CSC2231: DHT Geometries + P2P Replication

http://www.cs.toronto.edu/~stefan/courses/csc2231/05au

Stefan Saroiu

Department of Computer Science

University of Toronto

Question for you?

Outline

- **New DHT designs: CAN and Viceroy**
- **Flexibility as a DHT design requirement**
- **File-sharing replication**
  - Do current schemes work?
  - Could they work?

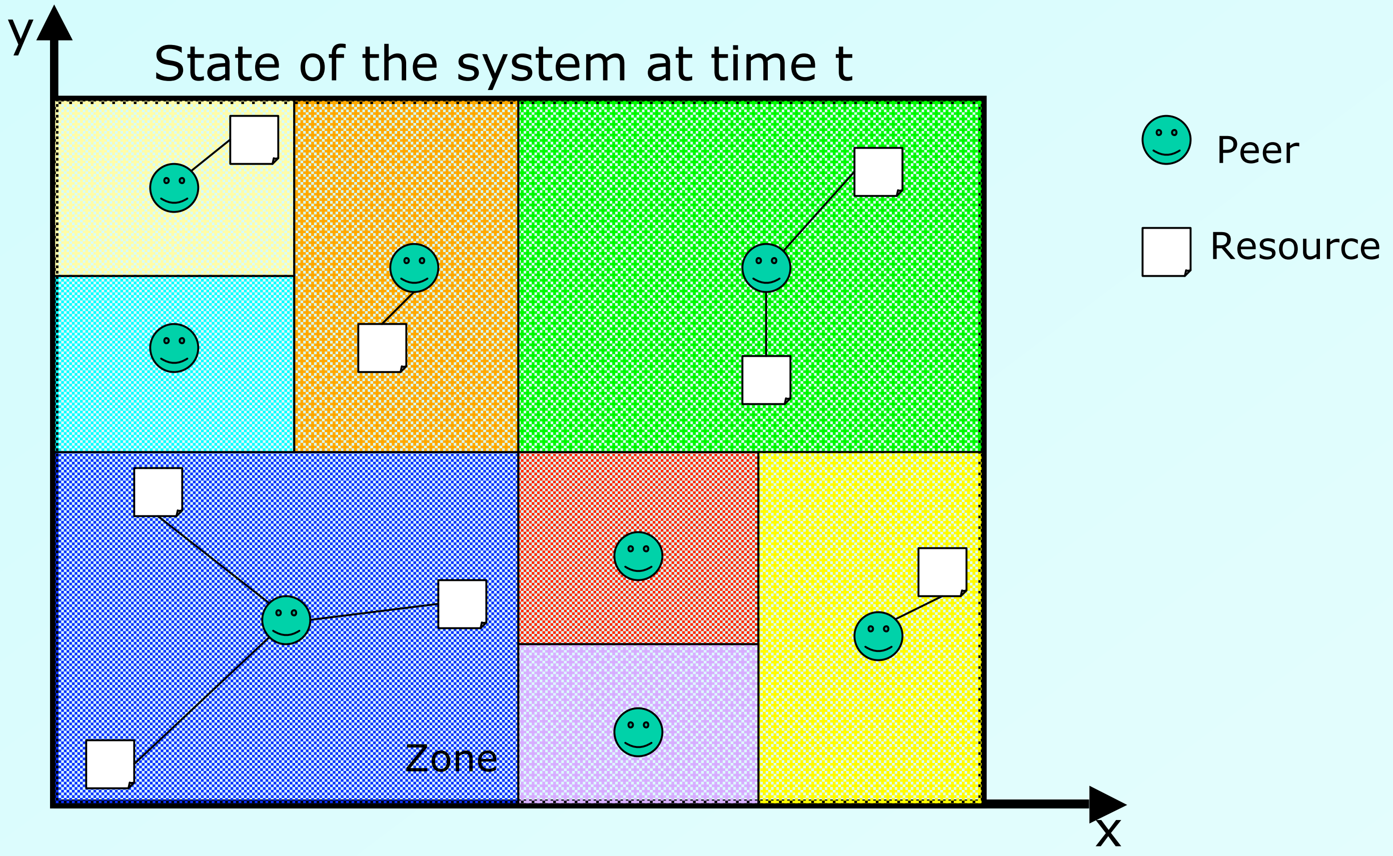
Outline

- **New DHT designs: CAN and Viceroy**
- **Flexibility as a DHT design requirement**
- **File-sharing replication**
  - Do current schemes work?
  - Could they work?

More DHTs

- **CAN:**
  - Route selection flexibility
  - Neighbor selection flexibility
- **Butterfly:**
  - Route selection flexibility
  - Neighbor selection flexibility

CAN at a High-Level

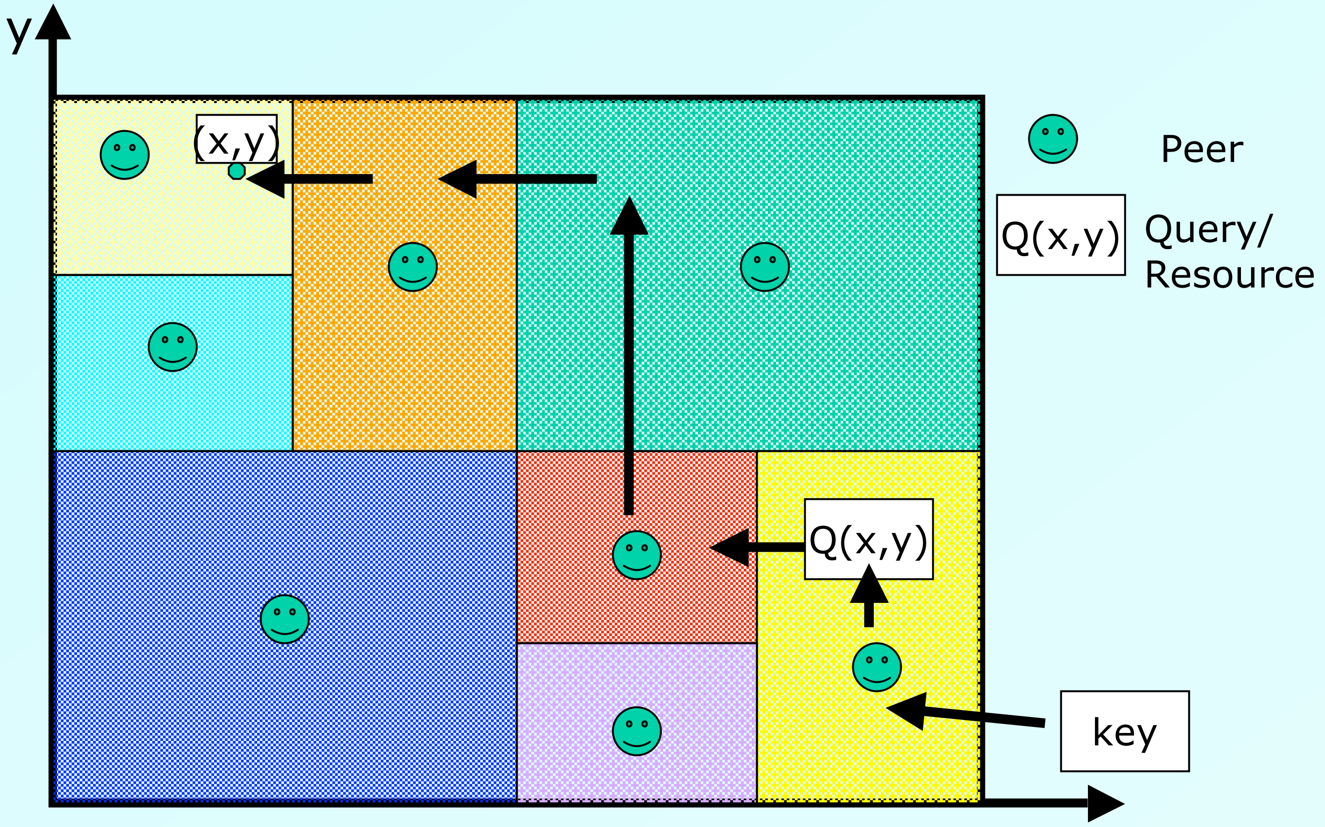


In this 2 dimensional space a key is mapped to a point (x,y)

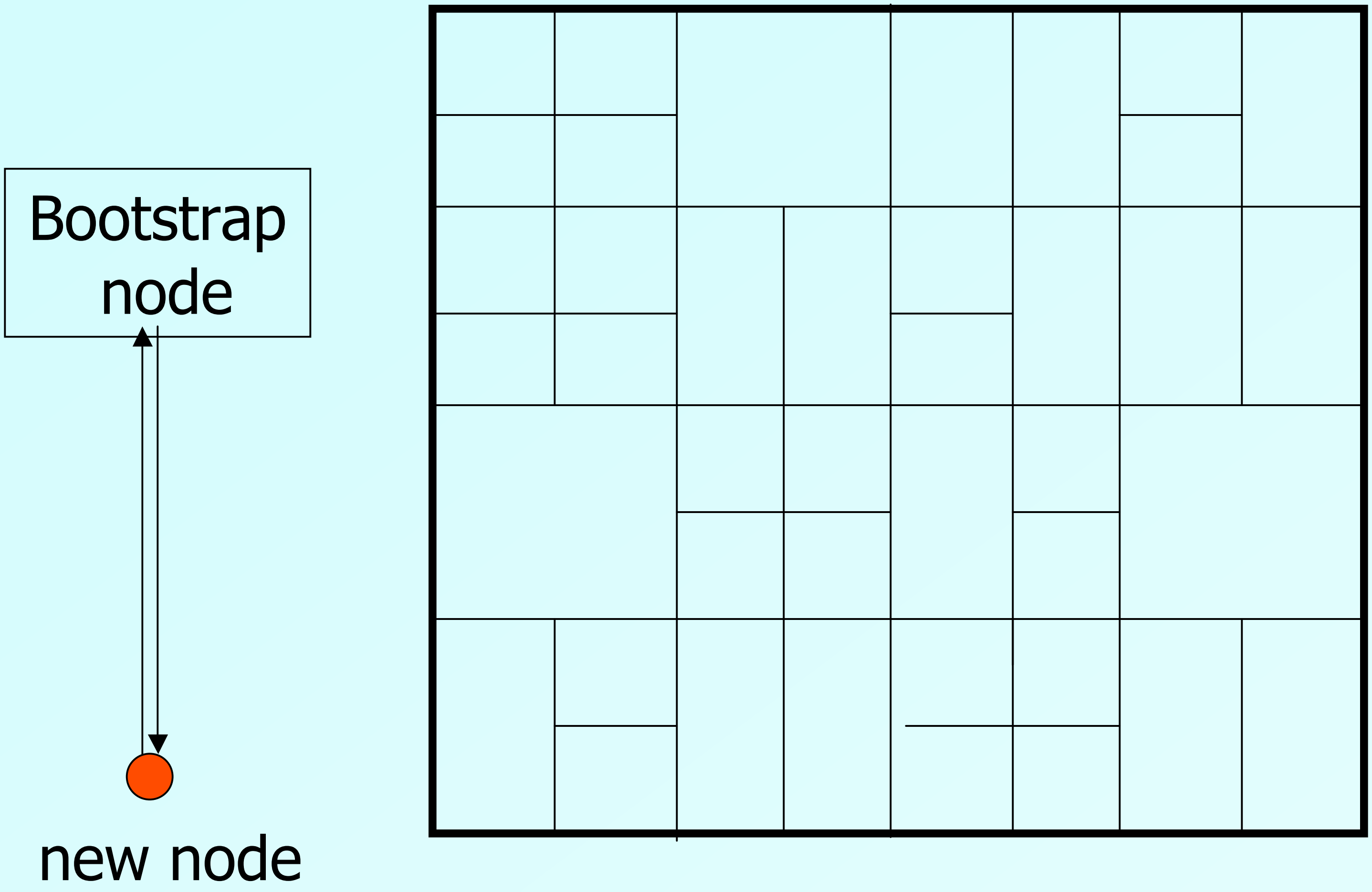
Routing

- □ d-dimensional space with n zones
- □2 zones are neighbor if d-1 dim overlap
- □Routing path of length: $(d/4)n^{1/d}$
- □Algorithm:

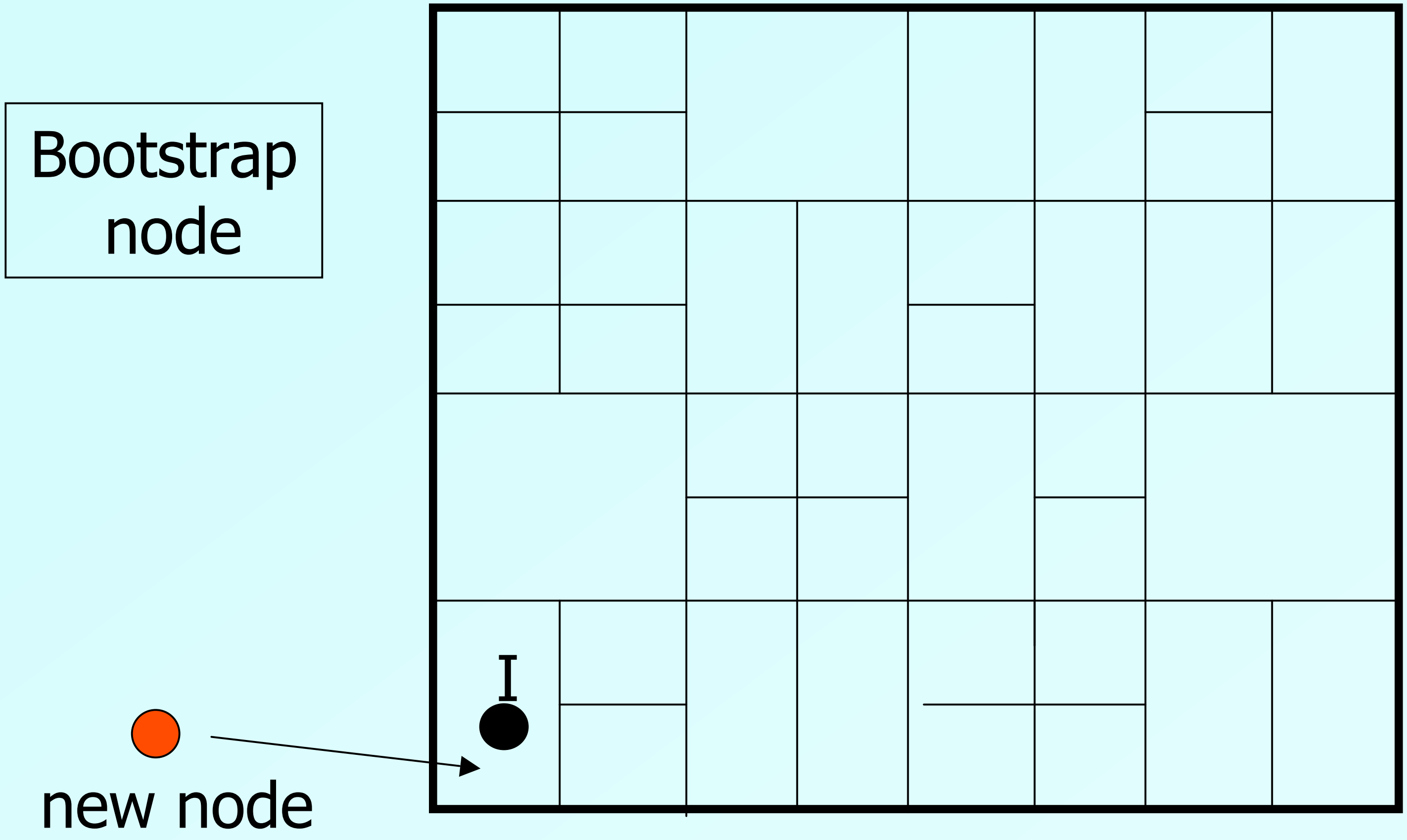
  Choose the neighbor nearest to the destination



CAN: construction



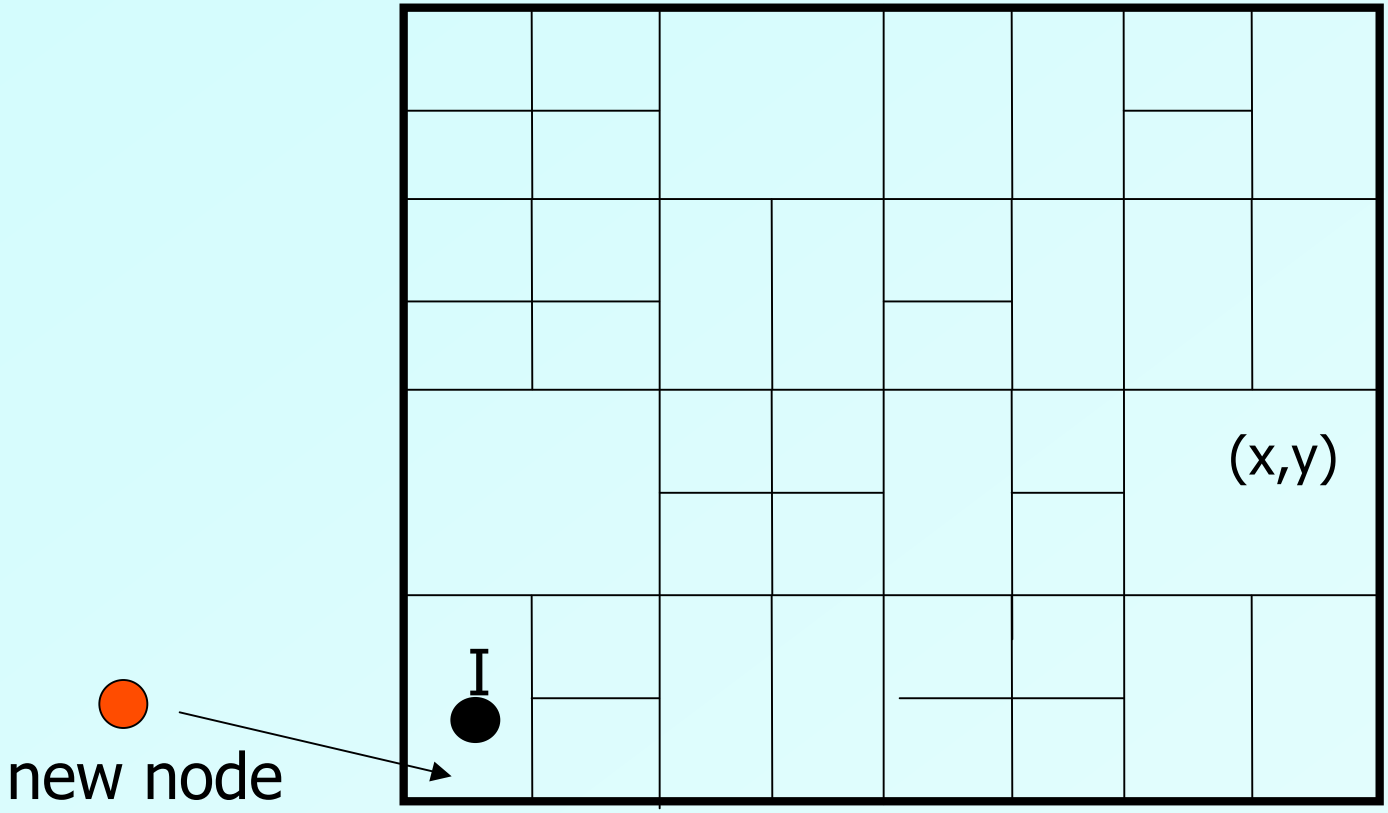
CAN: construction



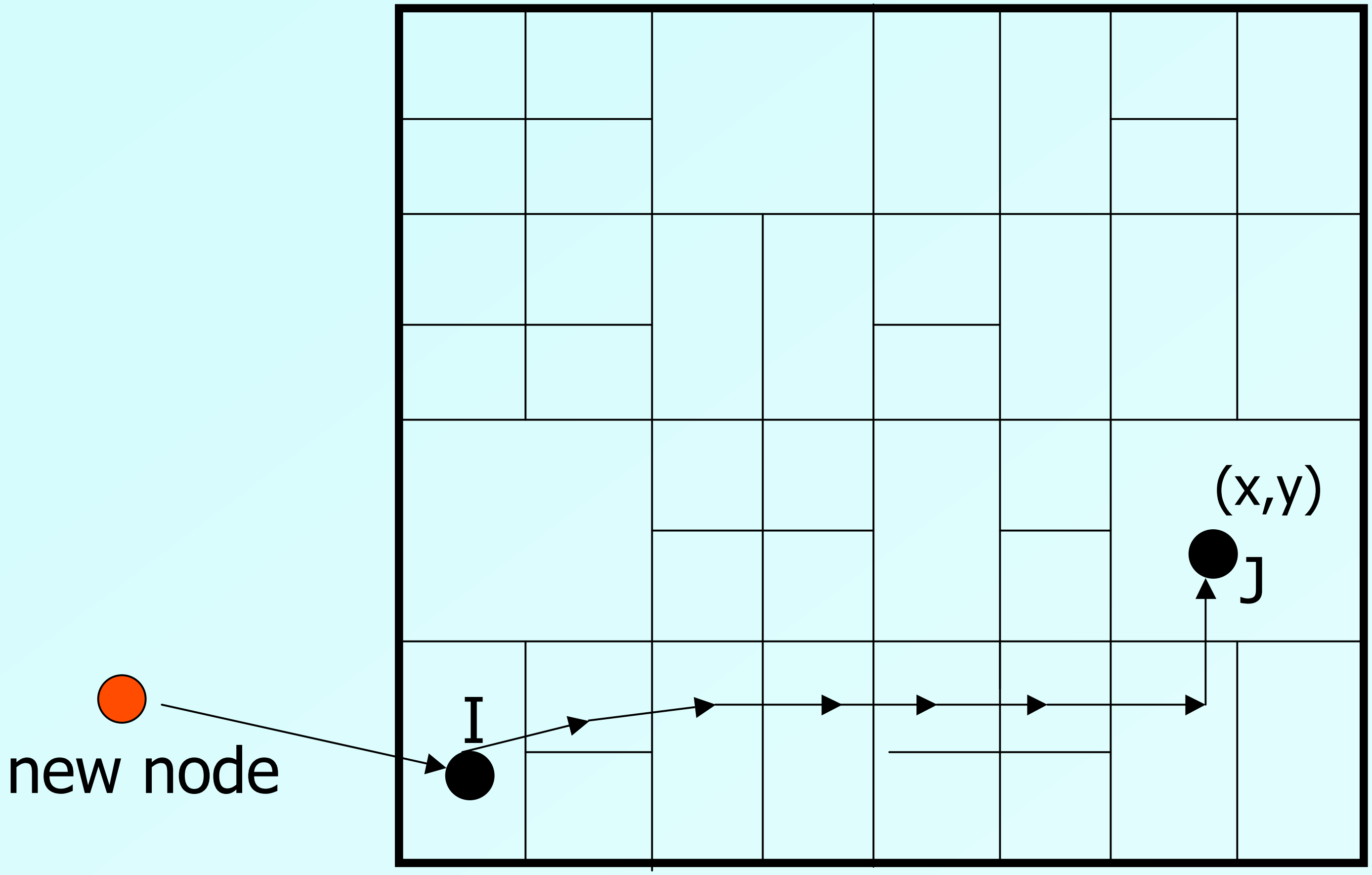
Bootstrap node

new node

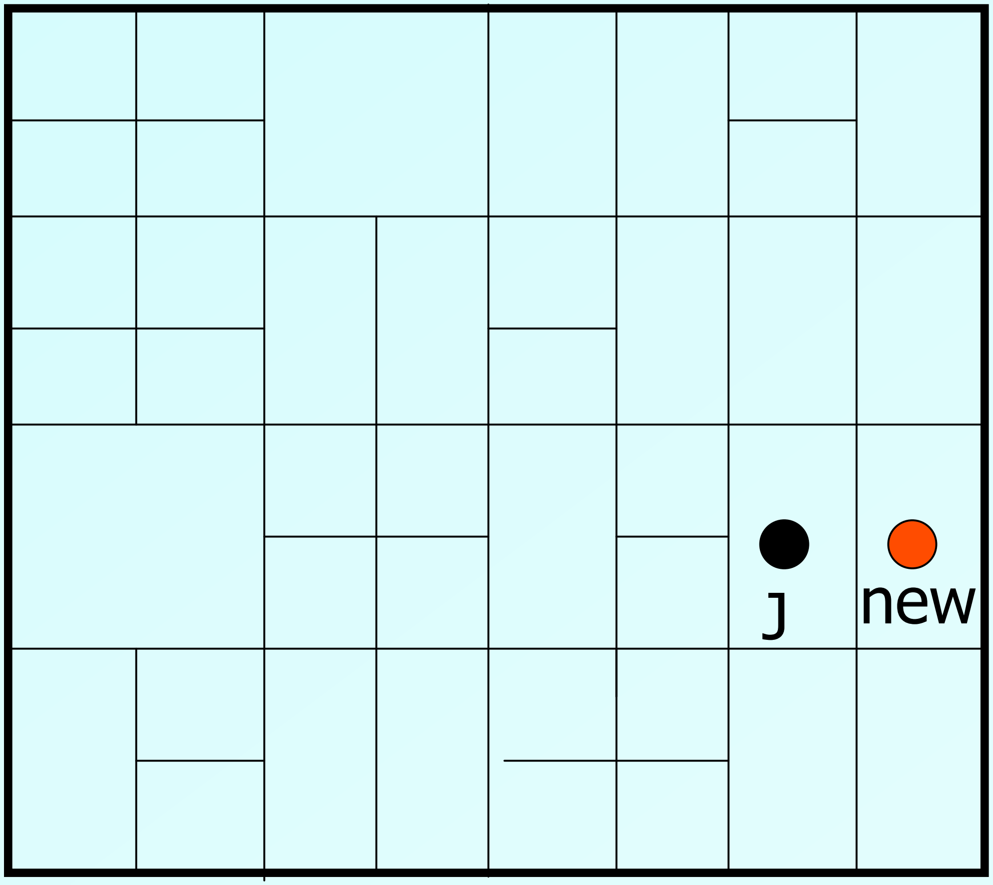
CAN: construction



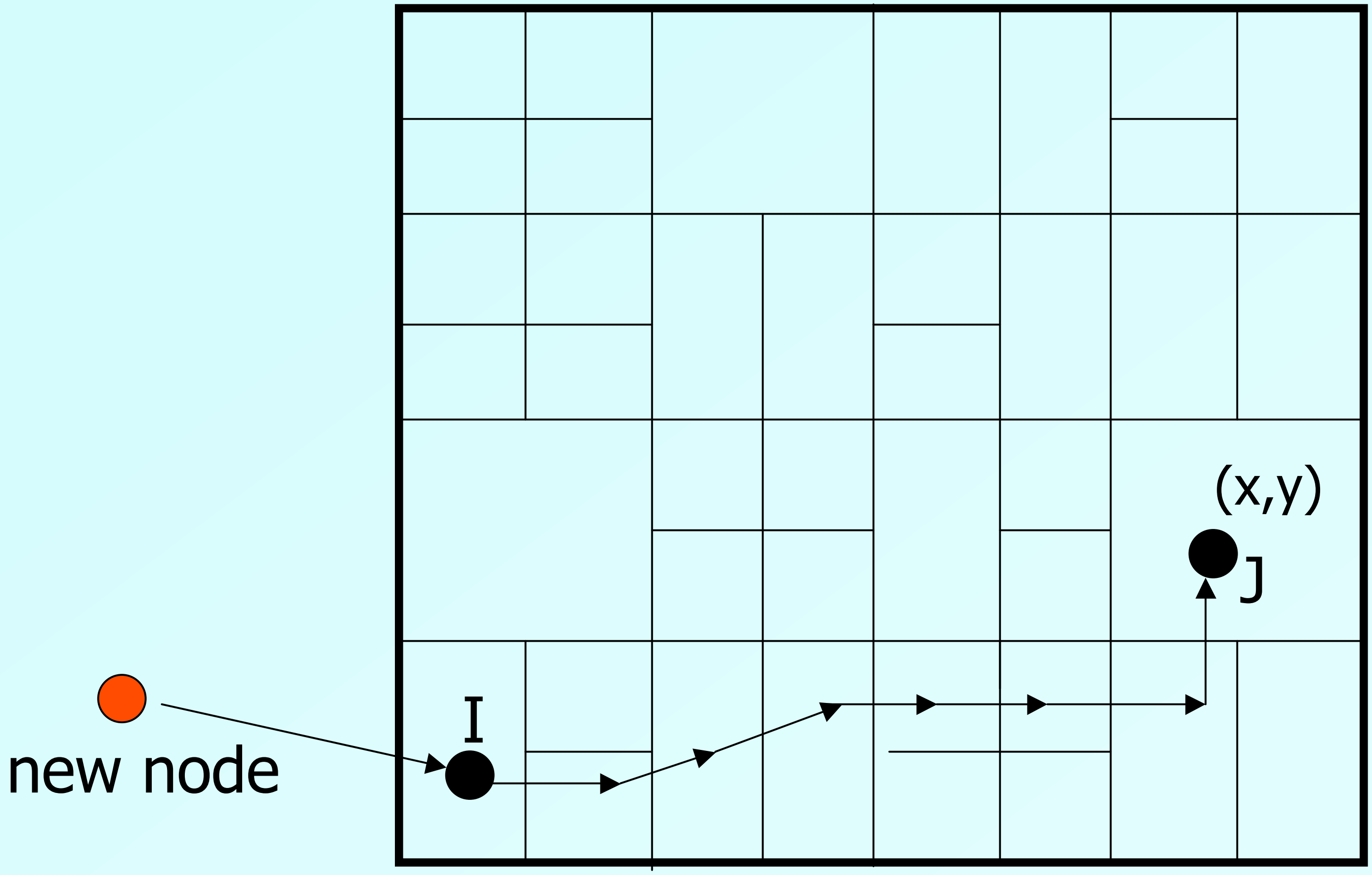
CAN: construction



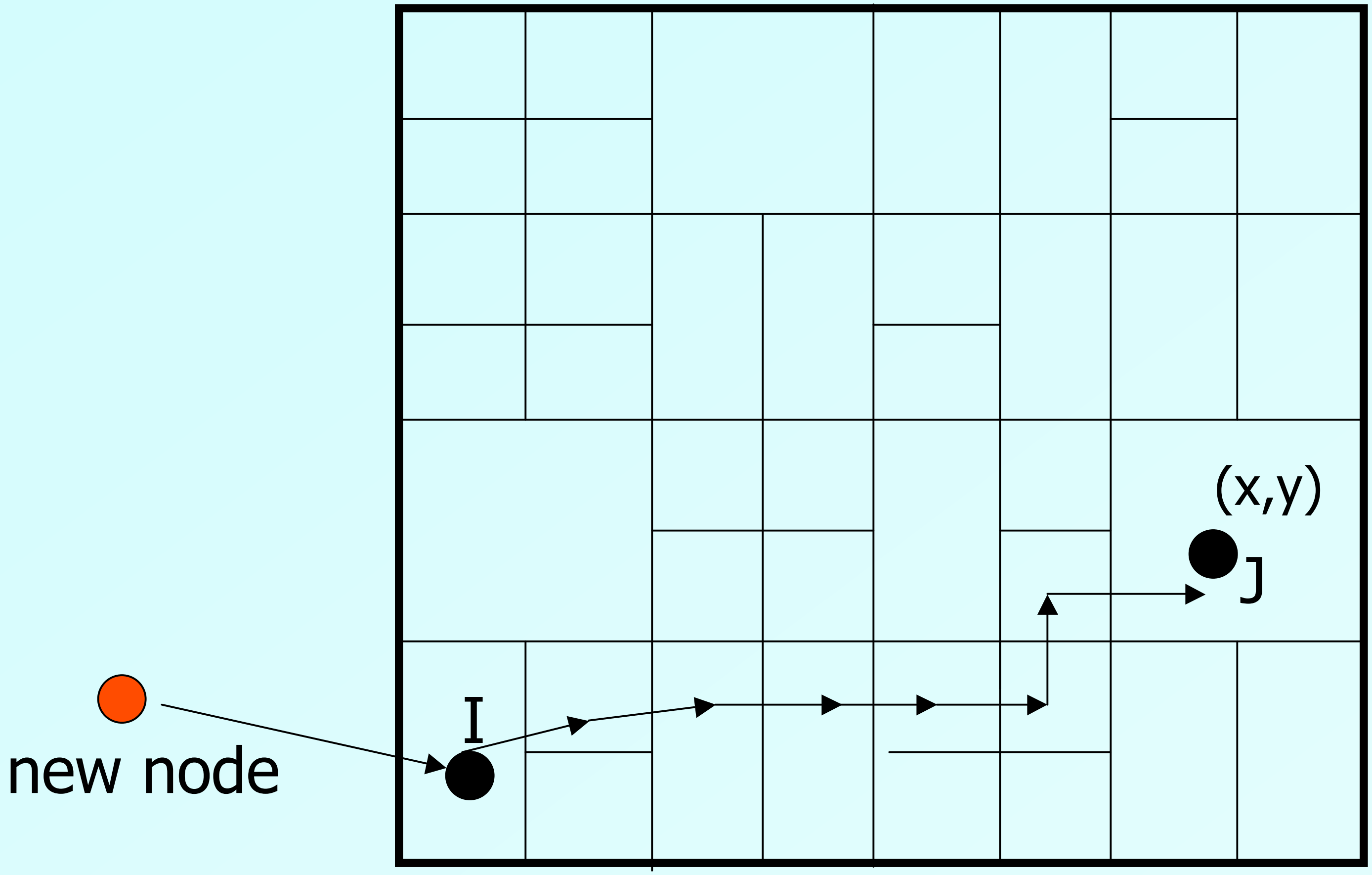
CAN: construction



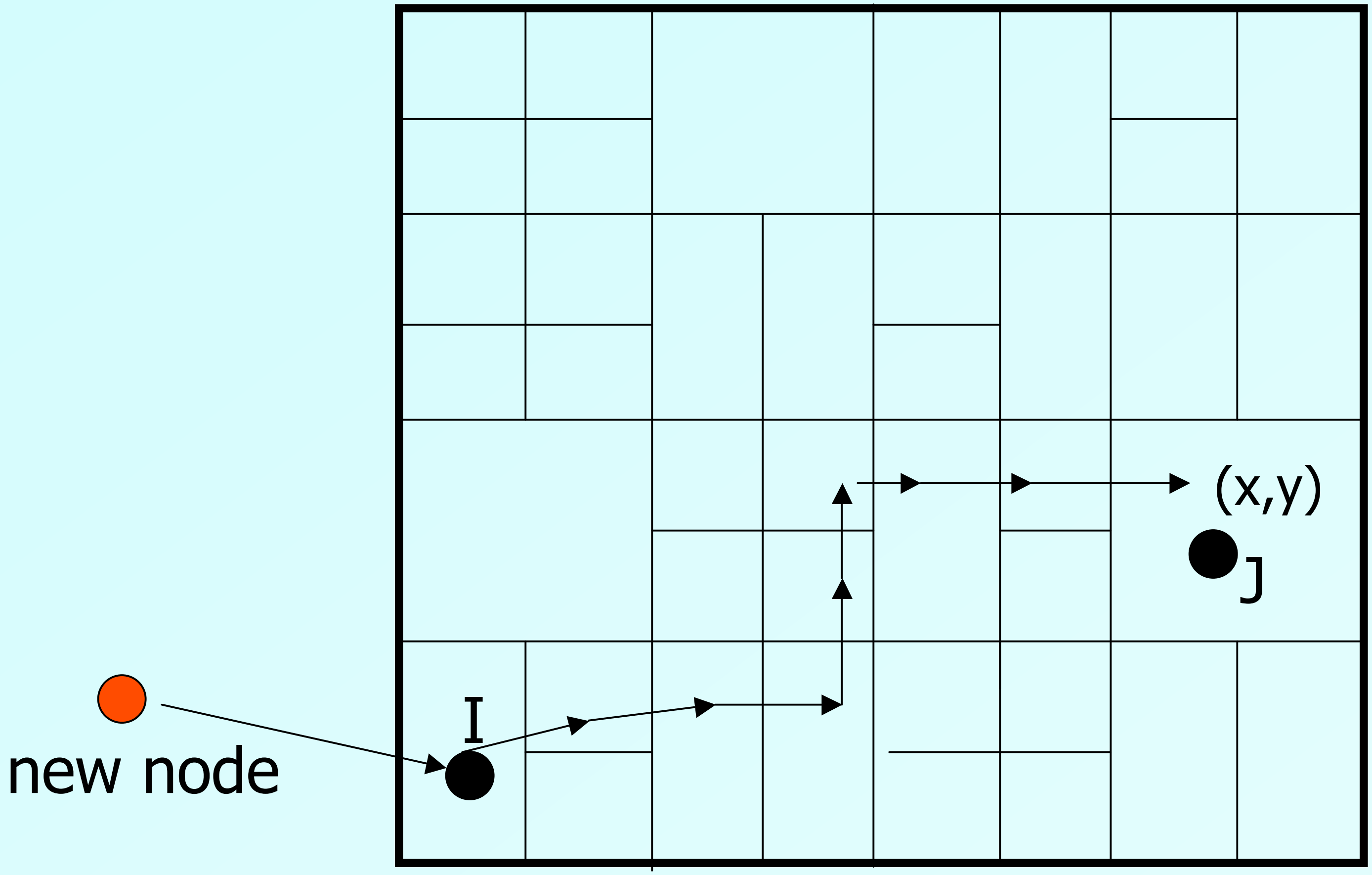
CAN: route selection flexibility



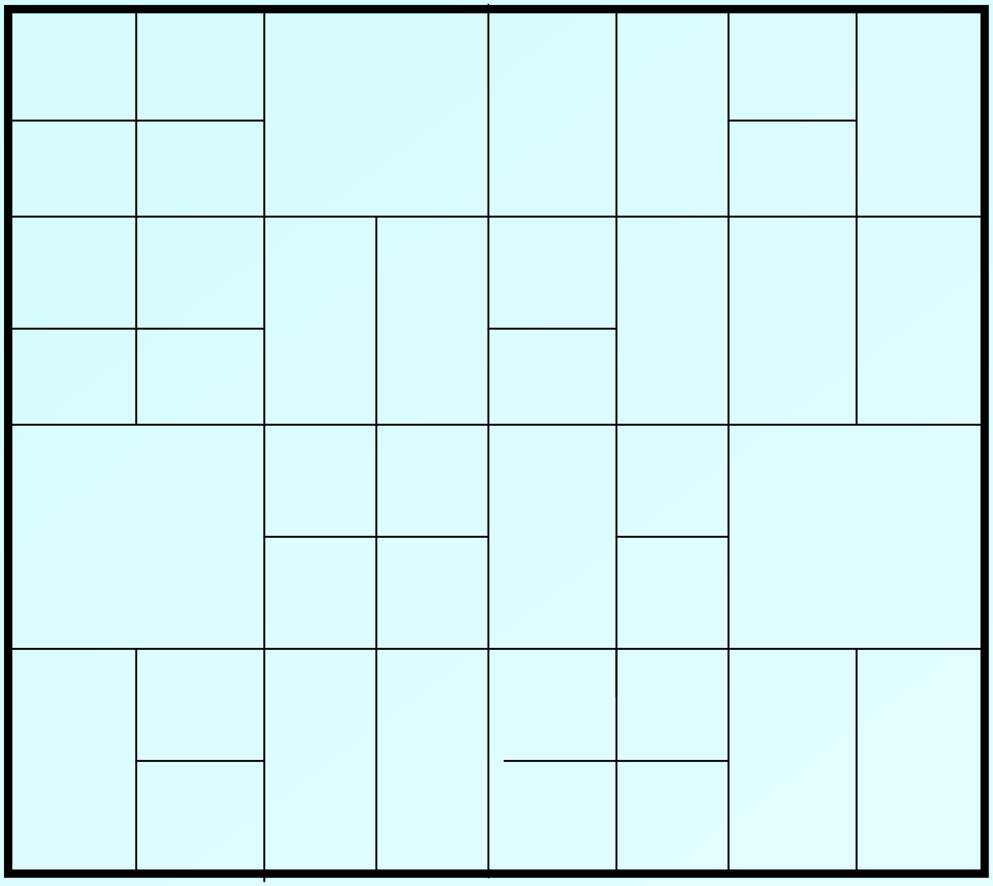
CAN: route selection flexibility



CAN: route selection flexibility



CAN: neighbor selection flexibility

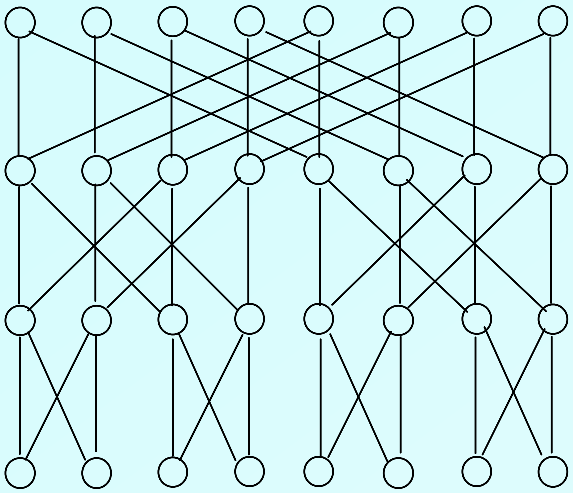


More DHTs

- **CAN:**
  - Route selection flexibility: great!
  - Neighbor selection flexibility: poor!
- **Butterfly:**
  - Route selection flexibility
  - Neighbor selection flexibility

Viceroy

- **Emulating the butterfly network**



level 1

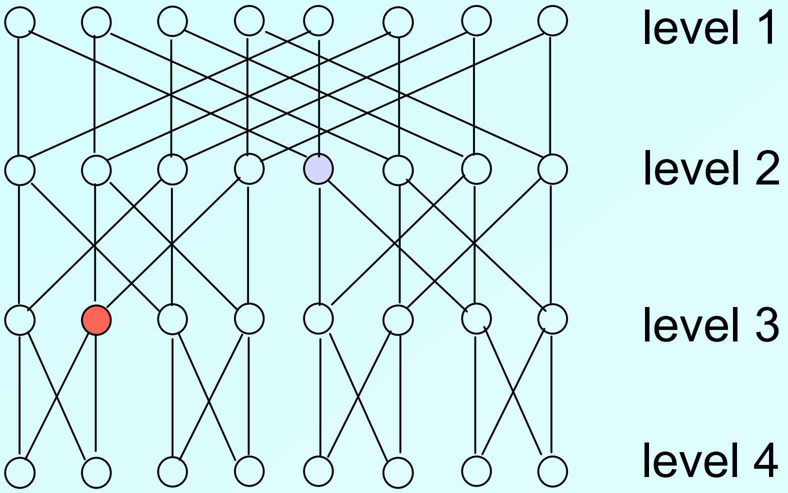
level 2

level 3

level 4

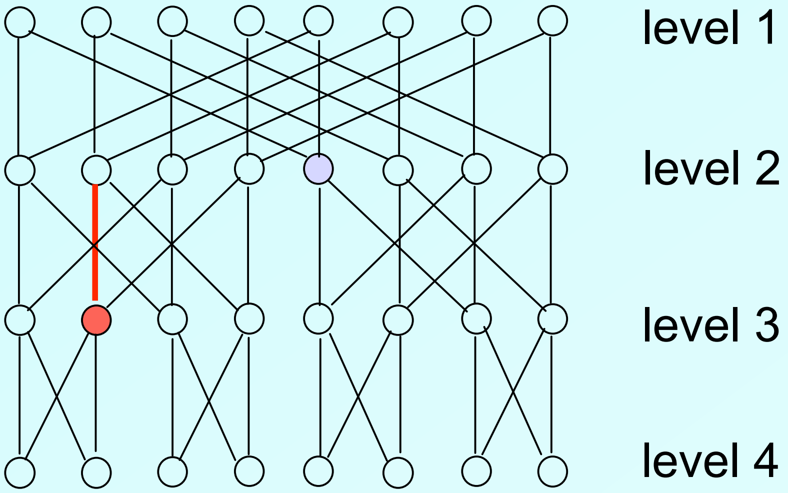
Viceroy

- **Emulating the butterfly network**



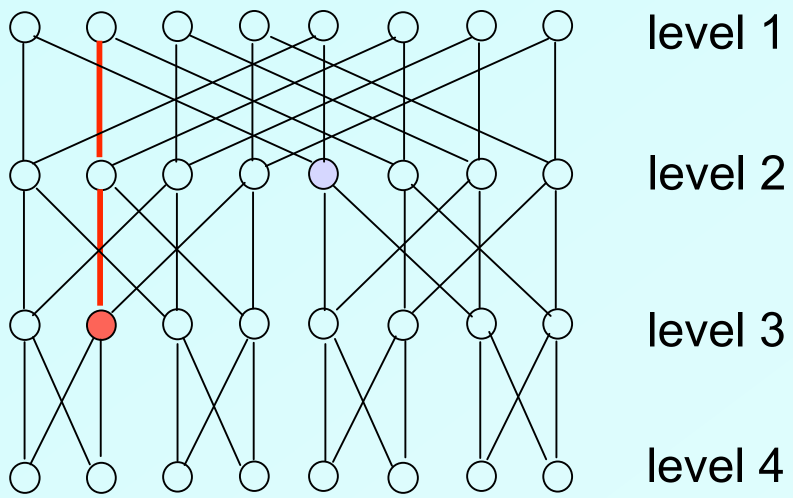
Viceroy

- **Emulating the butterfly network**



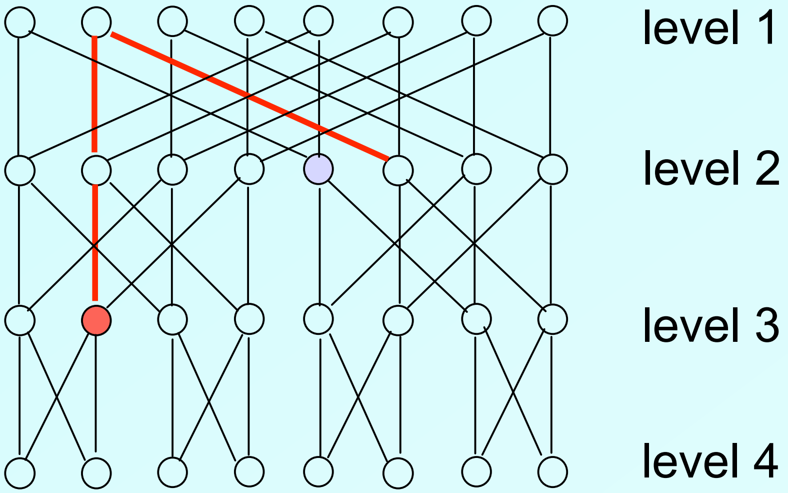
Viceroy

- **Emulating the butterfly network**



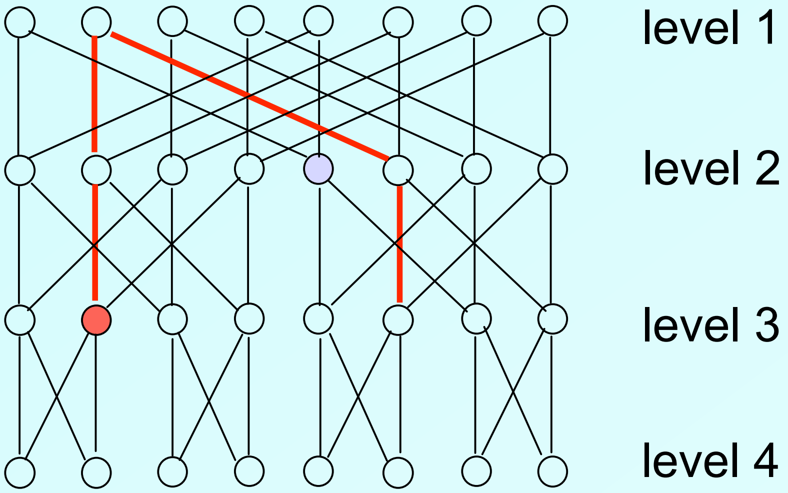
Viceroy

- **Emulating the butterfly network**



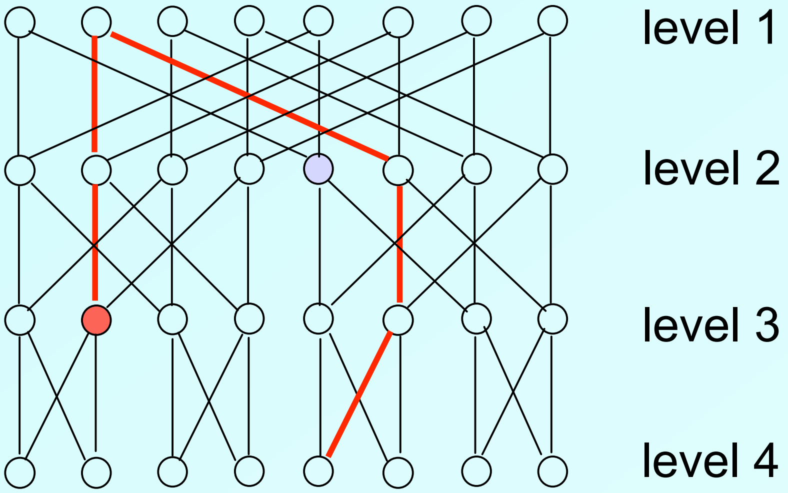
Viceroy

- **Emulating the butterfly network**



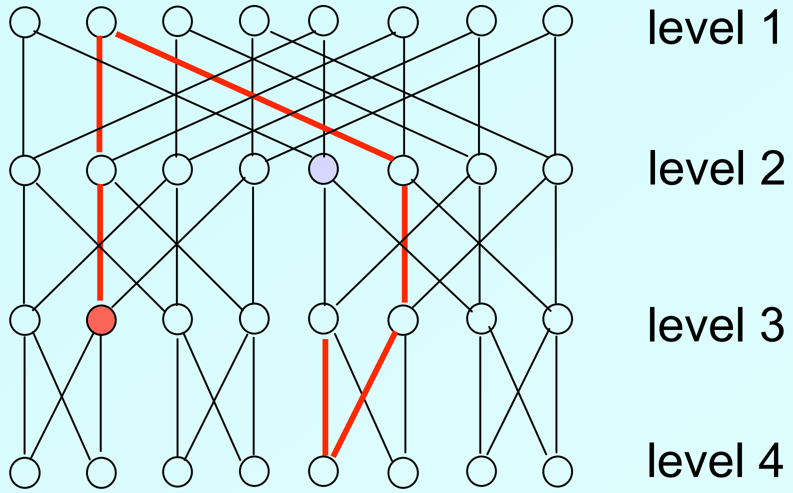
Viceroy

- **Emulating the butterfly network**



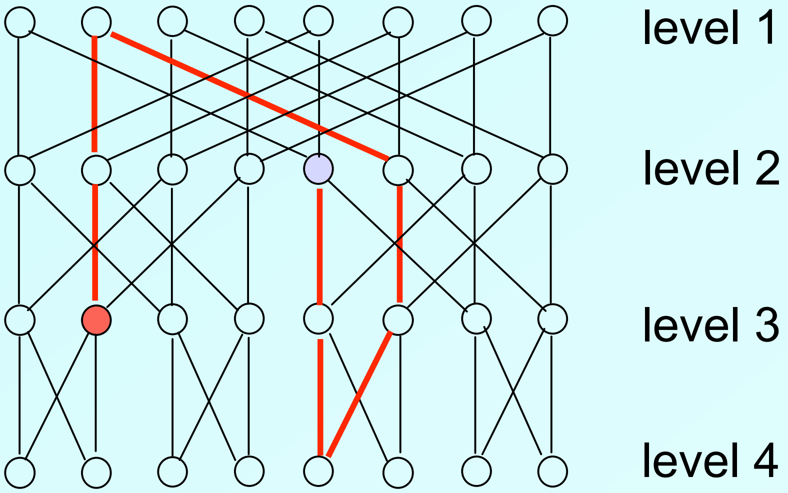
Viceroy

- **Emulating the butterfly network**



Viceroy

- **Emulating the butterfly network**



- **Logarithmic path lengths between any two nodes in the network**

More DHTs

- **CAN:**
  - Route selection flexibility: great!
  - Neighbor selection flexibility: poor!
- **Butterfly:**
  - Route selection flexibility: poor!
  - Neighbor selection flexibility: poor!

Outline

- New DHT designs: CAN and Viceroy
- **Flexibility as a DHT design requirement**
- File-sharing replication
  - Do current schemes work?
  - Could they work?

Summary of flexibility analysis

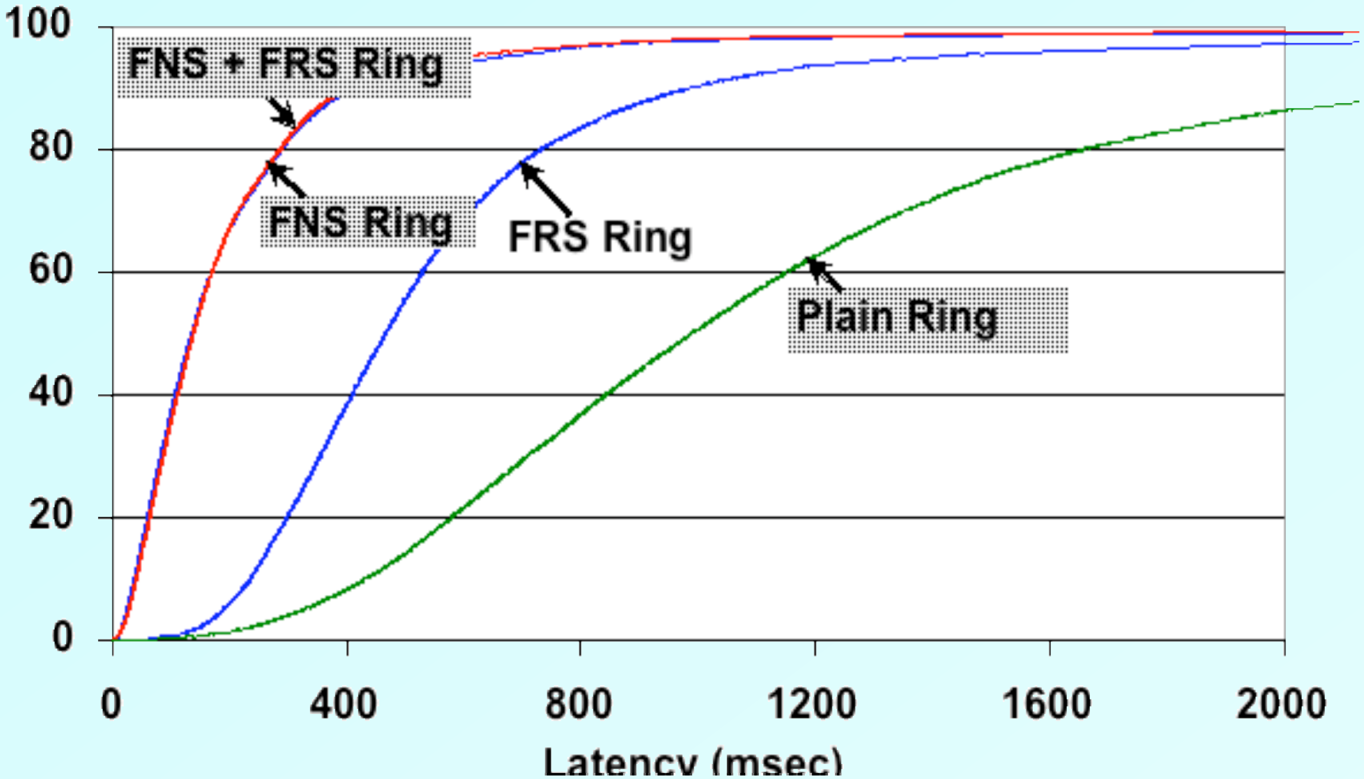
Flexibility	Ordering of Geometries
Neighbors	Hypercube << Tree, XOR, Ring, Hybrid **(1)** **(2^{i-1})**
Routes	Tree << XOR, Hybrid, Hypercube, Ring **(1)** **(logN/2)**

How relevant is flexibility for DHT routing performance?

Analysis of *Overlay Path Latency*

- **Goal: Minimize end-to-end overlay path latency**
  - not just the number of hops
- **Both flexibility in neighbor selection (FNS) and route selection (FRS) can reduce latency**
  - Tree has FNS, Hypercube has FRS, Ring & XOR have both

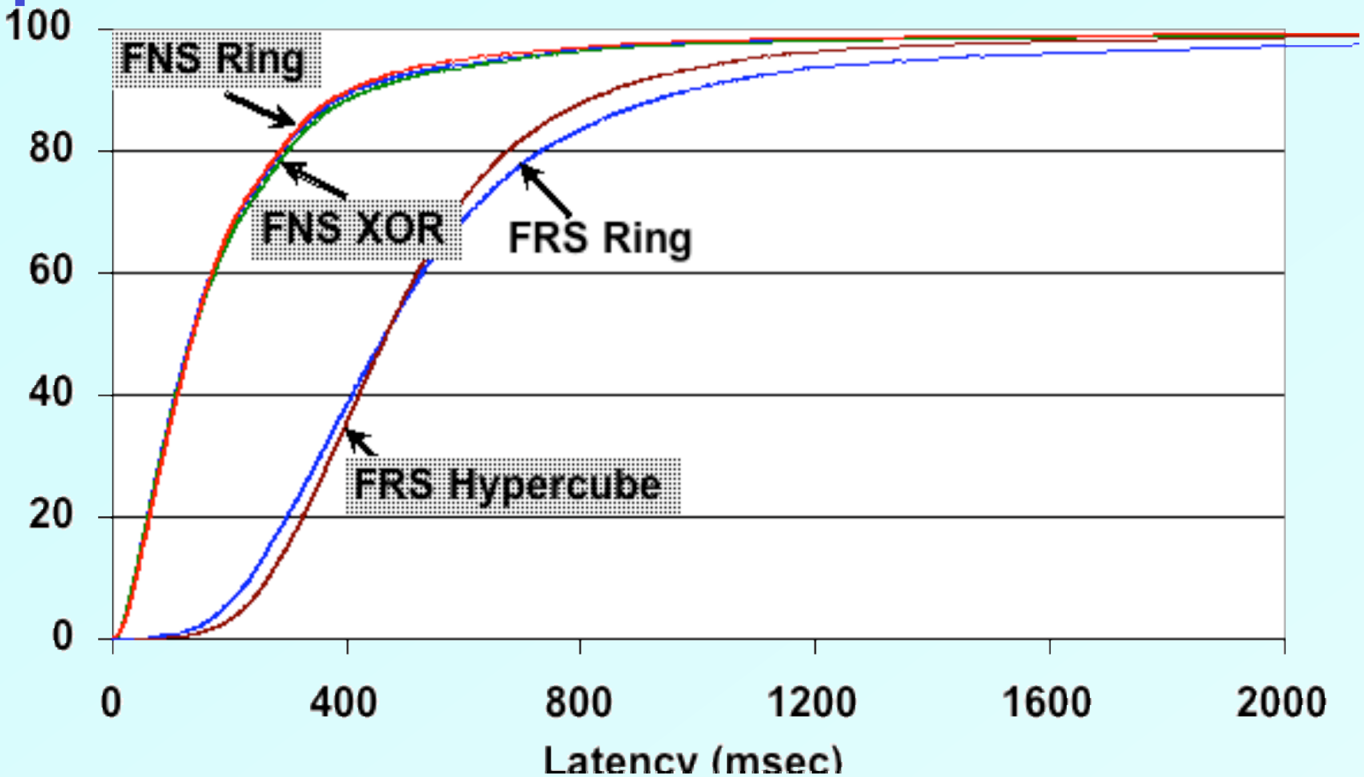
Which is more effective, FNS or FRS?



Plain << FRS << FNS ≈ FNS+FRS

Neighbor Selection is much better than Route Selection

Does Geometry affect performance of FNS or FRS?



No, performance of FNS/FRS is independent of Geometry

A Geometry's support for neighbor selection is crucial

Summary of results

- **Flexible routing selection matters for Static Resilience**
  - Ring has the best resilience
- **Both flexible routing and neighbor selection reduce Overlay Path Latency**
- **But, neighbor is far more important than routing**
  - Ring, Hybrid, Tree and XOR have high flexible neighbor selection

Outline

- New DHT designs: CAN and Viceroy
- Flexibility as a DHT design requirement
- **File-sharing replication**
  - Do current schemes work?
  - Could they work?

Increasing Availability

- **Availability is driven by 3 factors:**
  - Machine availability
    - How often nodes fail?
    - How often nodes recover?
  - Content availability
    - Degree of data redundancy

- **3 ways to increase availability in any distrib. sys.:**
  1. Increase MTTF
  2. Reduce MTTR
  3. Increase data redundancy

What is Availability?

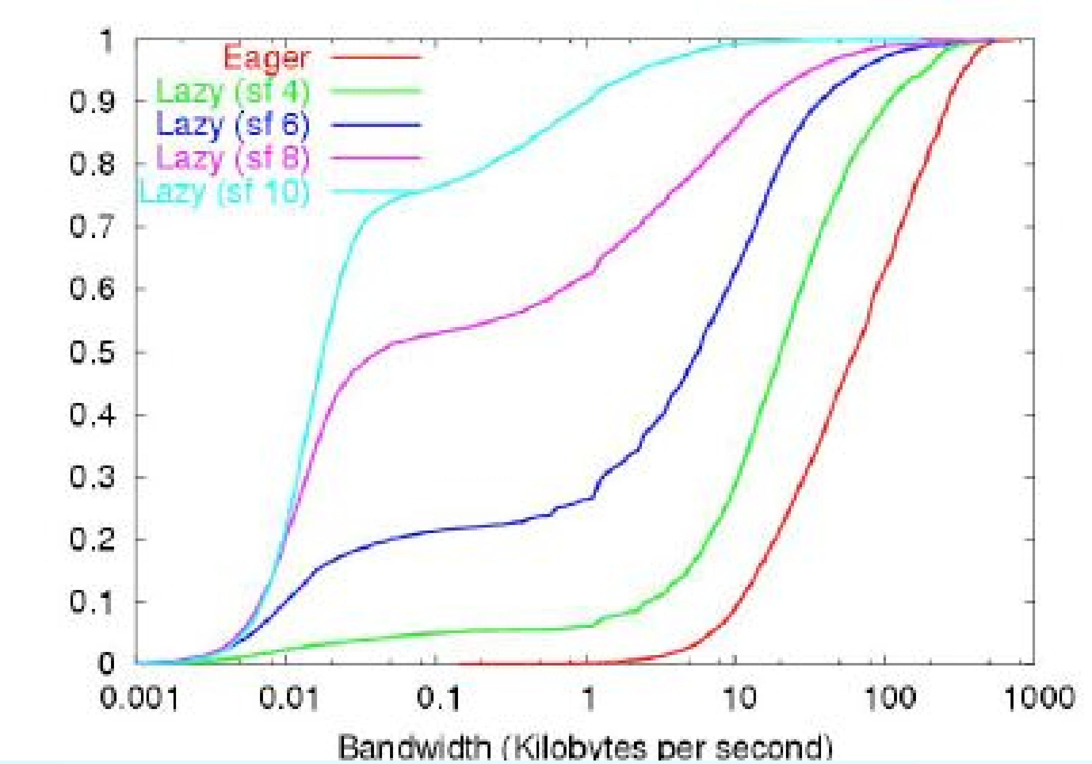
- **Availability is driven by 3 factors:**
  - Machine availability
    - How often nodes fail?
    - How often nodes recover?
  - Content availability
    - Degree of data redundancy

- **3 ways to increase availability in any distrib. sys.:**
  1. Increase MTTF
  2. Reduce MTTR
  3. Increase data redundancy

TotalRecall

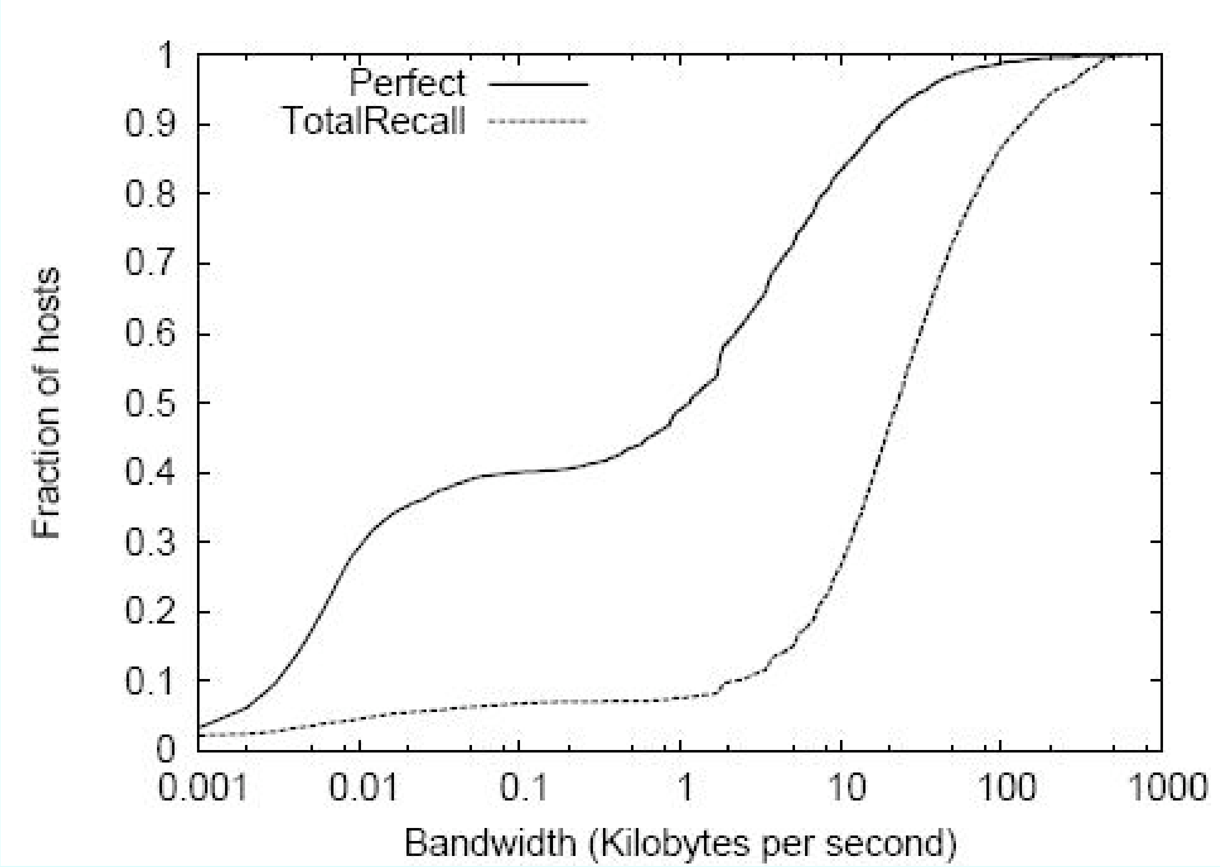
- **Observation:**
  - – Assuming independent failures, one could replicate data enough to maintain a desired level of availability
- **Trade-offs:**
  - – The more replicas there are, the more lazy the replication scheme used
  - – Replication vs. Erasure coding

Does It Work?



No.

Could It Work?



Yes.

Glacier: Highly durable, decentralized storage despite massive correlated failures

- **Observation:**
  - Failures are correlated. If file locations use a hash function, lookup failures are independent.
- **Problem:**
  - Still doesn’t solve the bandwidth problem

My take on P2P replication

- **There are three kinds of content:**
  - Popular content
    - Inherently well replicated
  - Unpopular content
    - Makes little sense to replicate
  - Grey area
    - Here is where it matters
- **Questions I have:**
  - How many files are in the grey area?
  - How much savings could a clever algorithm give us?