Scalable routing

Brighten Godfrey
CS 538 October 3 2013
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,

- $\approx 100$–$10,000x$ 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

(8, 23) → (3, 46)
Recipe for scaling

1. Convert **name** to **address**
   - **name**: arbitrary
   - **address**: hint about location
   - conversion uses distributed database (e.g., DNS)

2. Nodes have **partial view** of network

3. To route, combine partial 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!
Scalable routing in **structured** networks

- Manhattan routing
- Greedy routing
- NIRA

Scalable routing in **arbitrary** networks

- Hierarchy
- Compact routing
Structured networks
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: \( \text{dist}(s,t) \)
- Forwarding at node \( v \): send to neighbor \( w \) which minimizes \( \text{dist}(v,w) + \text{dist}(w,d) \)

What structure does this require?

- Compact addresses that can “summarize” location
- Good estimator of distance \( \text{dist}(s,t) \)
#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

If local minima in \(\text{dist}()\), we’re stuck!
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 destination $D$ than we are!
- Then planarize graph and traverse perimeter of void
"Small world" effect demonstrated by Milgram [’67]

Kleinberg’s model: $n \times n$ 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”
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

Up-graphs are small

[Calculation using 2013 CAIDA Internet topology data]
NIRA key ideas

Up-graphs are small

Union of source and dest subgraphs is all we need
  • exploits Internet’s current structure to find good paths

Address is effectively a subgraph, not just a number!
  • here “address” means “destination-specific location info”

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?
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)
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:
  \[
  \max_{s,t} \frac{\text{s \rightarrow t route length}}{\text{s \rightarrow t shortest path length}}
  \]
- Reasonably small packet headers (e.g., \(O(\log n)\))
Compact routing theory


Name-dependent
Addresses assigned by routing protocol

Name-independent
Arbitrary (“flat”) names
e.g., DNS or MAC address
Compact routing theory

Let's take a look at this scheme...

Space per node

$\tilde{\Theta}(n)$

$\tilde{\Theta}(\sqrt{n})$

$\tilde{\Theta}(n^{1/3})$

$\tilde{\Theta}(kn^2/(k+1))$

Unattainable

Worst-case stretch

1 3 5 k
Everyone knows shortest path to landmarks.

Used to define address:

\[ \text{addr}(t) = (L(t), b, t) \]

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

\[ V(s) = \text{nodes } t \text{ s.t. } d(s,t) < d(t,L(t)) \]

\[ V(s) = \text{nodes } t \text{ s.t. } d(s,t) < d(s,L(s)) \]

How big are \( V(t) \)?

Need a landmark in my vicinity.

\[ \tilde{\Theta}(\sqrt{n}) \text{ random landmarks: } \tilde{\Theta}(\sqrt{n})\text{-size vicinities} \]
"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
Show that any node \( v \) always has \( \sim \ln n \) 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 \( X_i = 0 \)

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

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

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

\[
= c \ln n
\]

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

Increase \( c \) to make this arbitrarily small.
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

CDF Over Src-Dest Pairs

Path Stretch

Disco Later
Disco First
S4 Later
S4 First

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 $\sim n^{1/2}$, 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
Distributed compact routing

• How do you compute FIBs without global view?

How to handle interdomain routing policies

• no one knows!
Conclusion
Growth of the Internet

There's occasional concern about increasing routing table size on the Internet...

But we seem to manage one way or another. What really matters here?
Simple shortest-path routing cannot scale

Internet has to do something better than that

- And it does!: Hierarchy (e.g., routing by IP prefix)

Fundamental tradeoff between scalability and stretch of paths

- Internet’s use of hierarchy gets us down to “only” 450k forwarding entries at the cost of some latency inflation
Steven M. Bellovin, Columbia U

- “Lawful Hacking: Using Existing Vulnerabilities for Wiretapping on the Internet”
- 4 pm today in CSL B02

Next week

- SDN