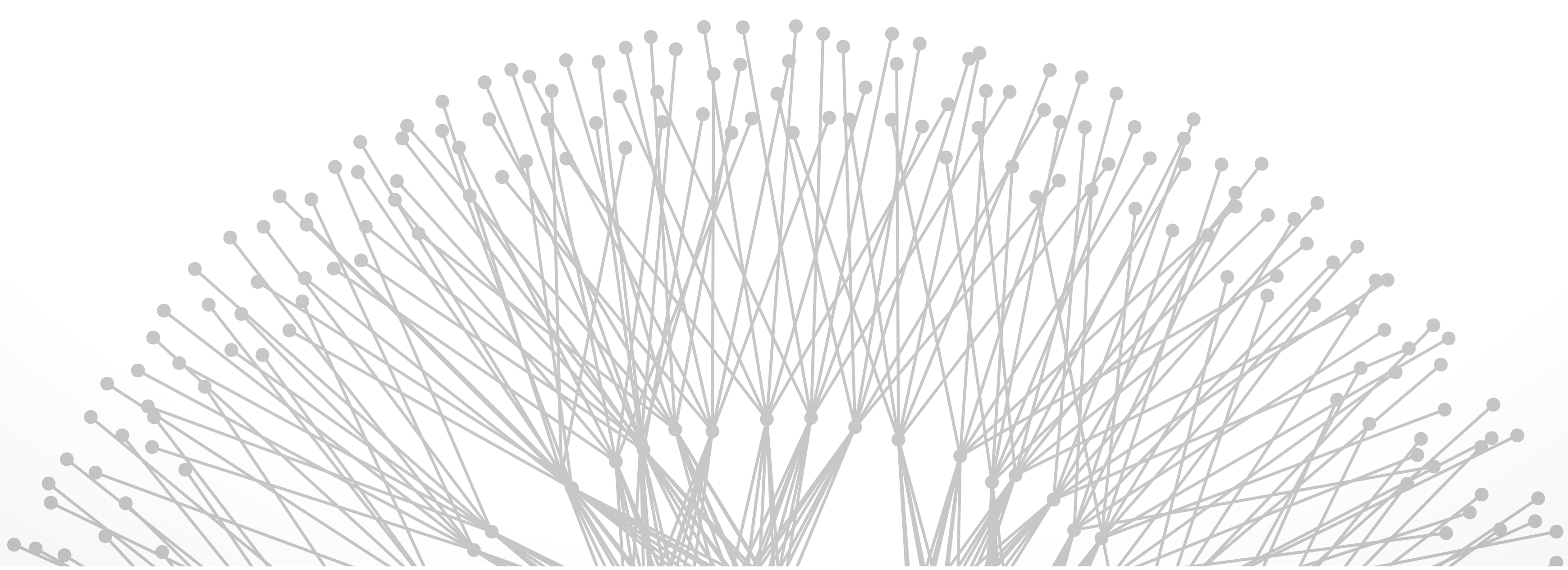
Scalable routing

Brighten Godfrey

CS 538 September 27 2012



How do we route in really big networks?

Classic shortest-path routing

$\Omega(n)$ memory per node

- at least store next hop to *n* destinations

$\Omega(n)$ messages per node per unit time

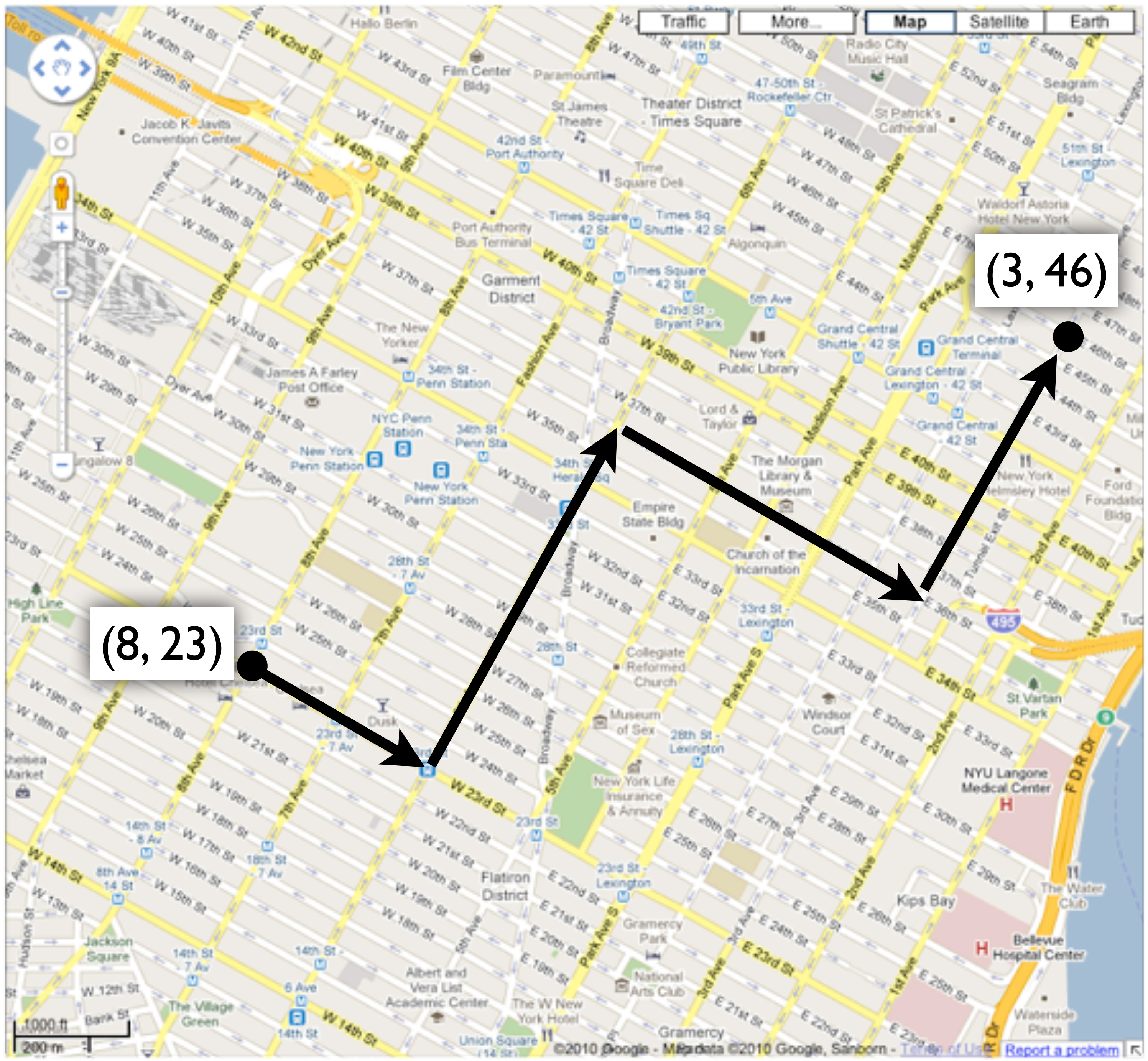
- assuming each node moves once per unit time
- also must recompute routes each of these times

if n = 1,000,000,000 and "unit time" = one day,

- ≈100–10,000x more fast path mem. than routers today
- 11,600 updates per second
- 4.4 Mbit/sec if updates are 50 bytes

How can we scale better than $\Omega(n)$ per node?

Routing in Manhattan



Recipe for scaling

1. Convert name to address
   - name: arbitrary
   - address: hint about location
   - conversion uses distributed database (e.g., DNS)
2. Nodes have incomplete local view of network
3. To route, combine local view with dest. address

Challenge: how do we summarize the network in the partial view and address?

- And what *exactly* are we trying to achieve?

Key goals

Addresses are small

Node state is small

Routes are short

- stretch = $\frac{\text{route length}}{\text{shortest path length}}$

How does Manhattan routing do?

- Assume square grid of *n* nodes $(\sqrt{n} \times \sqrt{n})$
- Address is (street, avenue); nodes store neighbors' addr.
- Address size: $2\log_2(\sqrt{n}) = \log_2 n$
- Node state: $\approx 4\log_2 n$
- Route length: shortest (stretch 1) *if we know address!*

Outline

Scalable routing in structured networks

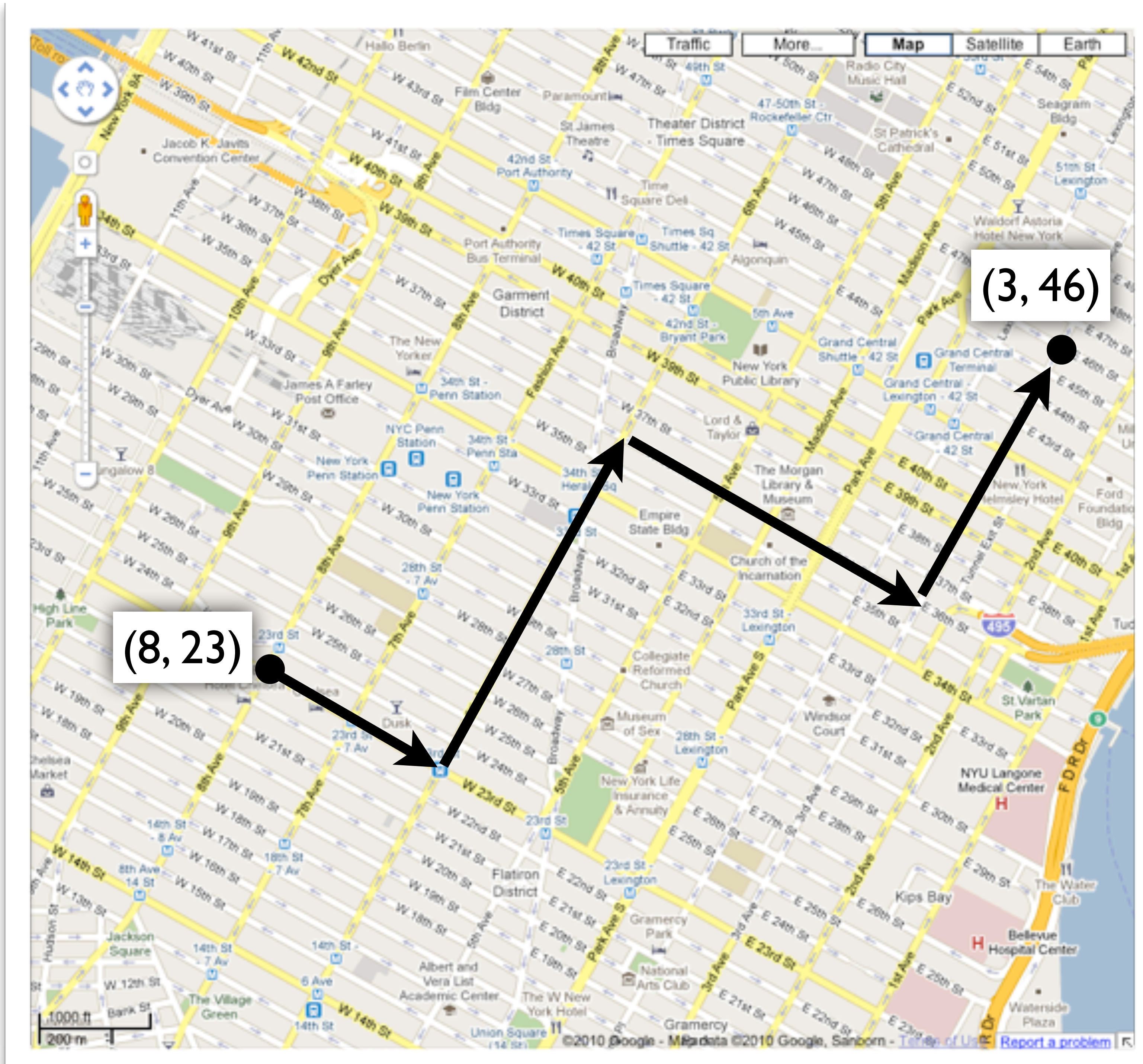
- Manhattan routing
- Greedy routing
- NIRA

Scalable routing in arbitrary networks

- Hierarchy
- Compact routing

Structured networks

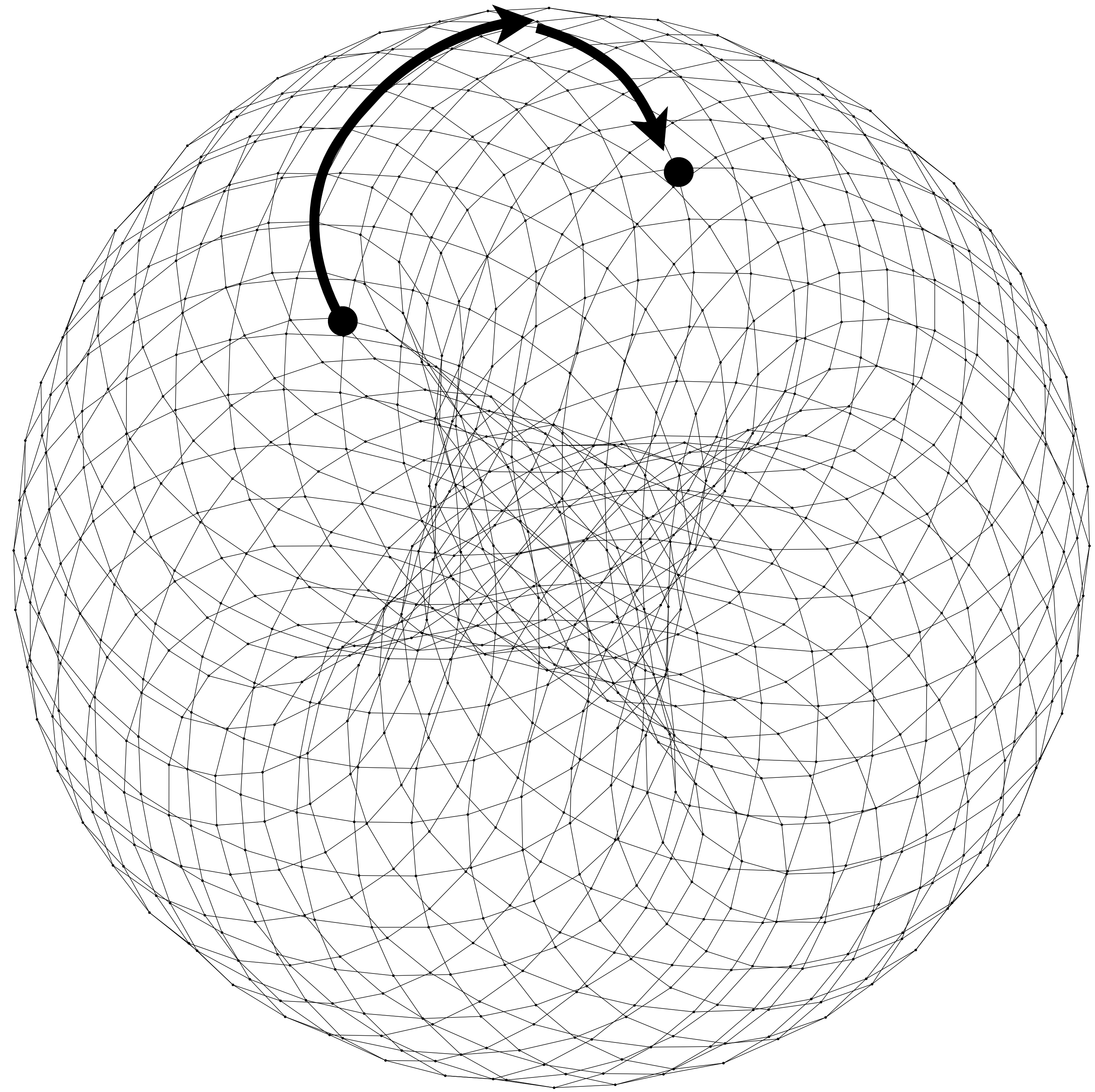
Grid



Torus

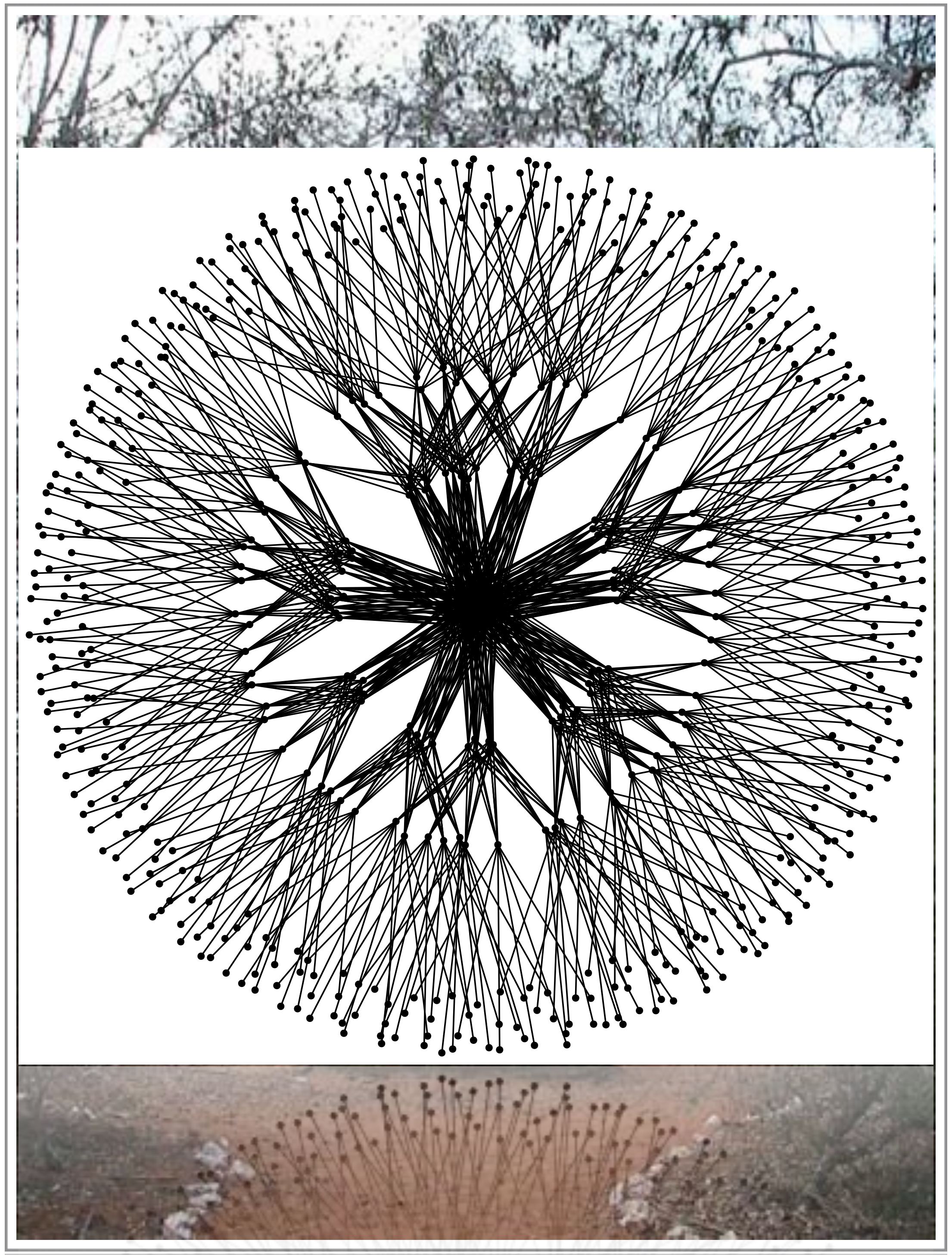
3D Torus

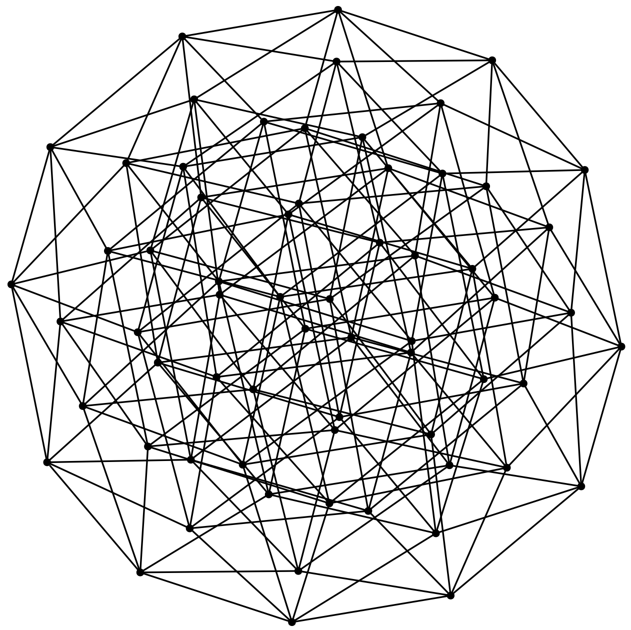
[Cray T3D press shot via hexus.net]



2D Torus

A plethora of structured graphs!





Hypercube

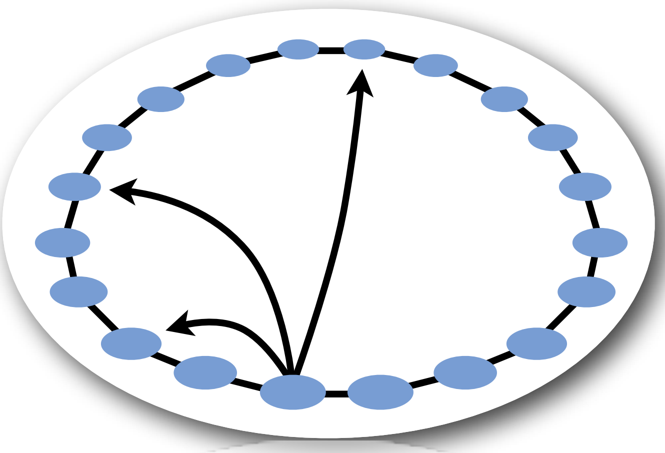
Supercomputers, distributed hash tables

Fat tree

Supercomputers, data centers

Small world

distributed hash tables



Greedy routing

Technique common in many structured networks

Scheme:

- Each node knows addresses of itself & neighbors
- Given two addresses, can estimate "distance" between them: dist(*s*,*t*)
- Forwarding at node *v*: send to neighbor *w* with lowest distance to destination *d* (minimize dist(*w*,*d*))

What structure does this require?

- Compact addresses that can "summarize" location
- Good estimate of distance dist(*s*,*t*) given two addresses
  - No local minima in dist()! (Q: Why could there be?)

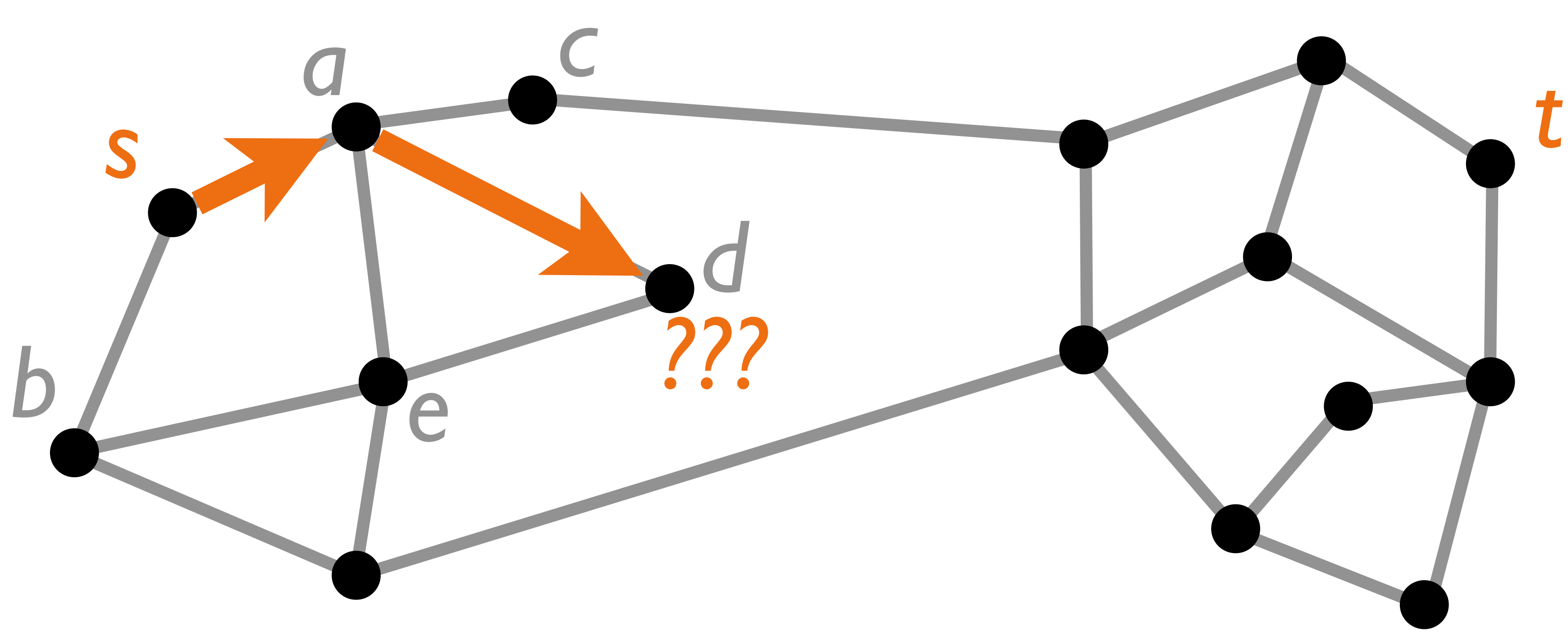
Greedy routing examples

#1: Manhattan routing

- Address: (x, y) coordinate on grid
- Distance 'estimation' of (x, y) to $(x', y') = |x-x'| + |y-y'|$

#2: Greedy geographic routing

- Address: physical location (e.g., (x,y) coord. from GPS)
- Distance estimation: Euclidean distance



Greedy Perimeter Stateless Routing

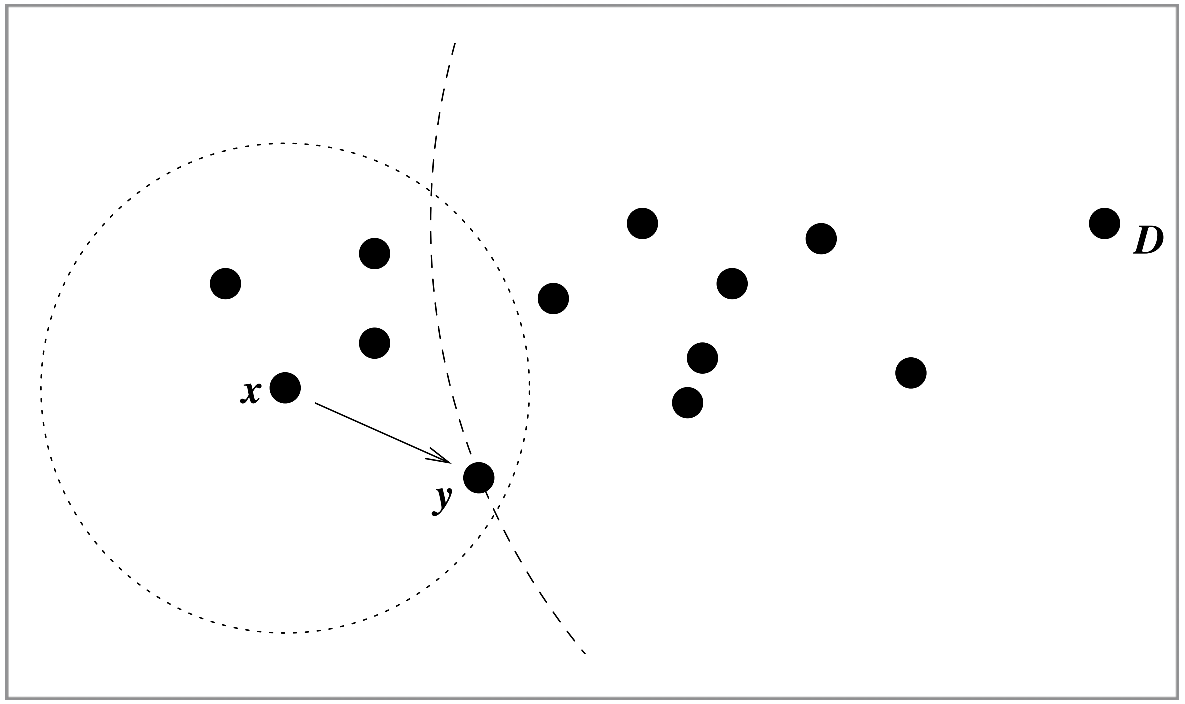
[Karp, Kung, MobiCom '00]

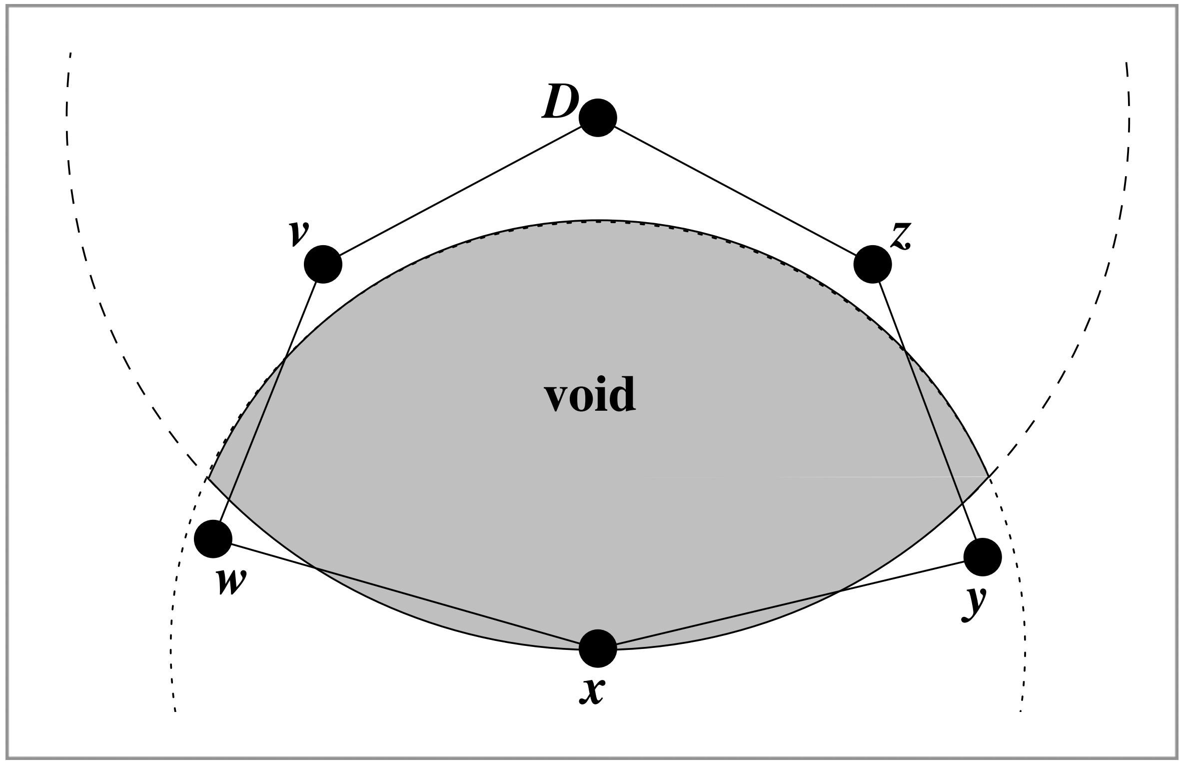
Address is physical location, e.g., from GPS

Distance estimate is Euclidean distance

If we get stuck...

- = no neighbor is closer to *x* than we are!
- Then planarize graph and traverse perimeter of void

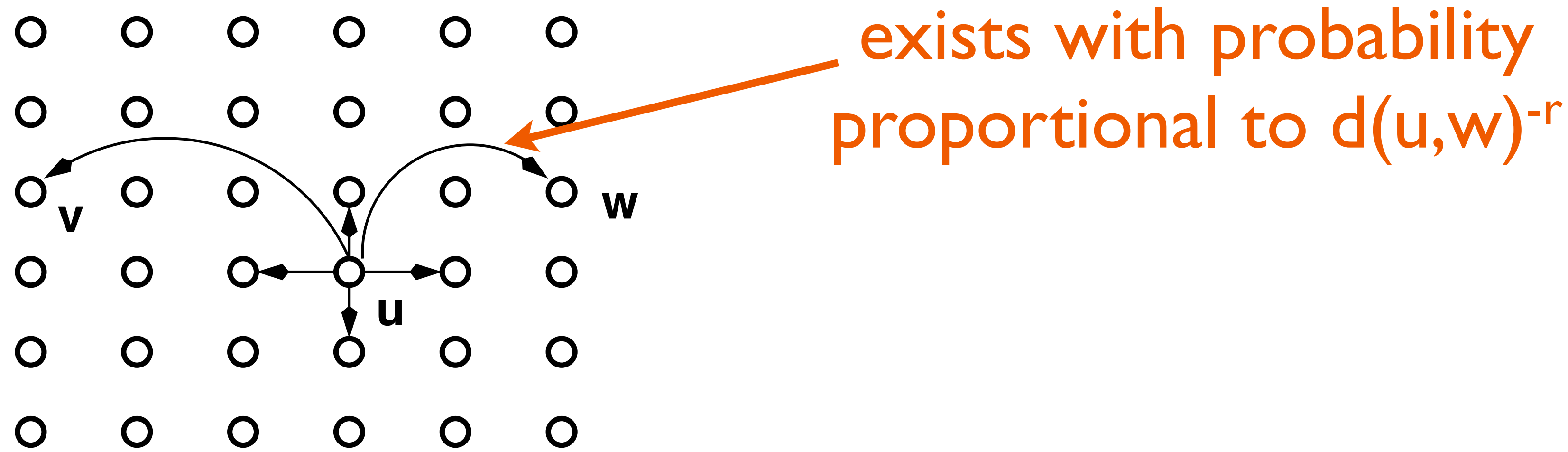




Greedy example #3: Small worlds

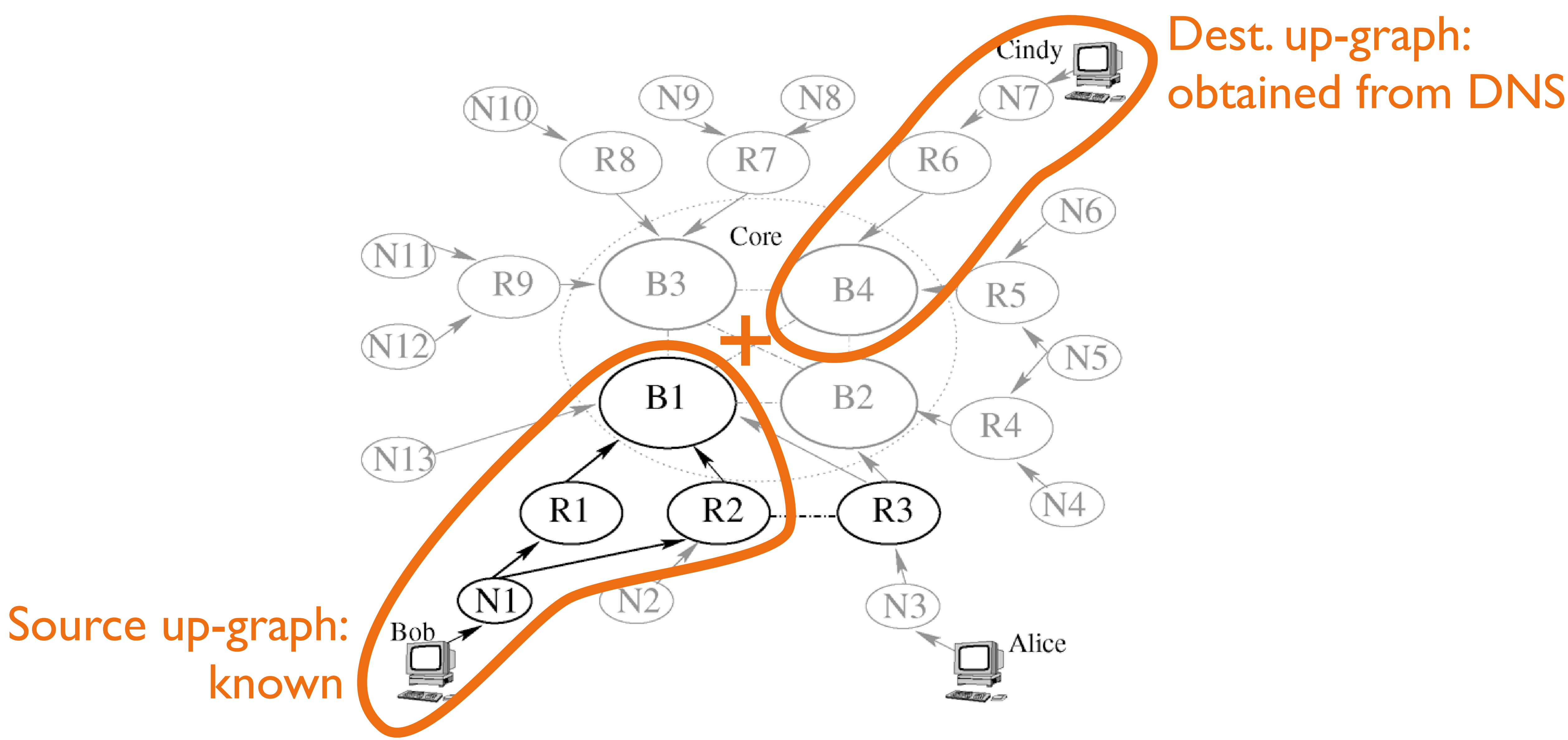
"Small world" effect demonstrated by Milgram ['67]

Kleinberg's model: *n* x *n* lattice, plus long range edges



Result: greedy routing finds short ($O(\log^2 n)$) paths with high probability if and only if $r = 2$

Non-greedy: NIRA [Yang et al '07]



Assumes a graph with a "core"

- routes go up to core (provider links), over (peering links), and down (customer links)
- i.e., valley-free

NIRA key ideas

Address is effectively a subgraph, not just a number!

- here "address" means "destination-specific location info"

Up-graphs are small

Union of source and dest subgraphs is all we need

- exploits Internet's current structure to find good paths

Q: How well does NIRA satisfy our goals?

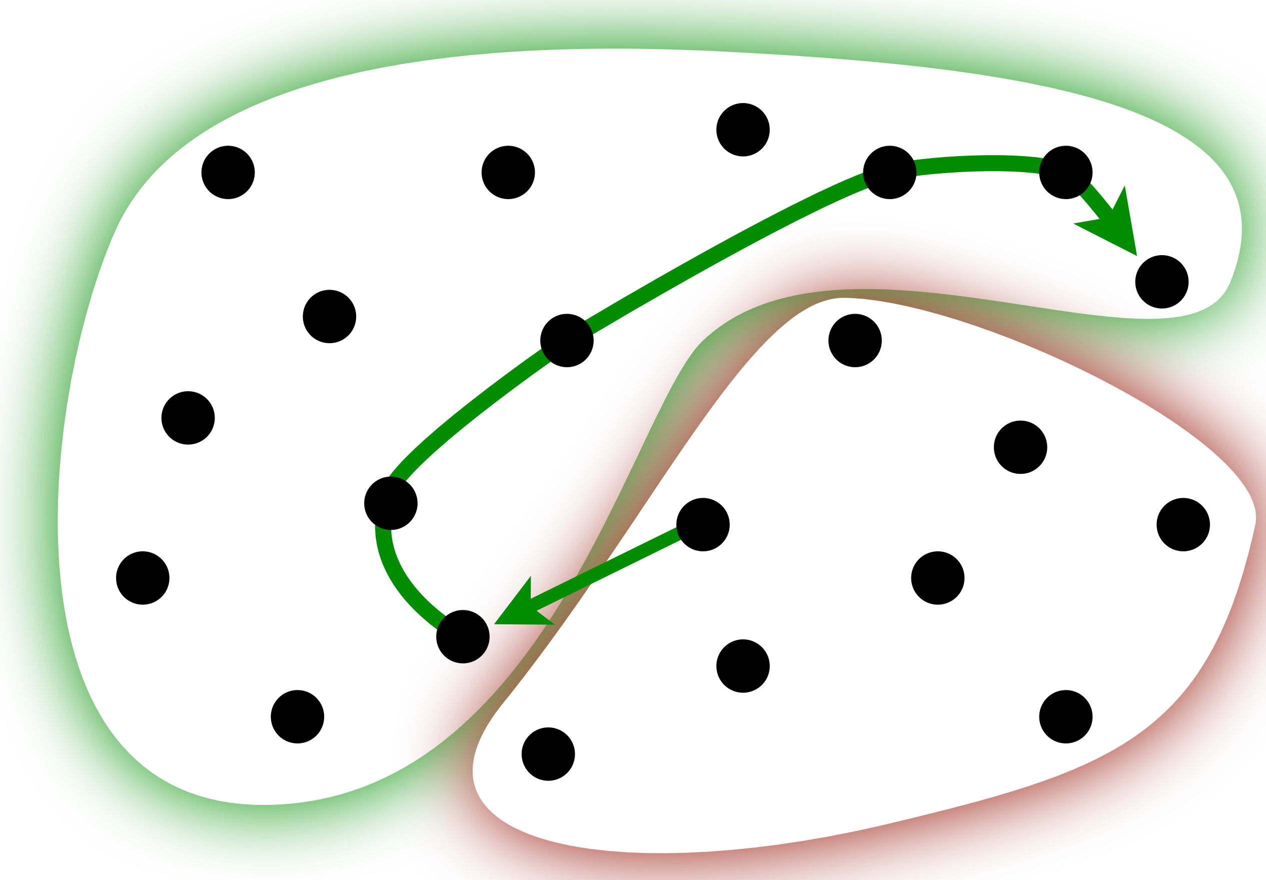
- small address
- small node state
- low stretch

But what if our network
does not have a
“special” structure?

Technique in practice: Hierarchy

No structure? Make one!

- 2-level hierarchy: nodes in clusters
- each node knows how to reach one node of each cluster and all nodes in its own cluster



Problems:

- Some paths very long
- Location-dependent addresses (as in earlier techniques)

128.112.128.81

Fundamental tradeoffs

Can we achieve our key goals?

- Low state
- Low stretch (short paths)
- Short addresses

Or, does scalability force us to give something up?

Compact routing theory

Given arbitrary graph, scheme must:

- Construct state (forwarding tables) at each router
- Specify forwarding algorithm:
  - Input: Forwarding table, incoming packet
  - Output: Packet's next hop (+ optionally change header)

Goals:

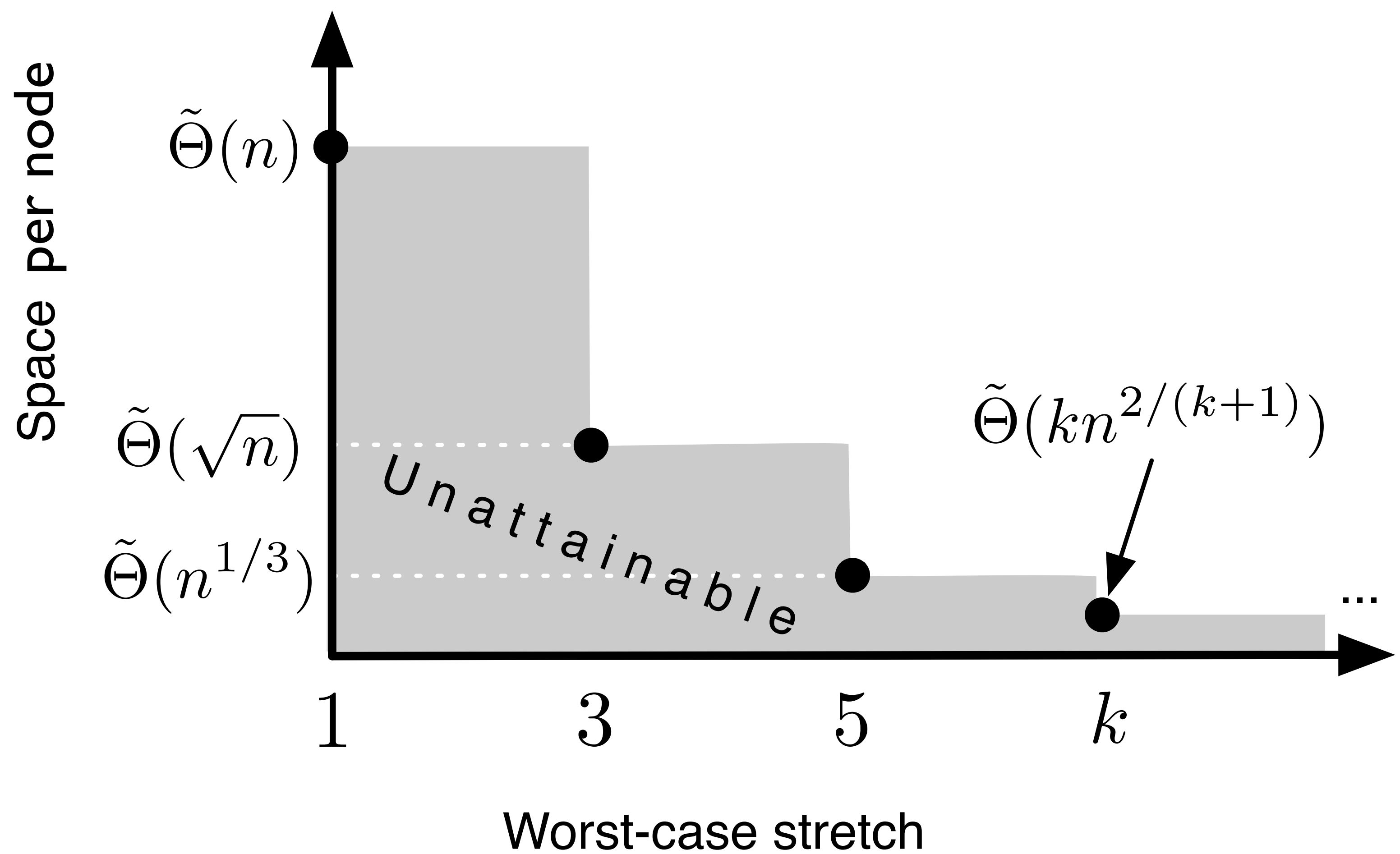
- Minimize maximum state at each router (FIB memory)
- Minimize maximum stretch:

$$\max_{s,t} \frac{s \rightsquigarrow t \text{ route length}}{s \rightsquigarrow t \text{ shortest path length}}$$

- Reasonably small packet headers (e.g., O(log *n*))

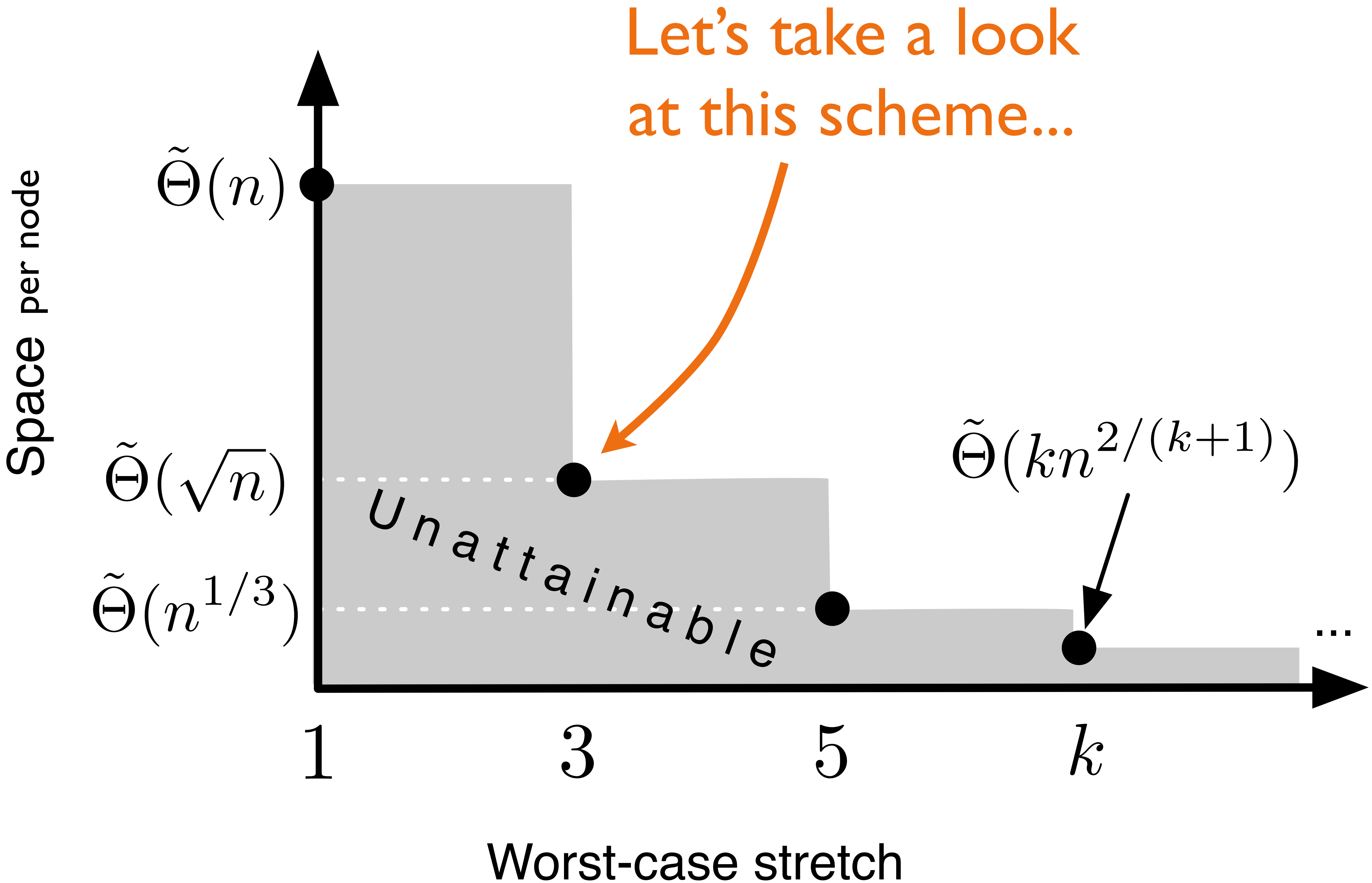
Compact routing theory

[Peleg & Upfal '88,
Awerbuch et al. '90,
...,
Cowen '99,
Thorup & Zwick '01,
Abraham et al. '04]



Name-dependent	Addresses assigned by routing protocol
Name-independent	Arbitrary ("flat") names e.g., DNS or MAC address

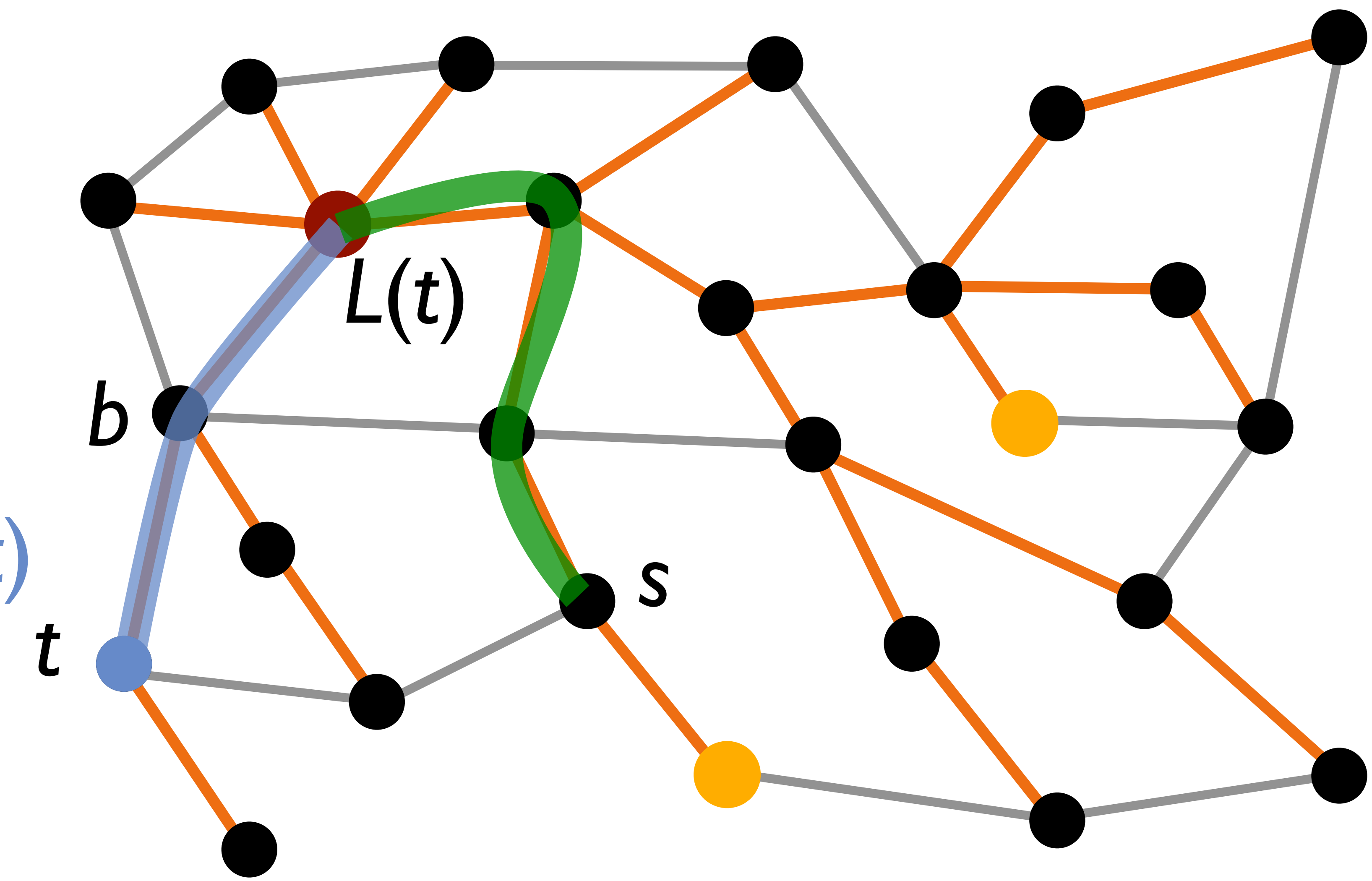
Compact routing theory



Landmarks

Everyone knows shortest path to landmarks.
Enable approximately shortest paths.

addr(t) = ($L(t)$,b,t)



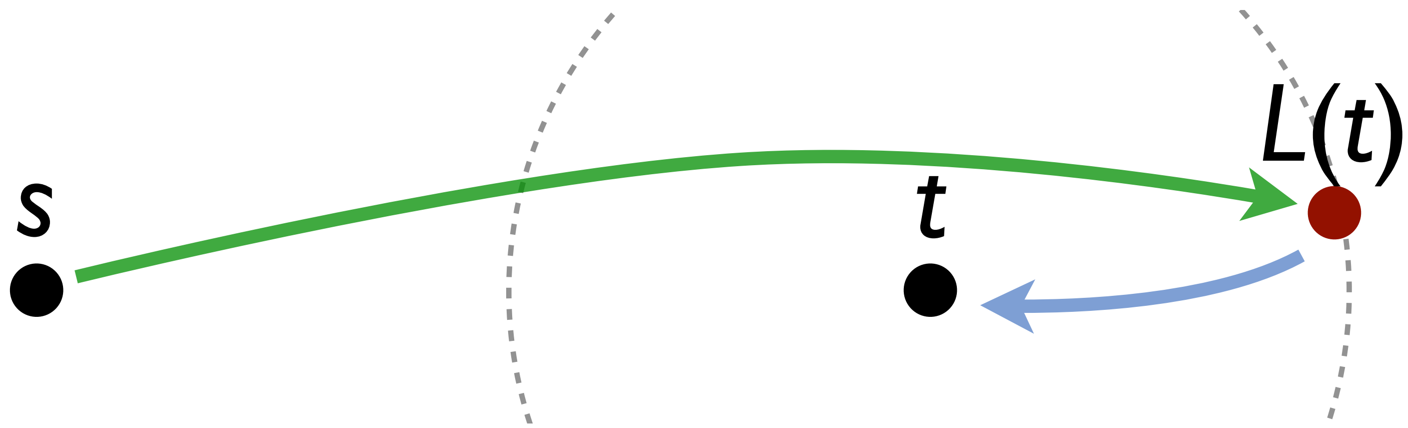
route length = dist. to landmark + dist. to t

$\leq d(s,t) + d(t,L(t)) + d(L(t),t)$

triangle inequality

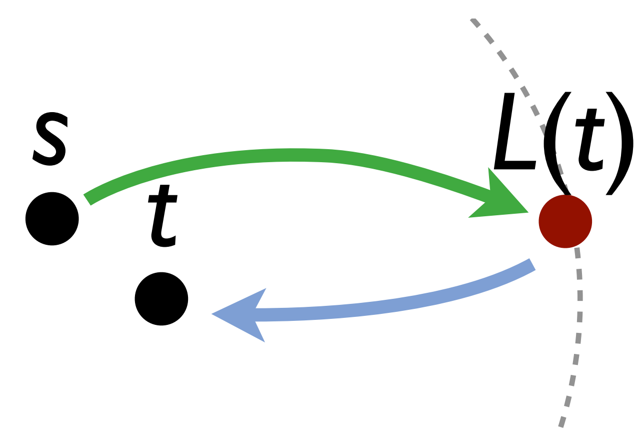
Stretch analysis

Case 1: $d(s,t) \geq d(t,L(t))$: further than landmark



- route length $\leq d(s,t) + d(t,L(t)) + d(L(t),t) \leq 3d(s,t)$

Case 2: $d(s,t) < d(t,L(t))$: closer than landmark



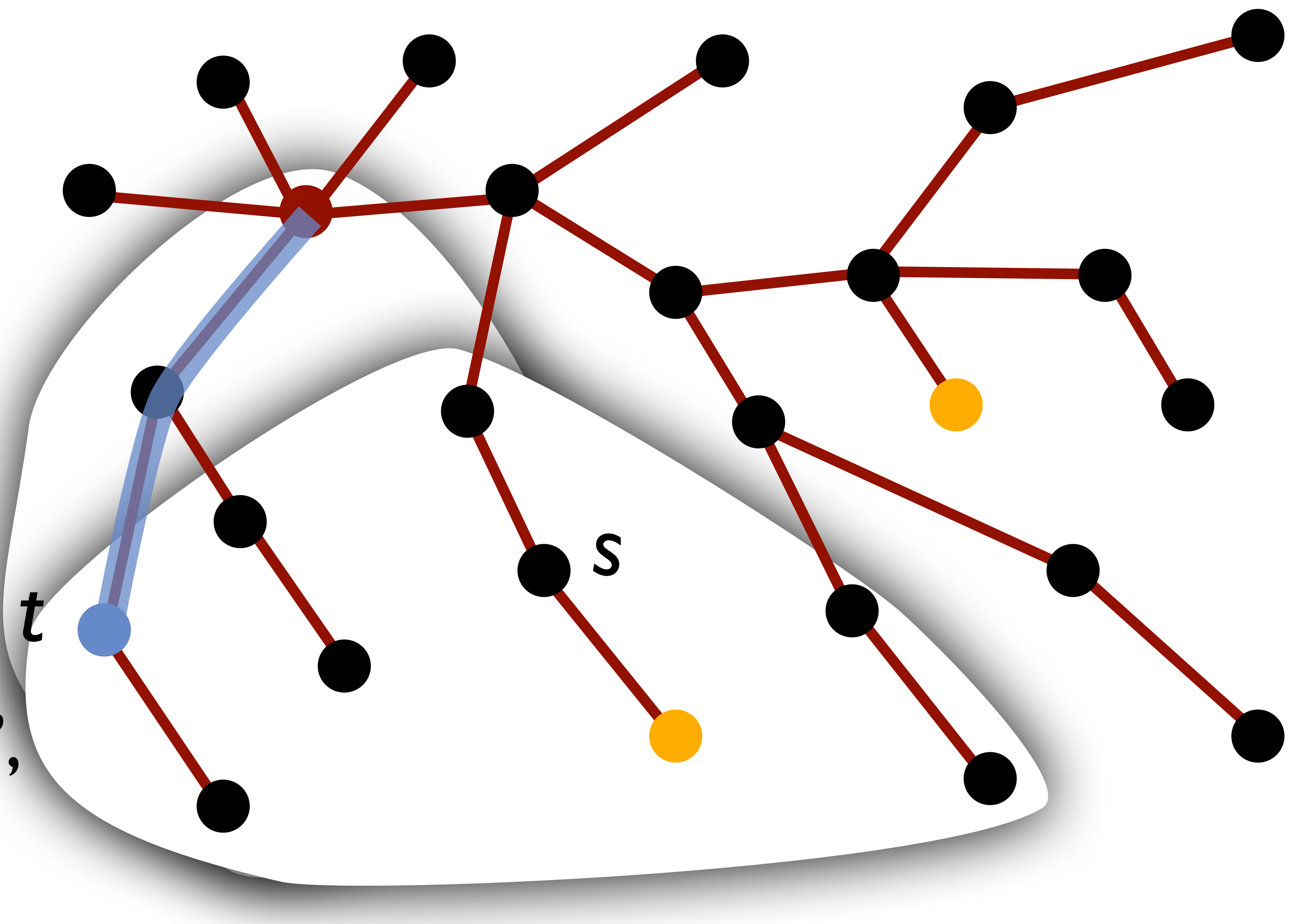
- Trouble!
- Idea: in Case 2, just remember the shortest path.

Vicinities

~~$V(s)$ = nodes t s.t. $d(s,t) < d(t,L(t))$~~

$V(s)$ = nodes t s.t. $d(s,t) < d(s,L(s))$

Requires "handshaking", but convenient to implement



How big are $V(t)$?

Need a landmark in my vicinity.

$\tilde{\Theta}(\sqrt{n})$ random landmarks: $\tilde{\Theta}(\sqrt{n})$-size vicinities

Tool: Chernoff bound

"The sum of many small independent random variables is almost always close to its expected value."

X_i = m independent (0,1) random variables

$X = \sum X_i$, $E[X] = \mu$

For any $0 \leq \delta \leq 2e - 1$,

$$\Pr[X < (1-\delta)\mu] < e^{-\mu\delta^2/2}$$
$$\Pr[X > (1+\delta)\mu] < e^{-\mu\delta^2/4}$$

See, e.g., Motwani & Raghavan, Theorems 4.1 - 4.3

How many landmarks are enough?

Show that any node *v* always has ~ln *n* landmarks in its vicinity if we use about $\sqrt{c \cdot n \ln n}$ landmarks

$X_i = 1$ if *i*th closest node to *v* is landmark, else $X_i = 0$

$$\Pr[X_i] = \frac{\sqrt{c \cdot n \ln n}}{n}$$

$$E[X] = (\text{Number of nodes in vicinity}) \cdot \Pr[X_i]$$

$$E[X] = \sqrt{c \cdot n \ln n} \cdot \frac{\sqrt{c \cdot n \ln n}}{n}$$

$$= c \ln n$$

Increase *c* to make this arbitrarily small

$$\Pr\left[X < \frac{1}{2} c \ln n\right] < e^{-(c \ln n) \cdot \frac{1}{4} \cdot \frac{1}{2}} = e^{\ln n^{-c/8}} = n^{-c/8}$$

Chernoff bound

Analysis

Stretch

- $\leq$ 3 if outside vicinity (after "handshake")
- = 1 if inside vicinity

State (data plane)

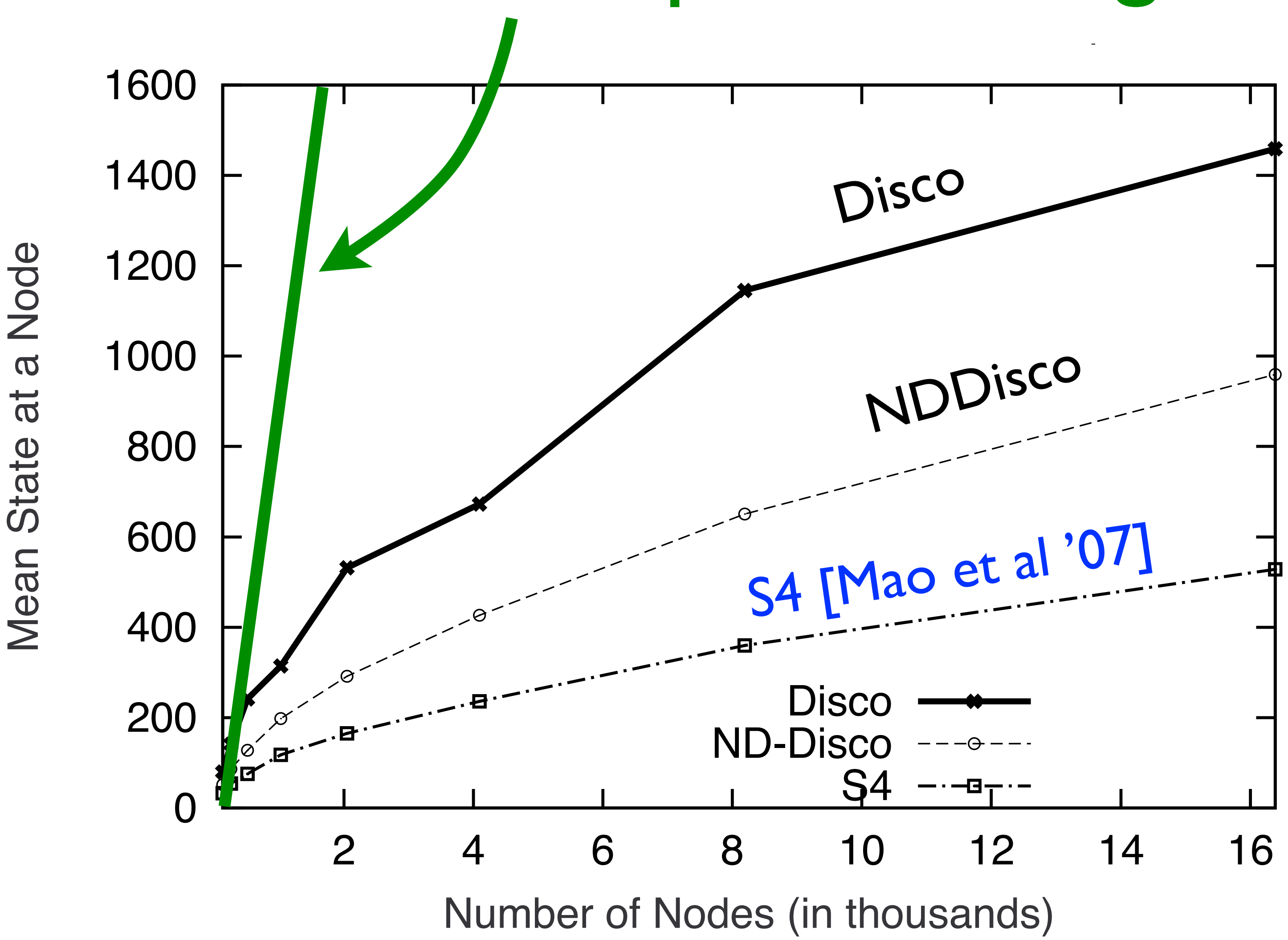
- Routes to landmarks: $O(\sqrt{n \log n} \cdot \log n) = \tilde{\Theta}(\sqrt{n})$
- Routes within vicinity: $O(\sqrt{n \log n} \cdot \log n) = \tilde{\Theta}(\sqrt{n})$

Address size

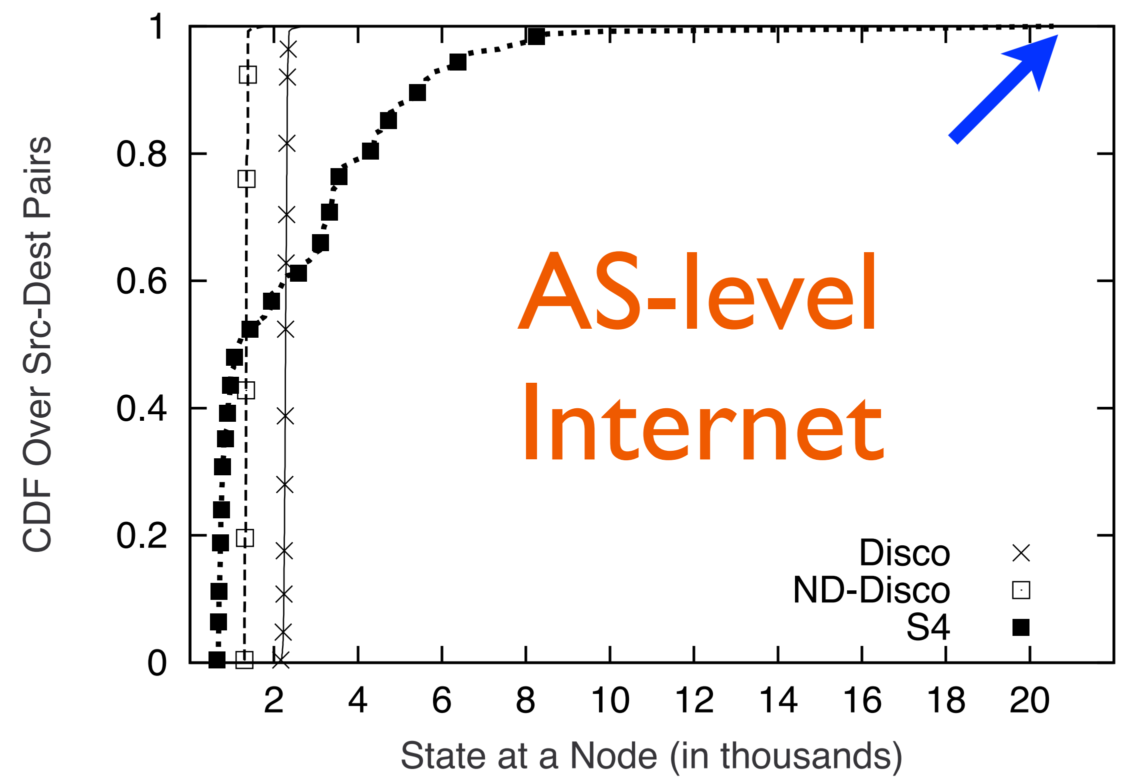
- Simple implementation: depends on path length, but very short in practice
- More complicated/clever storage of route from landmark to destination: $\Theta(\log n)$

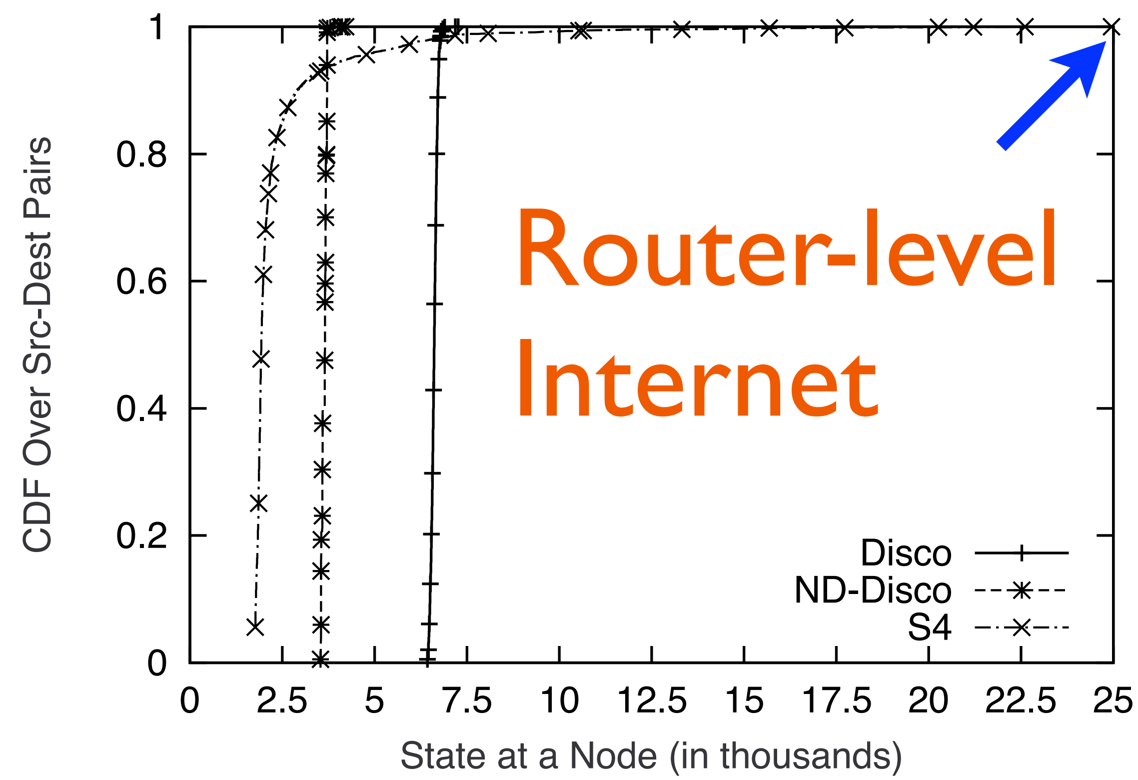
State in example networks

Shortest path routing

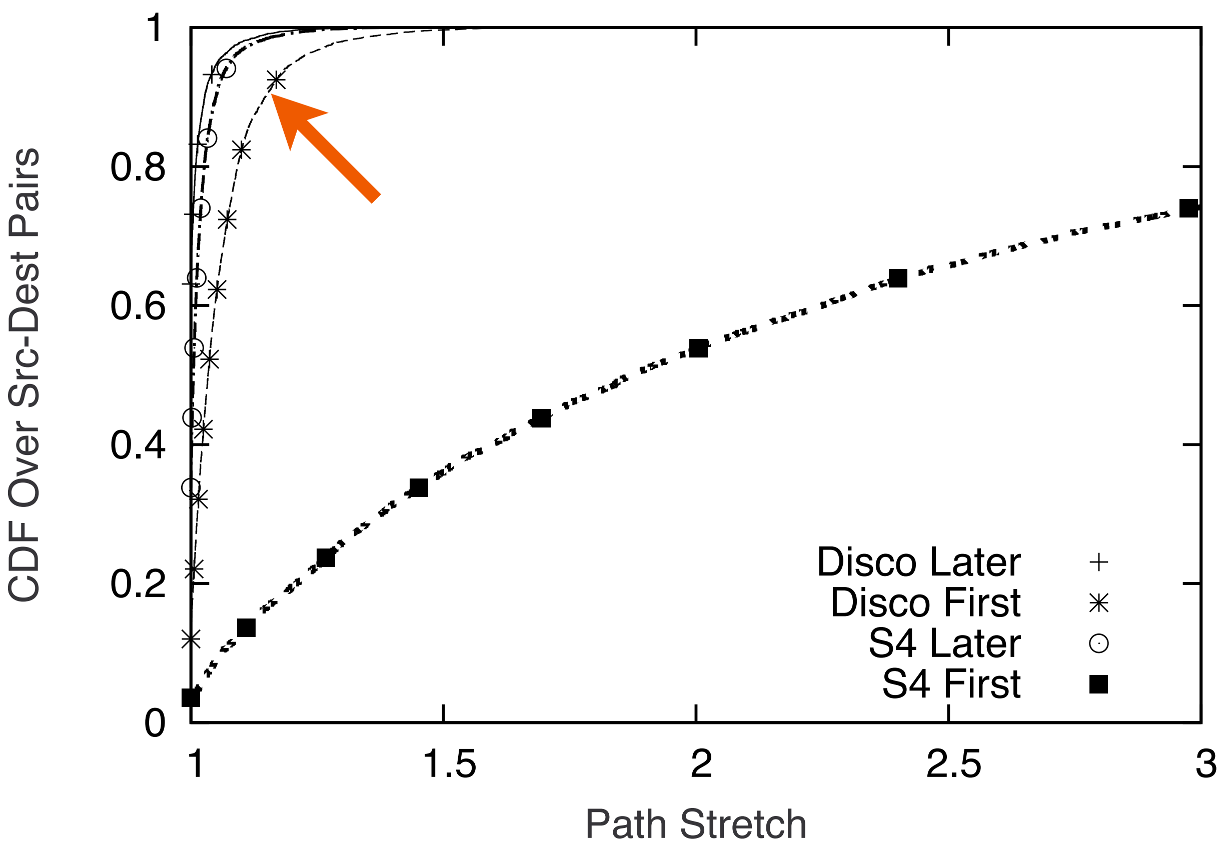


Geometric random graphs

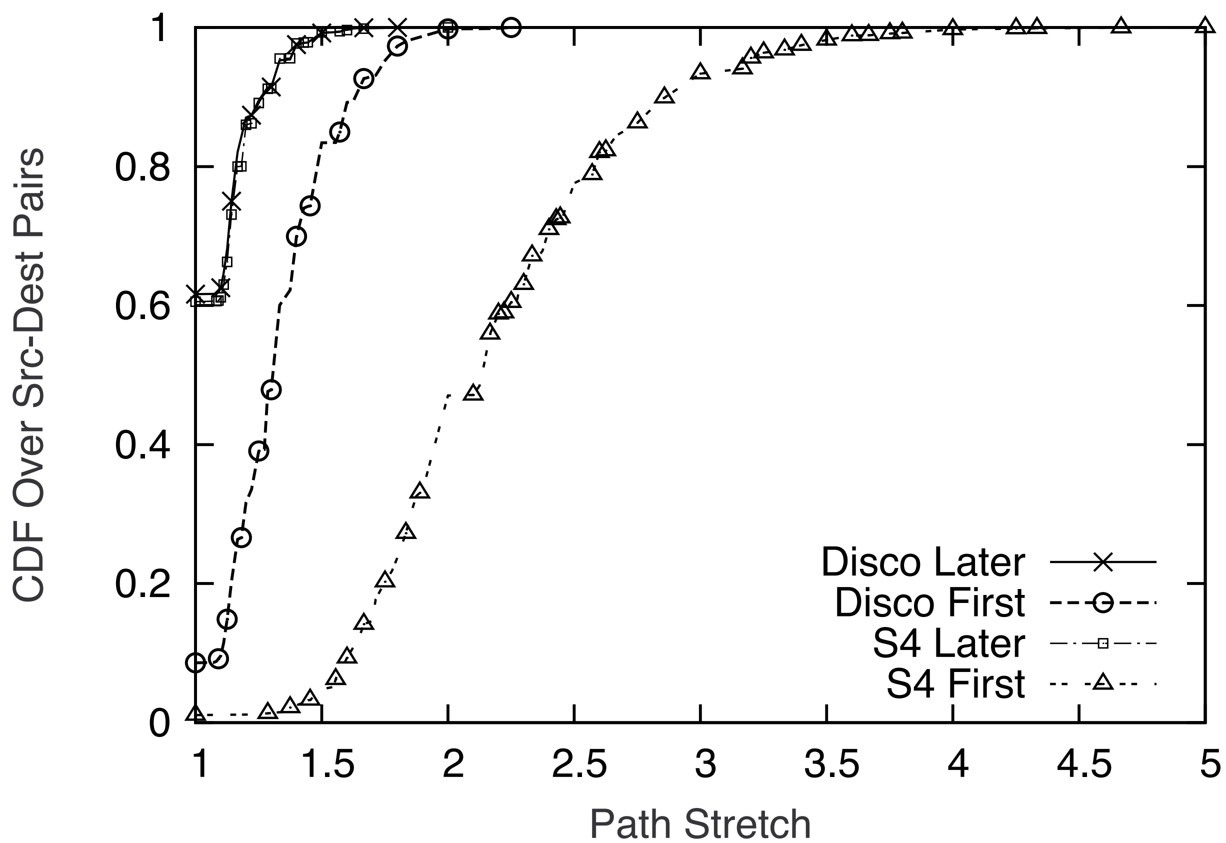




Stretch in example networks



16,000-node Geometric random graph



Router-level Internet topology

What we're not seeing

Routing on flat names with low stretch and state

- we assumed source knows destination address

Other points state-stretch tradeoff space

- we saw state ~$n^{1/2}$, stretch 3

Why you cannot do better than this

- ...in the general case (dense graphs)

Why you can do better than this

- ...if the network is sparse (few edges), as essentially all real networks are

What we're not seeing

Distributed compact routing

- How do you compute FIBs without global view?

How to handle interdomain routing policies

- no one knows!

Tuesday: Invited Lecture

Theo Benson, Princeton / Duke

- *"Demystifying and controlling the performance of data center networks"*
- In 2405 SC, not here

Thursday

- Two readings on reliability