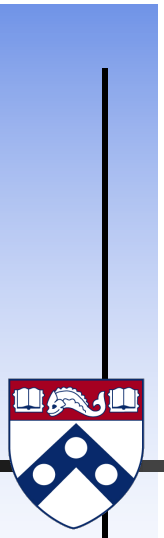


CIS 5530: Networked Systems

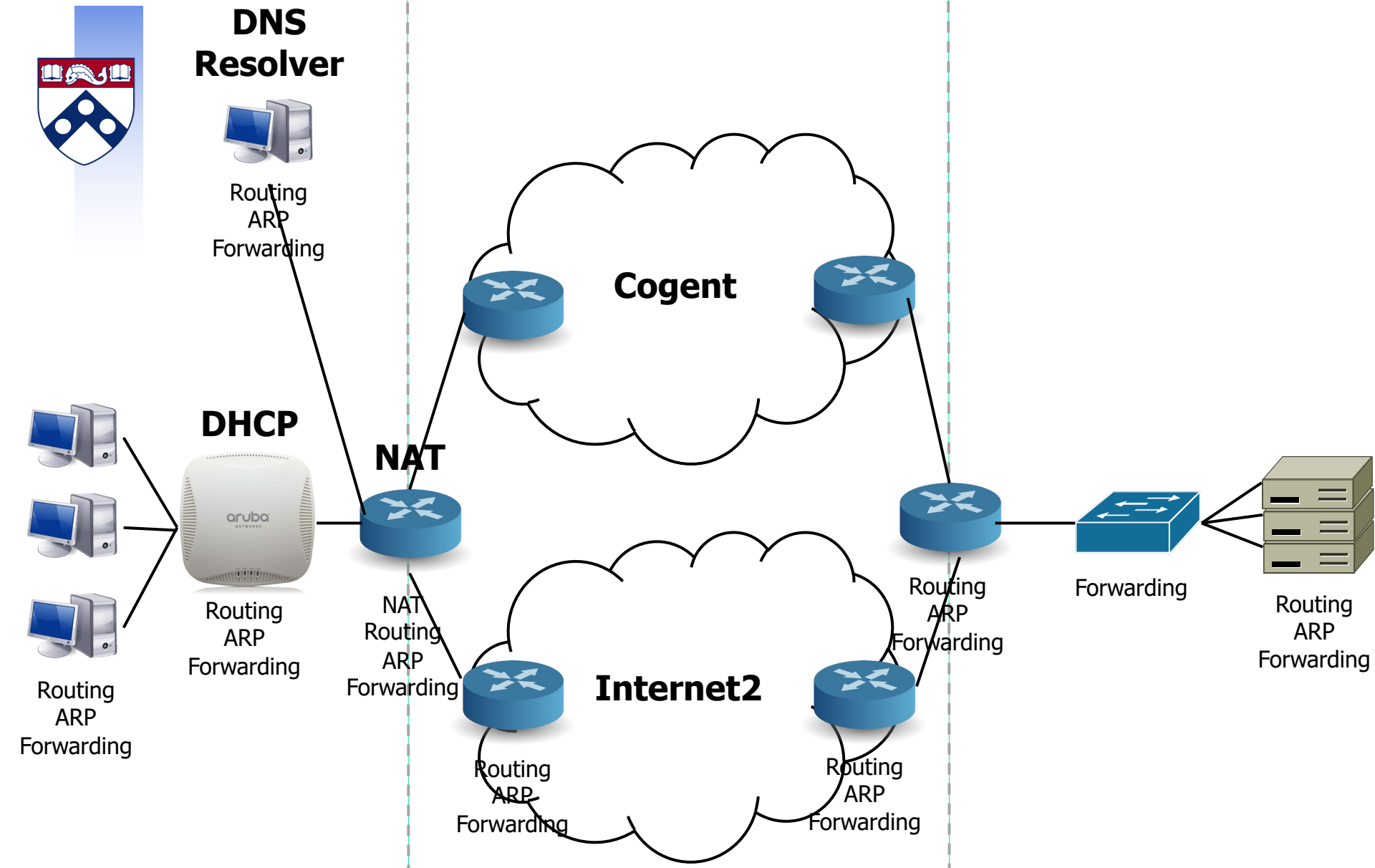
Overlay Networks

February 27, 2023

Agenda

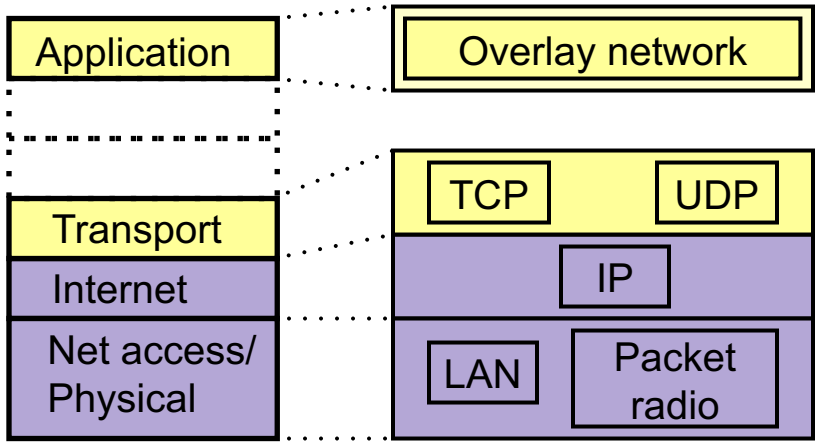
- Putting it all together 
- Overlay Networks 

  - DHTs
  - Content Addressable Networks

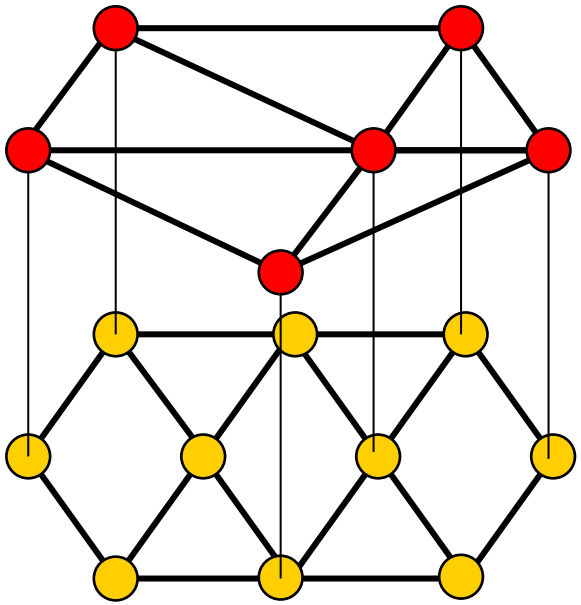

DNS Resolver
Routing
ARP
Forwarding
Cogent
DHCP
NAT
aruba
Routing
ARP
Forwarding
NAT
Routing
ARP
Forwarding
Routing
ARP
Forwarding
Routing
ARP
Forwarding
Internet2
Routing
ARP
Forwarding
Routing
ARP
Forwarding
Forwarding
Routing
ARP
Forwarding

What Is an Overlay?

- "Overlay network"
  - Network layer at the application level
  - Each node is an end-host. Forms a network among end-hosts.
- Examples: Bittorrent, Tor, etc.
- Why overlay networks?
  - Add new functionalities. E.g. route by content rather than addresses, route to many nodes (multicast)
  - Address existing limitations - reliability, security, performance



Overlay Network



Overlay

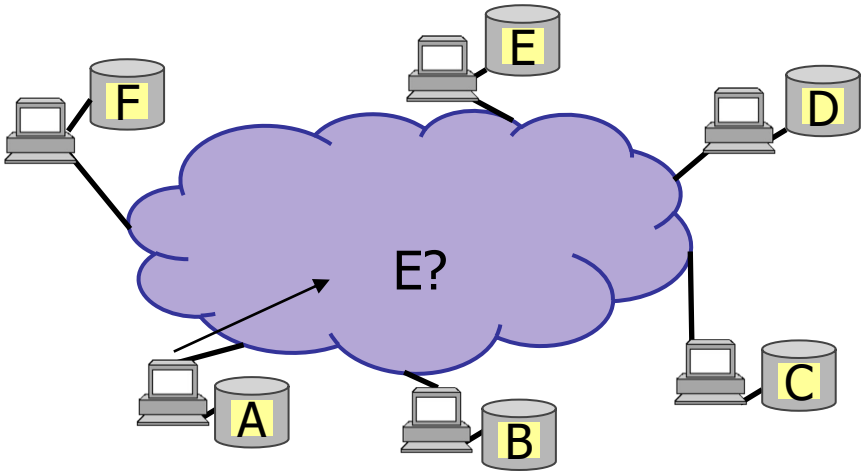
Internet

Overlay API
E.g. route(key), send(group,Msg), etc.

Transport API:
send(Addr:Port, Packet)

Example: P2P Keyword Search

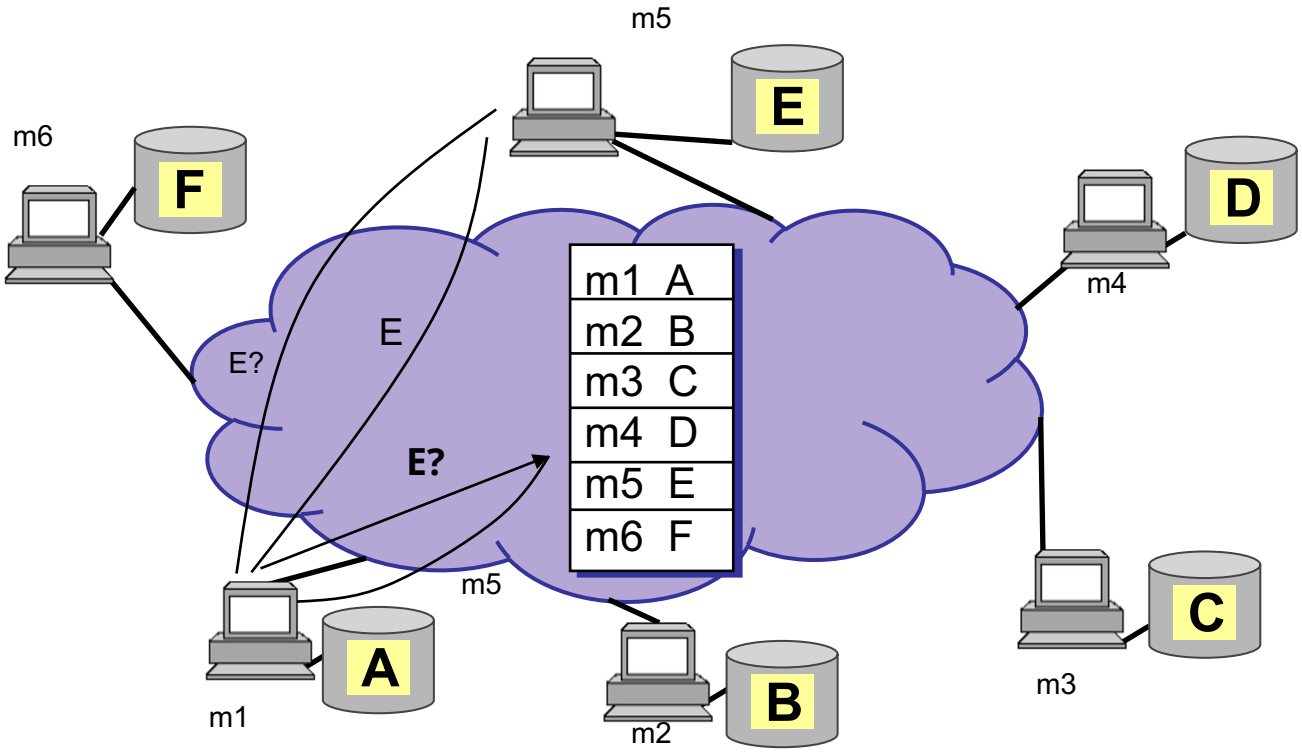
- Each user stores a subset of items (files). Each item has a number of keywords associated with it.
- Given keyword (or keywords), find where a particular item is stored based on its name



Solution 1: Centralized Index

- Example: Napster
- Assume a centralized index system that maps files to machines that
  are alive
- How to find a file
  - Query the index system → return a machine that stores the required file
    - Ideally this is the closest/least-loaded machine
  - Download the file via FTP
- Advantages:
  - Simplicity, easy to implement sophisticated search engines on top of the index system
- Disadvantages:
  - Robustness, scalability

Example

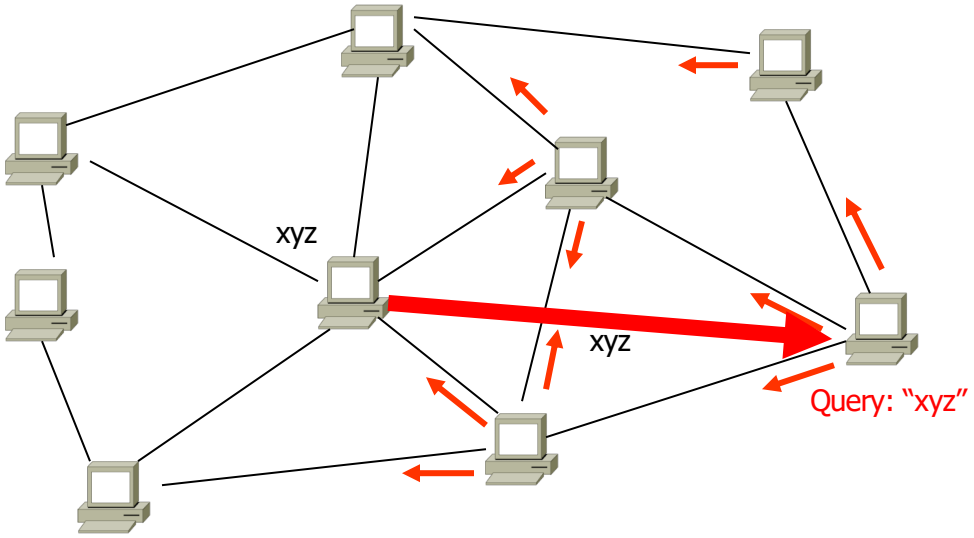


Solution 2: Flood-based Approach

- Example: Gnutella network
- Flood-based protocol:
  - Send request to all neighbors
  - Neighbors recursively forward the request
  - Eventually a machine that has the file receives the request, and it sends back the answer
- Advantages:
  - Totally decentralized, highly robust
- Disadvantages:
  - Not scalable; the entire network can be swamped with request (to alleviate this problem, each request has a TTL)

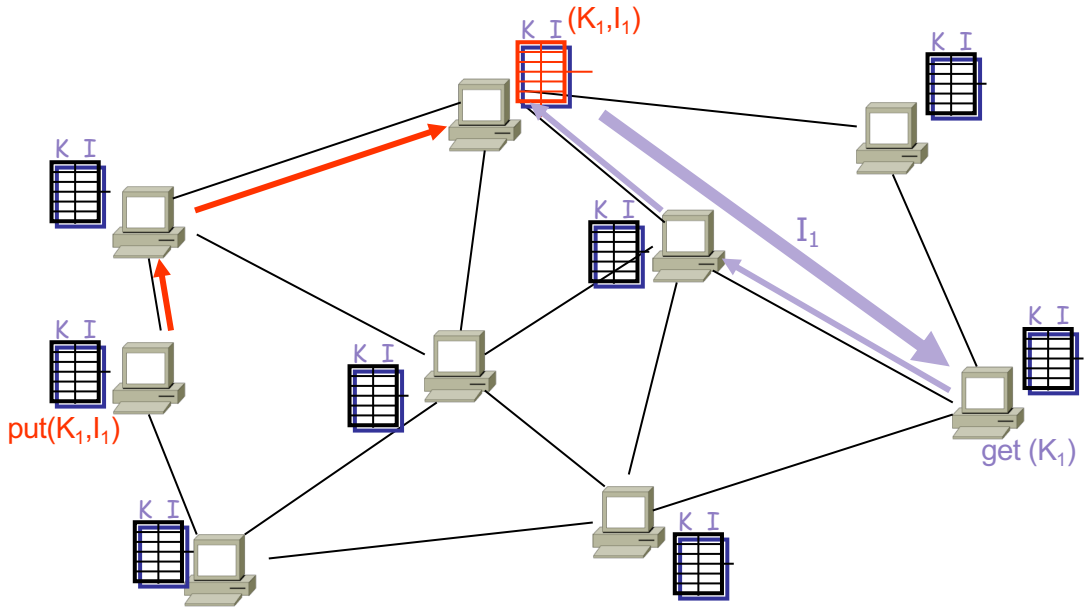
Flood-based Approach

- Ad-hoc topology
- Queries are flooded for bounded number of hops
- No guarantees on recall



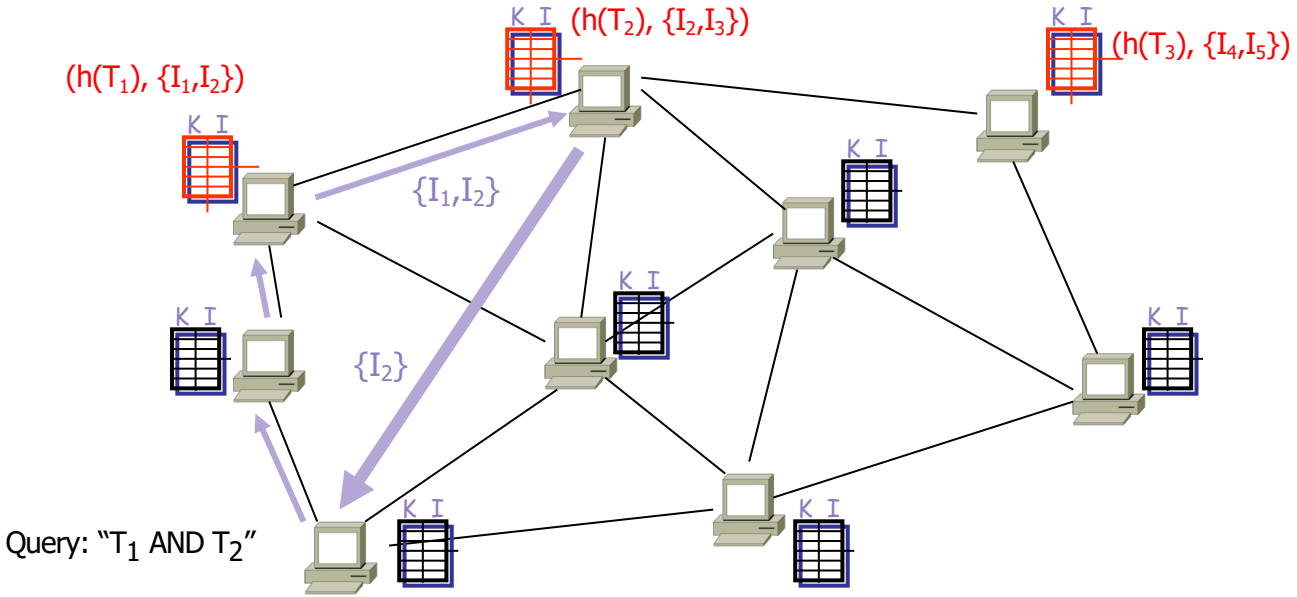
Solution 3: DHT-based Approach

- Distributed Hash Tables (DHTs)
- Hash table interface: put(key,item), get(key)
- O(log n) hops
- Guarantees on recall



Keyword Search Using DHTs

- Inverted Lists hashed by keyword (term) in the DHT



Agenda

- Putting it all together ✓
- Overlay Networks ✓
  - DHTs ✓
  - Content Addressable Networks ← NEXT
  - Chord

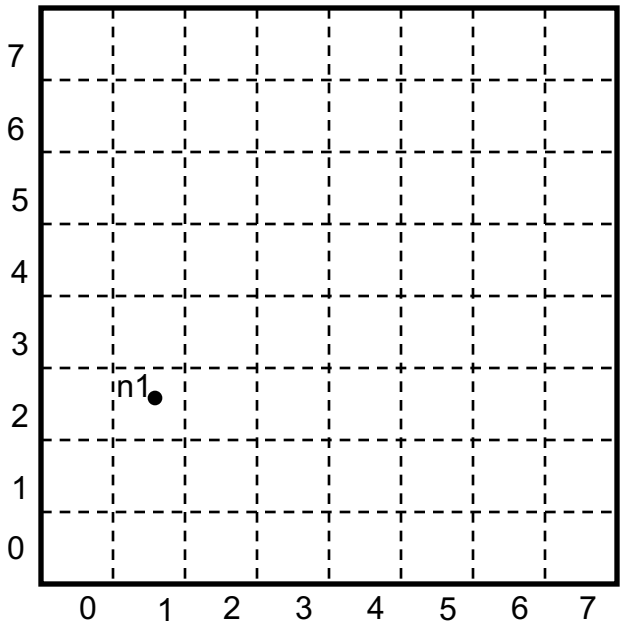
Content Addressable Network (CAN)

- Associate to each node and item a unique id in an d-dimensional Coordinate space

- Properties
  - Routing table size O(d)
  - Guarantees that a file is found in at most d*n1/d steps, where n is the total number of nodes

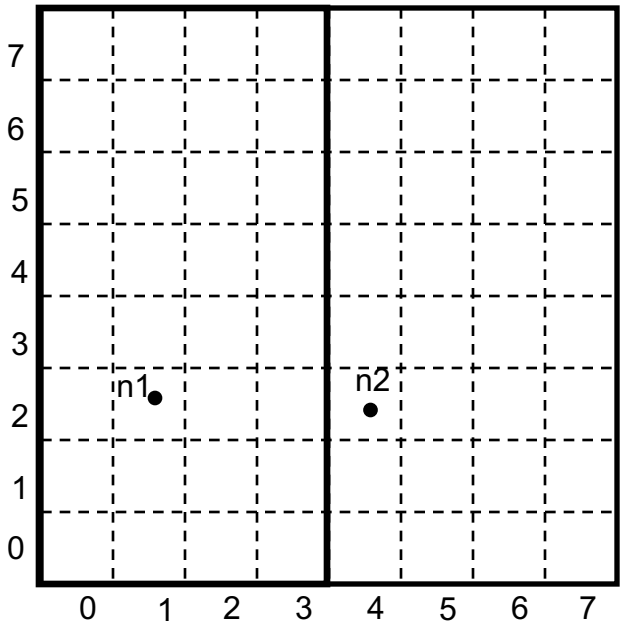
CAN Example: Two Dimensional Space

- Space divided between nodes
- All nodes cover the entire space
- Each node covers either a square or a rectangular area of ratios 1:2 or 2:1
- Example:
  - Node n1:(1, 2) first node that joins → cover the entire space



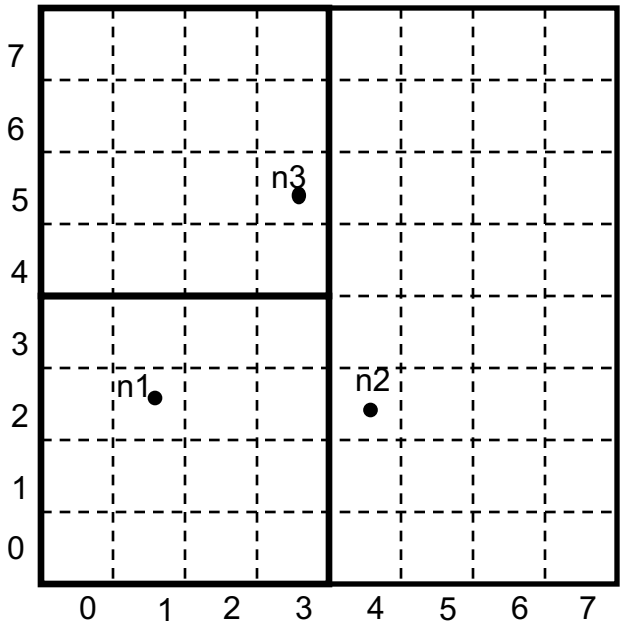
CAN Example: Two Dimensional Space

- Node n2:(4, 2) joins → space is divided between n1 and n2



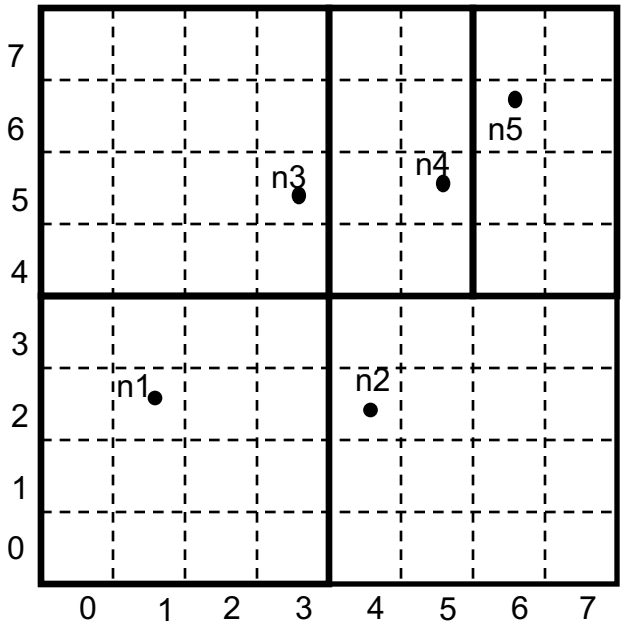
CAN Example: Two Dimensional Space

- Node n3:(3, 5) joins → space is divided between n1 and n3



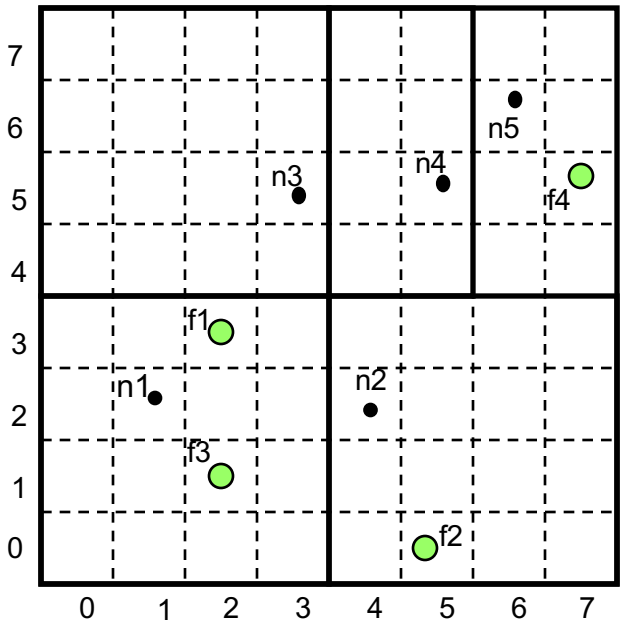
CAN Example: Two Dimensional Space

- Nodes n4:(5, 5) and n5:(6,6) join



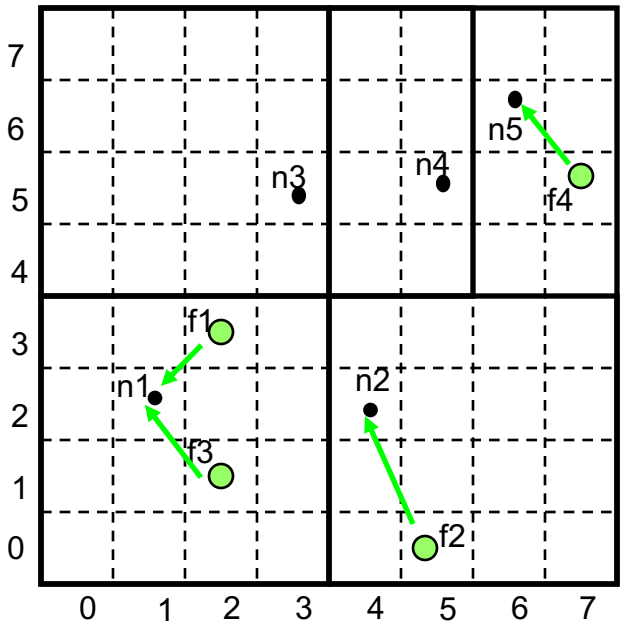
CAN Example: Two Dimensional Space

- Nodes: n1:(1, 2); n2:(4,2); n3:(3, 5); n4:(5,5);n5:(6,6)
- Items: f1:(2,3); f2:(5,1); f3:(2,1); f4:(7,5);



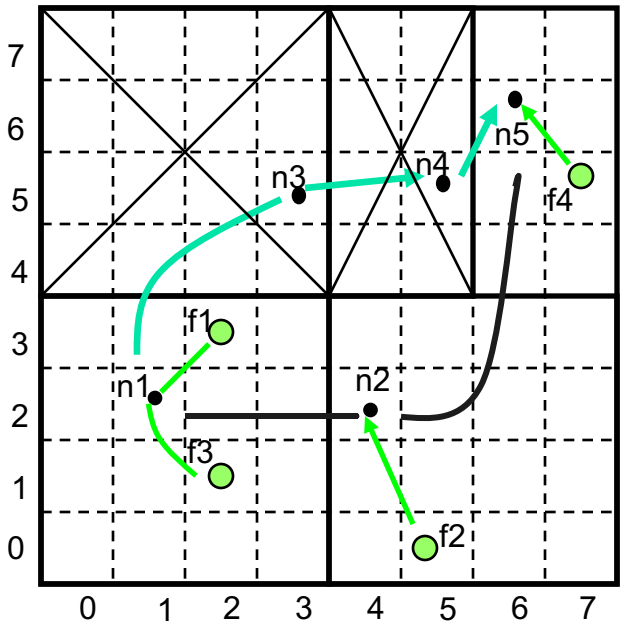
CAN Example: Two Dimensional Space

- Each item is stored by the node who owns its mapping in the space

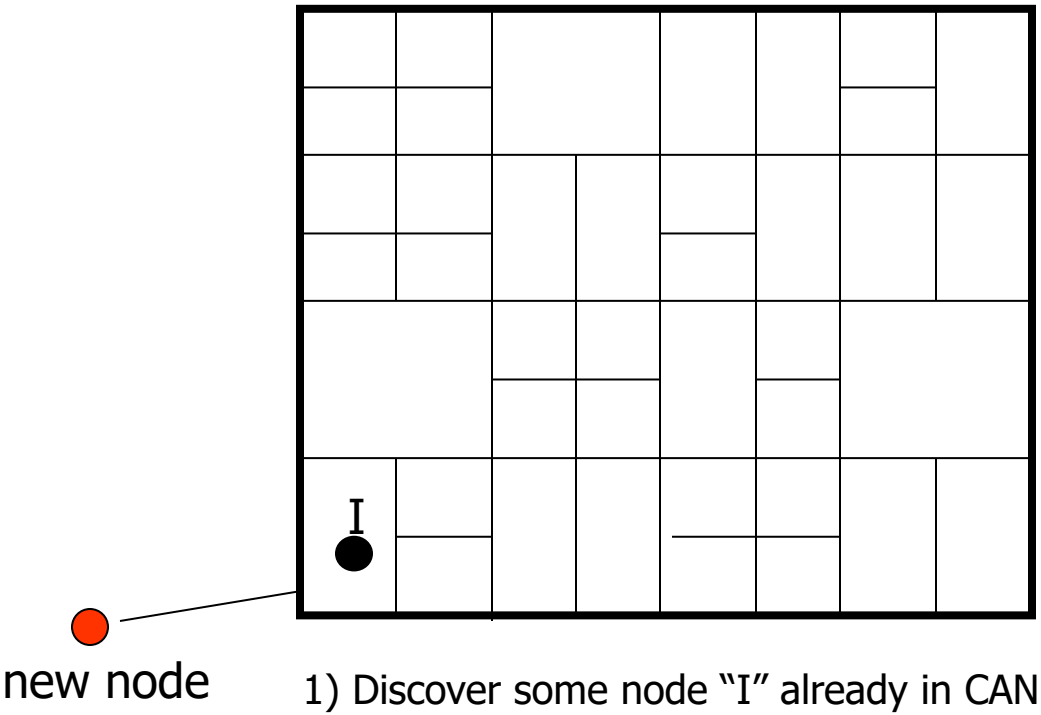


CAN: Query Example

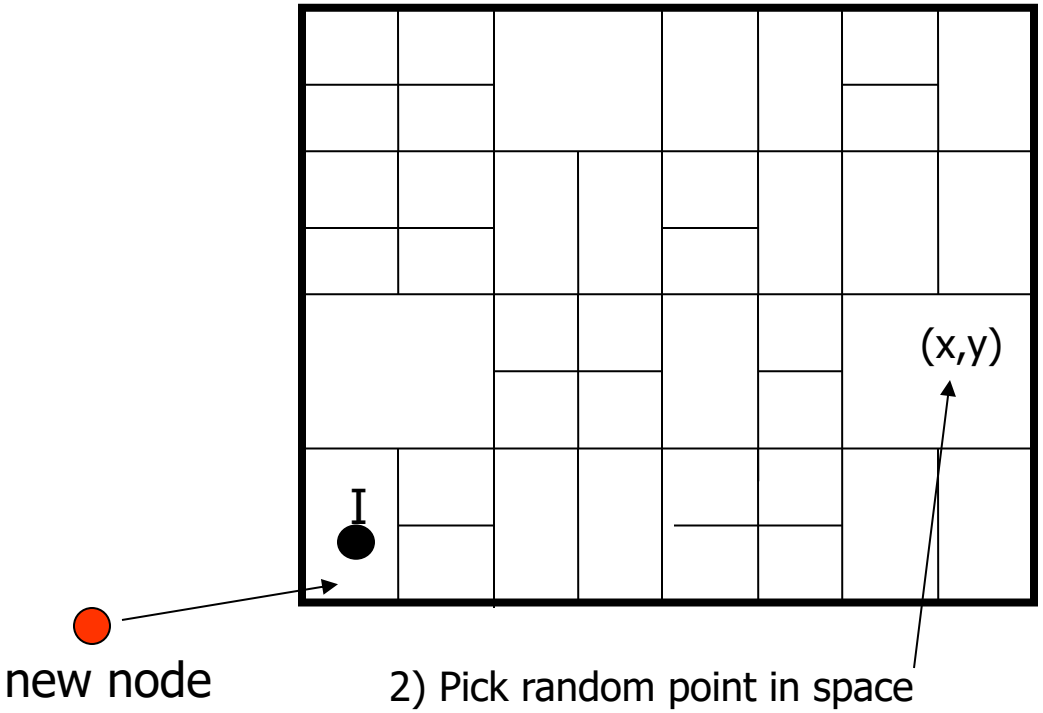
- Each node knows its neighbors in the d-space
- Forward query to the neighbor that is closest to the query id
- Example: assume n1 queries f4
- Can route around some failures



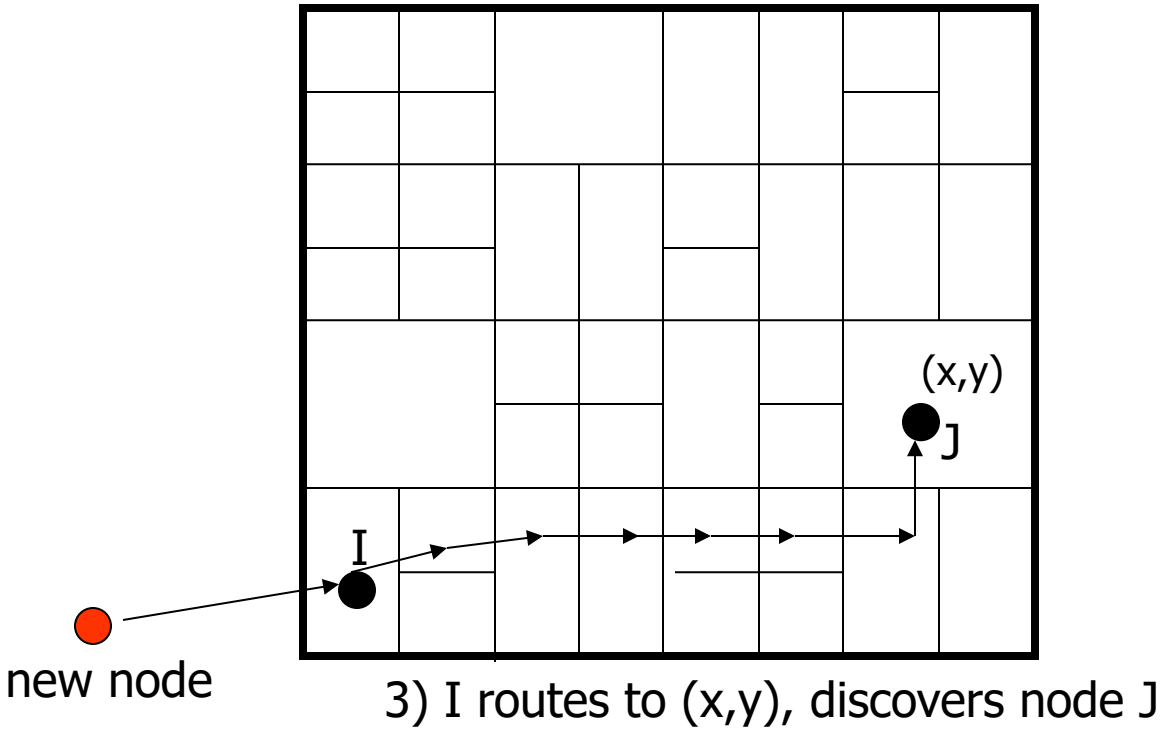
CAN: Node Joining



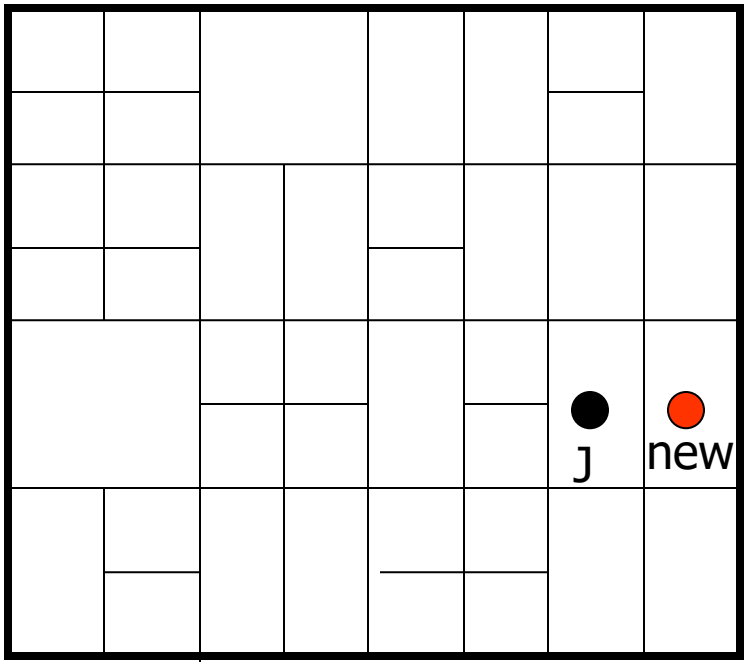
CAN: Node Joining



CAN: Node Joining



CAN: Node Joining



4) split J's zone in half… new node owns one half

Node Departure

- Node explicitly hands over its zone and the associated (key,value) database to one of its neighbors
- In case of network failure this is handled by a take-over algorithm
- Problem : take over mechanism does not provide regeneration of data
- Solution: every node has a backup of its neighbours

Chord

- Associate to each node and item a unique id in an uni-dimensional space 0..2m-1
- Key design decision
  - Decouple correctness from efficiency
- Properties
  - Routing table size O(log(N)) , where N is the total number of nodes
  - Guarantees that a item is found in O(log(N)) steps using its key

Review: Simple Hashing

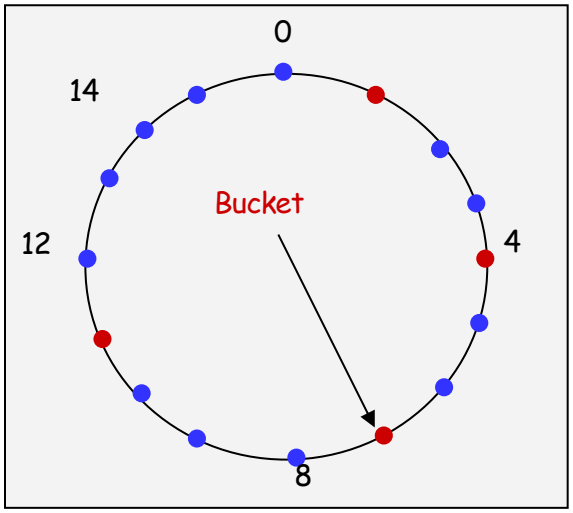
- Given document XYZ, we need to choose a server to use
- Suppose we use modulo
- Number servers from 1…n
  - Place document XYZ on server (XYZ mod n)
  - What happens when a servers fails? n → n-1
  - Why might this be bad?

Consistent Hash [Karger 97]

- David Karger, et al. Consistent Hashing and Random Trees: Tools for Relieving Hot Spots on the World Wide Web. STOC '97.
- Used by Akamai CDN for load balancing
- Desired features
  - Balanced – load is equal across buckets
  - Smoothness – little impact on hash bucket contents when buckets are added/removed

Consistent Hash [Karger 97]

- Construction
  - Assign each of C hash buckets to random points on mod 2n circle, where, hash key size = n.
  - Map object to random position on circle
  - Hash of object = closest clockwise bucket



- Smoothness → addition of bucket does not cause movement between existing buckets

- Balance → no bucket is responsible for large number of objects