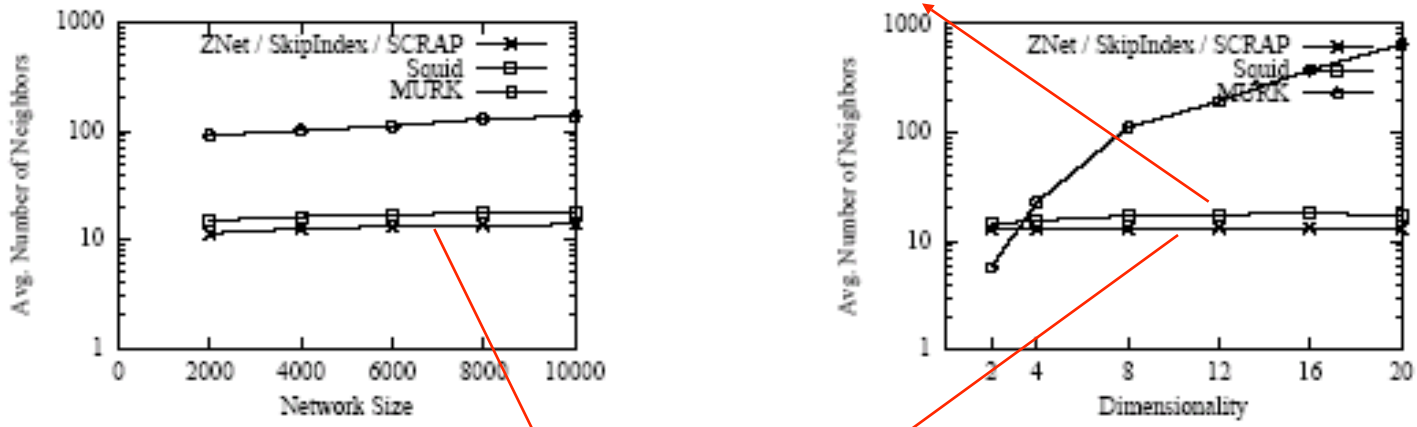
(b) Network size = 6000

MURK behaves badly when dimensionality is low

Maintenance Cost

- We measure maintenance cost in terms of the number of neighbors maintained by each node

Affected by data skewness



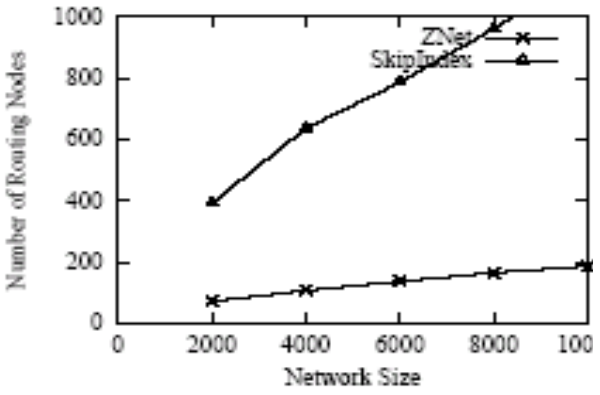
(a) Dimensionality = 8

(b) Network size=6000

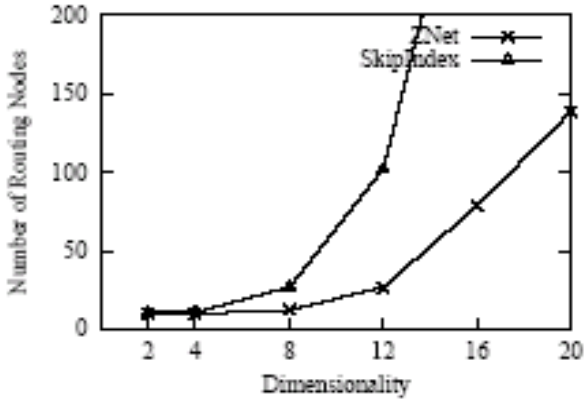
Increase logarithmically with the network size; independent of dimensionality

Range Search Cost

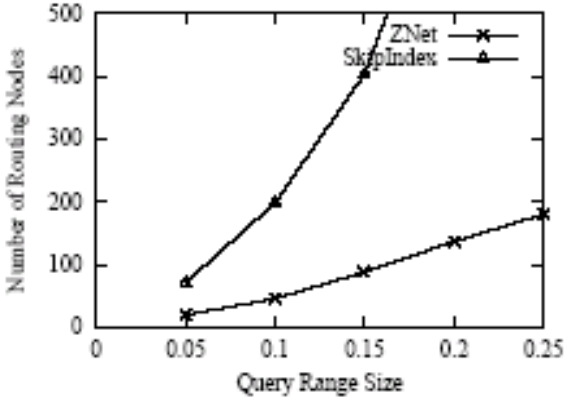
- Range search cost is measured by the number of routing nodes
- Four factors are involved: network size, dimensionality, query range size, and data distribution



(a)



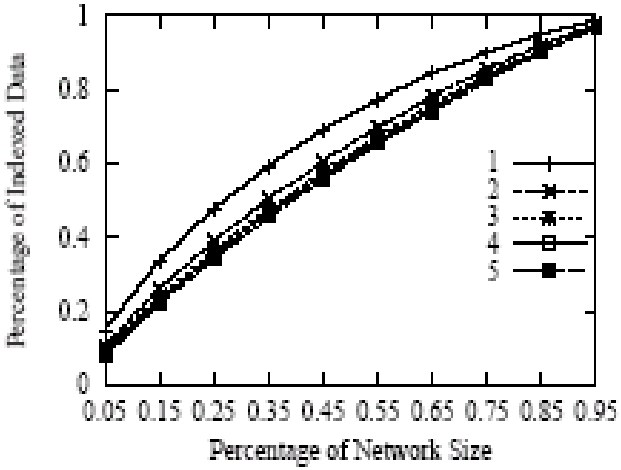
(b)

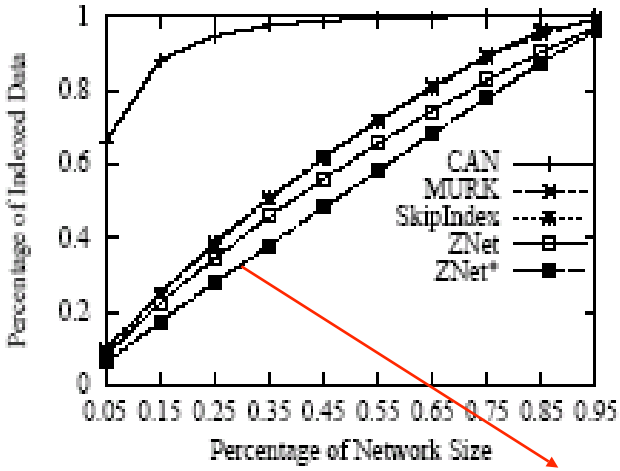


(c)

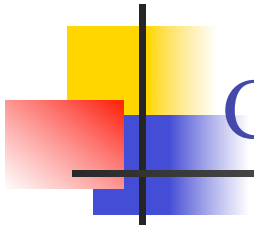
Load Balancing

- Load Balancing is measured by the amount of data indexed by each node





Znet*: dynamic load balancing in ZNet



Conclusion

- Znet possesses nearly all desirable properties, while others typically fail in one or another:
  - *has low routing cost and low maintenance cost*
  - *Increase logarithmically with the network size*
  - *independent of data dimensionality and distribuition*
  - *Low range search cost*
  - *Can achieve better load balancing*