

Advanced Computer Networks

Distributed Hash Table

Jianping Pan
Summer 2015

Course project proposal

- Feedback in progress
 - ↳ from the teaching team (TA and me)
- Also please open a thread on connex forum
 - ↳ “Discussion and Private Messages”
 - Course Projects Discussion
 - ↳ discuss your projects among your team members
 - ↳ with the rest of the class
 - ↳ and the teaching team
 - ↳ please keep one project in one thread. Thanks!

Search in P2P networks

- In client-server model, the server is well known
 - ↳ e.g., <http://www.google.com:80/>
- In P2P, everyone is a potential server
 - ↳ how to find “a” server for a request?
 - ↳ Napster
 - centralized directory server
 - ↳ Gnutella
 - scoped flooding search
 - ↳ scalability? reliability?

Abstract P2P networks

- Service primitives

- ↳ put (key, data); // insert data identified by key

- ↳ get (key); // retrieve data by key

- data: files, services, or any objects

- key: file name, service name, or any label

- applications: file swap, storage, content delivery, etc

- Design goals

- ↳ scalability

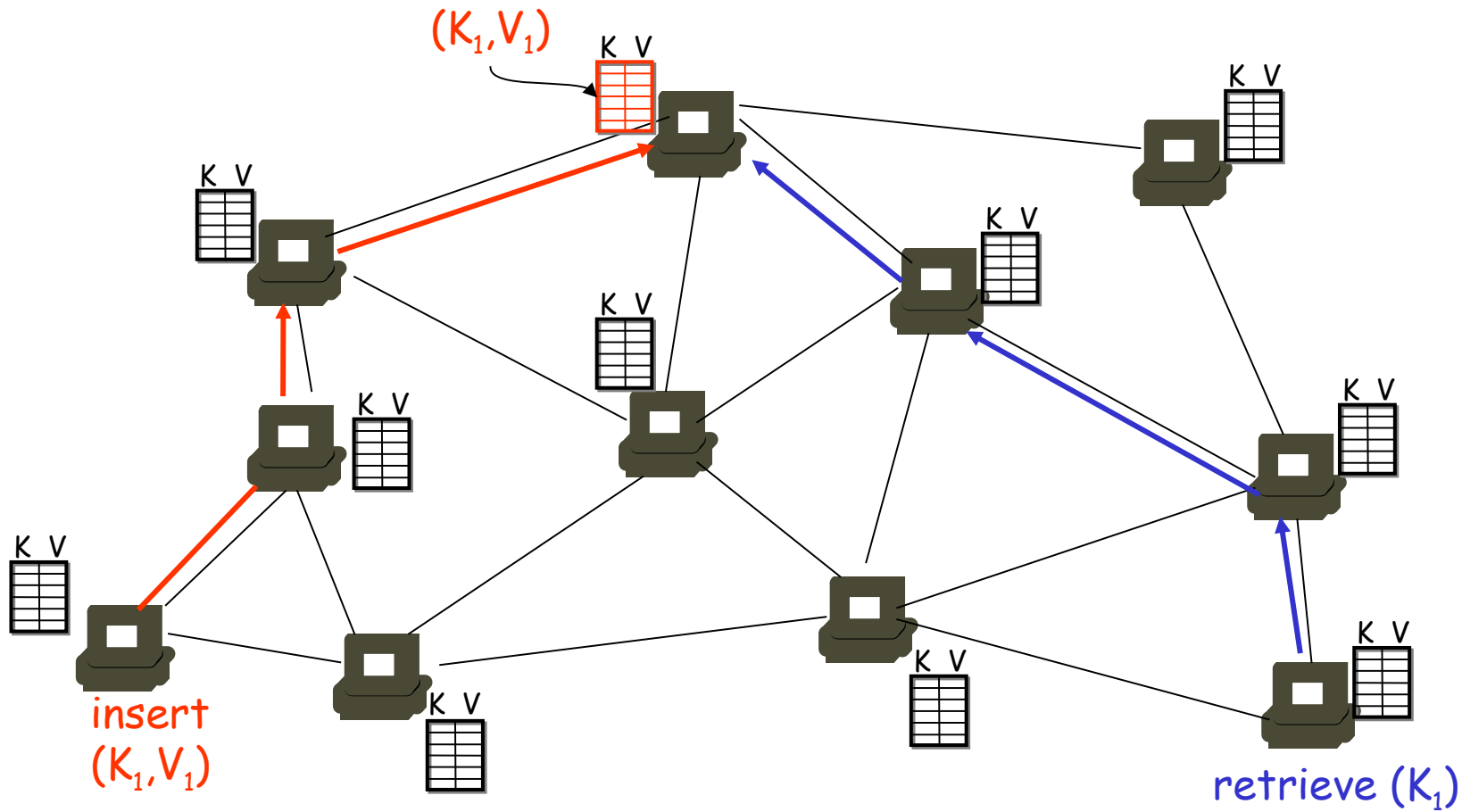
- ↳ robustness

- ↳ performance

Distributed Hash Table

- Hash table
 - ↳ map keys to values
 - ↳ by hash function
 - ↳ insert and lookup: usually $O(1)$
 - ↳ need to deal with hash collisions
- Distributed Hash table
 - ↳ no hash coordinator
 - consistent hashing
 - ↳ robustness: self-repair

Ideas



<http://www.icir.org/sylvia/sigcmm01.ppt>

Schemes

- Chord

- ↳ circular key space

- ↳ lookup: $O(N)$

- with $O(1)$ successor list

Lookup(my-id, key-id)

n = my successor

if my-id < n < key-id

 call Lookup(id) on node n // *next hop*

else

 return my successor // *done*

Chord

- Performance improvement

• \bar{H} lookup: $O(\log N)$ w/ “finger” table

- $O(\log N)$ finger entries

Lookup(my-id, key-id)

look in local finger table for

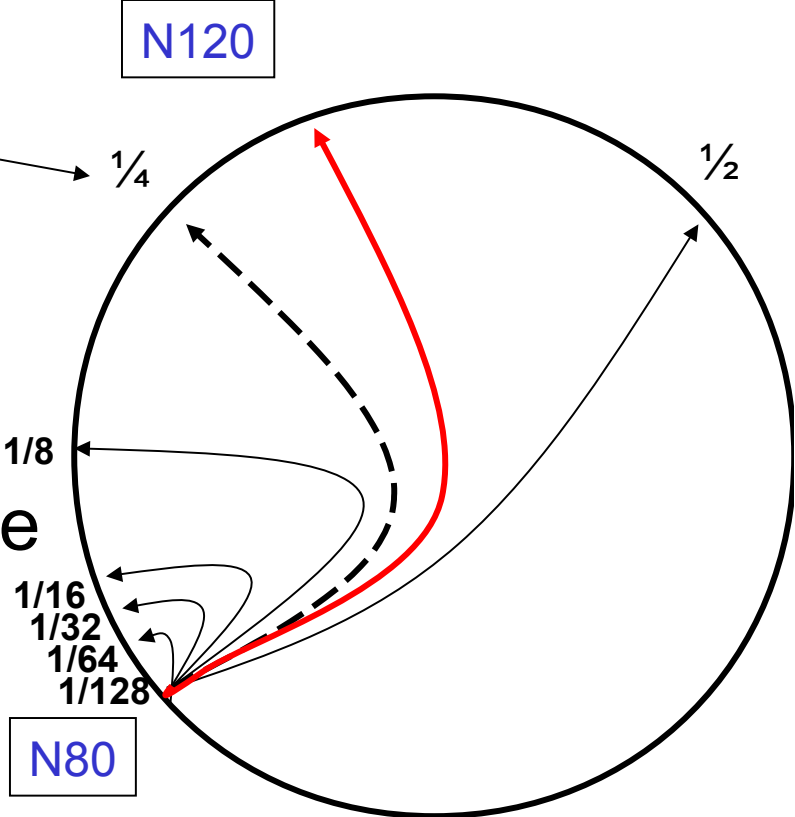
highest node n s.t. $\text{my-id} < n < \text{key-id}$

if n exists

call Lookup(id) on node n // *next hop*

else

return my successor // *done*



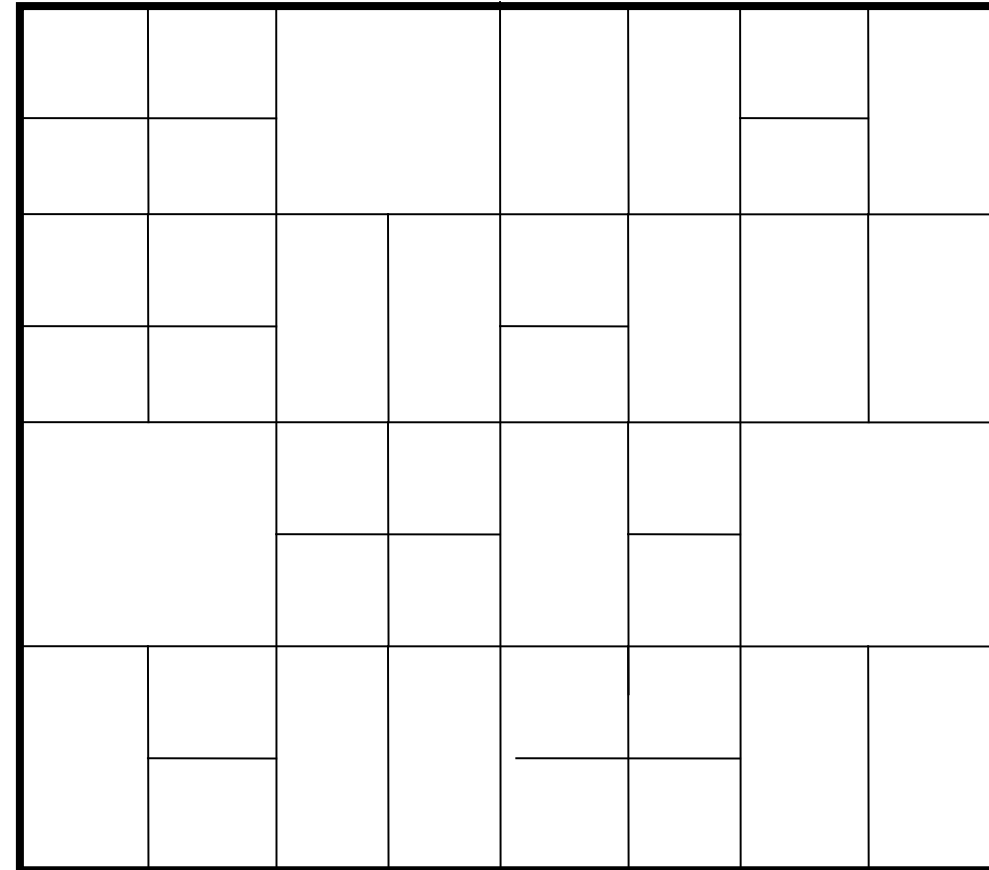
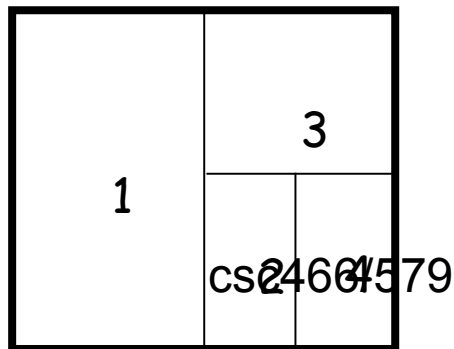
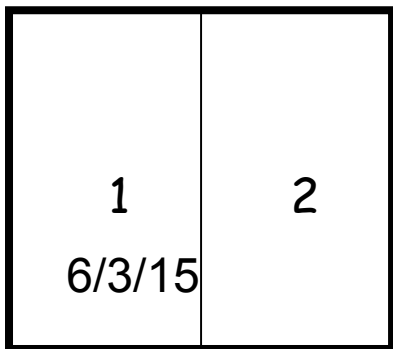
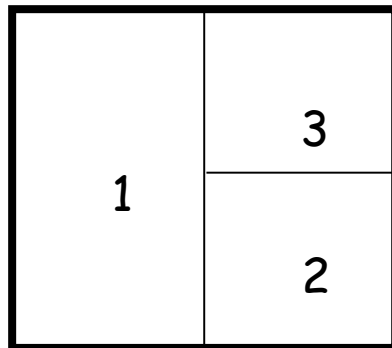
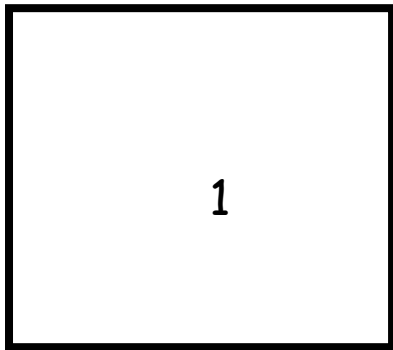
Content-addressable networks

- 2-d example

\mathcal{H} a virtual space torus

- $h_x(\text{key})$ and $h_y(\text{key})$

\mathcal{H} partitioned among nodes



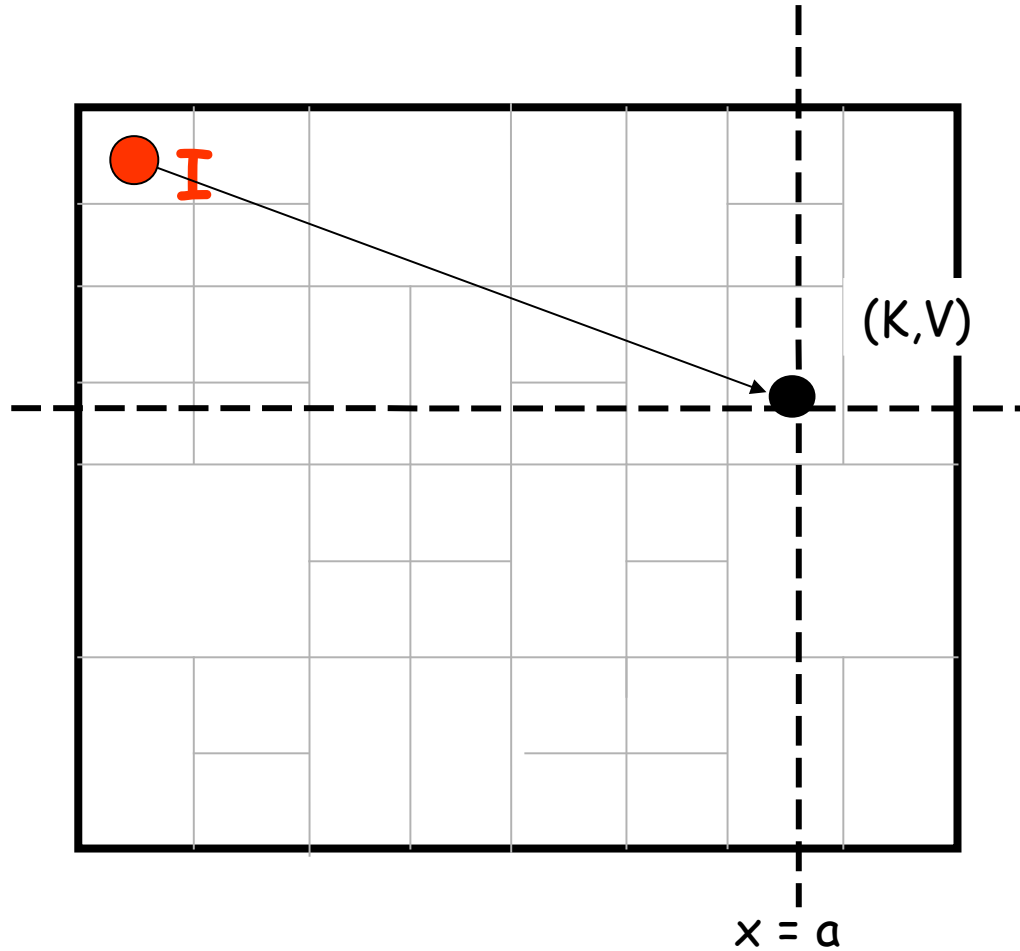
Put (key, data)

node I::insert(K,V)

(1) $a = h_x(K)$
 $b = h_y(K)$

(2) route(K,V) \rightarrow (a,b)

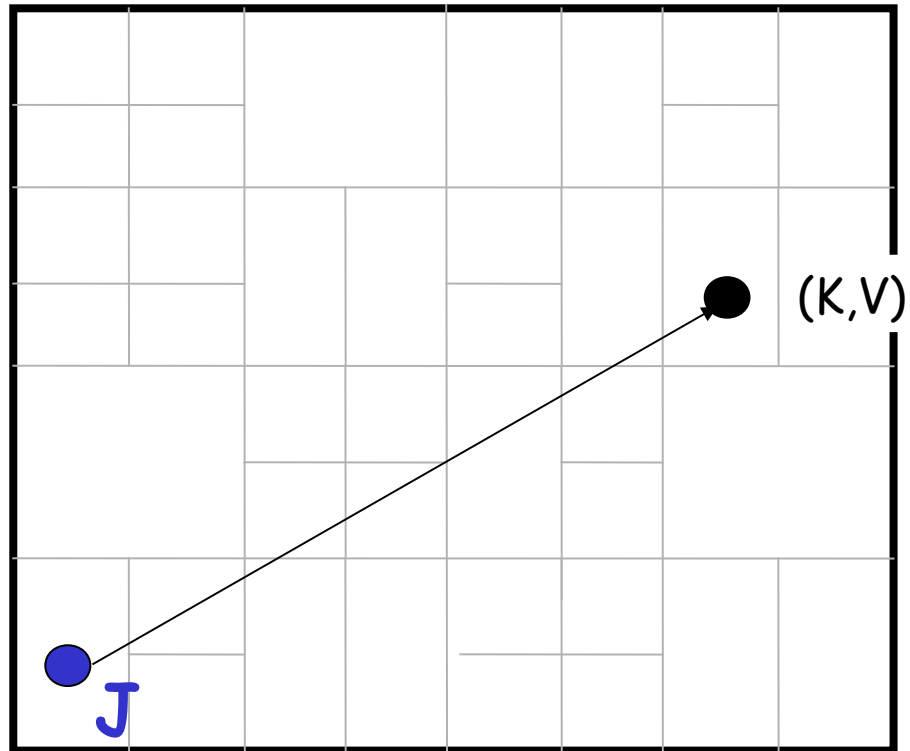
(3) (a,b) stores (K,V)



Get (key)

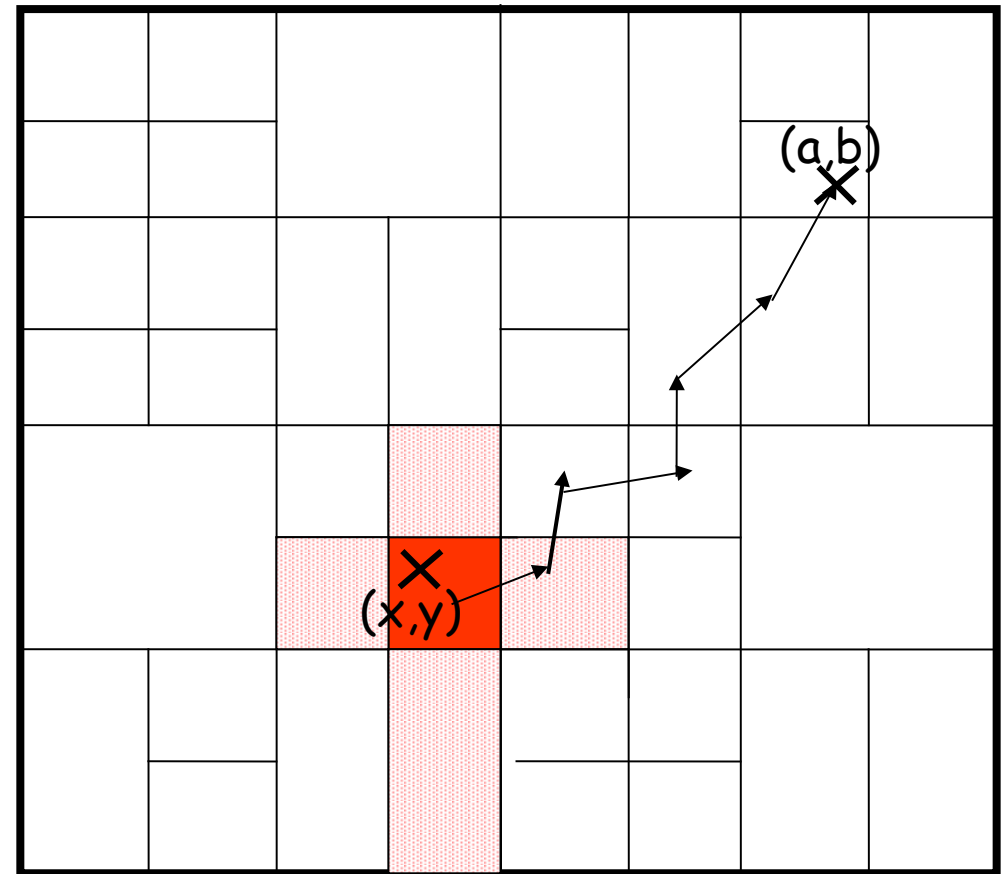
node J::retrieve(K)

(1) $a = h_x(K)$
 $b = h_y(K)$



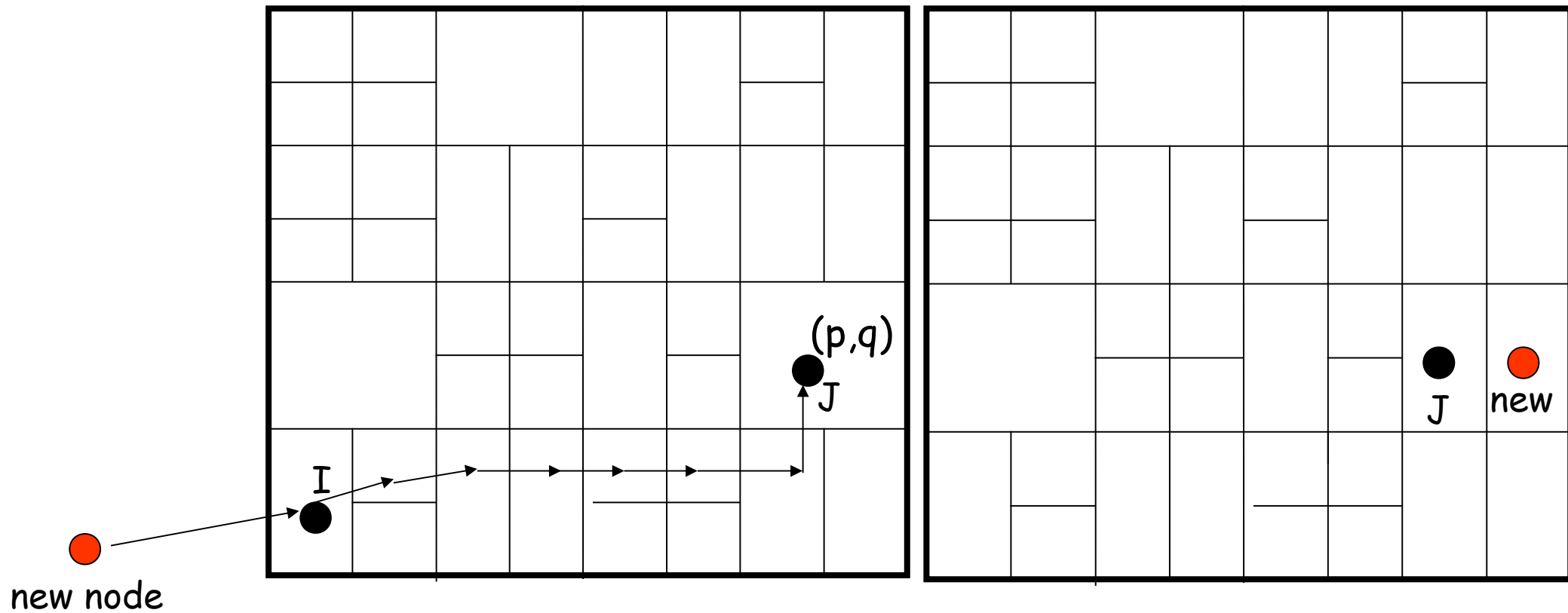
CAN routing

- Neighborhood routing
 - ↳ a node only knows how to reach its neighbors
- For 2-d space
 - ↳ 4 neighbors
 - ↳ or $O(d)$ in general
- Path length
 - ↳ $O(d n^{1/d})$
- Resilience
 - ↳ multi-path



Node join and leave

- Bootstrap node



Comparison

- CAN

- $O(d n^{1/d})$ hops

- $O(d)$ neighbors

- can be independent of n

- Chord

- $O(\log N)$ hops

- $O(\log N)$ neighbors

Next lecture

- Unstructured P2P
 - ↳ [CRBLS03] Yatin Chawathe, S. Ratnasamy, Lee Breslau, Nick Lanham, Scott Shenker, "Making Gnutella-like P2P Systems Scalable", Sigcomm 2003. Gnutella
- Reading list is online
 - ↳ reading summary schedule to be posted soon