FAN: A Scalable Flabellate P2P Overlay Supporting Multi-Dimensional Attributes

Wei Song, Ruixuan Li, Zhengding Lu, Guangcan Yu

College of Computer Science and Technology
Huazhong University of Science and Technology

P2P applications booming

- Most current P2P systems only support content searching over a single attribute (keyword, file name,…)
- Full blown P2P applications require an efficient resource lookup over multi-attributes
- Example applications:
  - Complex query over P2P
  - Semantic search
  - Large-Scale combinatorial search

Motivation

- Develop a scalable P2P overlay supporting multi-dimensional resource attributes
- High searching efficiency (logarithmic level)
- Low maintenance cost
- Stabilization with peer's fast joining and leaving

Our contributions

- Propose a novel P2P structured overlay supporting multi-dimensional attributes - FAN
- Improve a routing algorithm over FAN
- Give the network management methods for FAN
- Carry out the simulations to evaluate the performance of FAN

Related work

- Unstructured P2P network
  - Like Gnutella systems
- Structured P2P network
  - Algorithms supporting single attributes
  - Algorithms supporting multi-attributes

Resource searching in unstructured P2P network

- Flooding in Gnutella
- Like Gnutella systems try to address the issue of heavy network load
  - Incentive-Based Routing Systems [1]
  - Self Organizing Overlay Using Gossip-based Construction [2]

Resource searching in structured P2P network

- Resource searching over single attribute
  - Chord
  - Tapestry
- Resource searching supporting multi-dimensional attributes
  - CAN

Related work

- Improved algorithms over CAN
  - a routing algorithm [3] supporting multi-dimensional queries
- Multi-attribute Hub
  - Mercury [4]
- Improve division method of CAN
  - the Hilbert curve [5]

Outline

- Introduction
- FAN: Flabellate overlAy Network
- Adaptive Replication for FAN
- Performance Evaluation
- Conclusion

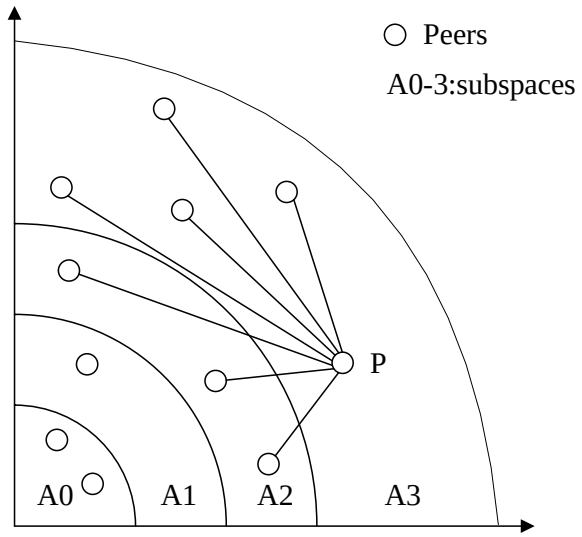
FAN: Flabellate overlAy Network

- The shared resources in FAN are described by *d*-dimensional attributes
- The resource searching and network management are based on the peer's distance

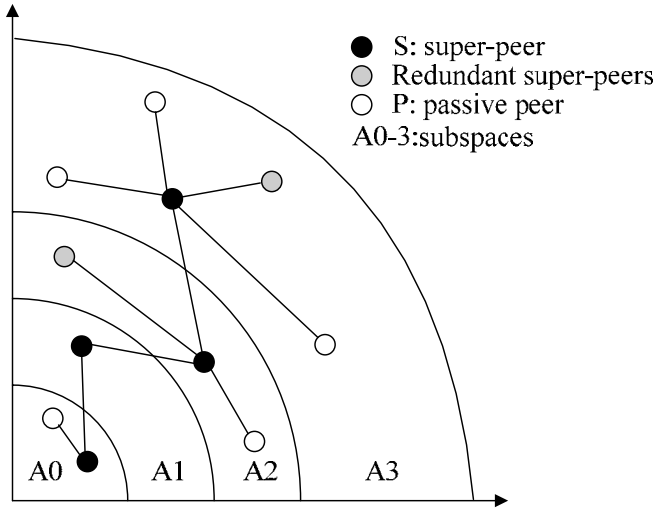
Some definitions in FAN

- Distance for peer $P(x_1,x_2,\ldots,x_d)$
  - $D_P=(x_1^2+x_2^2+\ldots+x_d^2)$
- Subspace
- Distance from subspace $A(a,b)$ to peer $P(x_1,x_2,\ldots,x_d)$
  - $Distance_{AtoP}= \begin{cases} 0 & P \in A \\ \min(|D_P-a|,|D_P-b|), & P \notin A \end{cases}$

FAN construction



Peer stores the routing information of its adjacent subspace



Use the super-peer to manage the routing information for its subspace

Routing table at peers

Table 1. FAN routing table

(a) Routing table in a super-peer

Peer coordinates	Peer address	Subspace range	
(1,0.5,1,1.5, 1, 1)	211.69.192.70:9705	(5,10)	Passive peers
(0,0.5,2,1.5, 1)	211.69.192.80:9705	(5,10)	
(1,1,1,1.4,0, 1.5)	211.69.202.18:9706	(5,10)	
(0.5,1,1.5,0, 1.5)	211.72.101.54:9706	(5,10)	
(1,0,0.5,0.5, 1)	211.80.102.79:9705	(3,5)	Adjacent subspaces
(2,2.5,1.5,0, 2)	211.82.101.78:9705	(10,18)	

(b) Routing table in a passive peer

Peer coordinate	Peer address	Subspace range	
(1,0.5,1,1.5, 1, 1)	211.69.192.70:9705	(5,10)	Super-peer
(0,0.5,2,1.5, 1)	211.69.192.80:9705	(5,10)	Backup super-peer

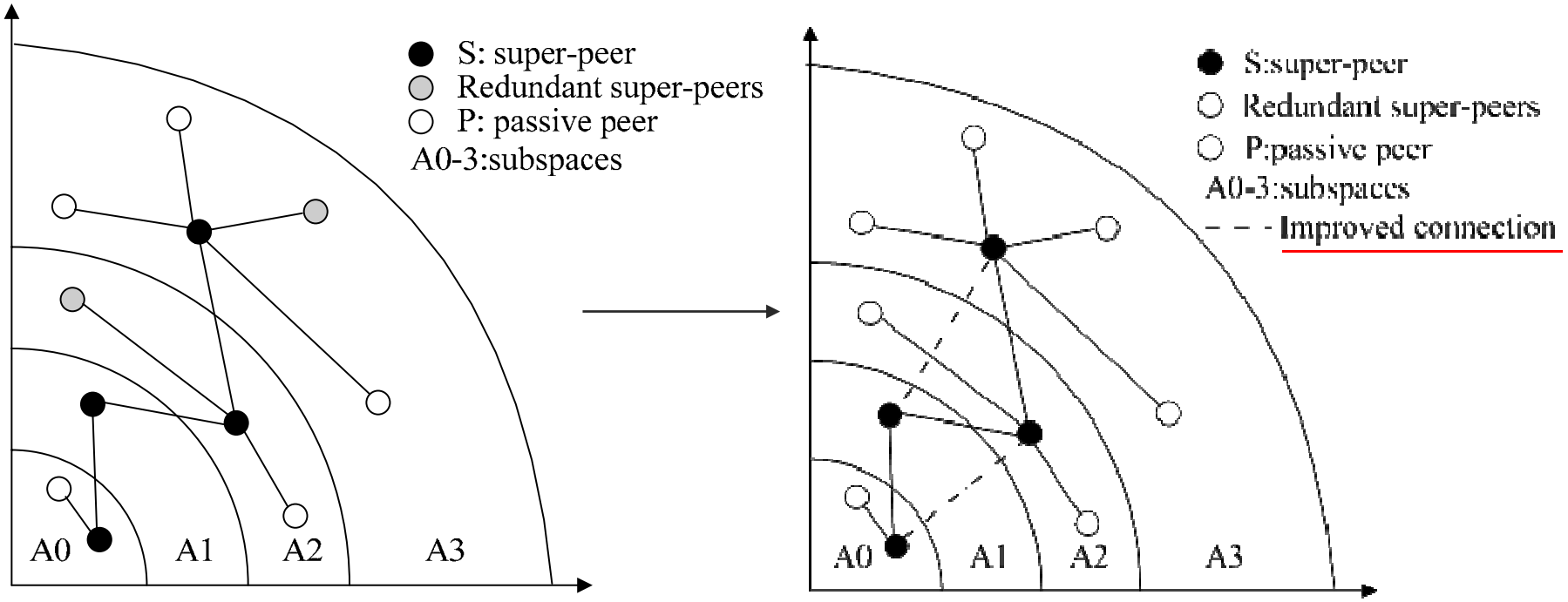
Basic routing algorithm

Input: *p*, source query peer; *r*, target resources.

Routing (*p*, *r*)

1. if (*r* is in *p*'s local resource list) then
2. return *r*;
3. elseif (*p* is a passive peer) then
4. *p* delivers the searching request to its super-peer;
5. else
6. *p* delivers the searching request to the nearest super-peer;
7. end if
8. end if

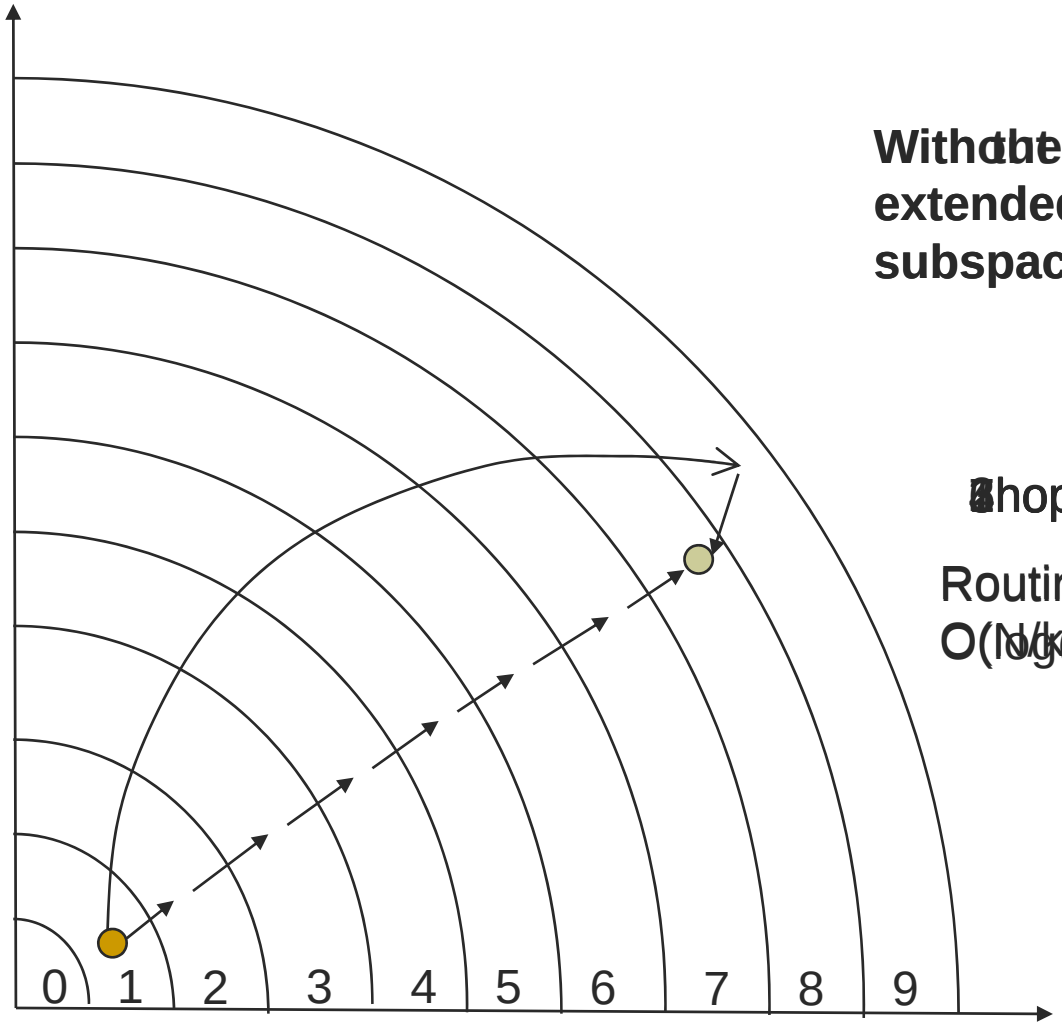
Improved FAN construction



Routing table at super-peer

Peer coordinates	Peer address	Subspace range
(1,0.5,1,1.5, 1, 1)	211.69.192.70:9705	(5,10)
(0,0.5,2,1.5, 1)	211.69.192.80:9705	(5,10)
(1,1,1,1.4,0, 1.5)	211.69.202.18:9706	(5,10)
(0.5,1,1.5,0, 1.5)	211.72.101.54:9706	(5,10)
(1,0,0.5,0.5, 1)	211.80.102.79:9705	(3,5)
(2,2.5,1.5,0, 2)	211.82.101.78:9705	(10,18)
(5,2.5,1.5,3,2.5)	**211.69.187.23:9701**	**(40,65)**
……		

Example of FAN routing



With the link state of the extended adjacent subspace

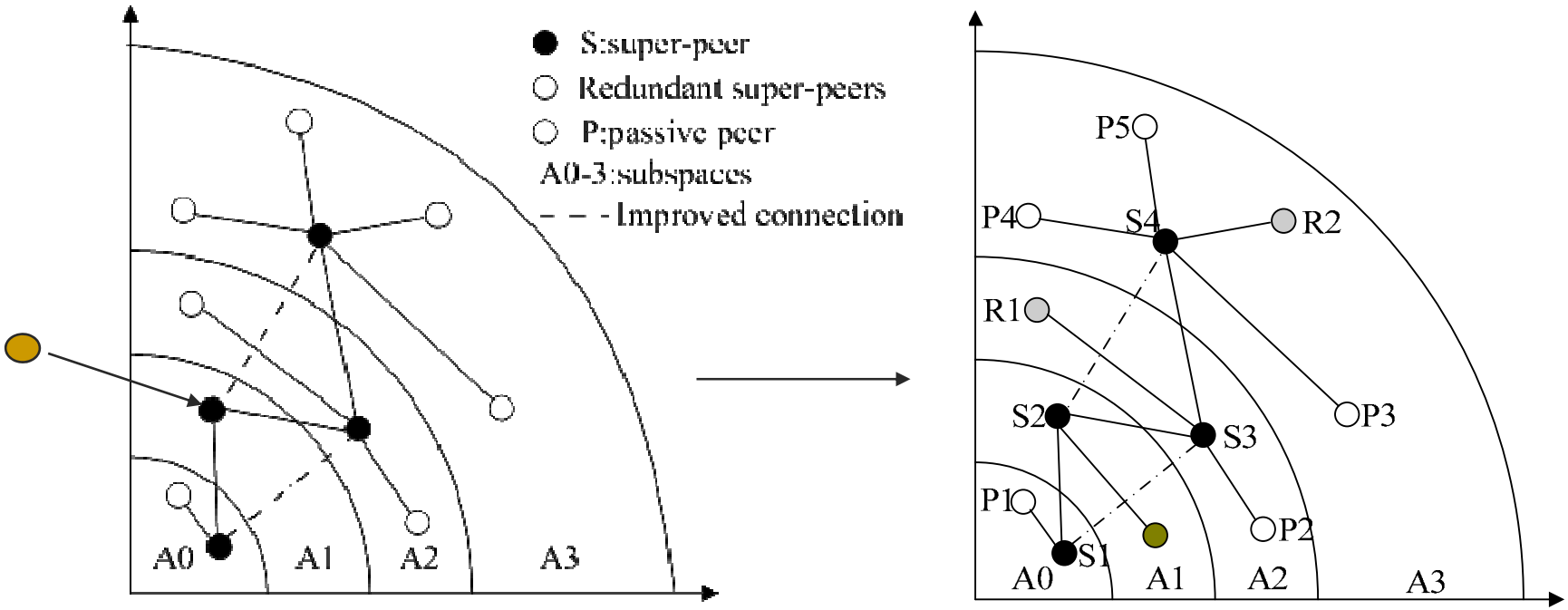
8 hops

Routing efficiency: O(N/k)

Peer joining

A peer joins a subspace without fulfillment

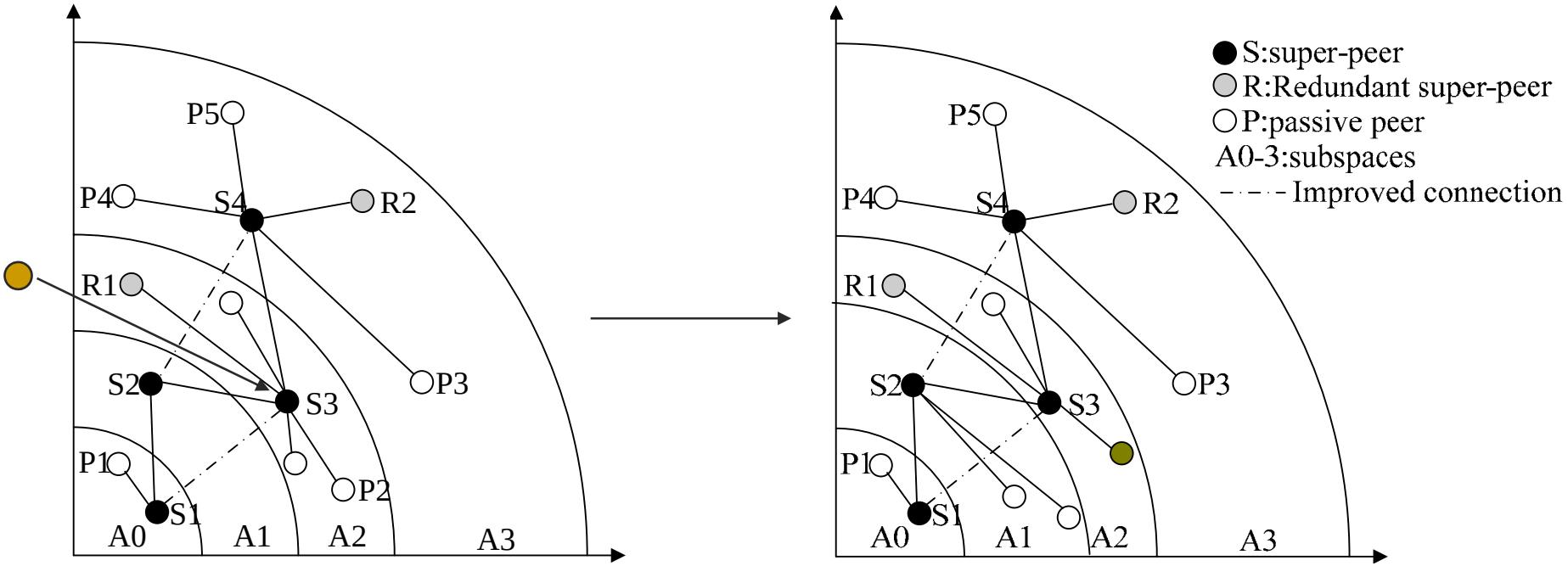
Directly join



Peer joining

A peer joins a full subspace

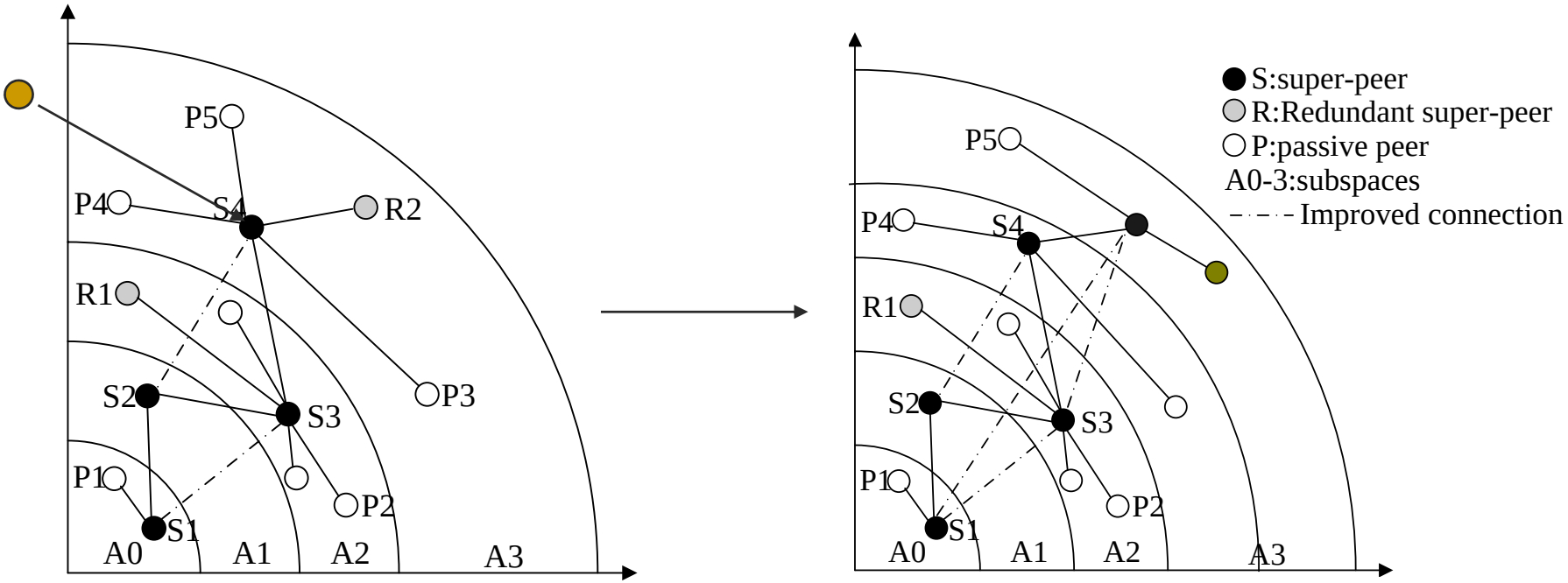
Adjust the subspace



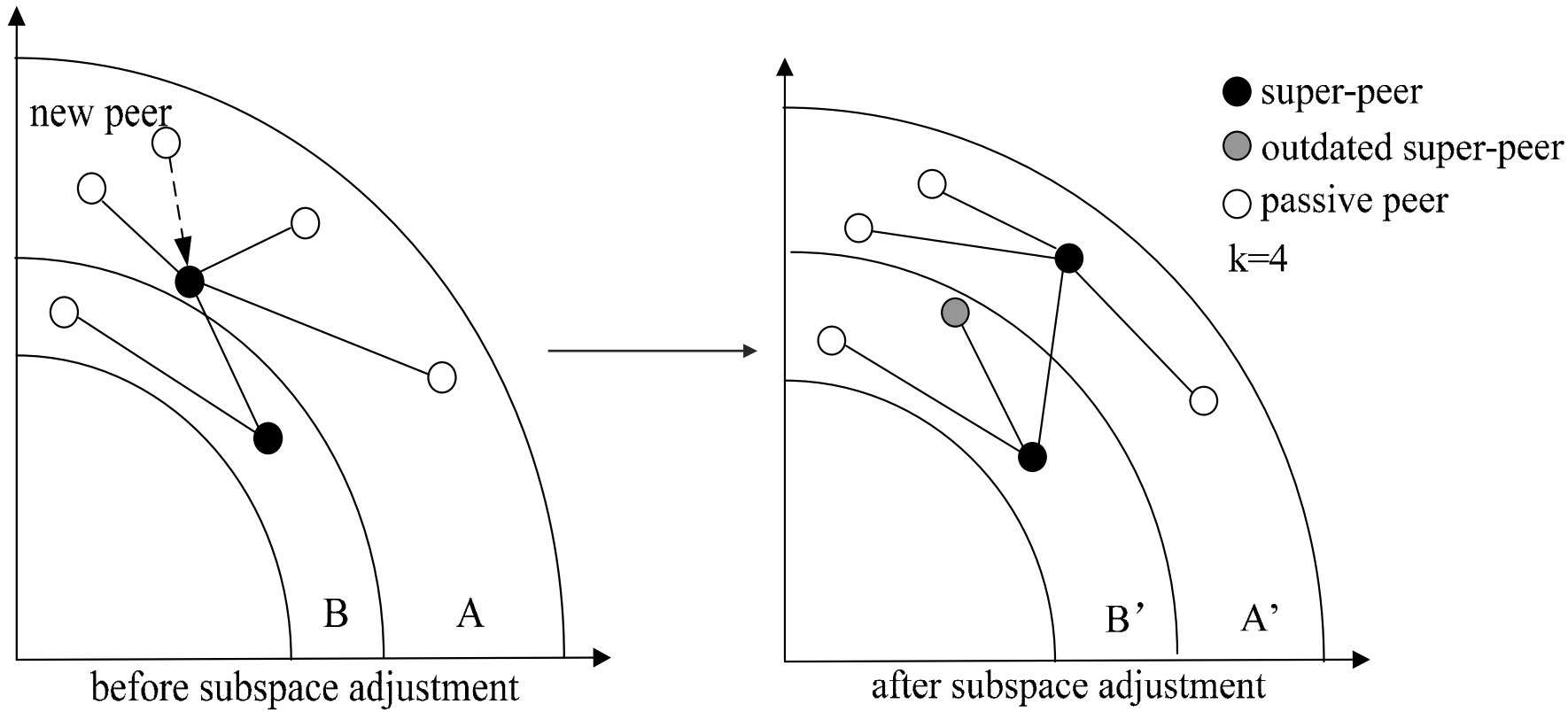
Peer joining

A peer joins a full subspace

Split the subspace



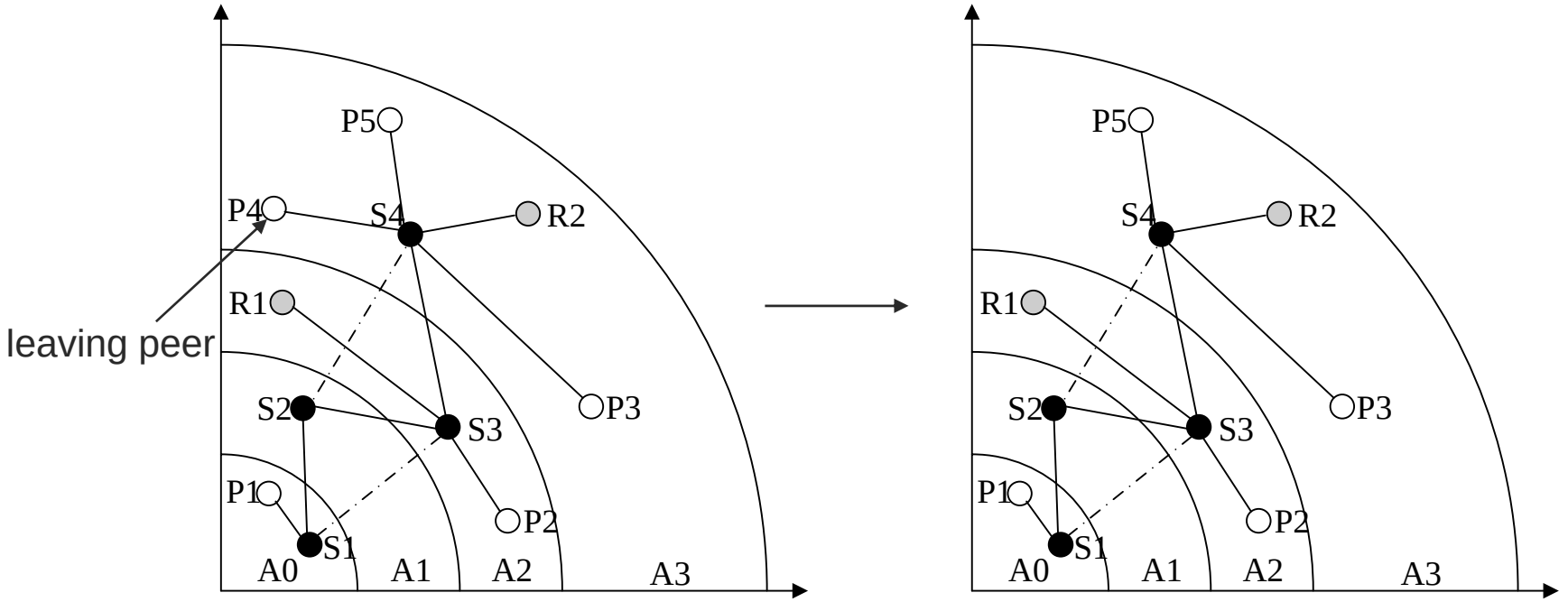
Subspace adjustment



Peer leaving

A passive peer leaves

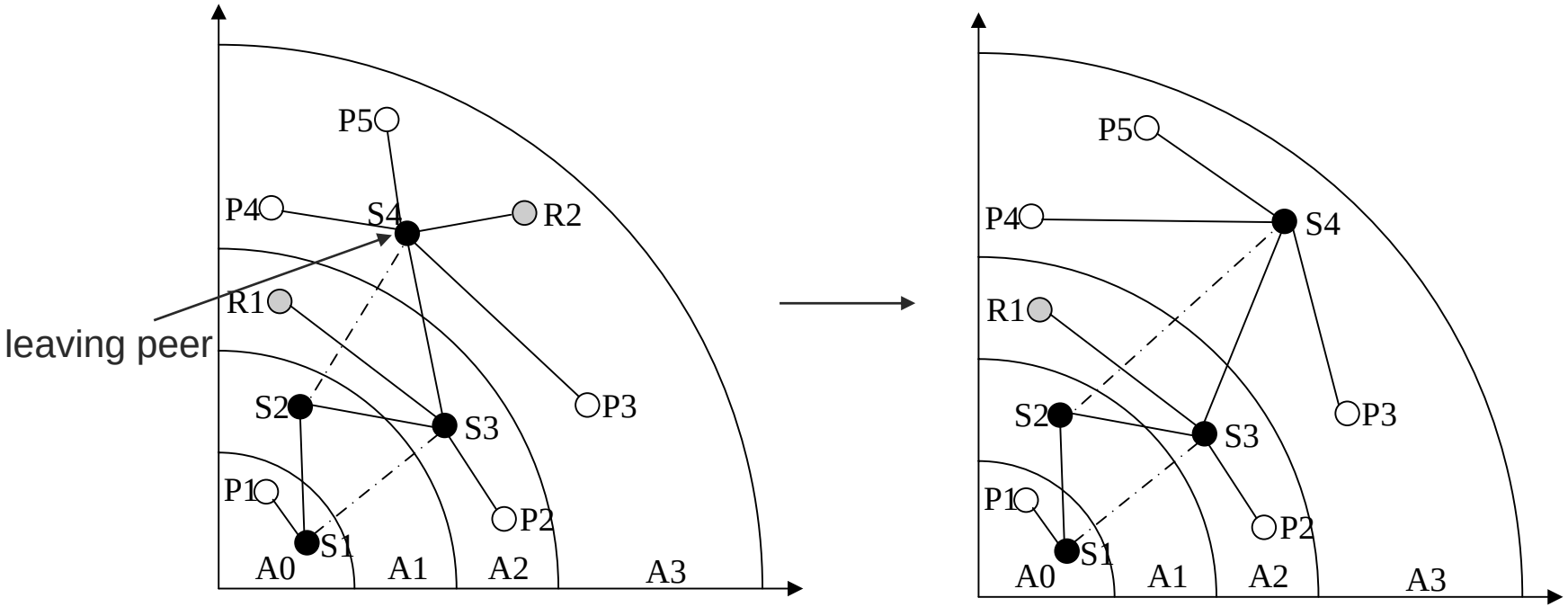
Directly leave



Peer leaving

A super peer leaves, but it is not the last one in that subspace

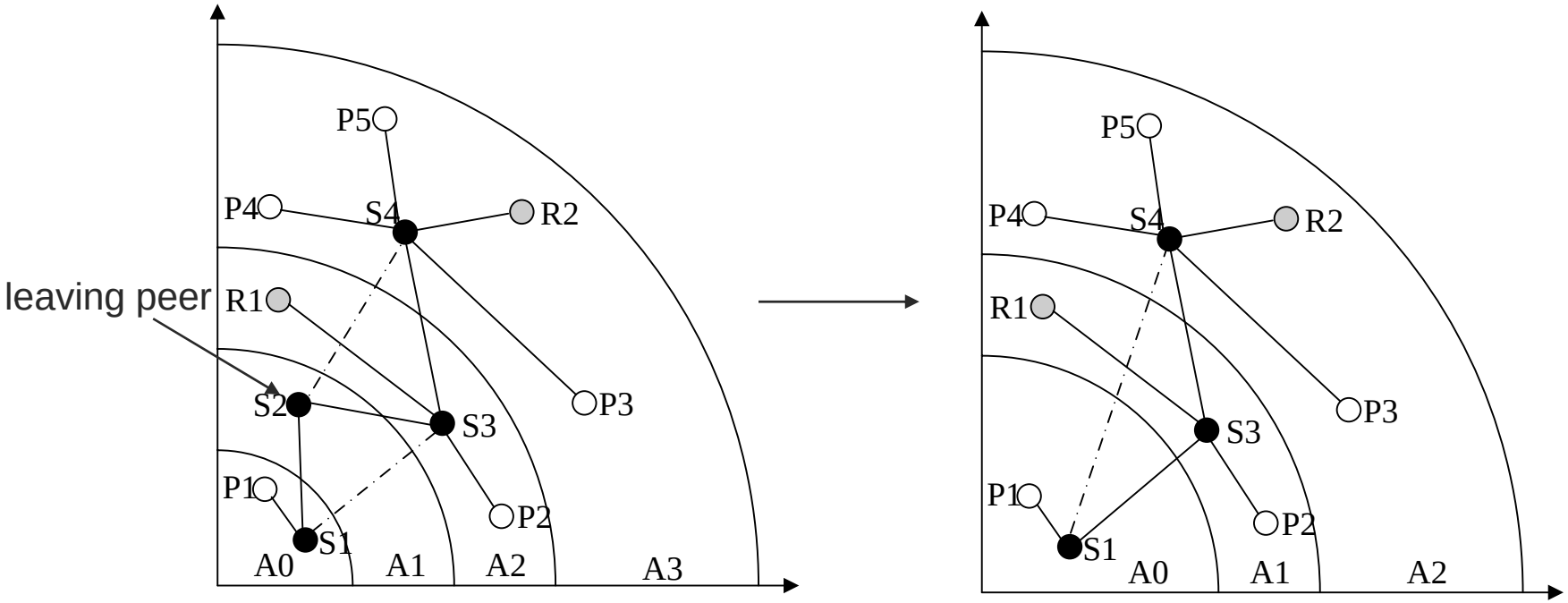
Changed to the backup super-peer



Peer leaving

The last peer in the subspace leaves

Merged to the adjacent subspace



Outline

- Introduction
- FAN: Flabellate overlAy Network
- Adaptive Replication for FAN
- Performance Evaluation
- Conclusion

FAN Adaptive replication

- The distribution of resources and accesses in a P2P network are both imbalanced
- Some nodes who host the popular resources become the hot nodes in the network
- We try to improve the replication algorithms to relieve the hot nodes in FAN

FAN Adaptive replication

- Create the replicas for the most popular resources in FAN
- Each super-peer keeps a query message table to analyze the popularity of local resources
- The nodes keep the replicas of a resource connecting each other to form a VRN (virtual replica network)

Query message table

count of queries	last node traveling through				average hops
resource $r_3(120)$	$n_0(50)$	$n_1(30)$	$n_2(20)$	$n_3(20)$	3.3
resource $r_2(90)$	$n_2(35)$	$n_1(30)$	$n_0(20)$	$n_3(5)$	2.8
resource $r_1(50)$	$n_1(20)$	$n_0(15)$	$n_2(10)$	$n_3(5)$	4.7

Adaptive replication algorithm

Input: *q*, query message; *r*, *q*'s target resource

Replication(*q*)

1. if (*q*'s forwarding count has reached *max_fwd*) then
2. if(*r* is top-*k* popular resource locally or *r*'s average hops are bigger than *max_hops*) then
3. create new replica via function new_replica();
4. return;
5. end if
6. end if
7. finding the least loaded node *n* from VRN;
8. if (*n* is in flow control model) then
9. insert *q* into the query queue;
10. create new replica via function new_replica();
11. else
12. *q*'s forwarding count++;
13. forward *q* to *n*;
14. return;
15. end if

Create new replica algorithm

new_replica()

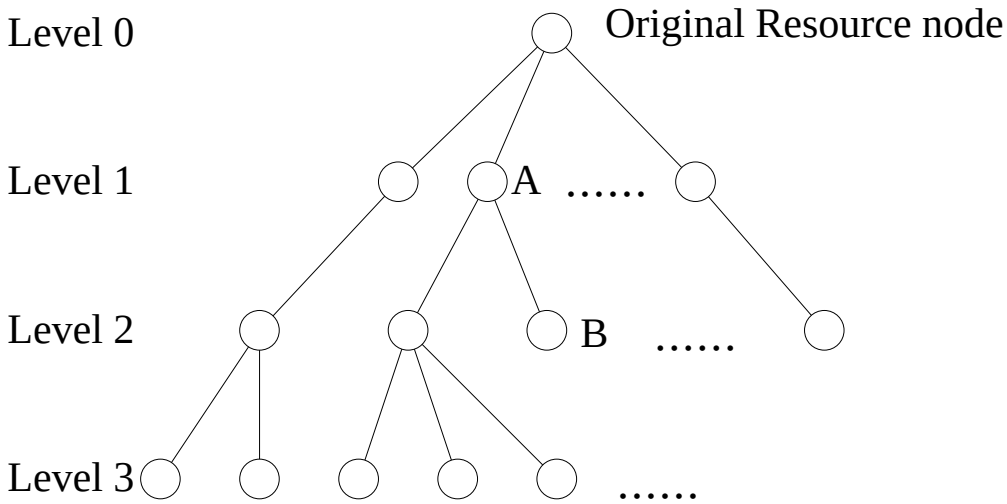
1. find the node *n* $\notin$ VRN from query message table with the maximum query message;
2. if found then
3. create a new replica at node *n*;
4. *n* joins the VRN;
5. else
6. randomly select a node *n*' at the average hops from FAN;
7. create a new replica at node *n*'
8. *n*' joins the VRN;
9. end if

Virtual replica network

- We choose the least load node to forward the query, so a peer needs to know the load information in the VRN

- If we use a complete figure to organize VRN, the cost of updating load is considerably high

Replica inheritor tree



VRN load update

- Based on the replica inheritor tree level, a peer *a* transfers its load updating massage to another node *b* by the probability *P*(*a*|*b*)

$$P(a \mid b) = C \times A^{-1-|l_a - l_b|}$$

where *A* and *C* are two constant parameters (0<*C*<1, *A*>1), l_a and l_b are the nodes' levels

VRN load update

- Assume that the size of a resource's VRN is *m* and a node *n*'s proportion threshold of the load change is p_n
  - Cost of updating is $E(\cos t) = \sum_{n \in VRN} p_n \cdot (m-1)$
- Base on the small world phenomenon, we use replica inheritor tree to update the load information
  - Cost of updating is $E(\cos t) = \sum_{n \in VRN} \sum_{k \in VRN, k \neq n} p_n \times C \times A^{-1-|l_n - l_k|}$

Outline

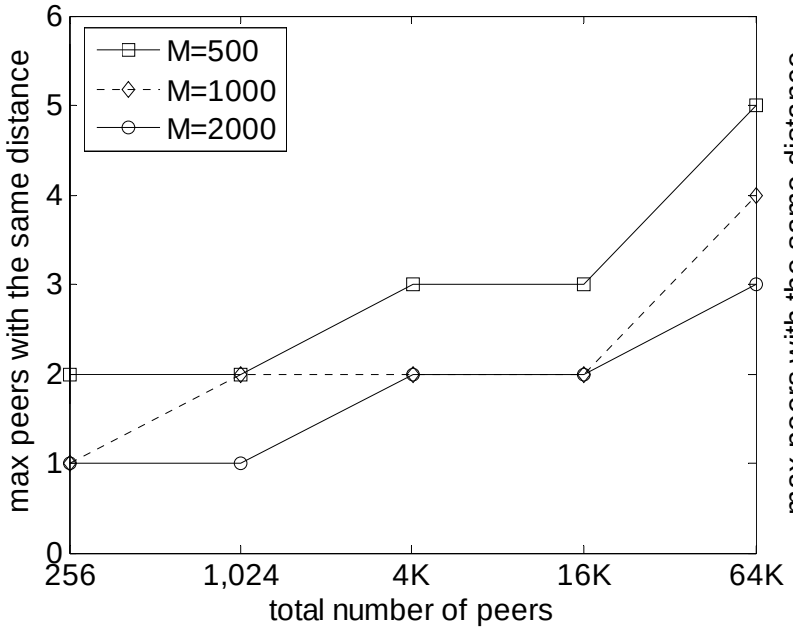
- Introduction
- FAN: Flabellate overlAy Network
- Adaptive Replication for FAN
- Performance Evaluation
- Conclusion

Simulation setup

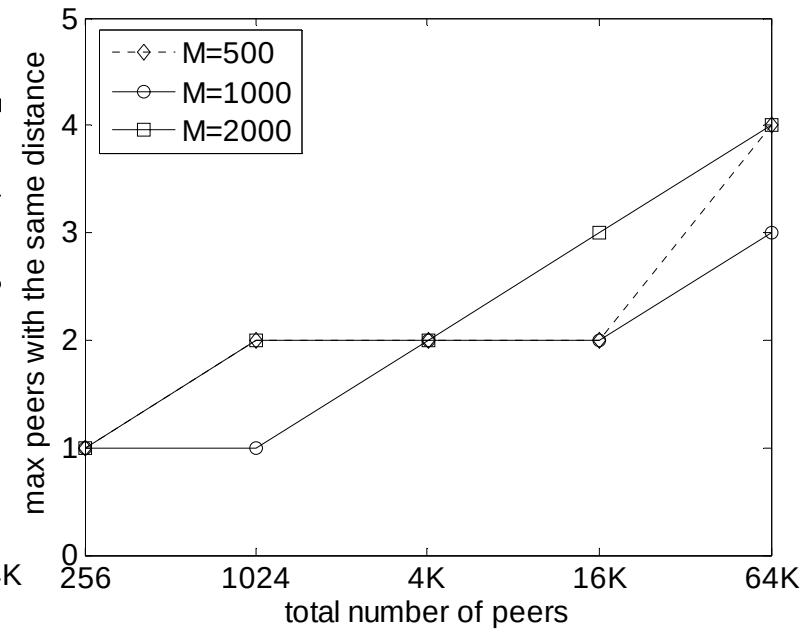
- Chord and FAN: PeerSim
- CAN: from references

	Parameter Descriptions	**Values**
N	The network size of FAN	256, 1024, 4096, 16K, 64K
d	mapping space dimensions	2, 3, 5, 7
S	super-peers in a subspace	1
k	The capability of a subspace	4, 8, 12, 16, 24
M	The range of each dimension value	500, 1000, 2000
w	The size of query queue	30

Maximal number of peers with the same distance



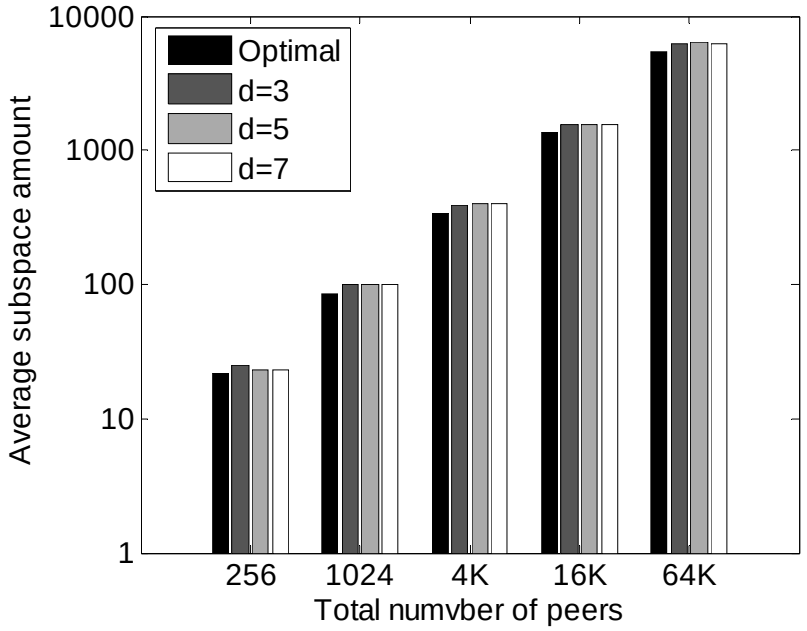
(c) *d*=5



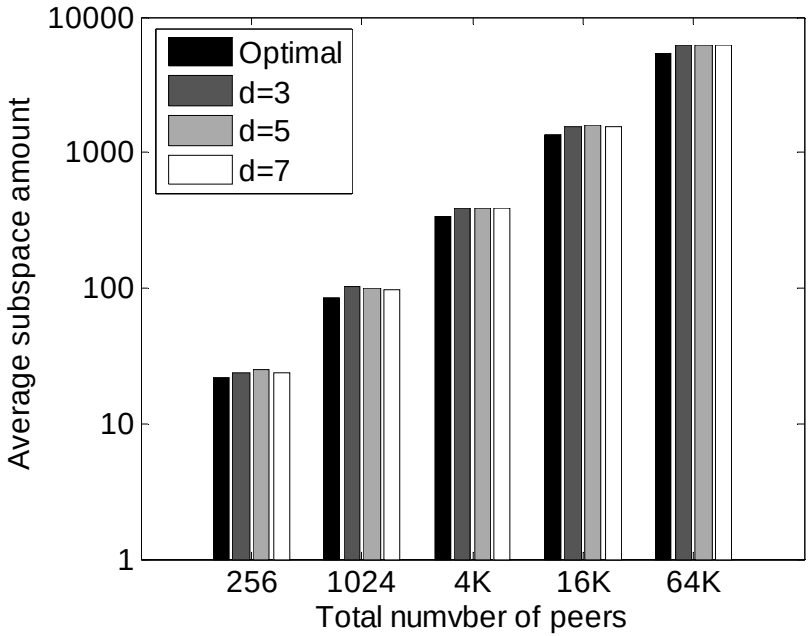
(d) *d*=7

the max number of peers with the same distance can not exceed the subspace capability (*k*)

Average amount of subspaces in FAN



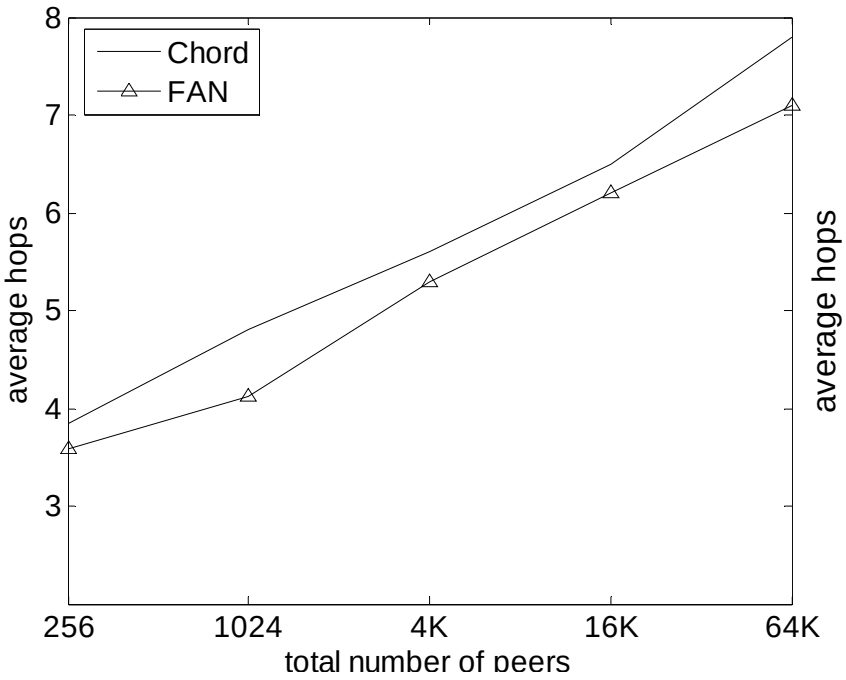
(c) k=8 M=1000



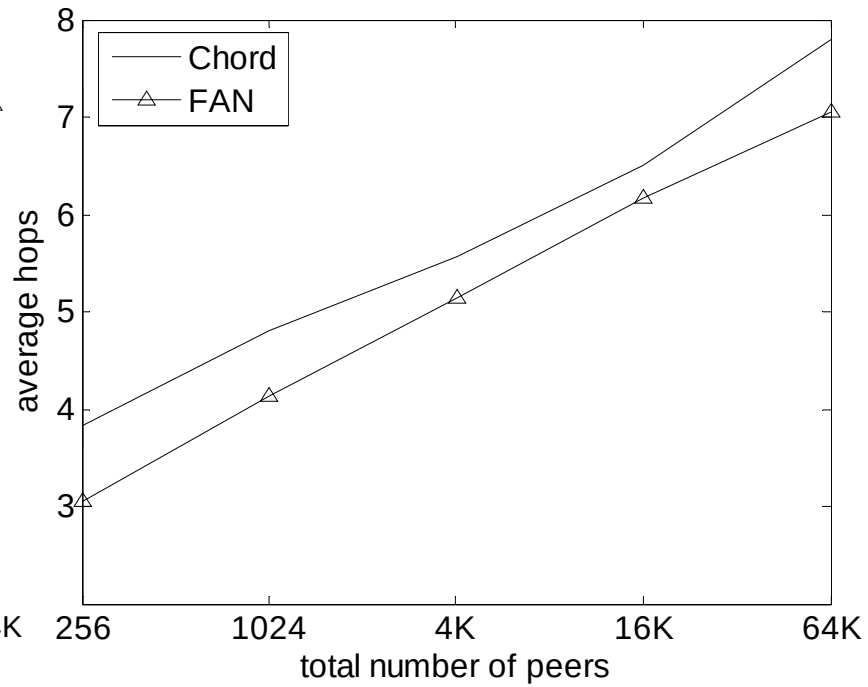
(d) k=8 M=2000

Slowly subspace increasing is crucial for the scalability and availability of FAN

Routing efficiency of single-dimensional FAN and Chord



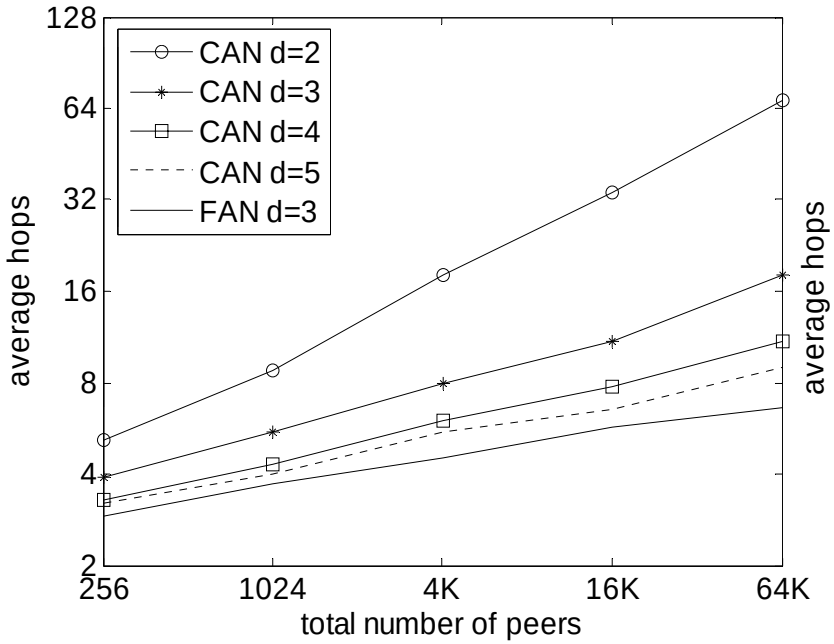
k=4



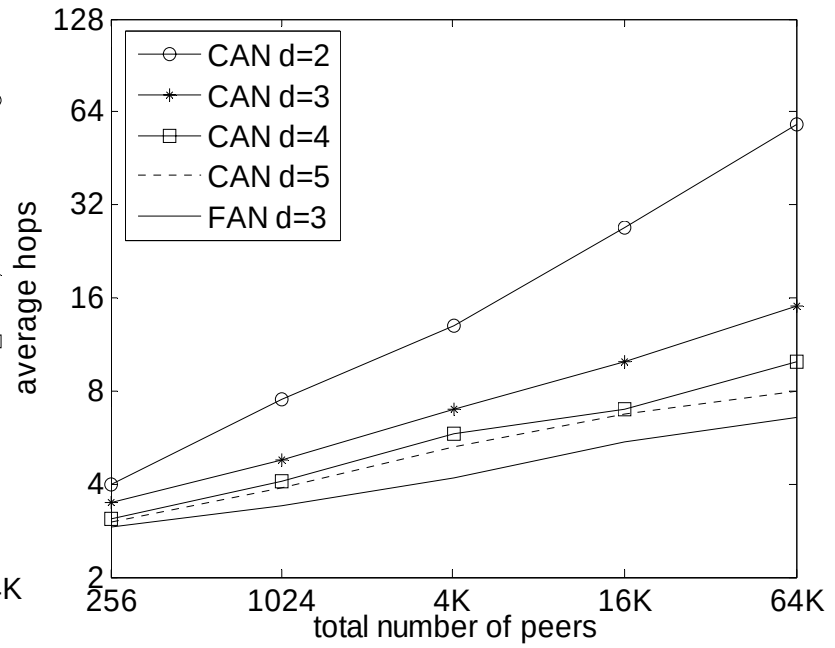
k=8

routing efficiency: the efficiency of the routing message spreading in the extended adjacent subspaces

Routing efficiency of FAN and CAN

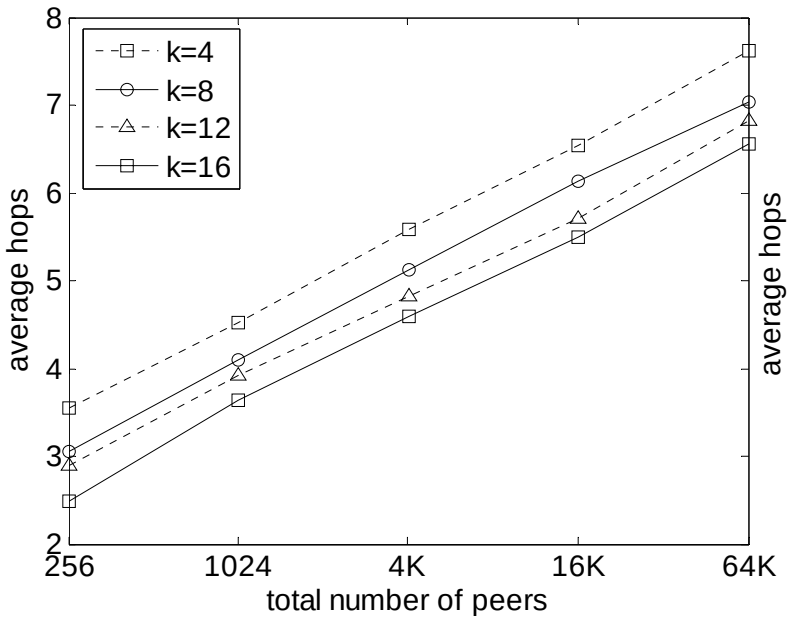


d=2

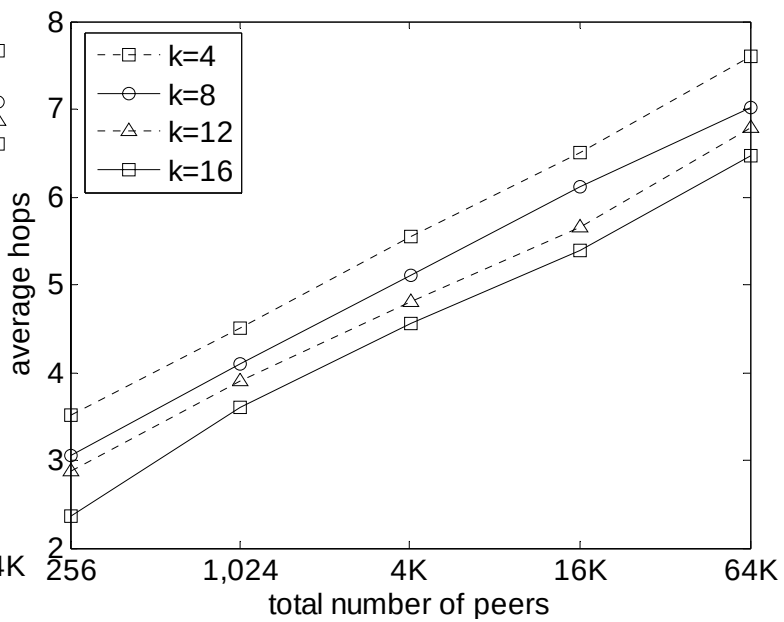


d=3

FAN routing efficiency with various *k* values

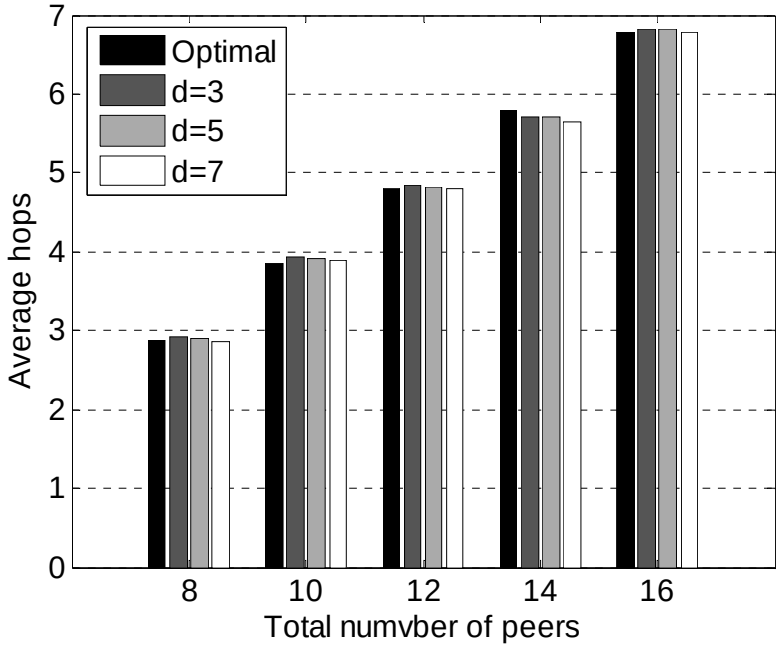


d=5

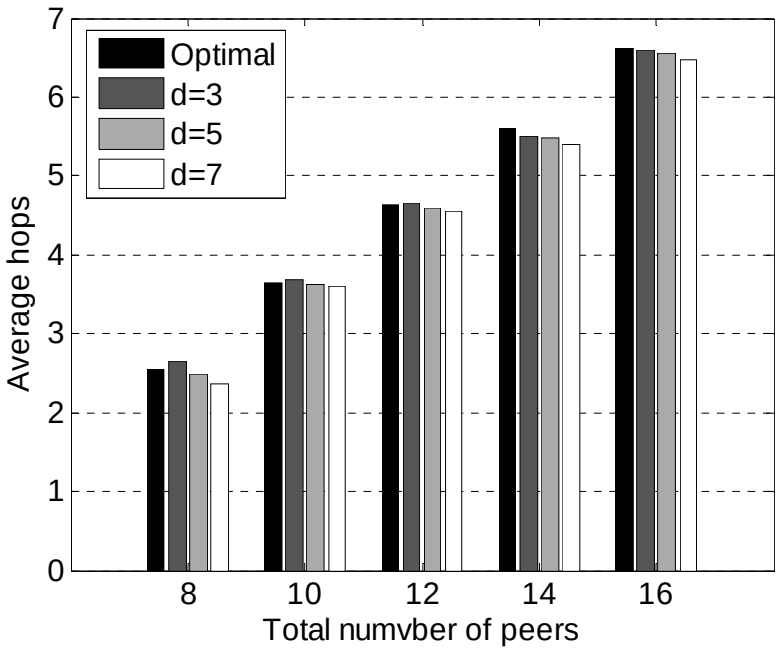


d=7

FAN routing efficiency with various *d* values

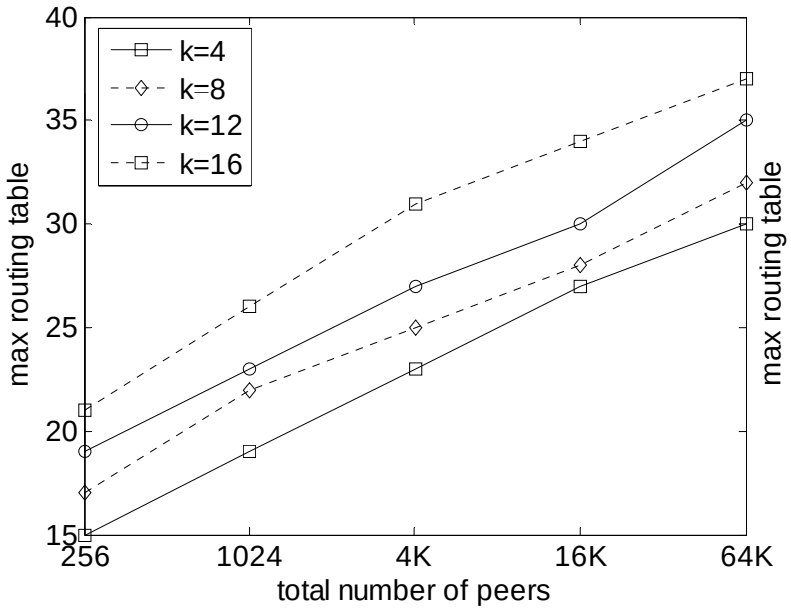


k=12

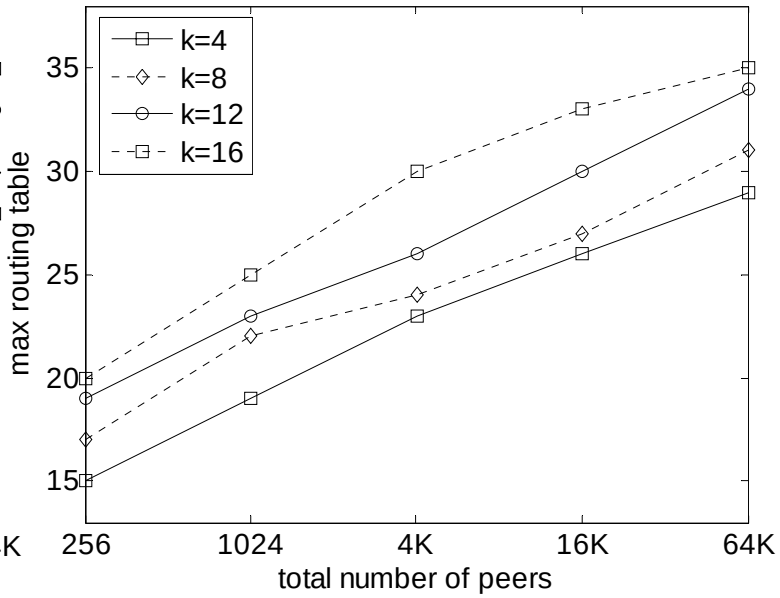


k=16

Maximal routing table items of super-peers in FAN



d=5

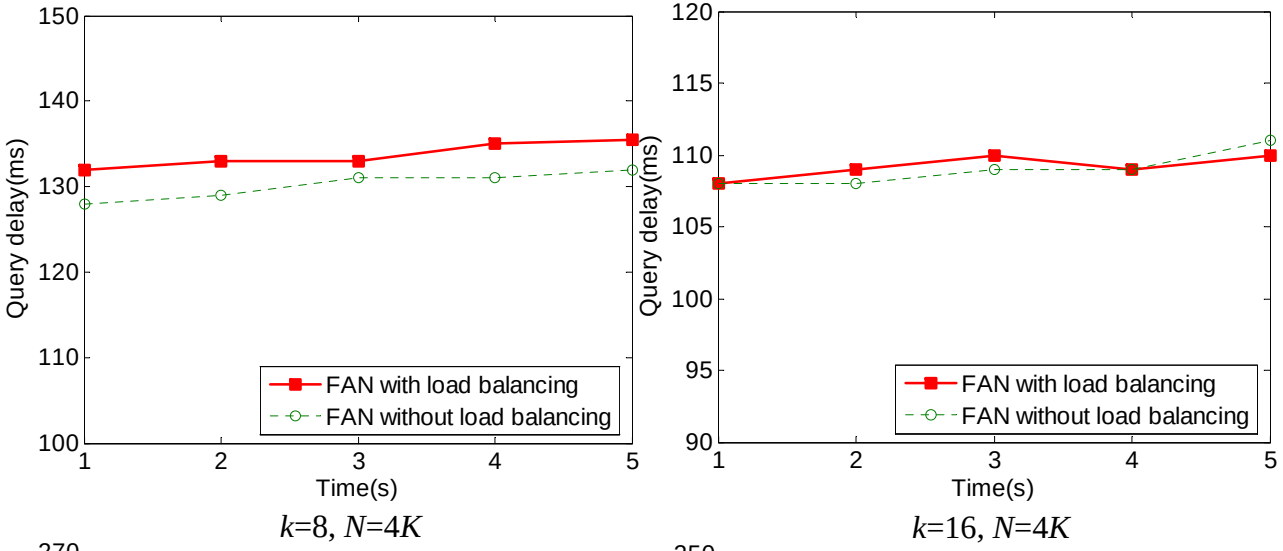


d=7

To test whether the routing messages will overburden the super-peers and influence FAN stability or not

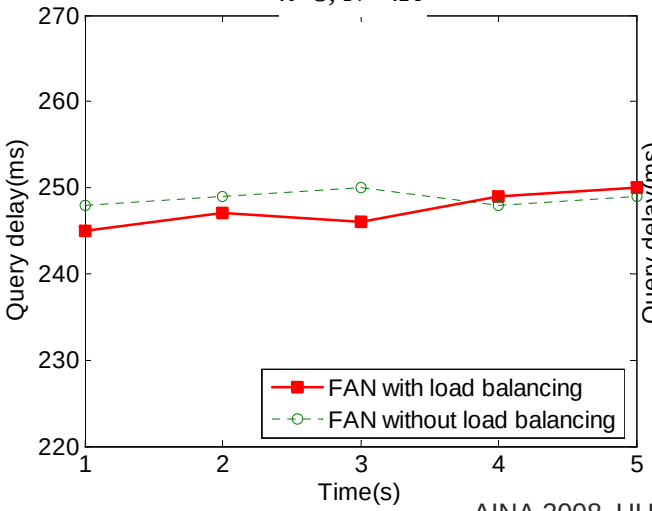
FAN query delay: light network load

each node issues only 1 query in a second

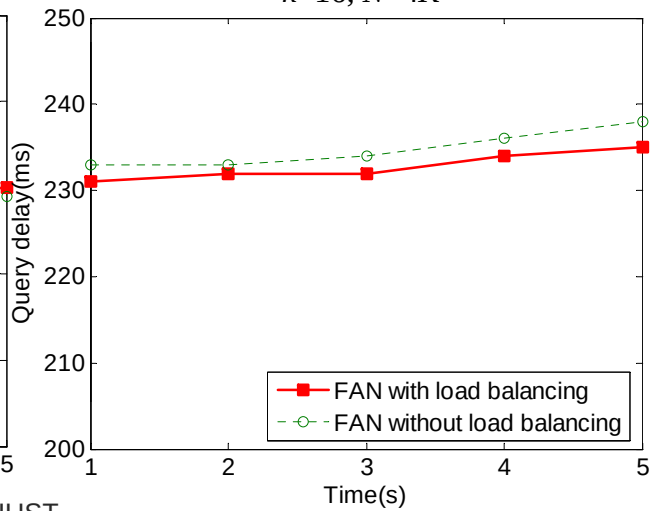


k=8, N=4K

k=16, N=4K



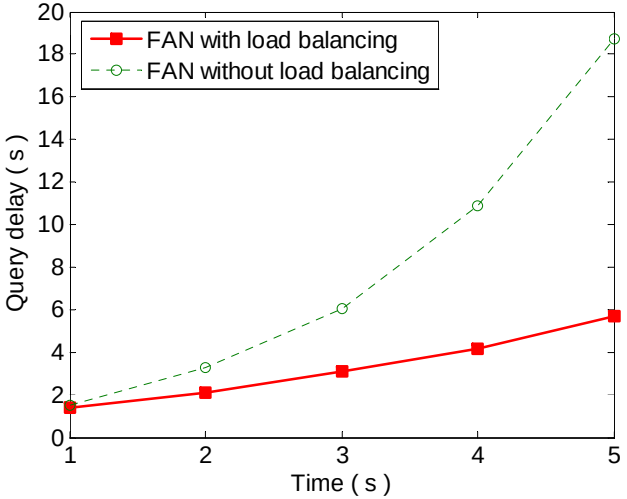
k=8, N=64K



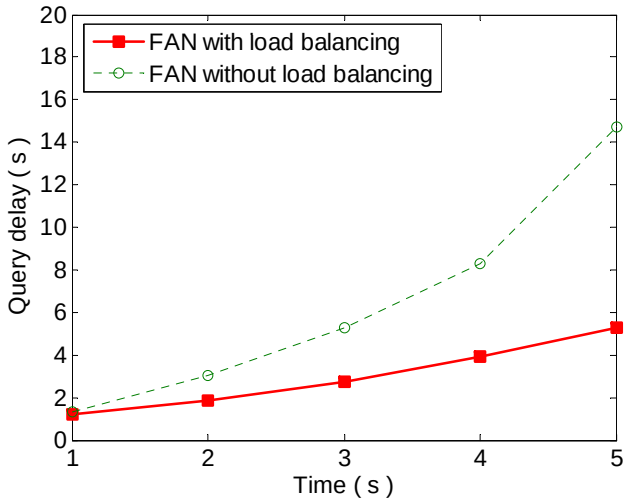
k=16, N=64K

FAN query delay: heavy network load

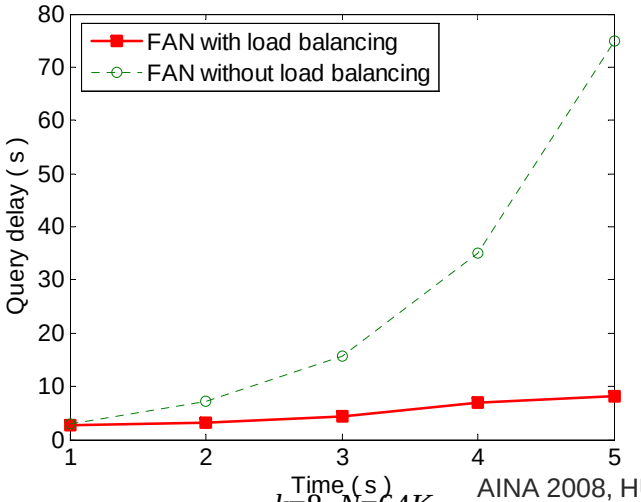
each node issues10 queries in a second



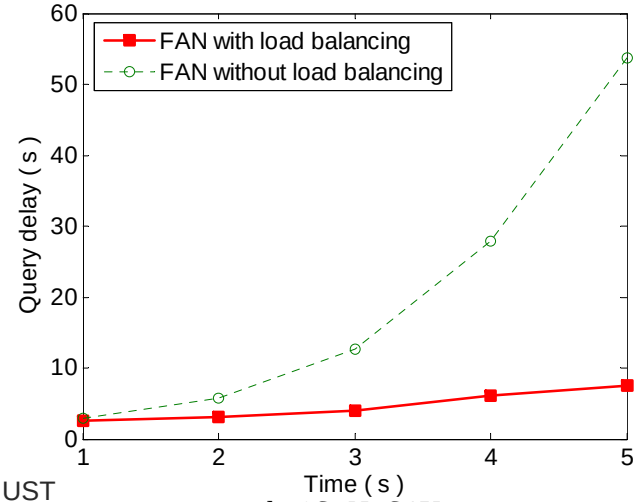
k=8, N=4K



k=16, N=4K



k=8, N=64K



k=16, N=64K

Conclusion and future work

- FAN is an underlying P2P overlay supporting multi-dimensional resource attributes
- Plan to facilitate FAN optimization in terms of the costs and different parameters
- Perform complex queries over FAN
- Apply to real applications

References

- [1] M. Afergan, “Using Repeated Games to Design Incentive-Based Routing Systems”, *In: Proc. of the 25th Annual IEEE Conference on Computer Communications (INFOCOM’06)*, 2006. pp. 1-13.
- [2] R. Guerraoui, S. B. Handurukande, and K. Huguenin, “GosSkip, an Efficient, Fault-Tolerant and Self Organizing Overlay Using Gossip-based Construction and Skip-Lists Principles”, In: *Proc. of the 6th IEEE International Conference on Peer-to-Peer Computing (P2P’06)*, 2006. pp. 12-22.
- [3] B. Liu, W.-C. Lee, and D. L. Lee, “Supporting Complex Multi-Dimensional Queries in P2P Systems”, In: Proc. of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS’05), NW Washington, DC USA, 2005, pp. 155-164.

References

- [4] A. R. Bharambe, M. Agrawal, and S. Seshan, “Mercury: Supporting Scalable Multi-Attribute Range Queries”, In: *Proc. of the 2004 ACM SIGCOMM Conference*, Portland, Oregon, USA, 2004, pp. 353-366.
- [5] Hamilton, Chris H.; Rau-Chaplin, and Andrew, “Compact Hilbert Indices for Multi-Dimensional Data”, In: *Proc. Of CICIS 2007*, pp. 139-146.

Thank you!

http://idc.hust.edu.cn