Fast and Scalable Information Retrieval in Decentralized Content Networks

Duc A. Tran
Computer Science Department
UMASS Boston

Roadmap

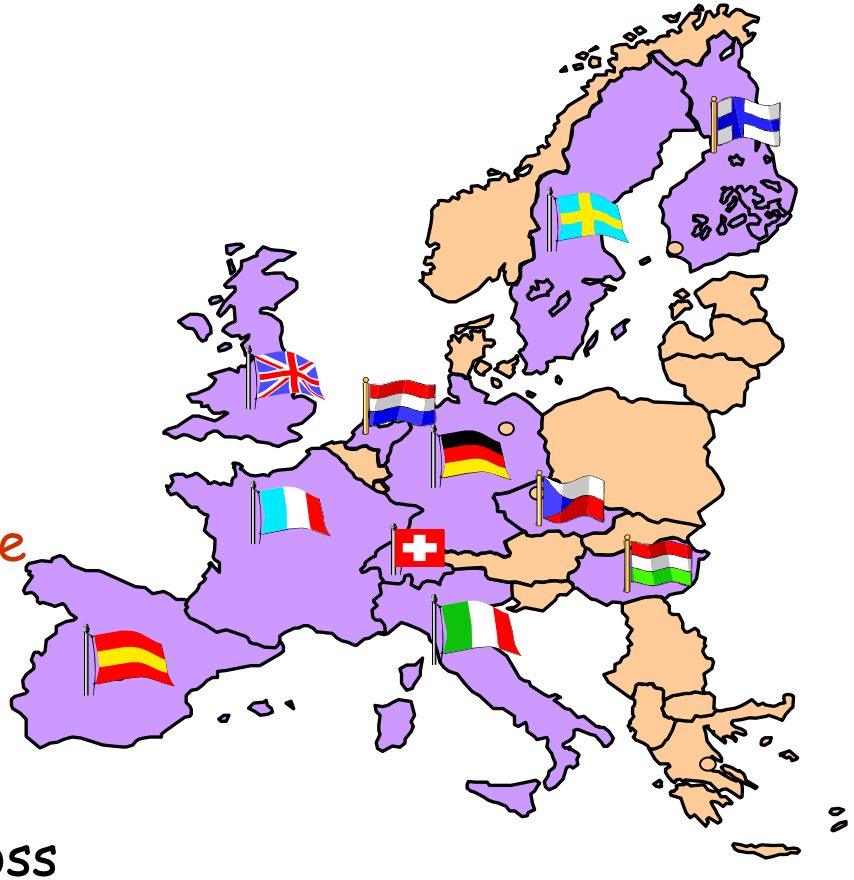
- Decentralized content networks (DCNs)
- Information retrieval in DCNs
- Proposed solution: EZSearch
- Performance study
- On-going research

Decentralized Content Networks

- An ad hoc network where nodes have the freedom to
  - Participate at any time
  - Publish content at any time
  - Access content at any time
  - Remove content at any time
  - Quit at any time
- Minimal (or zero) use of central servers for the main tasks
  - Storage
  - Networking
  - Management

DCN Example: Data Grids

- Grid
  - Multi-institution virtual organization
  - Institutions contribute data, storage, computing, and networking resources
- Data Grid:
  - Coordinated management of data distributed across remote resources
- Example: NIH Biomedical Informatics Research Network
  - Medical research centers across the US with data resources distributed



Courtesy of VLDB Tutorial 2003

DCN Example: Sensor Networks

- Sensors
  - Deployed in many places of interests
  - Capture important events in surrounding area
- Sensor data
  - Traffic: Info about vehicles that violate speed
  - Homeland security: Info about terrorist suspects detected based on face recognition
  - Environment: Info about the air conditions

DCN Example: Personal Web

Personal Web

Any end-user computing device can be a web server that stores its own contents.

URL = global ID of device + filename

Information Retrieval in DCNs

- IR: an indispensable service
- DCNs: rapidly growing in size and importance
- Problem: How to enable IR services in decentralized networks?
  - Find the best 10 computers in the grid each having about 20GB storage space available, CPU speed at least 3GHz, and usage cost between 5-10 USD/hour

What Approach to Use?

- Network size is huge
  - Kazaa: 140+ million nodes
  - Ask-everybody: do not work, too much traffic
- Volume of data is huge
  - Grow beyond the capacity of any central servers
  - Ask-server: do not work, too much resource
- ➔ need a **much better approach**

Design Goals

Query types

Exact queries
Top-K queries
Range queries

Scalable Solution

Goal: System

Resilient

Low control/storage overhead

Goal: Search

Accurate

Fast

Low search overhead

Proposed Solution: EZSearch
(Tran et. al. , Usenix05)

- Query cost and effectiveness
  - logN search time in worst case
  - Always accurate
  - Low search overhead
- System cost
  - Constant failure recovery overhead
  - Control overhead
    - Constant on average, logN worst-case
  - Storage overhead
    - No content replication
    - Indices is fairly distributed
    - Location of index is close to location of content

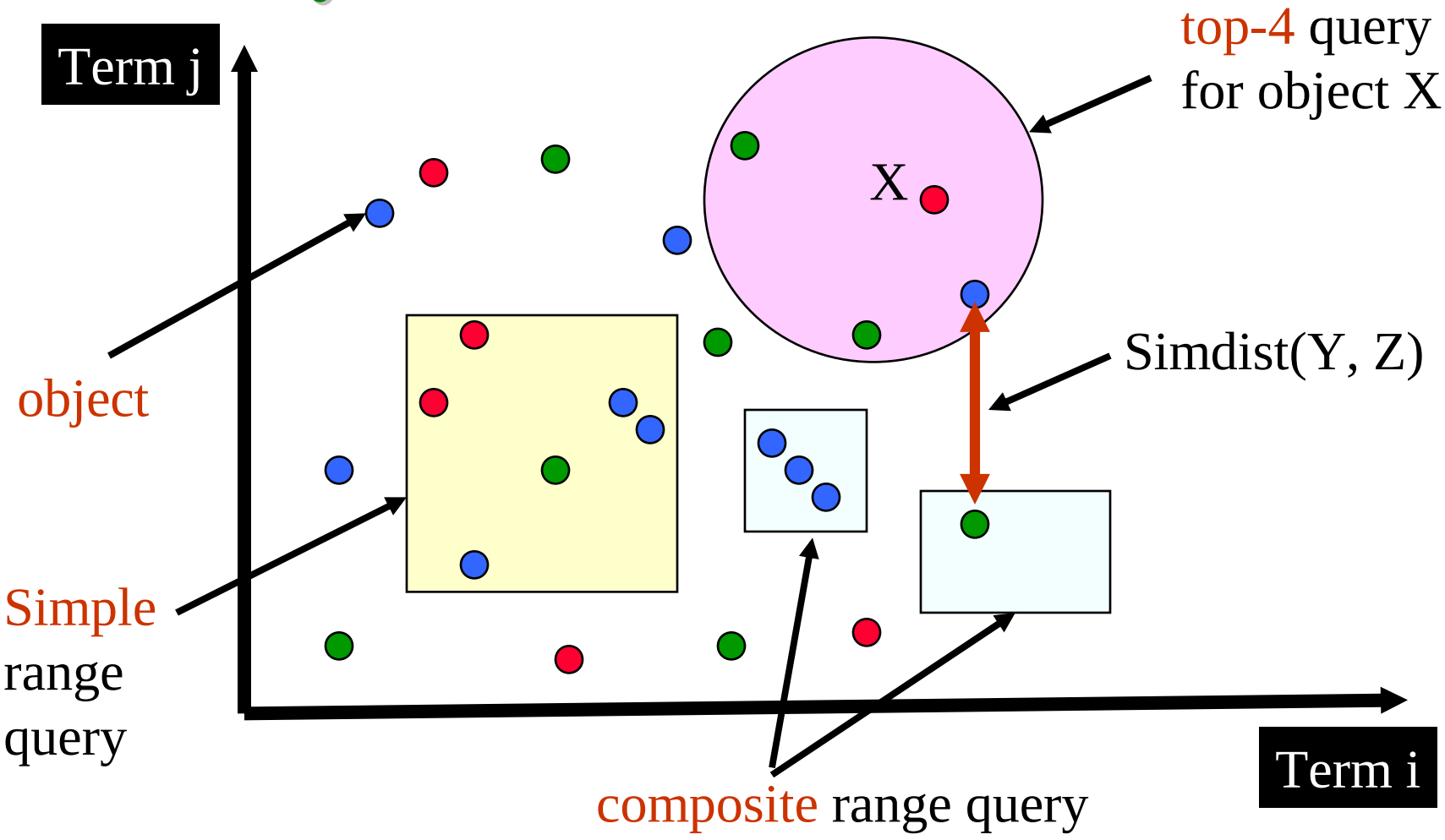
Existing Work

- SWAM
  - Banaei-Kashani and Shahabi, USC
  - ACM CIKM 2004
  - Logarithmic average-case search time
- EZSearch: Logarithmic worst-case search time

EZSearch: Overview

- Content and query description
- Node clustering
  - Low control overhead
- Index-to-cluster assignments
  - Monotonically approach query objects as we traverse the network
  - Indices of similar objects are in the same cluster
- Search algorithms
- Network construction and maintenance
- Performance evaluation

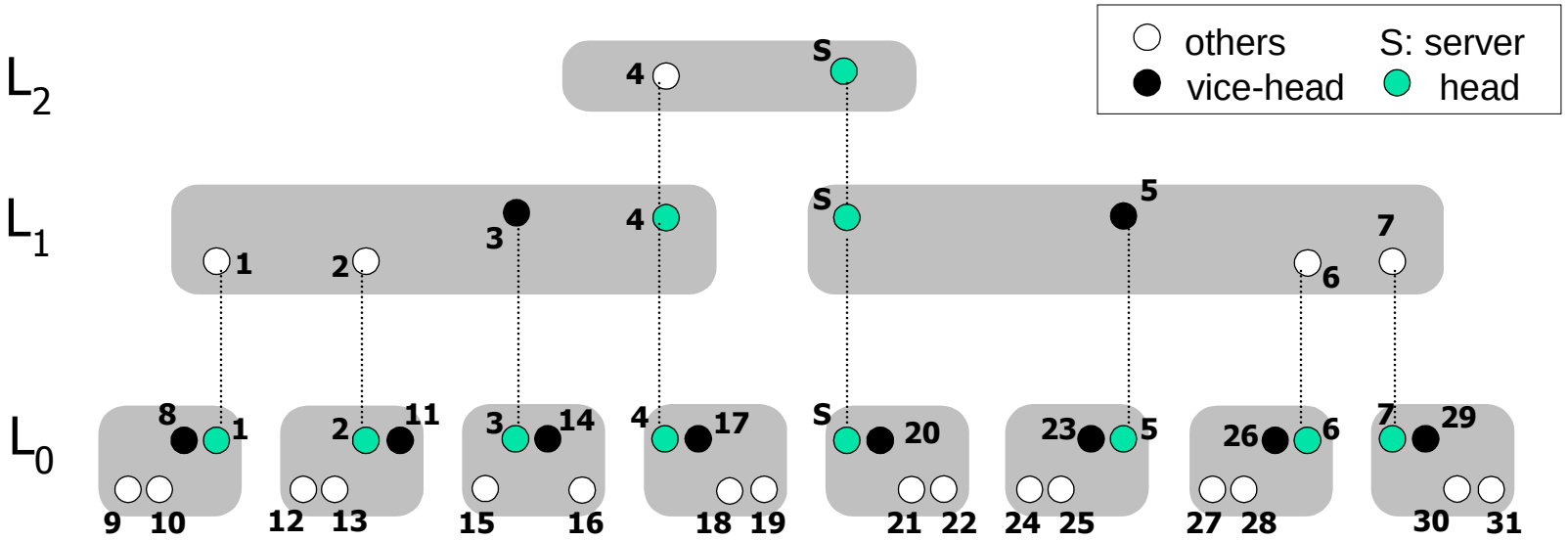
Objects and Queries



Object $x = (w_{1x}, w_{2x}, \ldots, w_{dx}) \in [0, 1)^d$

Node Clustering: Zigzag Hierarchy

(Tran et. al., Infocom03, JSAC 04)

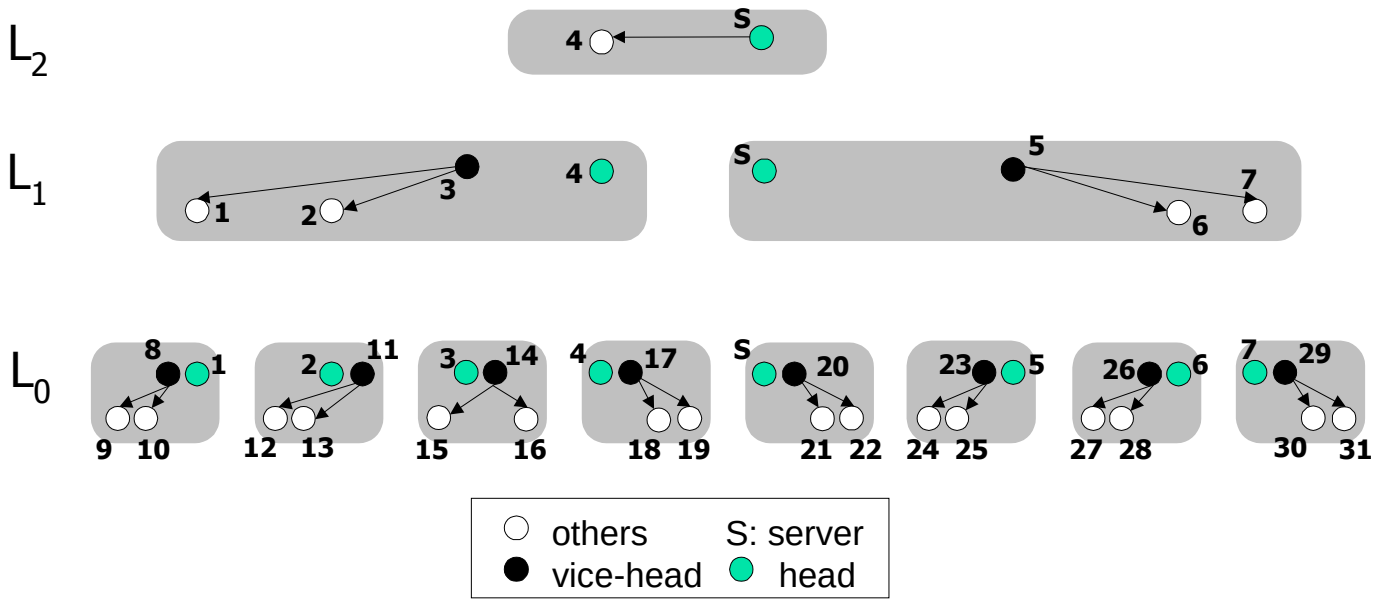


Cluster Size: [k, 3k]

Highest-layer cluster: [2, 3k] ($k \geq 5$)

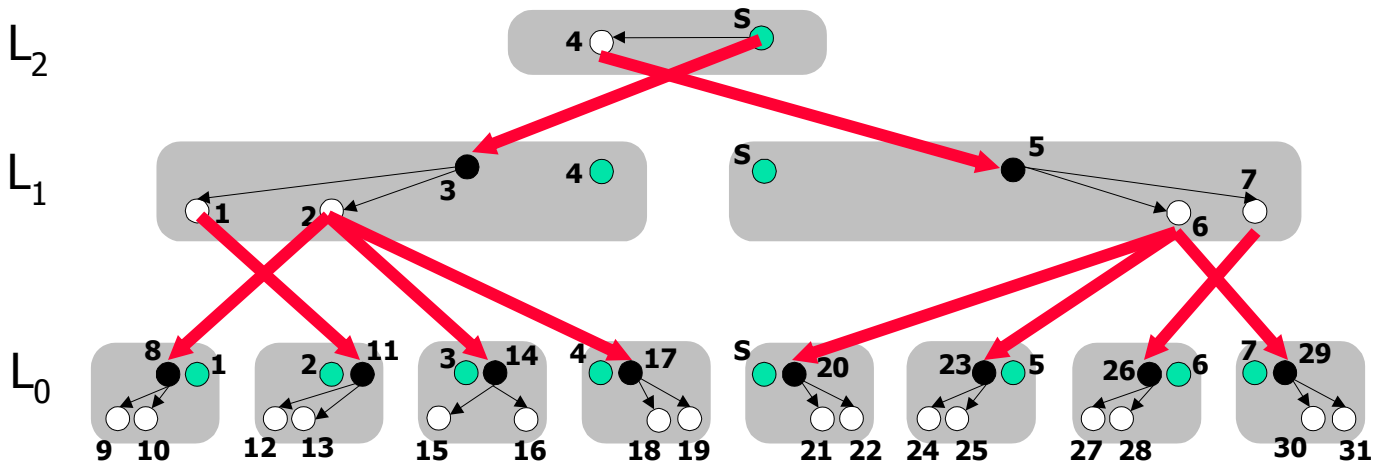
➔ No. layers $\in [\log_{3k} N, \log_k N+1]$

Intra-Cluster Connectivity



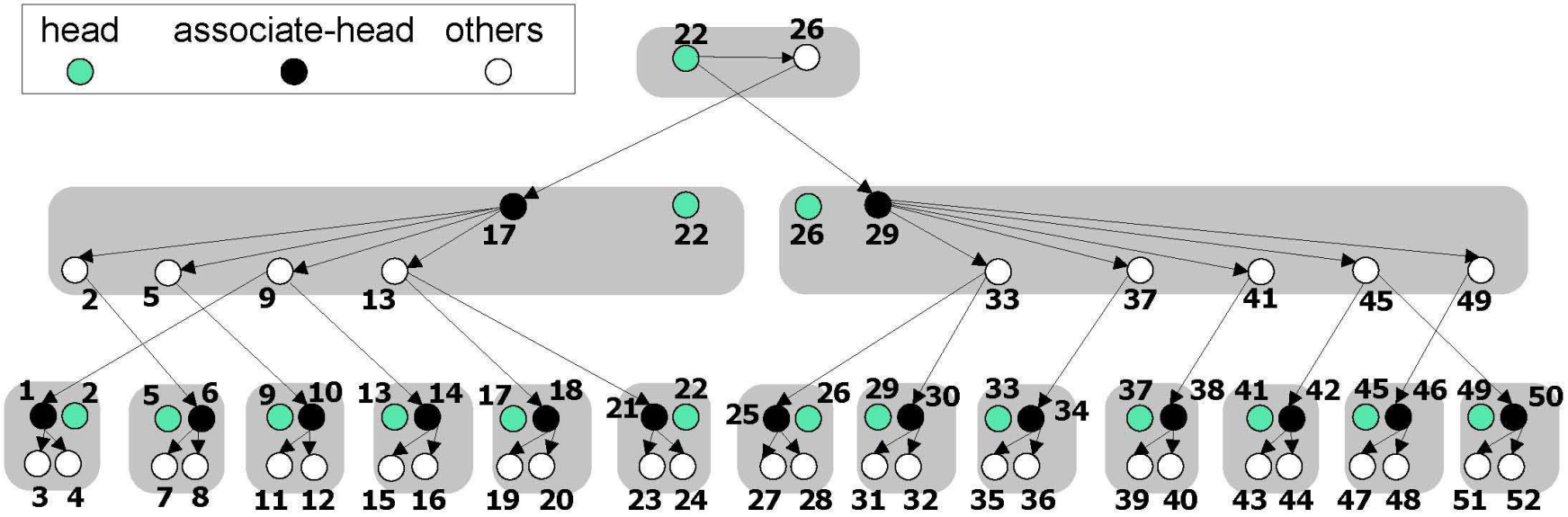
Non-head members of a cluster are children of their vice-head

Inter-Cluster Connectivity



Vice-head of a cluster is child to one of the foreign heads

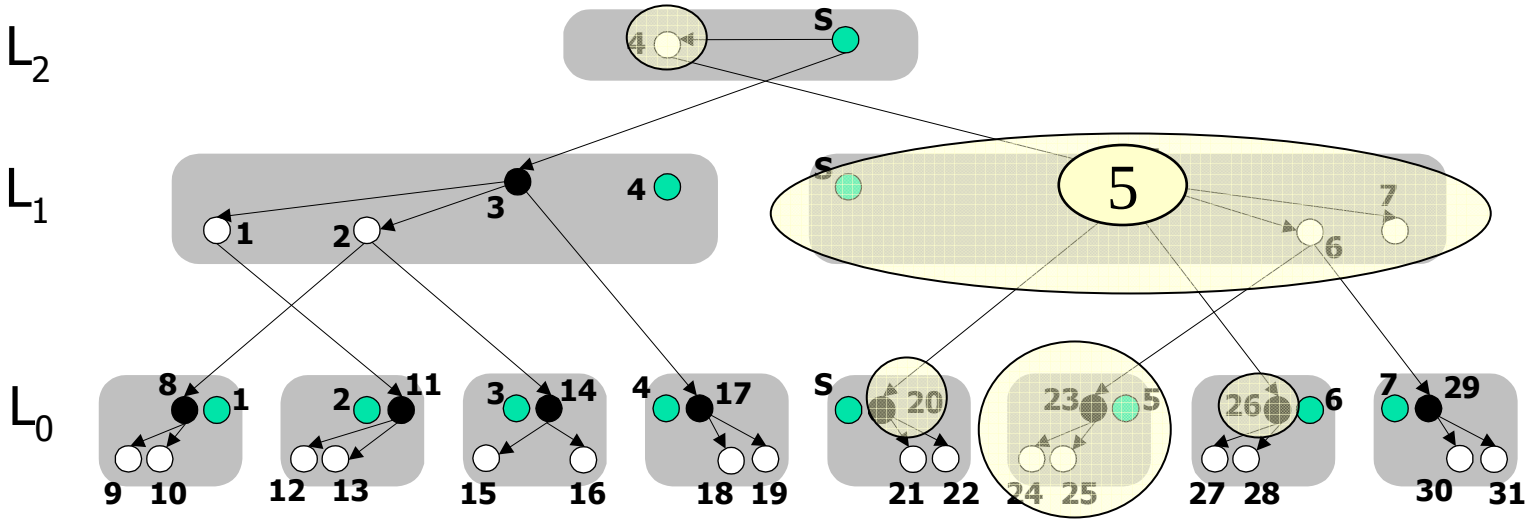
Zigzag Tree



Tree height is at most $2\log_k N+1$

Node degree is at most $6k-3$

Control Protocol



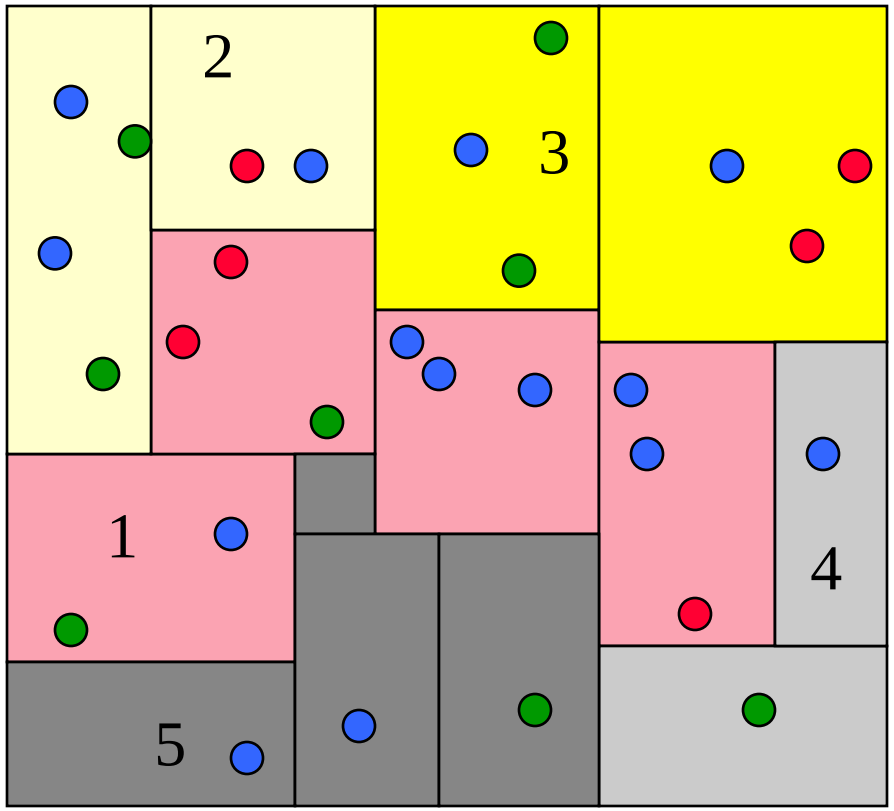
- A node exchanges soft-state information with its clustermates, parent, and children
- ➔ amortized worst-case: O(k)

Zigzag Hierarchy: Properties

- Max routing distance between any two nodes in the network is $4\log_k N+1$
- An average node keeps track of $O(k)$ other nodes
- Failure recovery requires $O(k)$ reconnections
- Cluster split/merger requires $O(k)$ reconnections

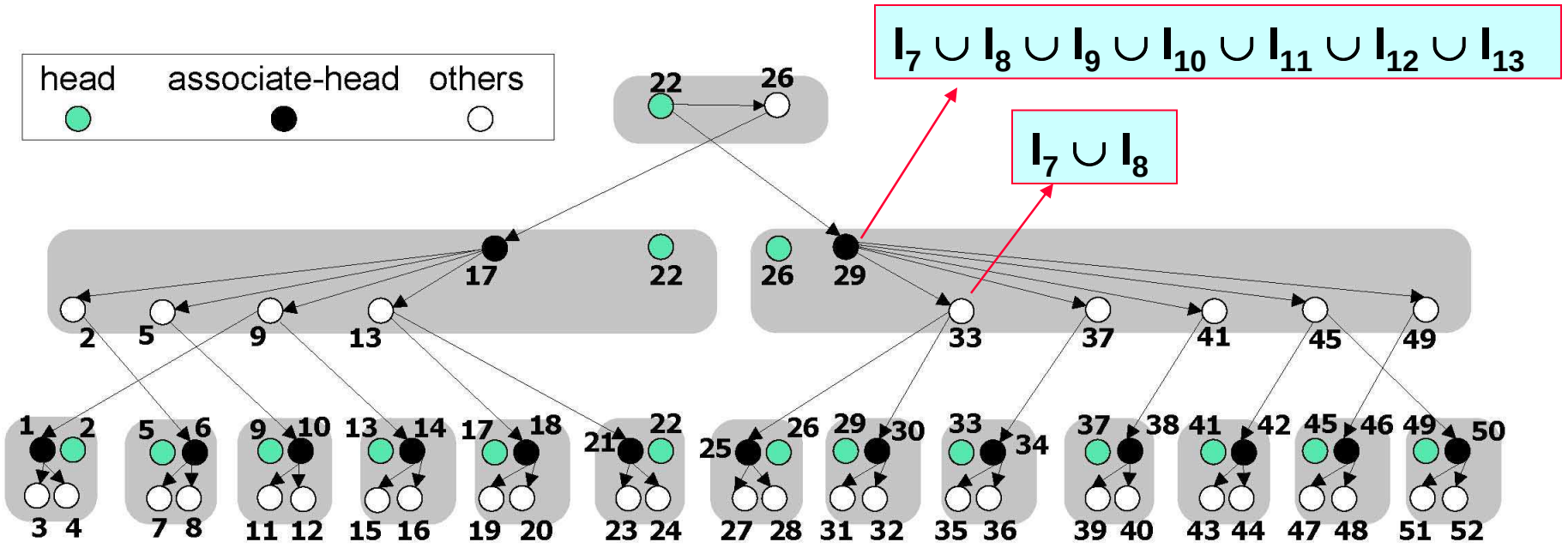
Zigzag is **excellent** for overlay-networking nodes in large-scale decentralized networks

Index Zones Assignments



- Each cluster C manages an index zone
  - A contiguous set of hyperrectangles
- Zone info kept at vice-head
- Union of all zones at layer-0 = $[0, 1)^d$

Zone Assignment: Example

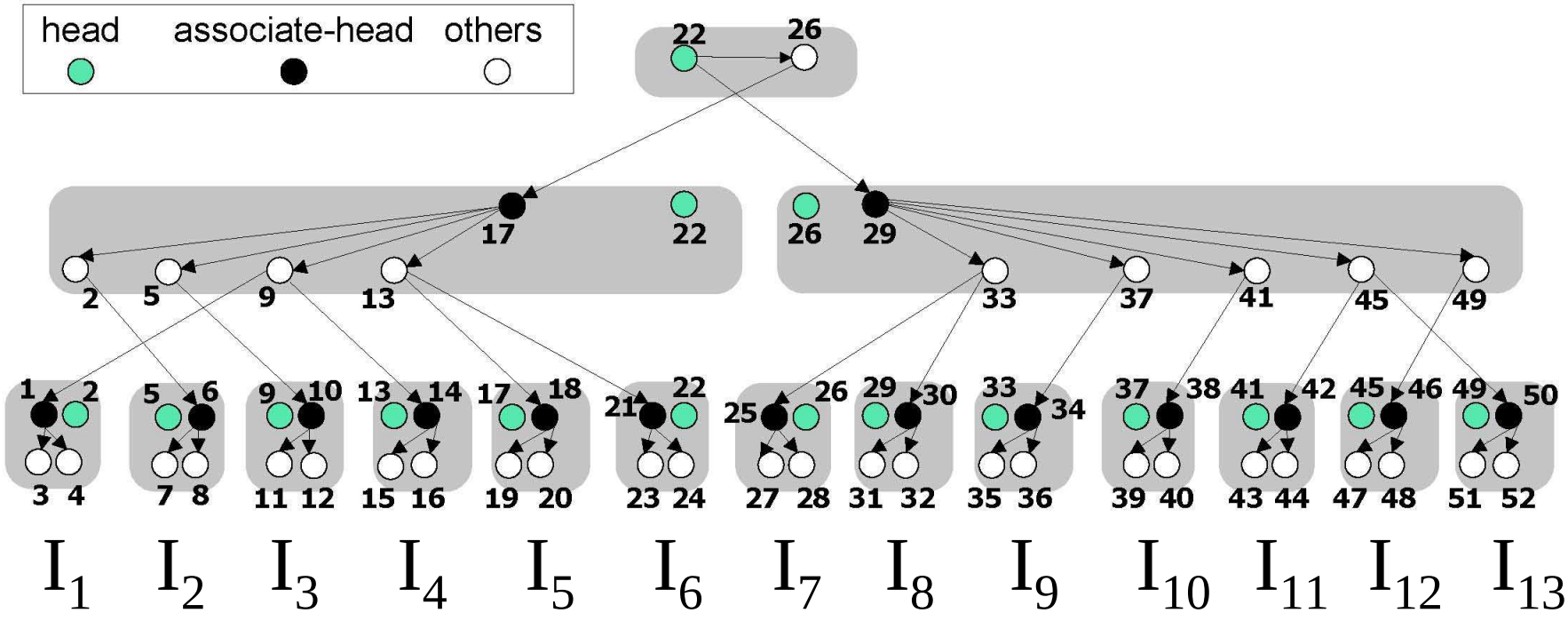


I_1 I_2 I_3 I_4 I_5 I_6 I_7 I_8 I_9 I_{10} I_{11} I_{12} I_{13}

$$I_1 \cup I_2 \cup \ldots \cup I_{13} = [0,1)^d$$

Parent zone = union of child zones

Where to Store Indices?



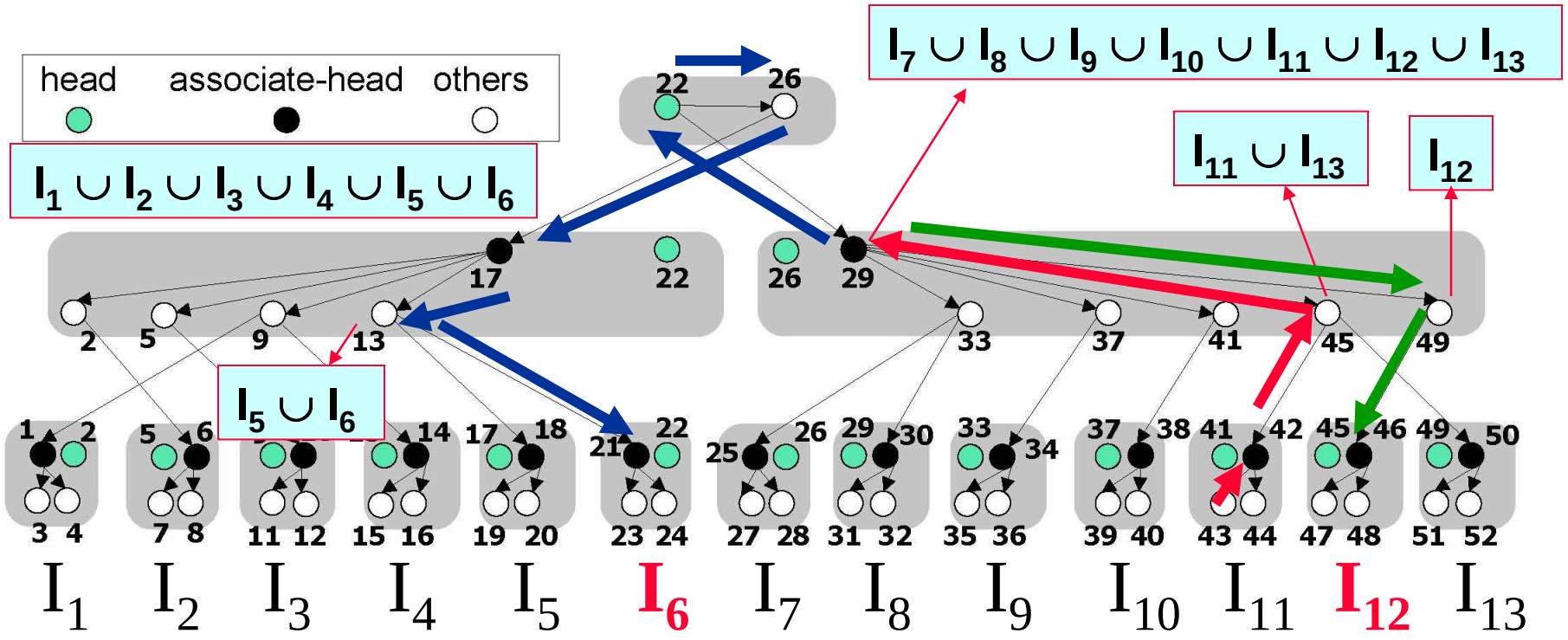
Node 7:
$x \in I_{13}$

Head 49: store index of x

Search Algorithms

- K-NN query (Q, K)
  - $Q = (w_{1Q}, w_{2Q}, \ldots, w_{dQ})$
  - K = number of best objects returned
- Exact query ~ 1-NN query
- Range query Q
  - Simple: $Q = [min_{1Q}, max_{1Q}] \times \ldots \times [min_{dQ}, max_{dQ}] \subset [0, 1)^d$
  - Composite: $Q = Q_1 \cup Q_2 \cup \ldots \cup Q_n \subset [0, 1)^d$

Range-Query Search



Longest path (9): 43, 42, 45, 29, 22, 26, 17, 13, 21. In general: $4\log_k N+1$
Nodes visited (10): 42, 45, 29, 49, 46, 22, 26, 17, 13, 21

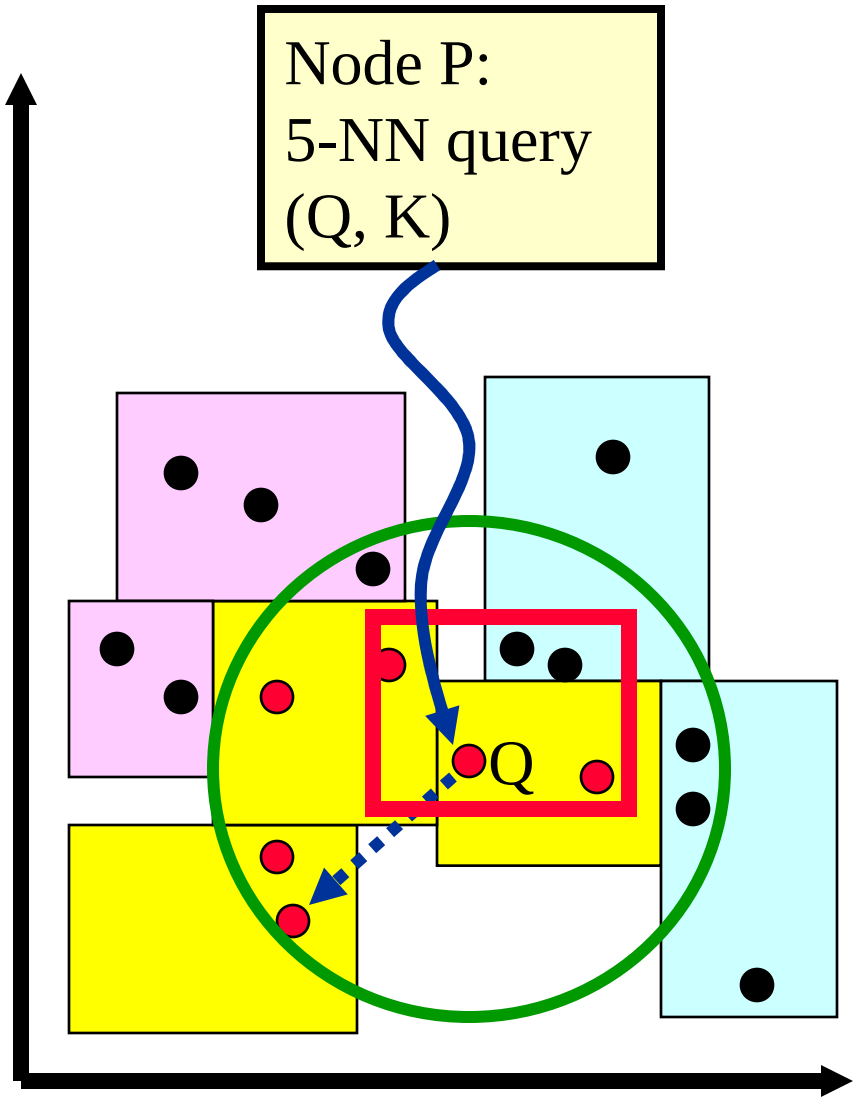
Node 43:
$Q \subset I_6 \cup I_{12}$

K-NN Search

- $\varepsilon = \max\{simdist(x, Q) | x \in zone(C)\}$

- $Q' = [w_1 - \varepsilon, w_1 + \varepsilon] \ldots [w_d - \varepsilon, w_d + \varepsilon]$
- Only accept objects x: $simdist(x, Q) < \varepsilon$

- Return best K objects among the accepted



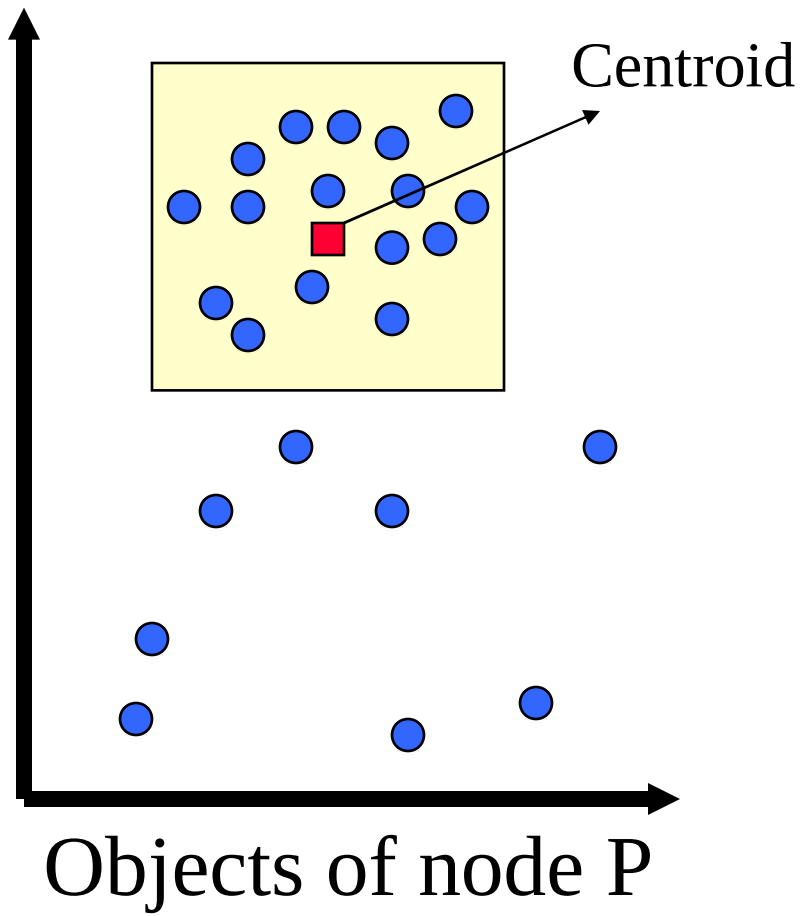
Object Publication

- Node P has a set of objects to **publish** $\{X_1, X_2, \ldots, X_m\}$
  - Create a "point" range query $Q_{pub} = \{X_1\} \cup \{X_2\} \cup \ldots \cup \{X_m\}$
- Apply search algorithm to find layer-0 clusters C_i : Zone(C_i) contains X_i
  - Head(C_i) stores index for X_i
- Analysis (worst-case)
  - Time: $4\log_k N+1$
  - Overhead: $h(4\log_k N+1)$
    - h = no. clusters C_i reached, h << m

Object Removal

- Same as object publication
- Instead of storing object, delete its index from the corresponding cluster head node

Node Join



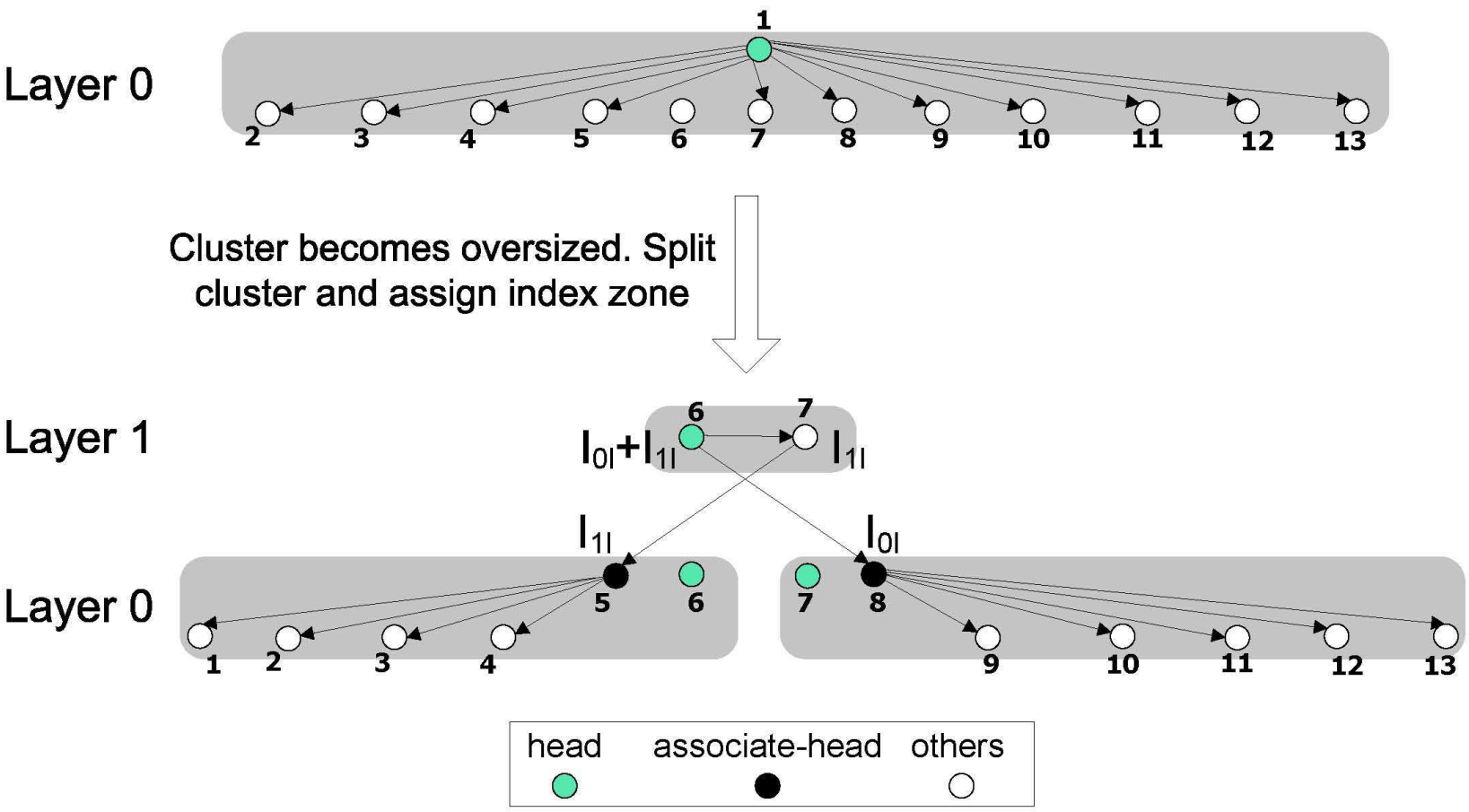
- ➔ low indexing overhead
- ➔ strong locality

- How?
  - P knows an existing node
  - Find object area of highest density
  - Compute centroid point $P_{centroid}$
  - Add P to layer-0 cluster C: $P_{centroid} \in$ zone(C)

Node Departure

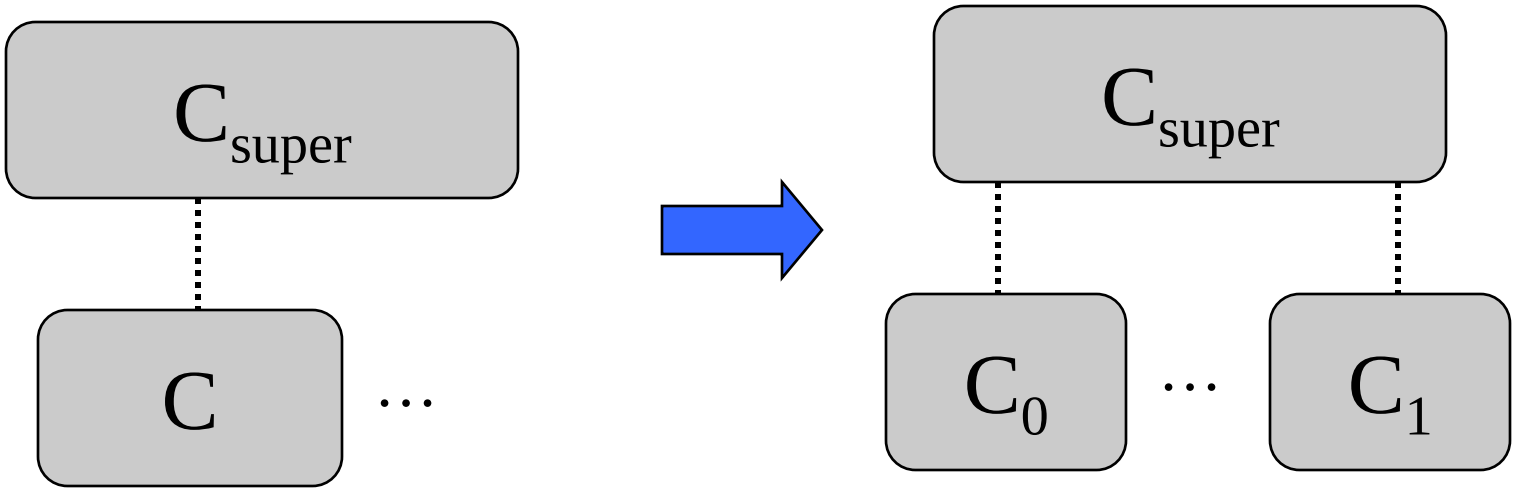
- P: failed node
- If P stores some indices ➔ indices are lost
  - Solution: Vice-head of P at layer-0 detects this failure and broadcasts "P not exist" along Zigzag tree. A node republishes objects if necessary
- If P has objects indexed at other nodes ➔ indices are invalid
  - Solution: Nodes that store indices for P will delete these indices when a query for objects of P arrives and contact with P fails

Hierarchy Construction



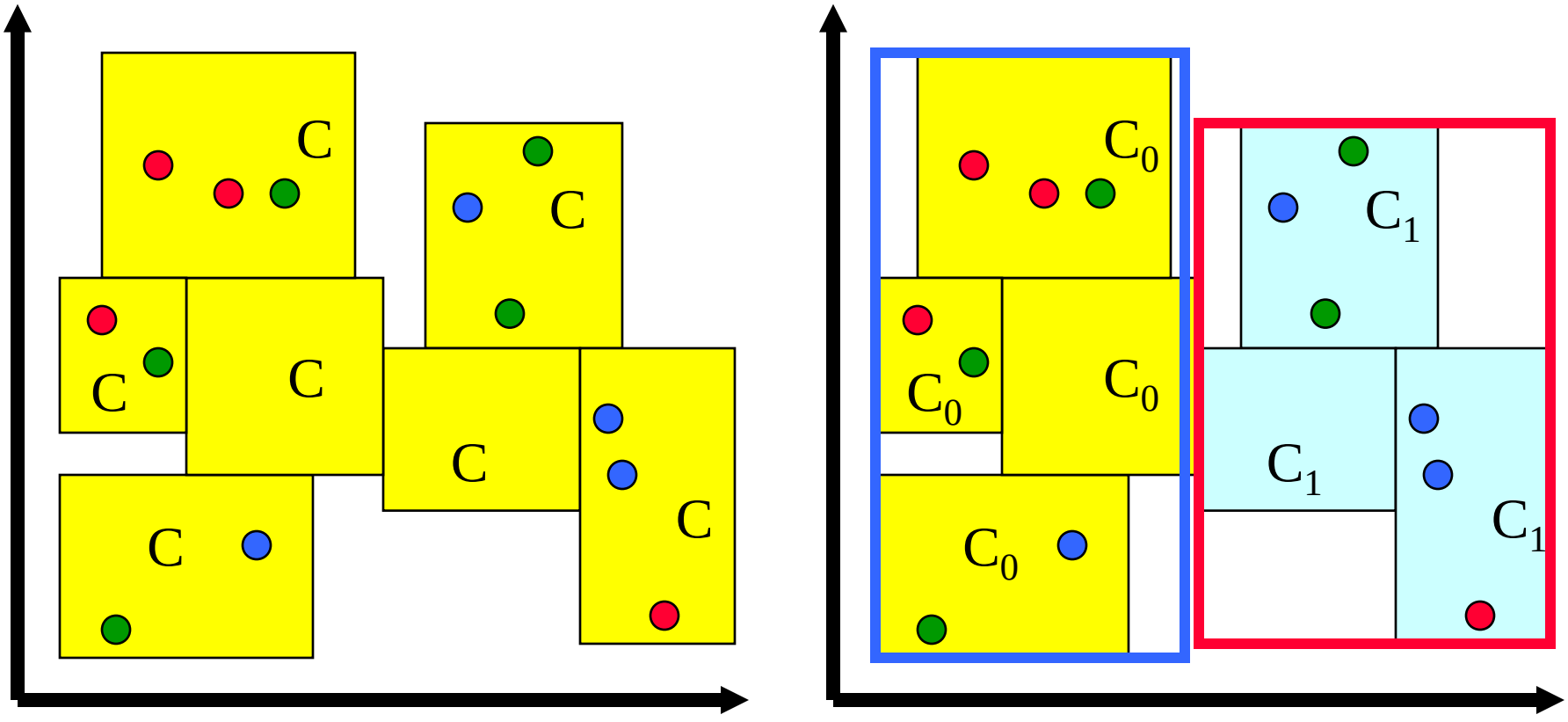
Cluster Split

- Cluster C is oversized ➔ need to split



zone(C_0) is contiguous, zone(C_1) is contiguous
$|C_0| \in [k, 3k]$, $|C_1| \in [k, 3k]$
Box(C_0) ∩ Box(C_1) minimal

Cluster Split



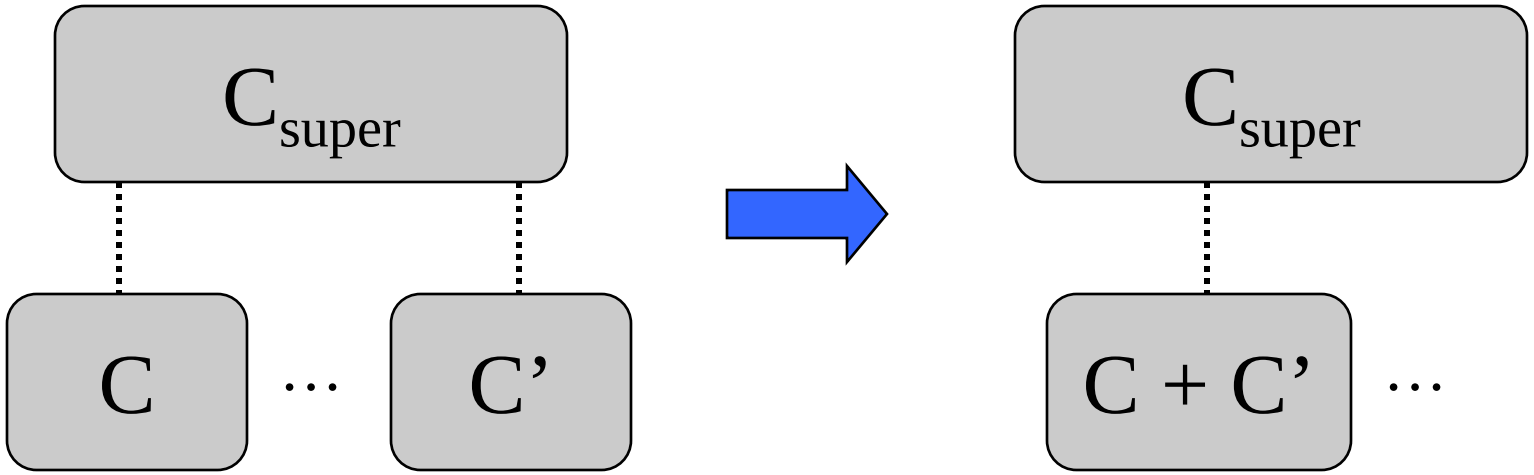
Before split

After split

Maximize **locality** of index zone

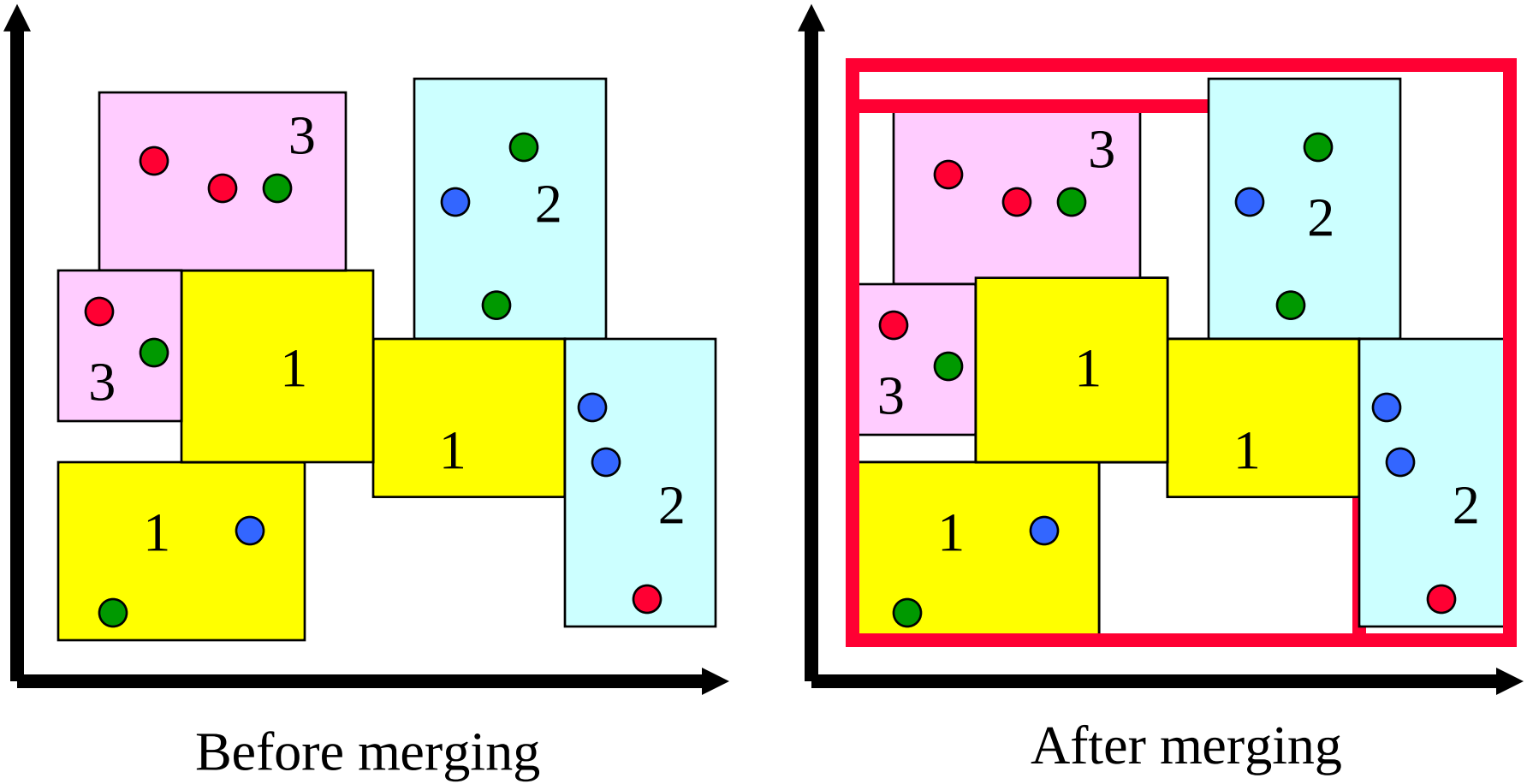
Cluster Mergence

- Cluster C is undersized ➔ need to merge



zone(C) ∪ zone(C') is contiguous
|C+C'| ∈ [k, 3k]
Box(C+C')-zone(C)-zone(C') minimal

Cluster Mergence



Before merging

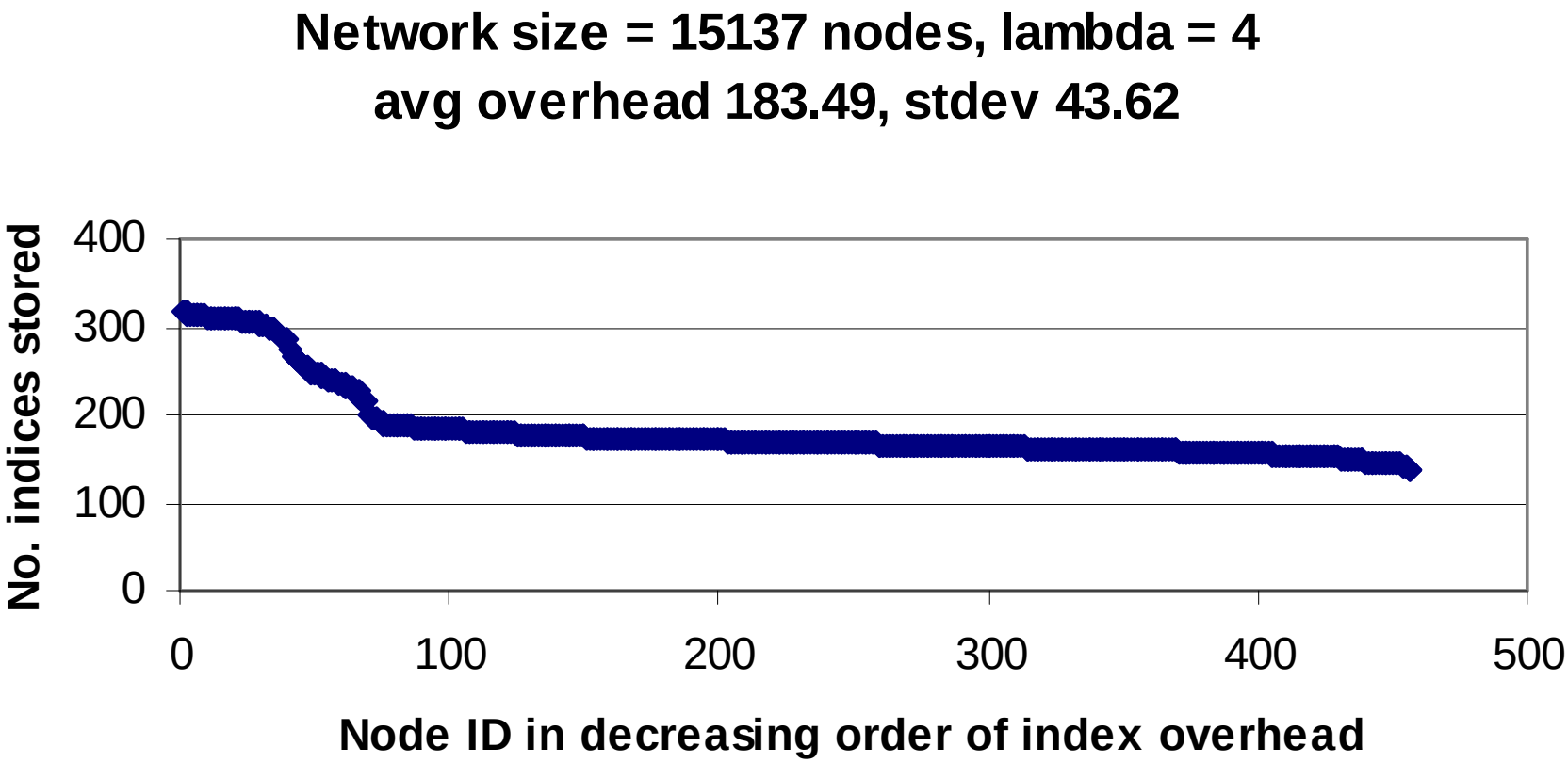
After merging

Maximize **locality** of index zone

Performance Evaluation

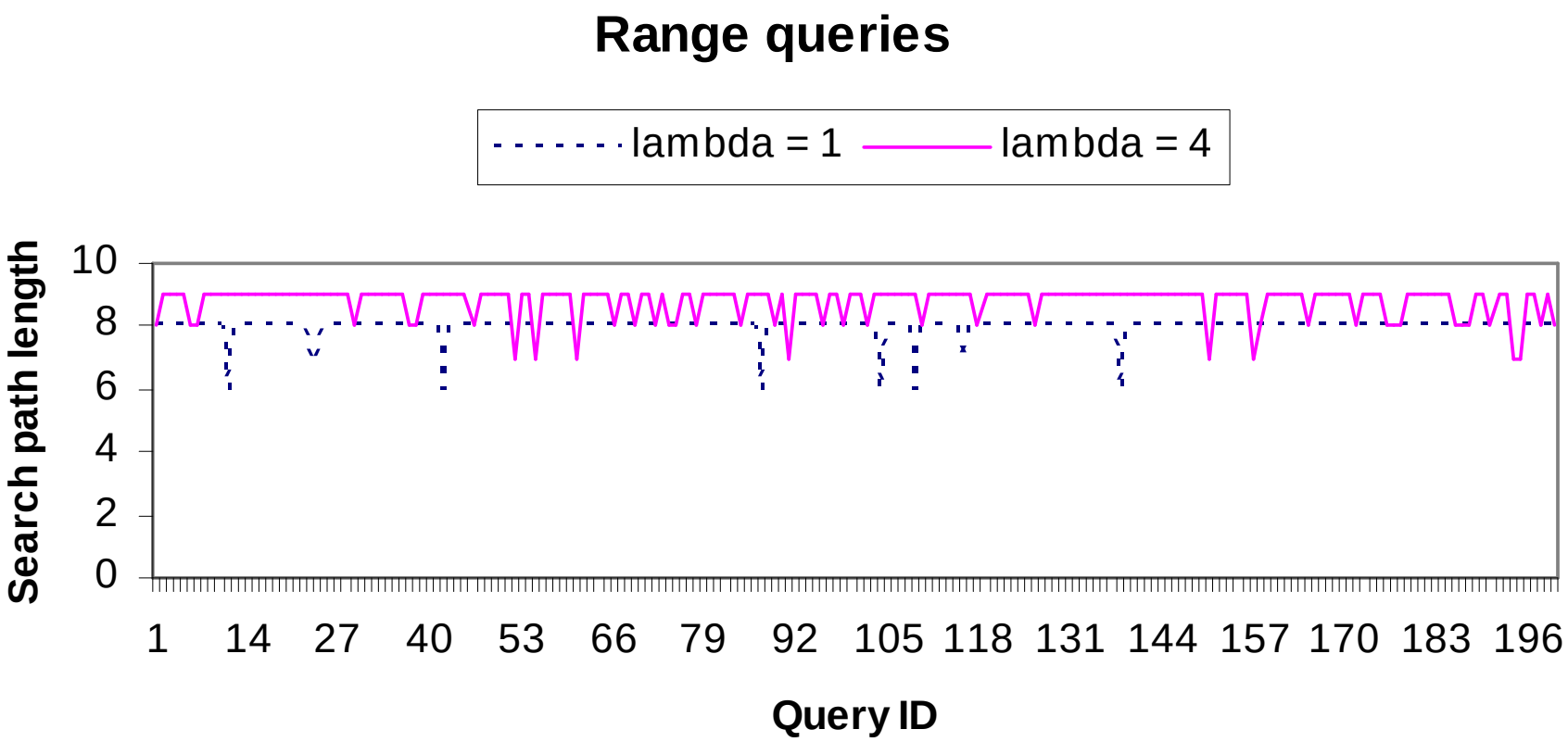
- Simulation period 5000 sec
- Nodes
  - Arrival: Poisson (default λ=2 node/sec)
  - Departure: anytime in random during simulation run
  - Random number of objects (at most 10)
- Objects
  - Uniform random points in $[0, 1)^d$
- Queries
  - 200 range: {0.1, 0.2, …, 0.9} with Zipf factor z = 0.8 (most queries are selective)
  - 200 K-NN: {10-NN, 20-NN, …, 40-NN} with Zipf factor z = 0.8 (most queries are selective)

Index Storage Overhead



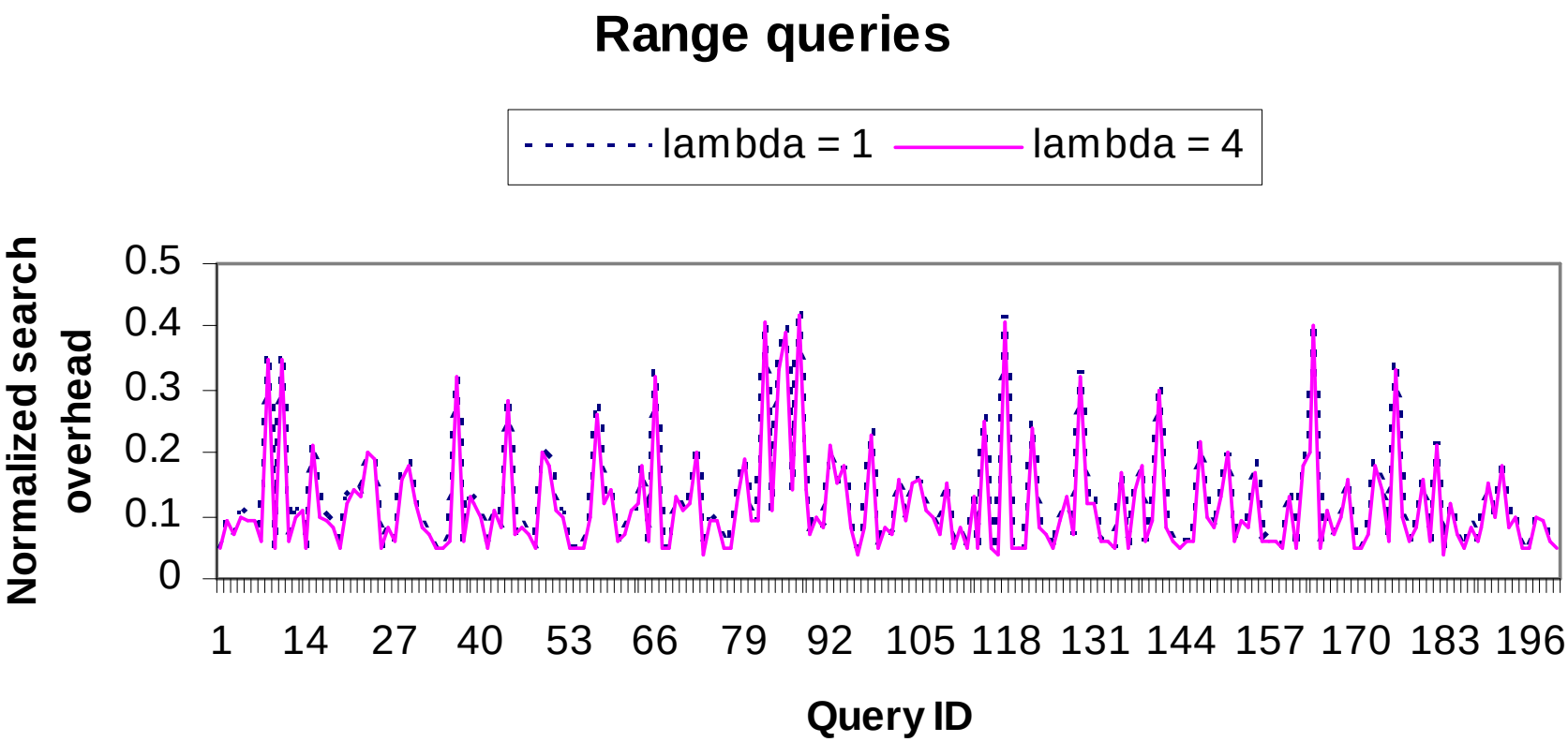
0.3% of data is indexed at such a node
Very fair distribution

Range Queries: Network Size on Search Time



Lambda=1: 3745 nodes, lambda=4: 15137 nodes
Very fast: visit no more than 9 nodes
Scalable with network and data size

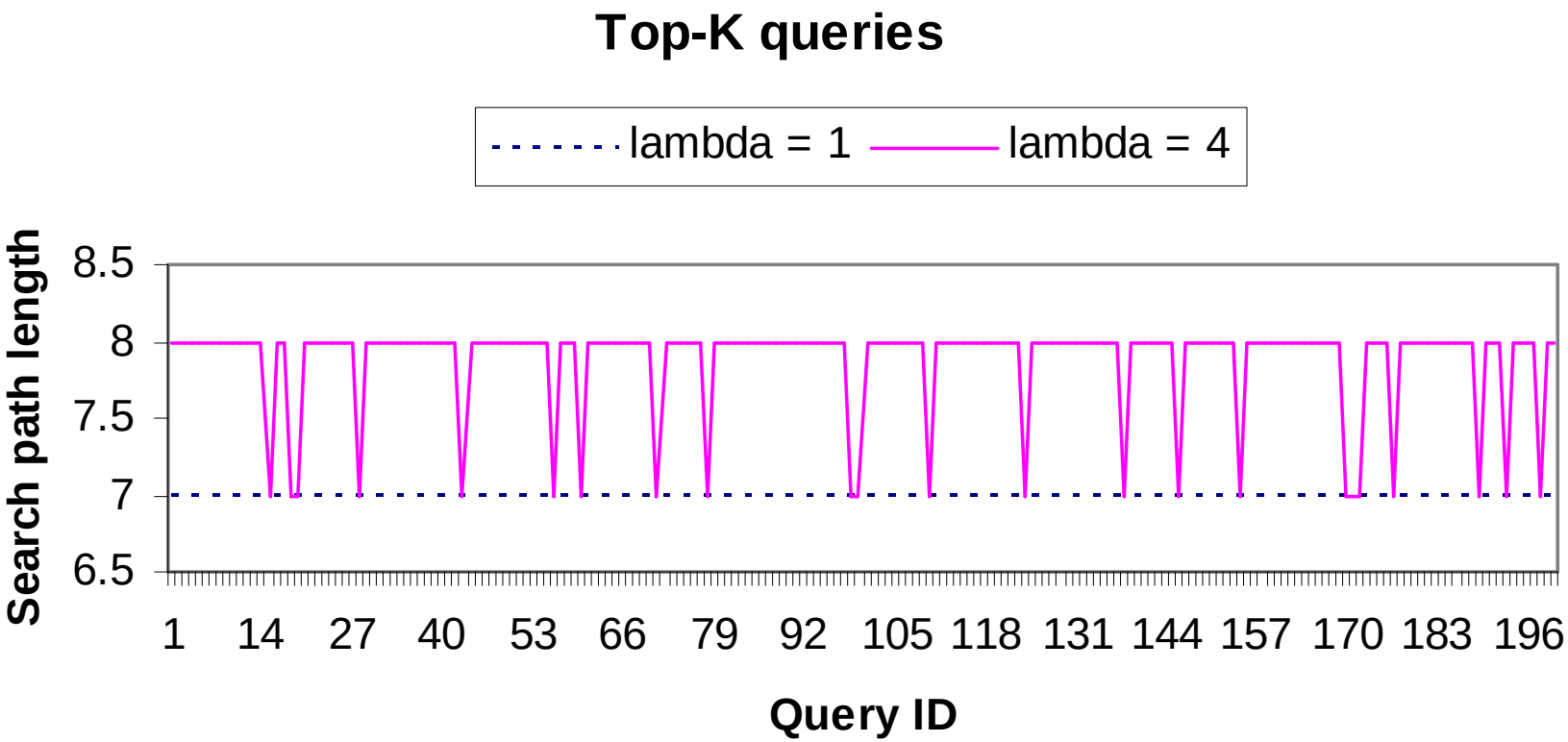
Range Queries: Network Size on Search Overhead



Lambda=1: 3745 nodes, lambda=4: 15137 nodes

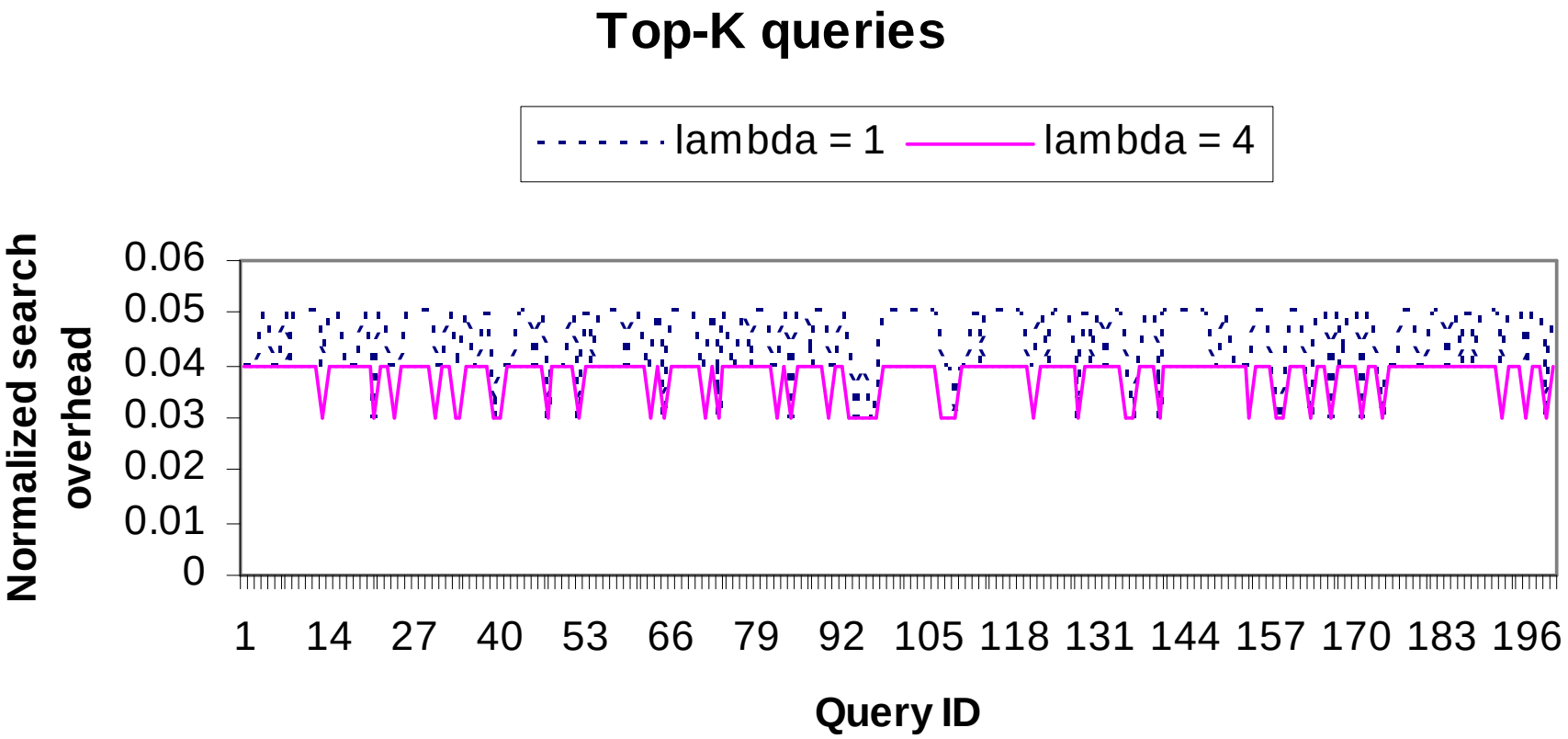
Very efficient and scalable

K-NN: Network Size on Search Time



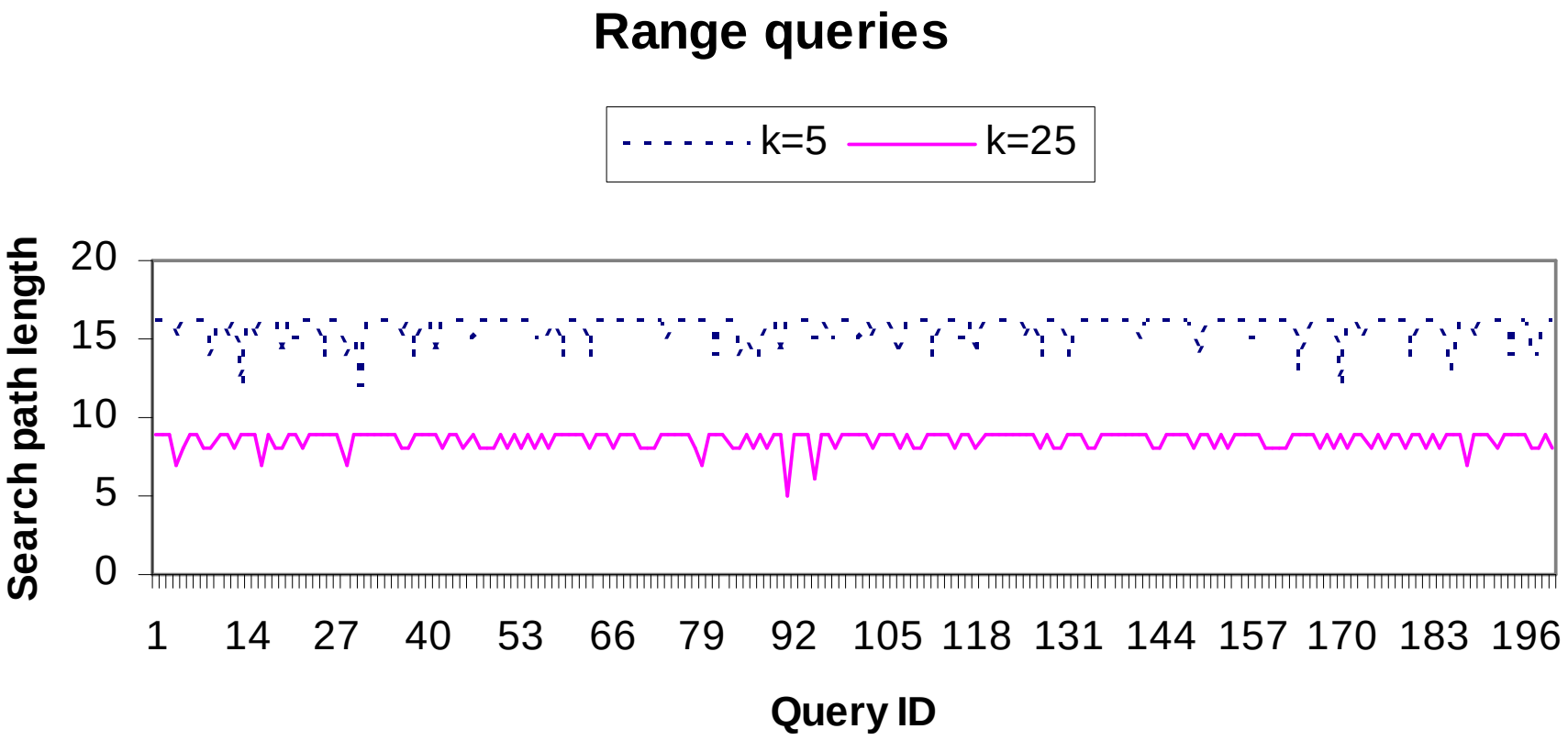
Very fast
Quadrupling size affects delay only slightly

K-NN: Network Size on Search Overhead



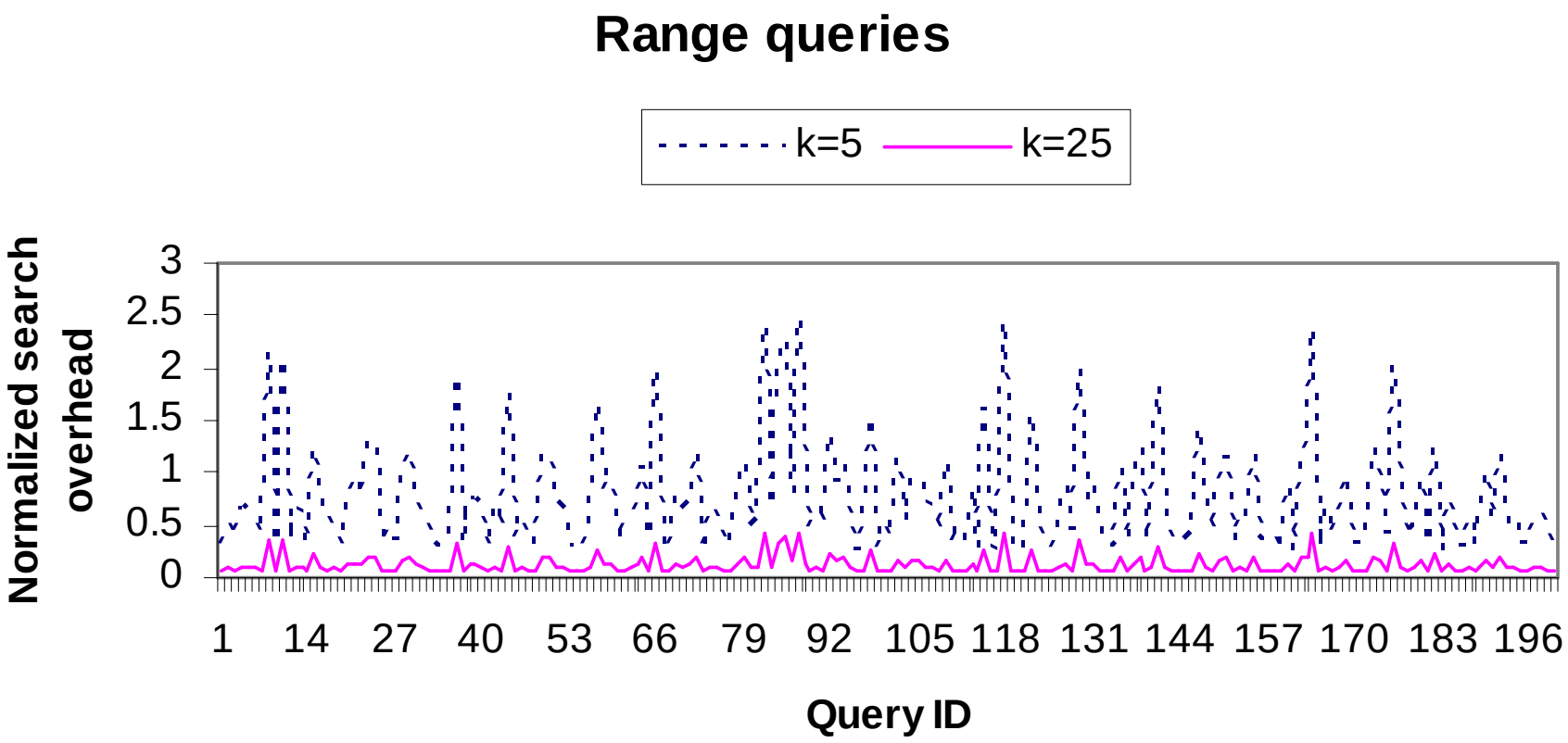
Quadrupling network size reduces search overhead

Range Queries: Cluster Size on Search Time (~7500 nodes)



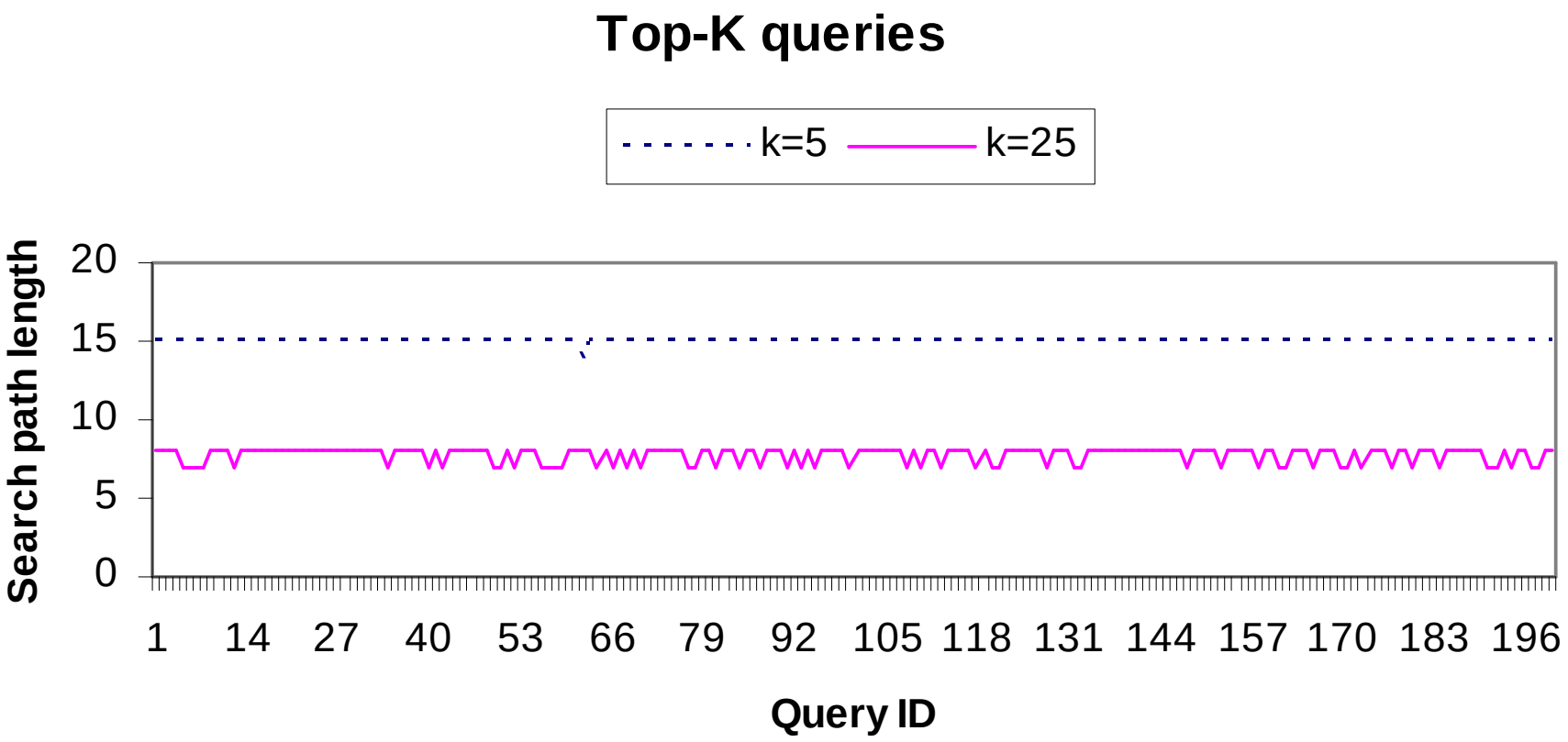
A larger k is recommended

Range Queries: Cluster Size on Search Overhead (~7500 nodes)



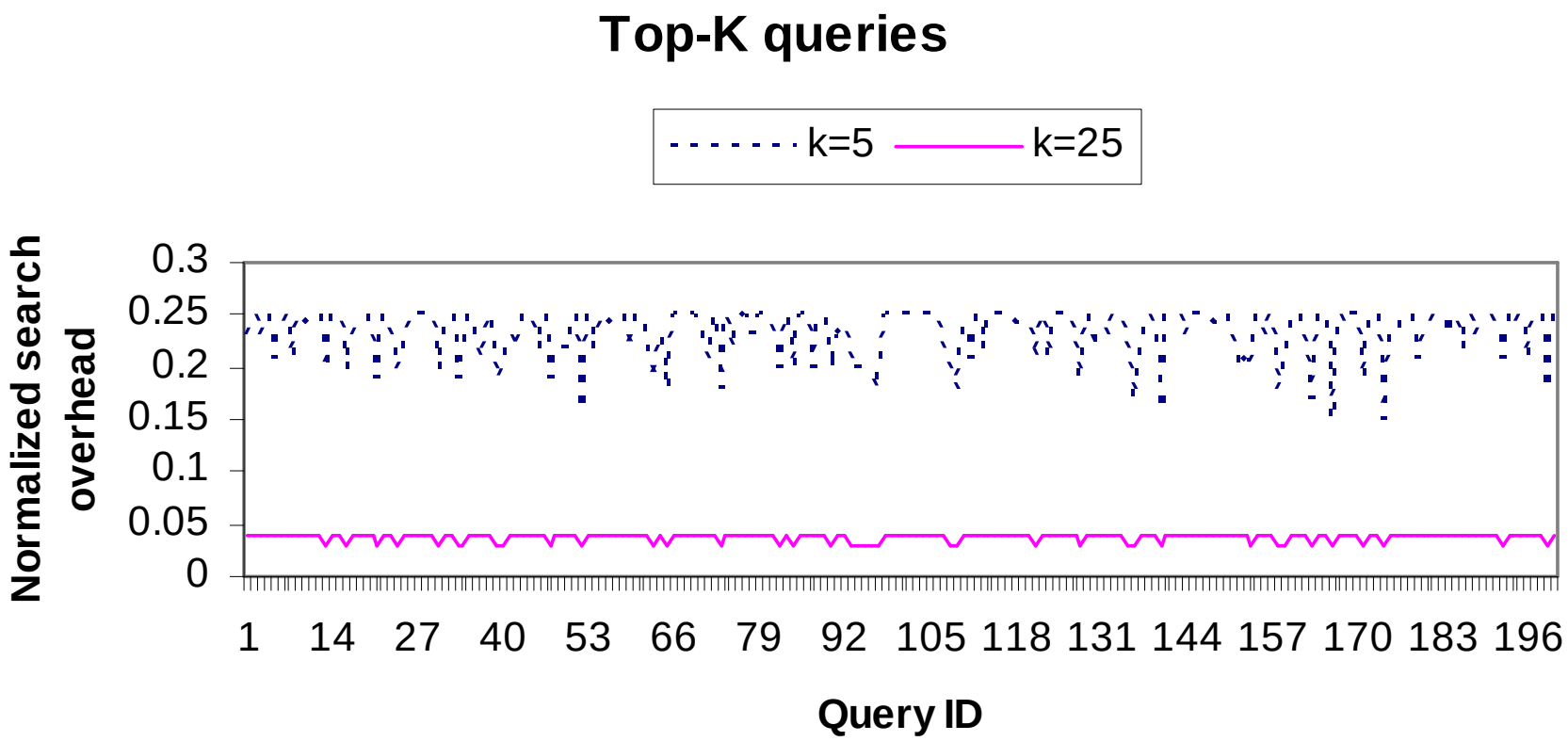
A larger k is recommended

K-NN: Cluster Size on Search Time



A larger k is recommended

K-NN: Cluster Size on Search Overhead

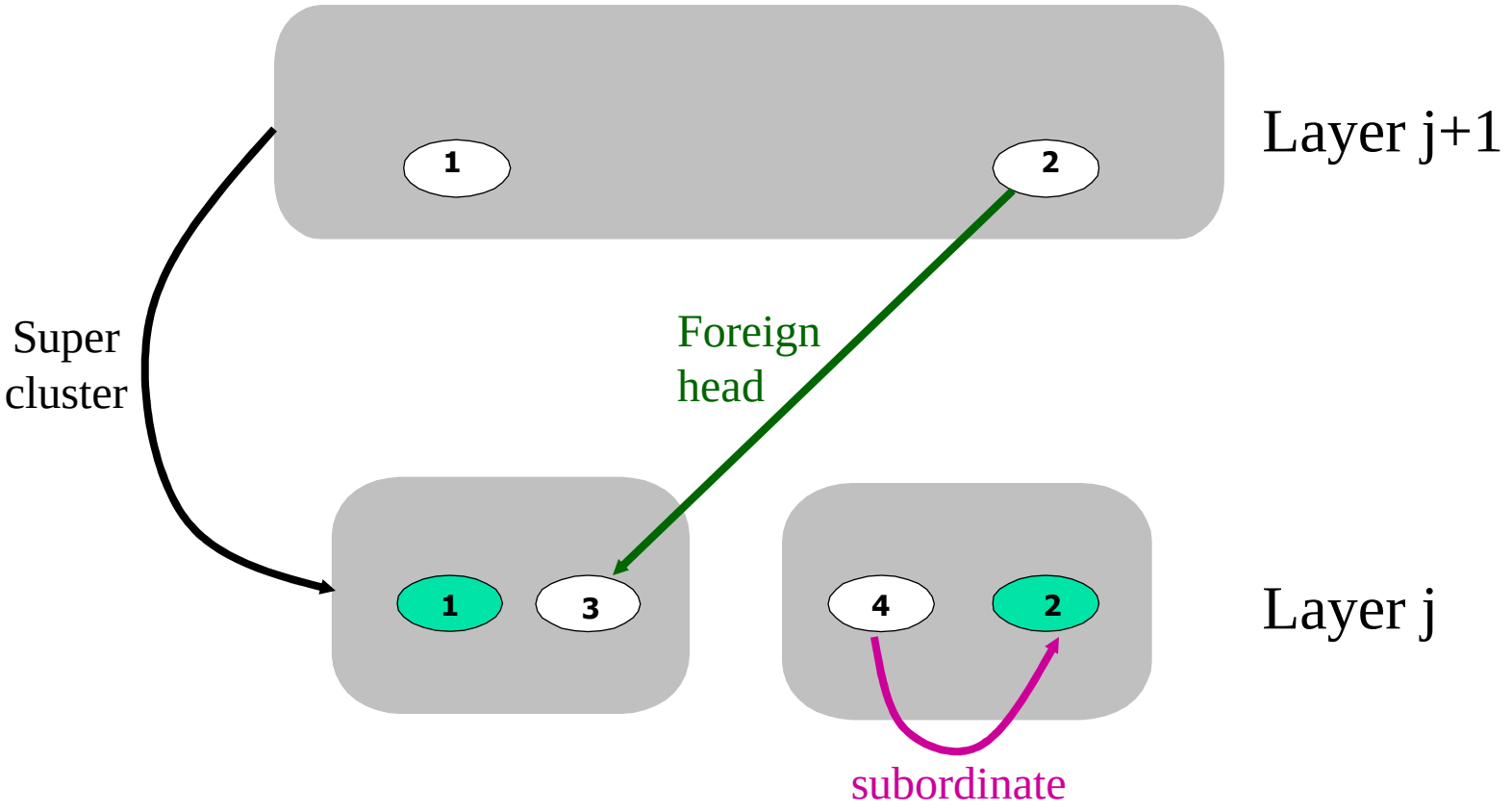


A larger k is recommended

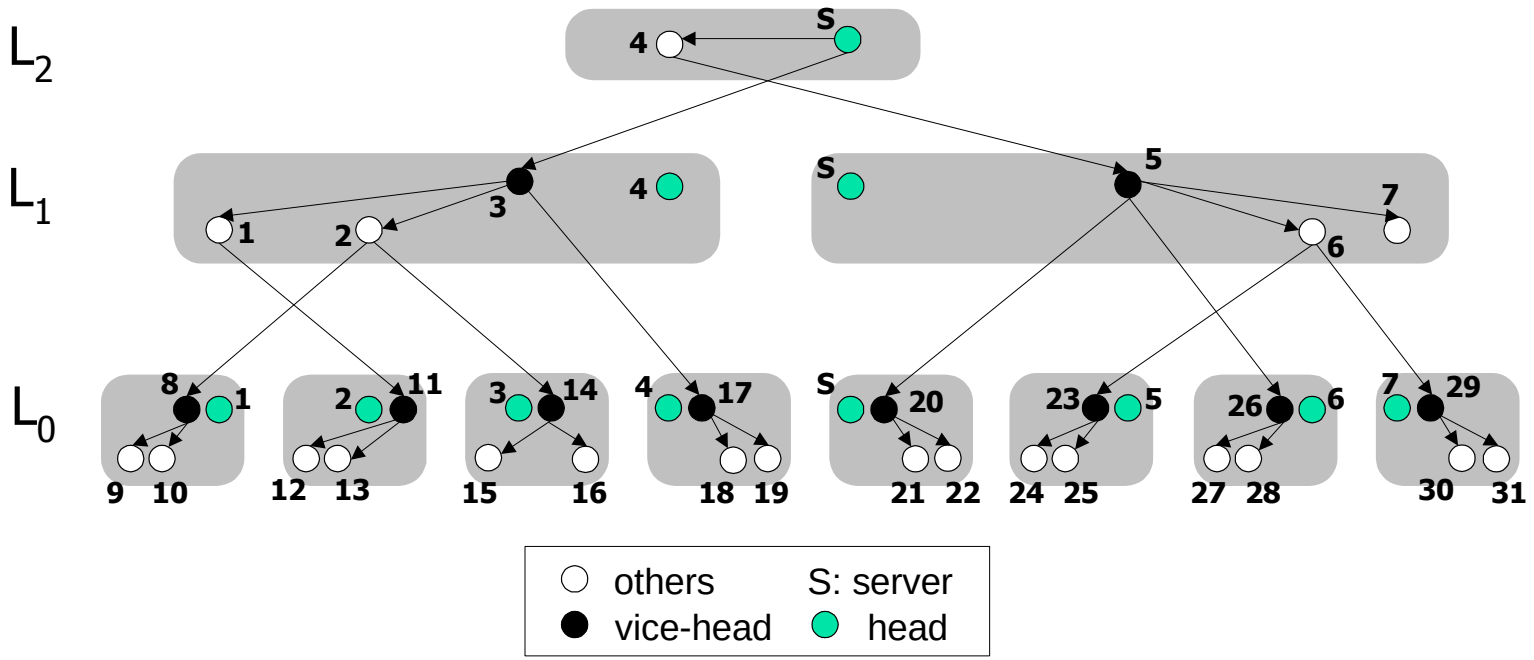
Summary

- Information retrieval: very important
- Decentralized networks: rapidly growing in size, popularity, and importance
- IR in DCNs is very challenging
- EZSearch: First solution that can handle exact, K-NN, and range queries
  - Fast: logarithmic worst-case search time
  - Accurate: all requested documents are found precisely
  - Robust: Failure recovery requires constant overhead
  - Load balancing: indices fairly distributed
  - Scalable: Performance and overheads are slightly affected by increases of network size

Terms


Layer j+1
1
2
Super
cluster
Foreign
head
1
3
4
2
Layer j
subordinate

Connectivity Rules

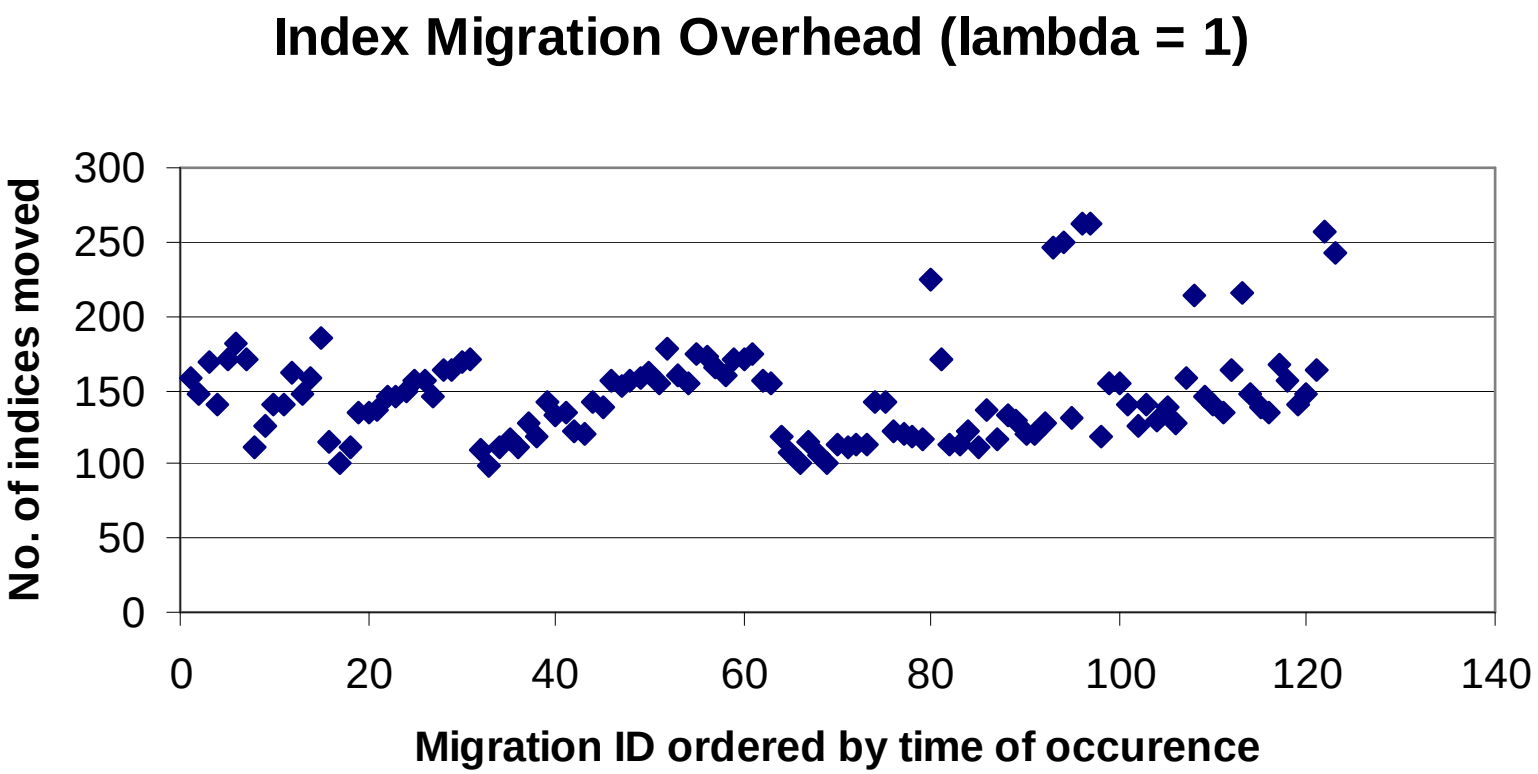


(In this example, k = 4)

Content Description

- Each node has a set of objects to be shared by the entire network
- Object $x = (w_{1x}, w_{2x}, \ldots, w_{dx}) \in [0, 1)^d$
  - w_{ix}: weight to reflect the significance of term (keyword, concept) t_i in object x
  - d: number of terms used to represent objects
- Simdist(x, y): semantic similarity between objects x and y
  - e.g., cosine distance function

Index Migration Overhead: 3745 Nodes



Index Migration Overhead: 15137 Nodes