SCOPE: Scalable Consistency Maintenance in Structured P2P Systems

IEEE INFOCOM 2005

Presented by 王靜嵐

2006/05/04

Outline

- Introduction
- Resolving Problems
- The SCOPE Protocol
- Performance Evaluation
- Conclusion

Introduction

- Newly-developed applications demand that P2P systems be able to manage dynamically-changing files.
- Some structured P2P-based applications, replication and caching have been widely used to improve scalability and performance.
- Maintaining consistency between frequently-updated files and their replicas is a fundamental reliability requirement for a P2P system.

Introduction

- SCOPE
  - A structured P2P system with replica consistency support, called Scalable COnsistency maintenance in structured PEer-to-peer systems.
  - Unlike existing structured P2P systems, SCOPE distributes all replicas' location information to a large number of nodes, thus preventing hot-spot and node-failure problems.

Resolving Problems

- Hot-spot problem
  - Due to the different objects' popularities, making the popular nodes heavily loaded while other nodes carry much less replicas.
- Node-failure problem
  - If the hashed node fails, update notifications have to be propagated by broadcasting.

The SCOPE Protocol

- The SCOPE protocol specifies:
  - Record the locations of all replicas
  - Propagate update notifications to related peers
  - A peer joins or leaves the system
  - Recover from a node's failure.

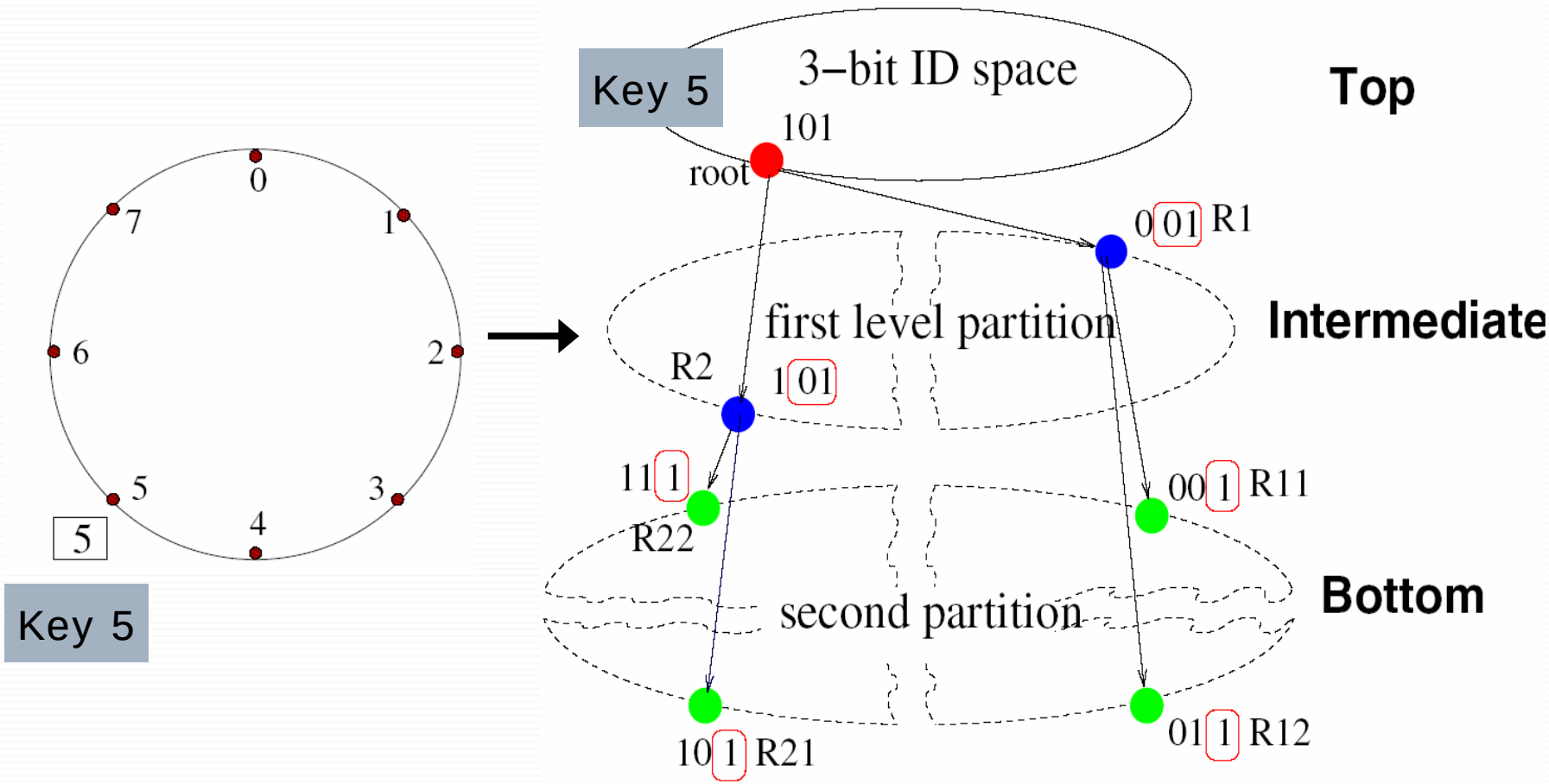
Record the locations of all replicas

- Partitioning Identifier Space
- Building Replica-Partition-Trees (RPTs)
  - Basic structure
  - Scalable RPT

Partitioning Identifier Space

- A consistent hash function (e.g. SHA-1) assigns each node and each key an *m*-bit identifier.
- The key identifier can be easily calculated by keeping a certain number of least significant bits.
- A partition can be further divided into smaller ones, and it records the existence of all keys in its sub-partitions.

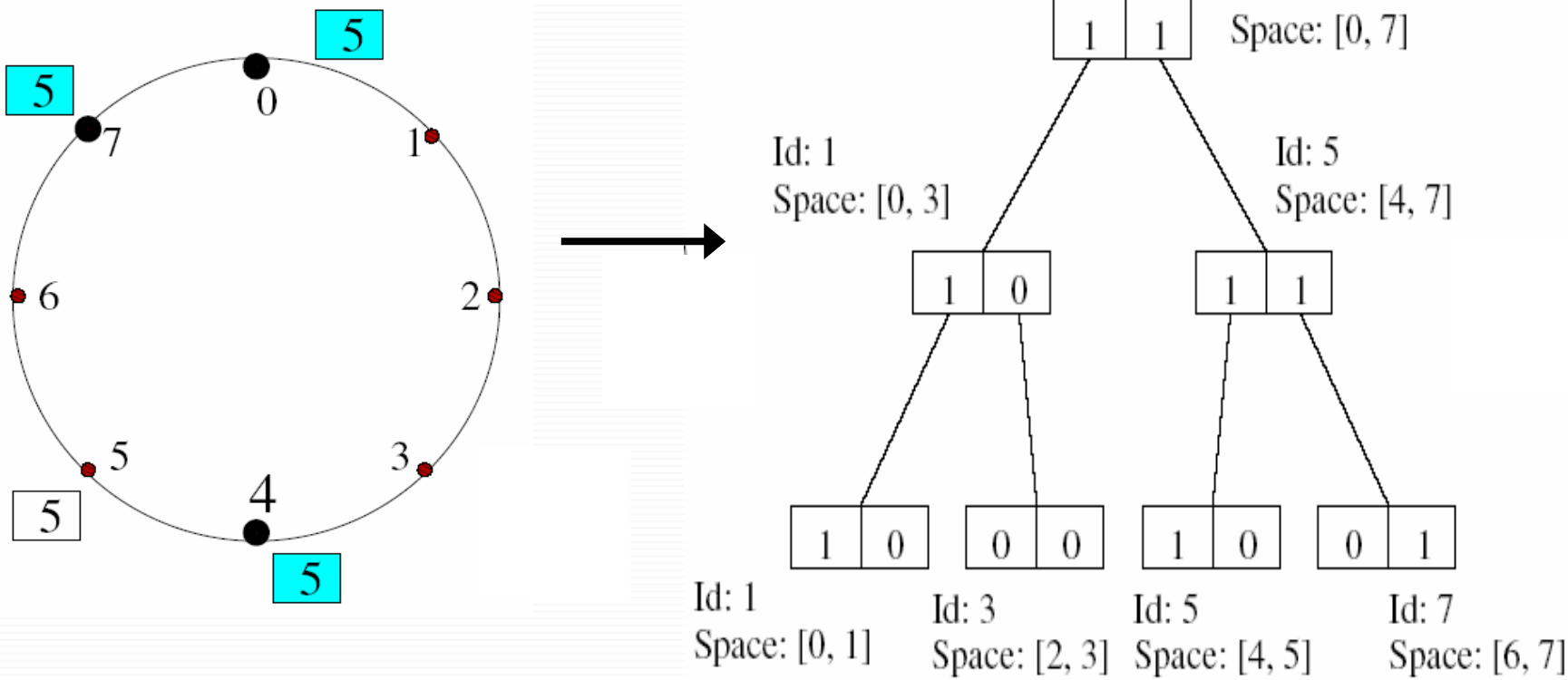
An identifier space with $m = 3$



Building Replica-Partition-Trees (RPTs)

- Building an RPT for each key by recursively checking the existence of replicas in the partitions.
- The primary node of a key in the original identifier space is the root of RPT(s).
- The representative node of a key in each partition, recording the locations of replicas at the lower levels, becomes one intermediate node of RPT(s).

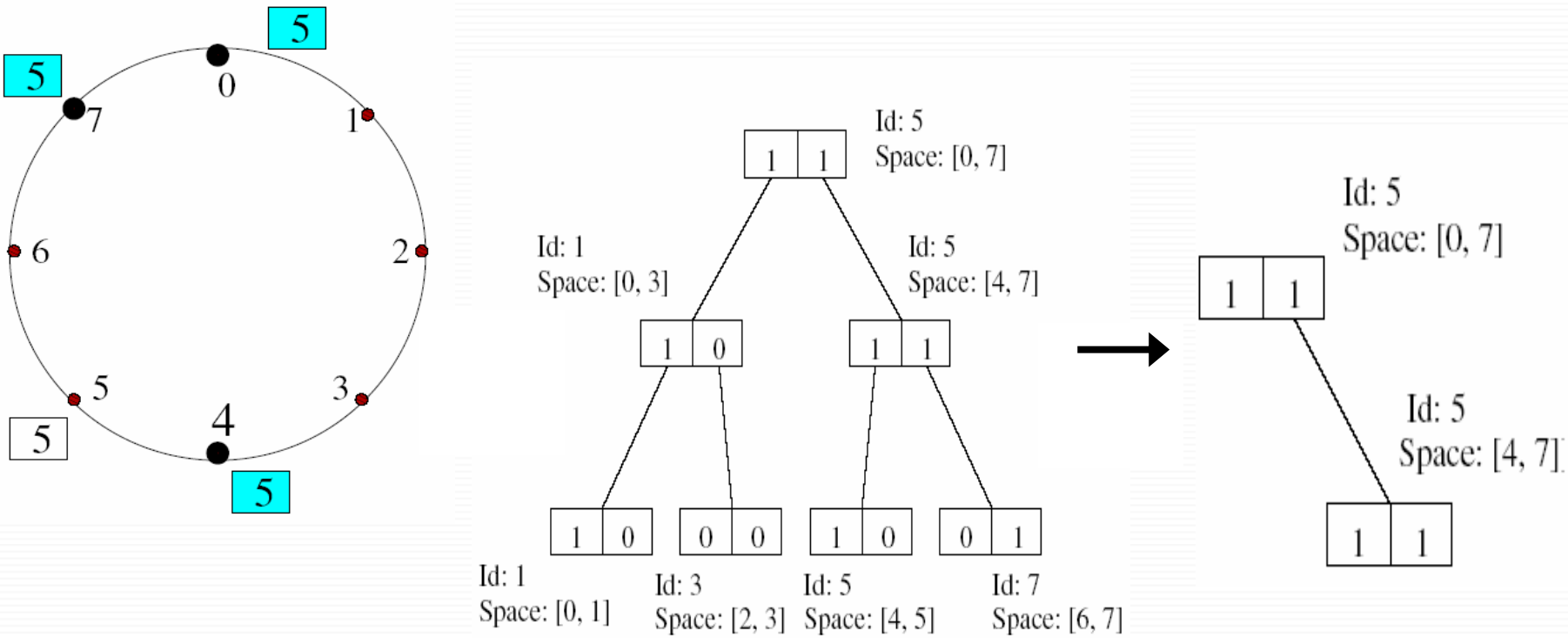
The RPT of key 5



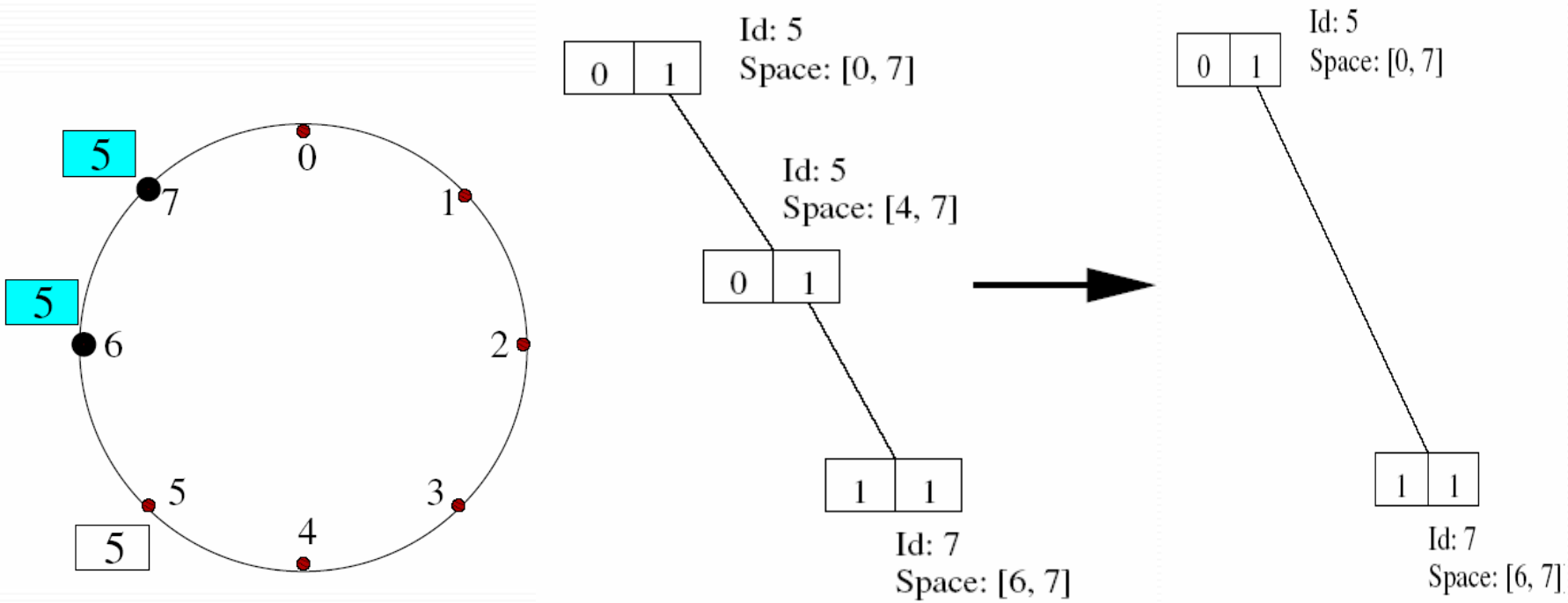
Scalable RPT

- Our goal is to reduce the heights of RPTs.
- By removing the redundant leaf nodes, we can build a much shorter RPT.
- If all nodes in a partition are clustered in one of its lower-level partitions, it is possible to reduce the intermediate nodes.

After removing redundant leaf nodes



After removing both redundant leaf nodes and intermediate nodes



Operation Algorithm

- New Operations
- Subscribe/Unsubscribe Operations
- Update notifications

New operations

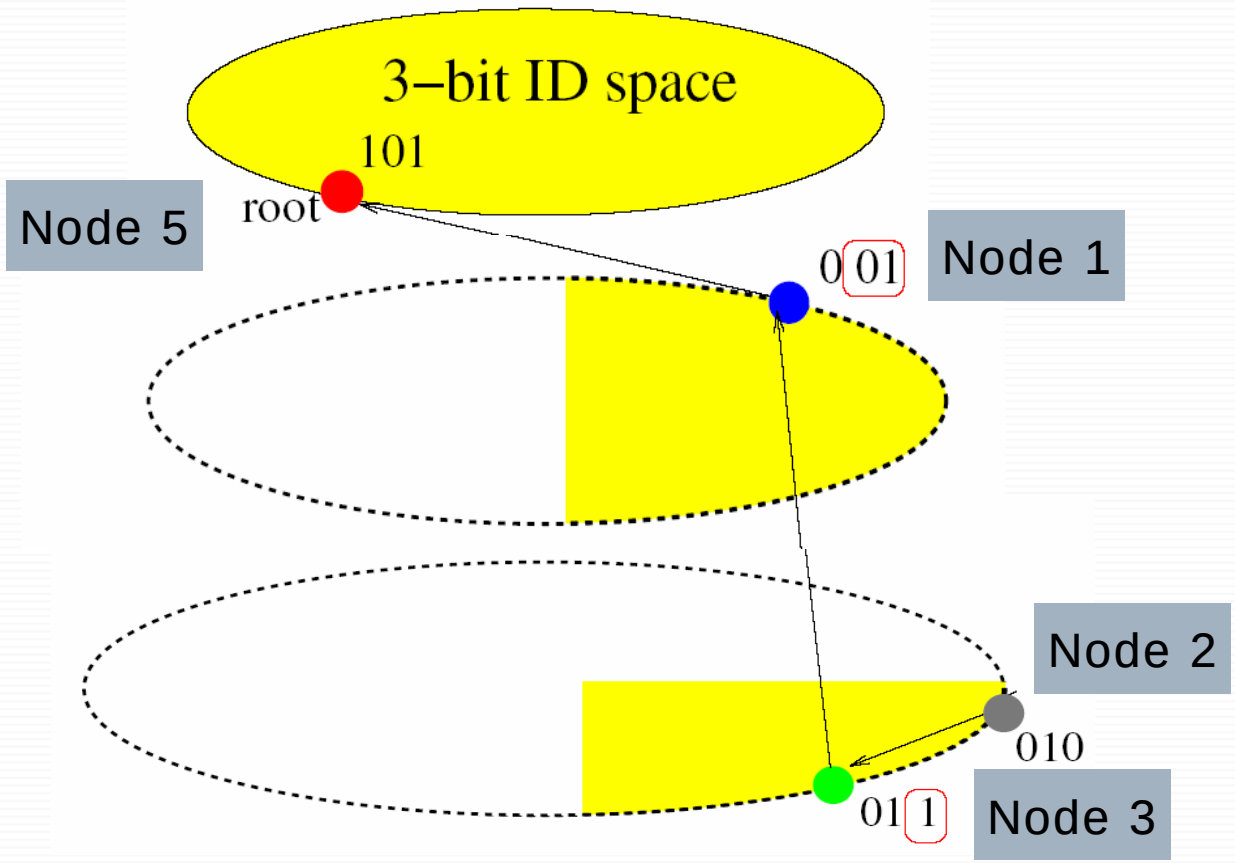
- Subscribe
  - when a node has an object and needs to keep it up-to-date, it calls *subscribe* to receive a notification of the object update.
- Unsubscribe
  - when a node neither needs a replica nor keeps it up-to-date, it calls *unsubscribe* to stop receiving update notifications.
- Update
  - when a node needs to change the content of an object, it calls *update* to propagate the update notification to all subscribed nodes.

Subscribe/Unsubscribe Operation

- In SCOPE, we implement the subscribe/ unsubscribe operations recursively for routing efficiency.

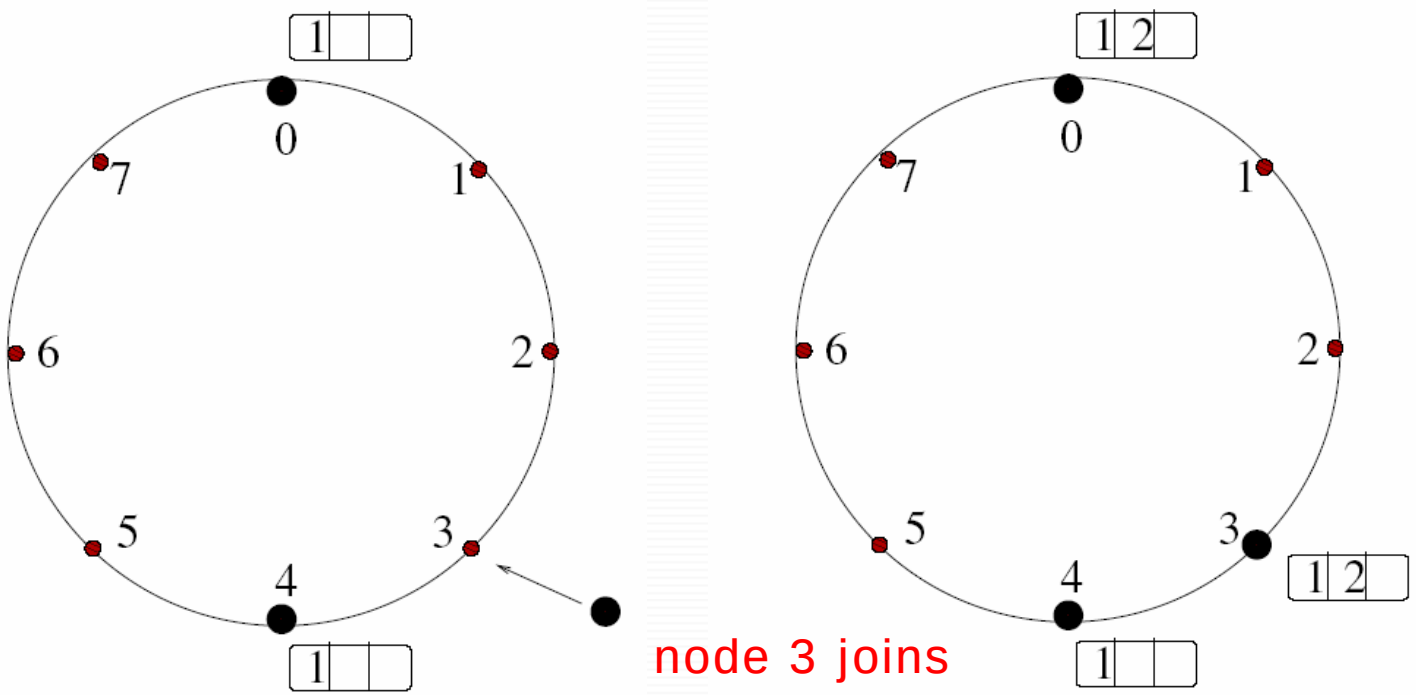
- In order to improve routing efficiency, every node maintains level indices to indicate the node's partitions at different levels.

Node 2 (010) subscribe Key 5 (101) in a 3-bit identifier space



Level Index

- The level index is used to record the change of the RPT height with the increase of partitioning.



Update notifications

- The root node first checks its vector of the key.
- It sends notifications to the representative nodes with corresponding bits set.
- Every intermediate representative is responsible for delivering the notifications to its lower-level representatives.
- When the notification reaches a leaf node, it is forwarded to the subscribers directly.

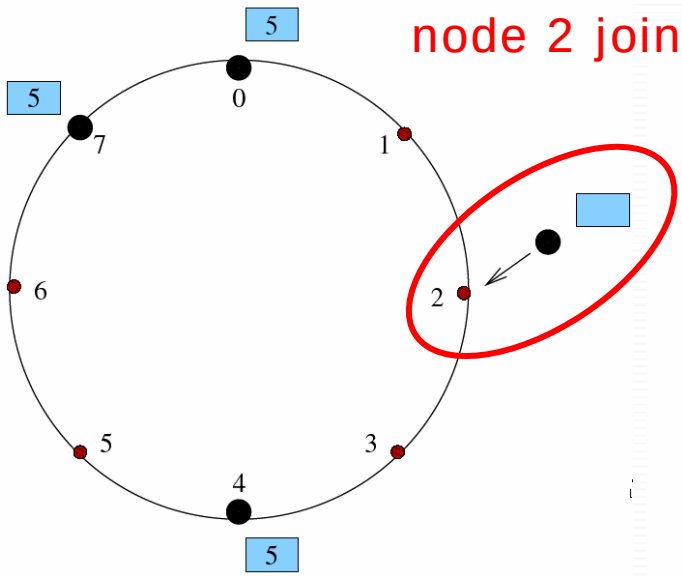
Maintenance and Recovery

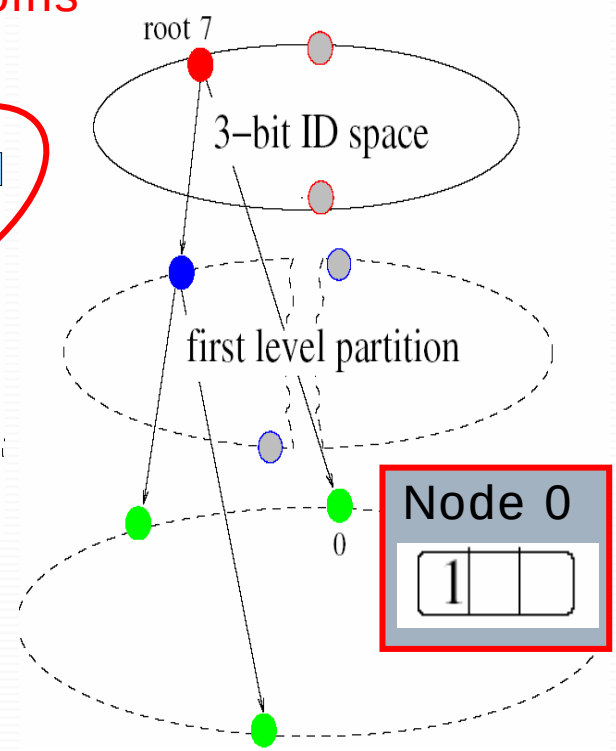
- Node Joining/Leaving
  - Only $O(1)$ nodes are updated when a node joins or leaves.
- Node Failure
  - only $O(\log_2 N)$ messages are transmitted to recover a node failure.

Node Joining/Leaving

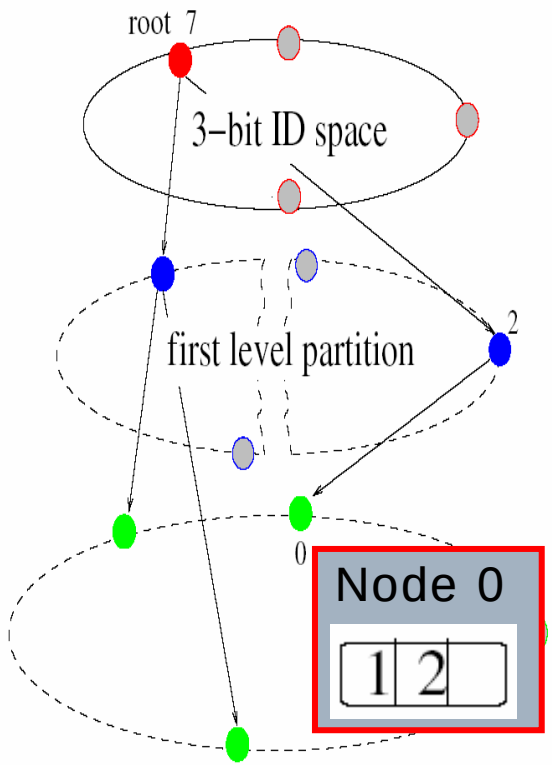
- A newly-joining node in SCOPE needs to take two actions to maintain :
  - transferring partition vectors
  - updating level indices

Node 2 (010) joins in a 3-bit identifier space





Before node 2 joins



After node 2 joins.

Node Failure

- The records at any level partition can be restored from its lower-level partitions.
- SCOPE has a recovery process invoked periodically after the stabilization process.
- When one peer fails, the recovery process is executed by the new node that takes over the failed one.

Maintenance and Recovery

- Node Joining/Leaving
  - Only $O(1)$ nodes are updated when a node joins or leaves.
- Node Failure
  - only $O(\log_2 N)$ messages are transmitted to recover a node failure.

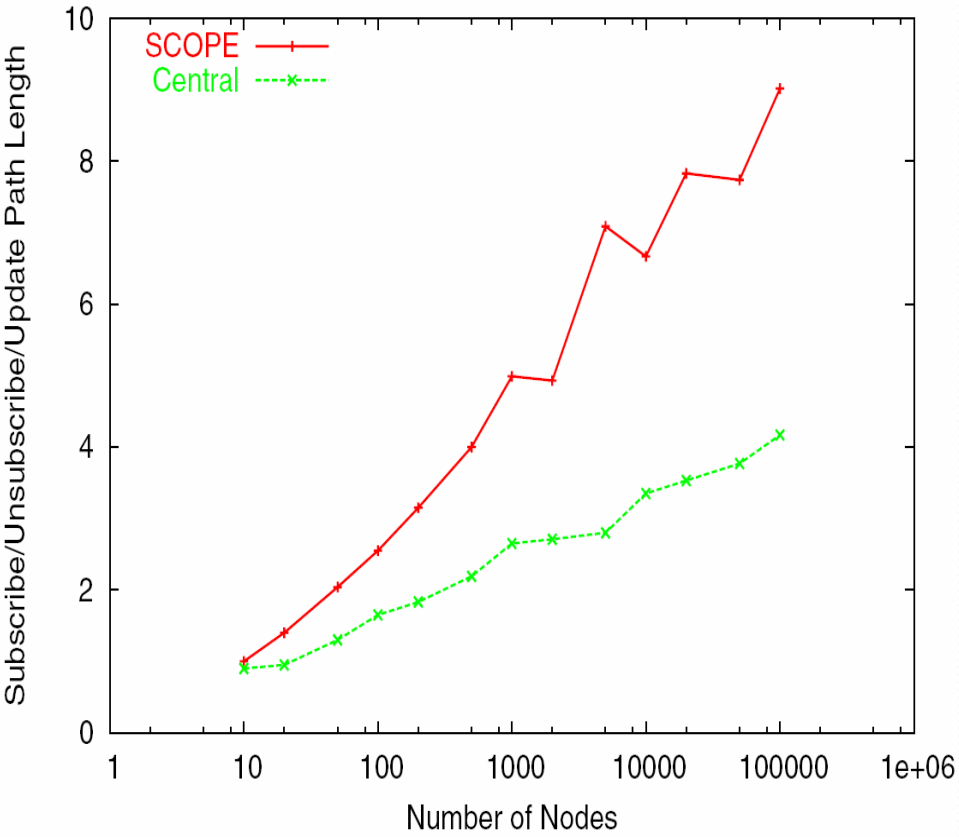
Performance Evaluation

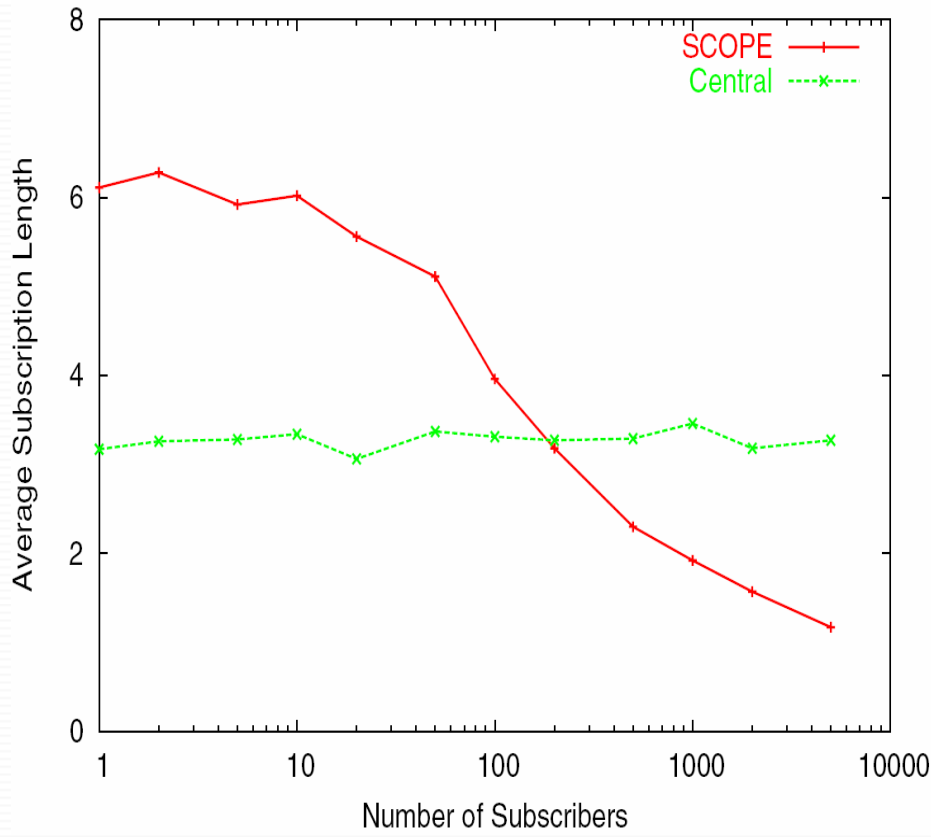
- Simulation Parameters
- Operation Effectiveness
- Maintenance Cost
- Failure Tolerance

Simulation Parameters

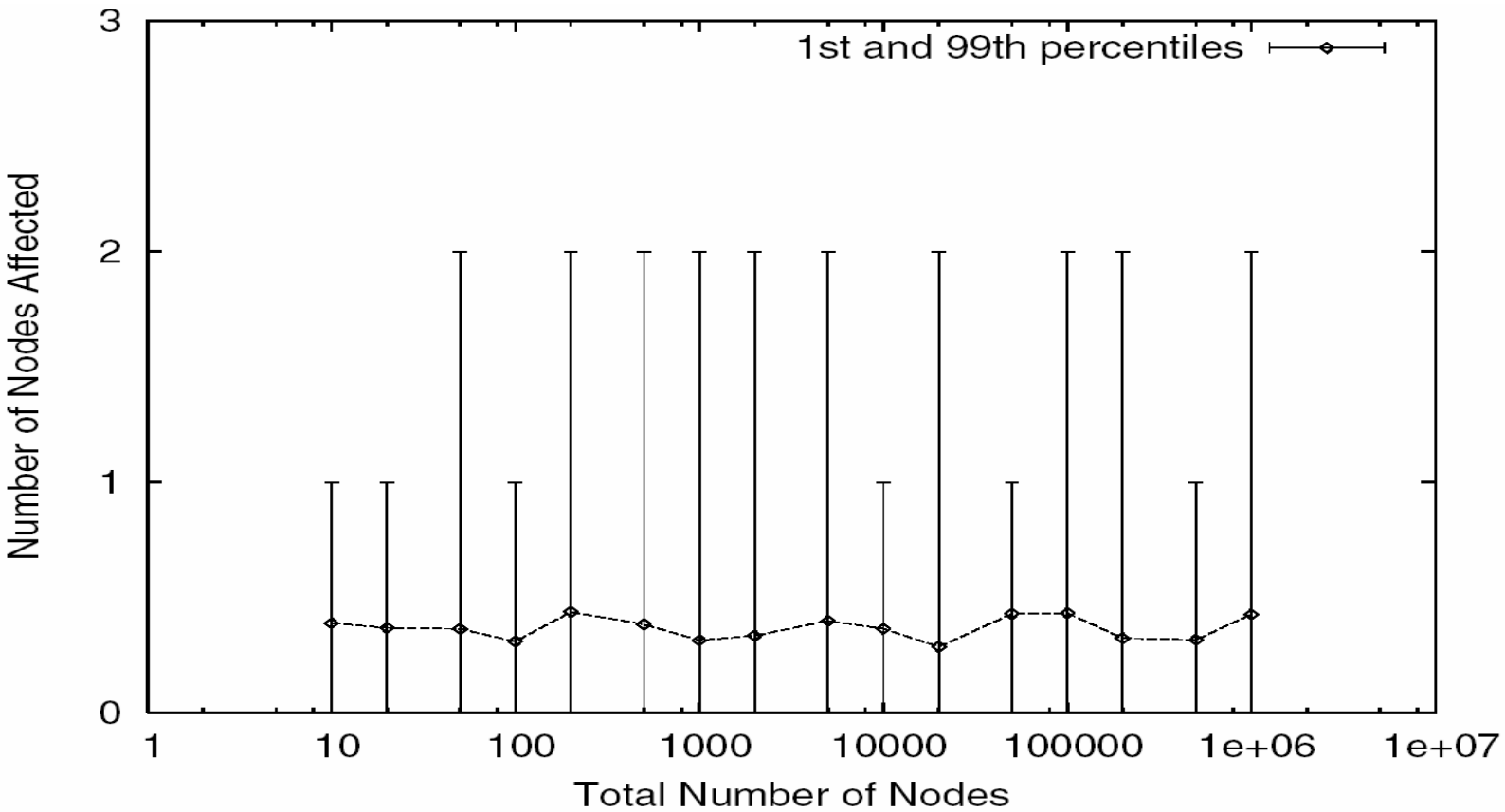
- All nodes and keys are randomly-selected integers.
- They are hashed to a 160-bit identifier space via SHA-1.
- The number of partitions at each level is 16.
- Specified as the Pastry default parameters,
  - The routing table of each node has 40 level
  - each level consists of 15 entries
  - the leaf set of each node has 32 entries.

Operation Effectiveness

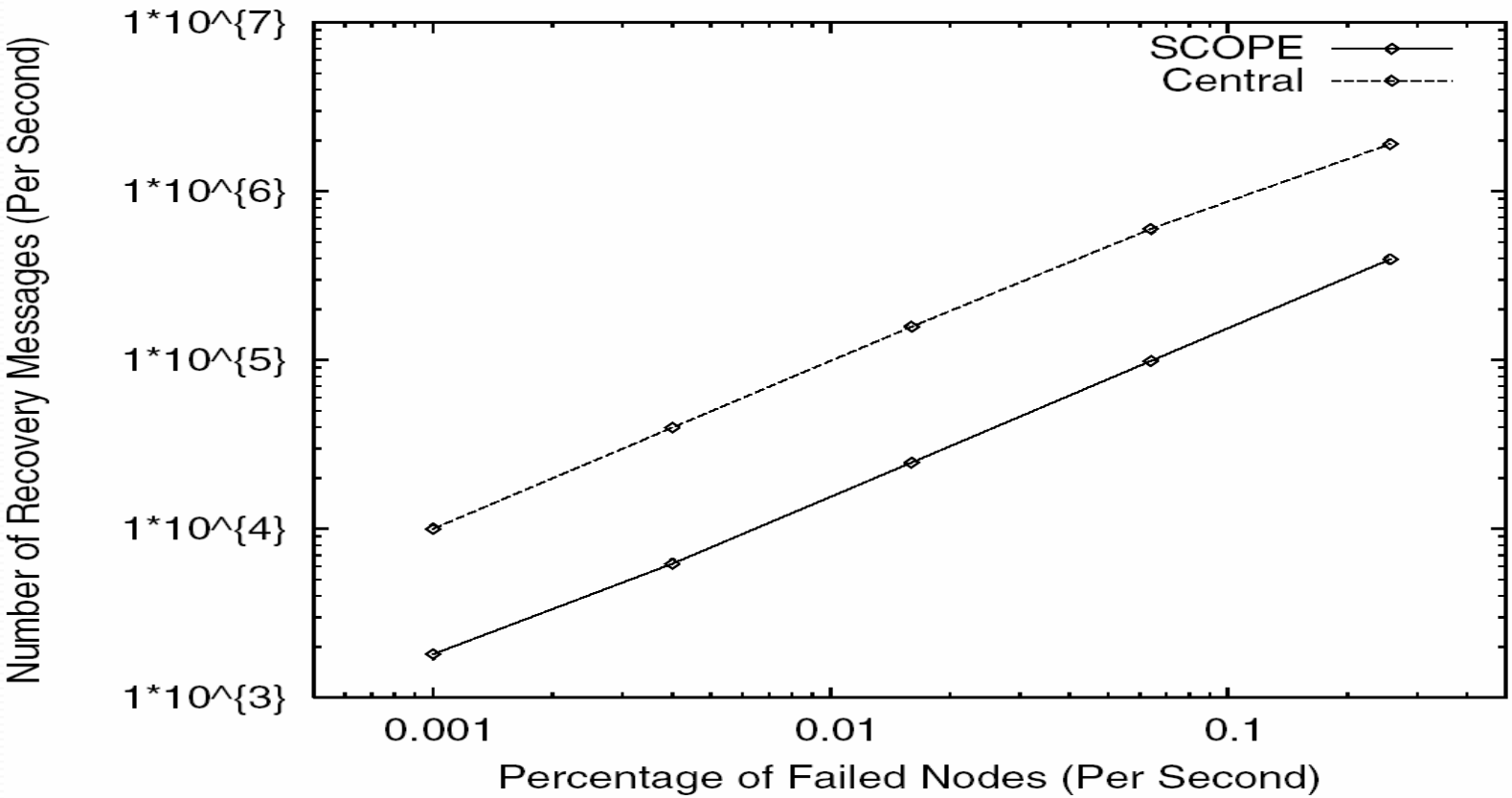




Maintenance Cost (A new node joins)



Failure Tolerance



Conclusion

- SCOPE builds a replica partition-tree for each key to distribute its replica location information among peers.
- Three new primitives subscribe/ unsubscribe/update, are introduced specifically to maintain the replica consistency.
- Due to hierarchical management, these operations can be committed efficiently with minimal maintenance cost.