Compact Routing with Slack

Michael Dinitz

Computer Science Department
Carnegie Mellon University

ACM Symposium on Principles of Distributed Computing
Portland, Oregon
August 13, 2007

Routing

- Routing in a network: nodes have IDs, packets have headers, routing decisions made based on packet header and local routing table

Routing

- Routing in a network: nodes have IDs, packets have headers, routing decisions made based on packet header and local routing table
- Tradeoff between size of routing table and optimality of routes

Routing

- Routing in a network: nodes have IDs, packets have headers, routing decisions made based on packet header and local routing table
- Tradeoff between size of routing table and optimality of routes
- Compact routing schemes try to have small stretch with small routing tables
- Stretch: $\max_{u,v} \frac{d_R(u,v)}{d(u,v)}$

Routing models

- Labels:
  - Labeled: scheme designer gets to choose labels (names) of nodes
  - Name-independent: node names assigned arbitrarily

Routing models

- Labels:
  - Labeled: scheme designer gets to choose labels (names) of nodes
  - Name-independent: node names assigned arbitrarily
- Ports:
  - Designer port: scheme designer assigns links to ports
  - Fixed port: links assigned to arbitrary ports

Previous Research

- Lots of work on routing in all of these models

Previous Research

- Lots of work on routing in all of these models
- Result that we use:
- Thorup-Zwick labeled schemes
  - Tree routing: Stretch 1 (exact routes), small labels
  - General routing: stretch $4k-5$, $\tilde{O}(n^{1/k})$ space, $o(k\log^2 n)$-bit labels, and $o(\log^2 n)$-bit headers

Previous Research

- Lots of work on routing in all of these models
- Result that we use:
- Thorup-Zwick labeled schemes
  - Tree routing: Stretch 1 (exact routes), small labels
  - General routing: stretch $4k - 5$, $\tilde{O}(n^{1/k})$ space, $o(k \log^2 n)$-bit labels, and $o(\log^2 n)$-bit headers
- Essentially matching upper and lower bounds for all of the models (see Cyril Gavoille's talk at LOCALITY for more)

What now?

- What if large worst case stretch is only caused by a few really bad pairs?

What now?

- What if large worst case stretch is only caused by a few really bad pairs?
- Would like to make claims of the form "ignoring a small number of pairs, have very small stretch on the rest"

What now?

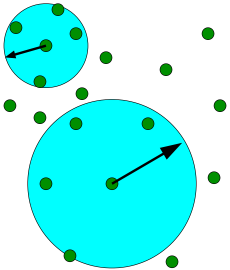
- What if large worst case stretch is only caused by a few really bad pairs?
- Would like to make claims of the form "ignoring a small number of pairs, have very small stretch on the rest"
- Studied before:
  - Metric embeddings: Kleinberg-Slivkins-Wexler '04, ABCDGKNS '05, Abraham-Bartal-Neiman '06, '07
  - Distance oracles/labels: Chan-D-Gupta '06, ABN '06
  - Spanners: CDG '06

ϵ-Neighborhoods

Definition

Given $0 < \epsilon < 1$, for any point $v \in V$, the ϵ-neighborhood $N_\epsilon(v)$ consists of the closest ϵn points to v

- $R(v, \epsilon) = \min\{r : |B(v, r)| \geq \epsilon n\}$
- v is ϵ-*far* from u if $d(u, v) > R(u, \epsilon)$



Slack definitions

- Given a weighted graph $G = (V, E)$

Slack definitions

- Given a weighted graph $G = (V, E)$

Definition (Uniform slack routing scheme)

A routing scheme R has ϵ-uniform slack and stretch α if $d_R(u, v) \leq \alpha d(u, v)$ for all $u, v \in V$ such that v is ϵ-far from u

Slack definitions

- Given a weighted graph $G = (V, E)$

Definition (Uniform slack routing scheme)

A routing scheme R has ϵ-uniform slack and stretch α if $d_R(u, v) \leq \alpha d(u, v)$ for all $u, v \in V$ such that v is ϵ-far from u

- One scheme that works for all ϵ:

Slack definitions

- Given a weighted graph $G = (V, E)$

Definition (Uniform slack routing scheme)

A routing scheme R has ϵ-uniform slack and stretch α if $d_R(u, v) \leq \alpha d(u, v)$ for all $u, v \in V$ such that v is ϵ-far from u

- One scheme that works for all ϵ:

Definition (Gracefully degrading routing scheme)

A routing scheme R is *gracefully degrading* with stretch α if for all $0 < \epsilon < 1$ it has ϵ-slack and stretch α

Our Results: Labeled Routing

- Uniform slack and gracefully degrading schemes in both fixed port and designer port models

Our Results: Labeled Routing

- Uniform slack and gracefully degrading schemes in both fixed port and designer port models

	ϵ-Uniform Slack	Gracefully Degrading
Stretch	$24k - 25$	$O(\log \frac{1}{\epsilon})$
Table Size	$O(\frac{1}{\epsilon^{4/k}} \frac{\log^{3-1/k} \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}} + \log n)$	$O(\log^4 n)$
Headers	$O(\frac{\log^2 n}{\log\log n})$	$O(\frac{\log^2 n}{\log\log n})$
Labels	$O(\frac{\log^2 n}{\log\log n}) + O(k \frac{\log^2 \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}})$	$O(\log^4 n)$

Our Results: Labeled Routing

- Uniform slack and gracefully degrading schemes in both fixed port and designer port models

	ϵ-Uniform Slack	Gracefully Degrading
Stretch	$24k - 25$	$O(\log \frac{1}{\epsilon})$
Table Size	$O(\frac{1}{\epsilon^{4/k}} \frac{\log^{3-1/k} \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}} + \log n)$	$O(\log^4 n)$
Headers	$O(\frac{\log^2 n}{\log\log n})$	$O(\frac{\log^2 n}{\log\log n})$
Labels	$O(\frac{\log^2 n}{\log\log n}) + O(k \frac{\log^2 \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}})$	$O(\log^4 n)$

- Note: setting $k = \Theta(\log \frac{1}{\epsilon})$ gives slack scheme with all parameters polylogarithmic in n and $\frac{1}{\epsilon}$

Our Results: Labeled Routing

- Uniform slack and gracefully degrading schemes in both fixed port and designer port models

	ϵ-Uniform Slack	Gracefully Degrading
Stretch	$24k - 25$	$O(\log \frac{1}{\epsilon})$
Table Size	$O(\frac{1}{\epsilon^{4/k}} \frac{\log^{3-1/k} \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}} + \log n)$	$O(\log^4 n)$
Headers	$O(\frac{\log^2 n}{\log\log n})$	$O(\frac{\log^2 n}{\log\log n})$
Labels	$O(\frac{\log^2 n}{\log\log n}) + O(k \frac{\log^2 \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}})$	$O(\log^4 n)$

- Note: setting $k = \Theta(\log \frac{1}{\epsilon})$ gives slack scheme with all parameters polylogarithmic in n and $\frac{1}{\epsilon}$
- Similar bounds discovered independently by Abraham, Bartal, and Neiman

Density Nets

- Intuition: Small set of points that approximates the distances well

Density Nets

- Intuition: Small set of points that approximates the distances well

Definition

An ϵ-*density net* is a subset N of V such that

1. For all $x \in V$, there is some $y \in N$ s.t. $d(x, y) \leq 2R(x, \epsilon)$
2. $|N| \leq \frac{1}{\epsilon}$
3. For all $u, v \in N$, $N_\epsilon(u) \cap N_\epsilon(v) = \emptyset$

Density Nets

- Intuition: Small set of points that approximates the distances well

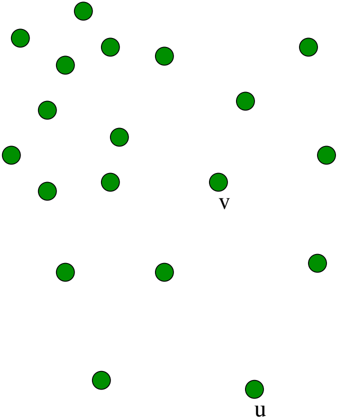
Definition

An ϵ-*density net* is a subset N of V such that

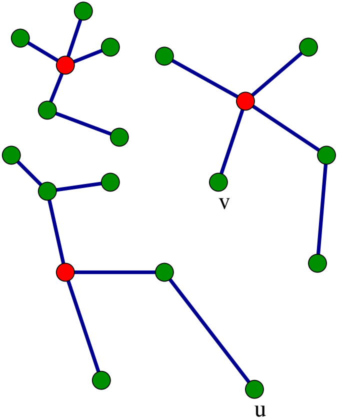
1. For all $x \in V$, there is some $y \in N$ s.t. $d(x, y) \leq 2R(x, \epsilon)$
2. $|N| \leq \frac{1}{\epsilon}$
3. For all $u, v \in N$, $N_\epsilon(u) \cap N_\epsilon(v) = \emptyset$

- Always exist, can be constructed in polynomial time [Chan D Gupta '06]

Slack Labeled Scheme

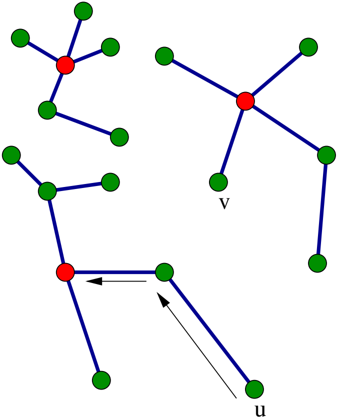


Slack Labeled Scheme



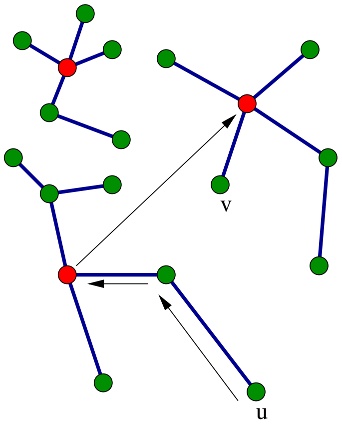
- Create an ϵ-density net N and shortest path trees out of net
- $u, v \in V$ such that v is ϵ-far from u

Slack Labeled Scheme



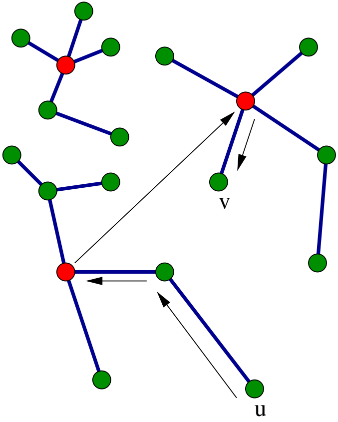
- Create an ϵ-density net N and shortest path trees out of net
- $u, v \in V$ such that v is ϵ-far from u
- Phase 1: Route to closest node in density net ($\leq 2R(u, \epsilon) \leq 2d(u, v)$)

Slack Labeled Scheme



- Create an ϵ-density net N and shortest path trees out of net
- $u, v \in V$ such that v is ϵ-far from u
- Phase 1: Route to closest node in density net ($\leq 2R(u, \epsilon) \leq 2d(u, v)$)
- Phase 2: Route "inside the net" to net node closest to v (name in label of v) ($\leq 6d(u, v)$)

Slack Labeled Scheme



- Create an ϵ-density net N and shortest path trees out of net
- $u, v \in V$ such that v is ϵ-far from u
- Phase 1: Route to closest node in density net $(\leq 2R(u, \epsilon) \leq 2d(u, v))$
- Phase 2: Route "inside the net" to net node closest to v (name in label of v) $(\leq 6d(u, v))$
- Phase 3: Use a tree routing scheme to go down the tree to destination $(\leq 3d(u, v))$

Routing Inside the Net

- Nodes in net (probably) not adjacent – what is "routing inside the net"?

Routing Inside the Net

- Nodes in net (probably) not adjacent – what is "routing inside the net"?
- Coppersmith-Elkin distance preserver trick:
  - Graph of shortest paths between net nodes
  - At most $O(\frac{1}{\epsilon^4})$ intermediate nodes of degree > 2
- Use Thorup-Zwick on these nodes + net nodes
- Degree 2 intermediate nodes handled trivially

Gracefully Degrading Scheme

- $\log n$ levels: $\epsilon_i = 1/2^i$
- Different density net at each level

Gracefully Degrading Scheme

- $\log n$ levels: $\epsilon_i = 1/2^i$
- Different density net at each level
- Create Thorup-Zwick distance labels for each net
- Every node's label includes distance label of closest net node – allows us to find out distance in each net
- Choose and route along the level with smallest total distance

Gracefully Degrading Scheme

- $\log n$ levels: $\epsilon_i = 1/2^i$
- Different density net at each level
- Create Thorup-Zwick distance labels for each net
- Every node's label includes distance label of closest net node – allows us to find out distance in each net
- Choose and route along the level with smallest total distance
- This scheme has constant average stretch (implied by gracefully degrading)

Our Results: Name-Independent Routing

- Uniform slack scheme in designer-port model, lower bound for uniform slack in fixed-port model

Our Results: Name-Independent Routing

- Uniform slack scheme in designer-port model, lower bound for uniform slack in fixed-port model

	Designer Port	Fixed Port
Stretch	27	$2k-1$
Table Size	$O(\frac{1}{\epsilon}\log^2 n + \log^4 n)$	$\Omega(n^{1/k})$
Headers	$O(\frac{\log^2 n}{\log\log n})$	

Name-Independent Difficulties

- Problem: how do we know which net node is closest to destination?

Name-Independent Difficulties

- Problem: how do we know which net node is closest to destination?
- Solution: distributed data structure to find label of destination

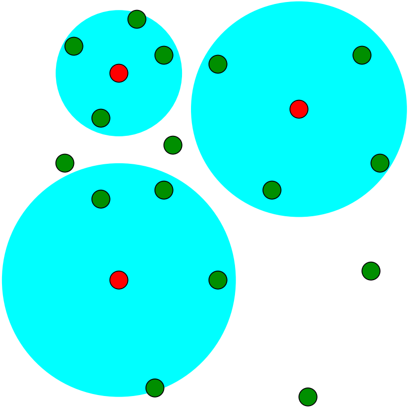
Name-Independent Difficulties

- Problem: how do we know which net node is closest to destination?
- Solution: distributed data structure to find label of destination
- Problem: even tree routing is hard – how do we make such a data structure?

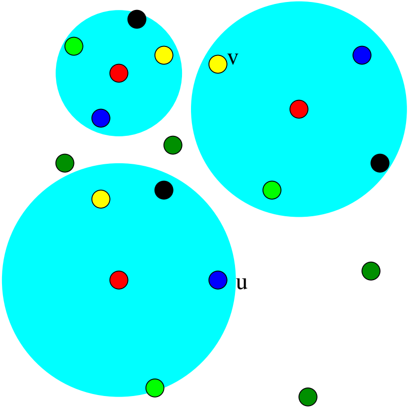
Name-Independent Difficulties

- Problem: how do we know which net node is closest to destination?
- Solution: distributed data structure to find label of destination
- Problem: even tree routing is hard – how do we make such a data structure?
- Solution: additive tree routing scheme

Distributing Labels (cont'd)

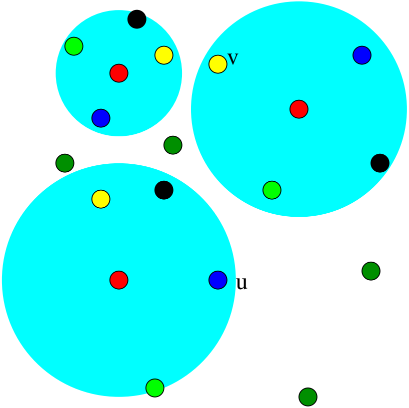


Distributing Labels (cont'd)



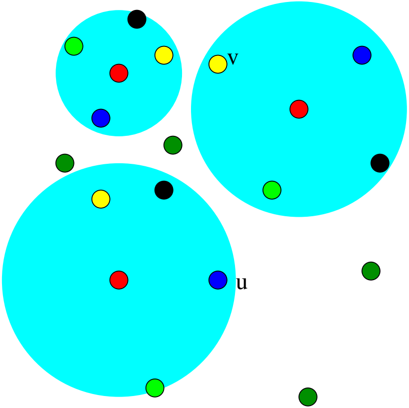
- For each $x \in N$, assign each node in $N_\epsilon(x))$ a different "color" in $[\epsilon n]$

Distributing Labels (cont'd)



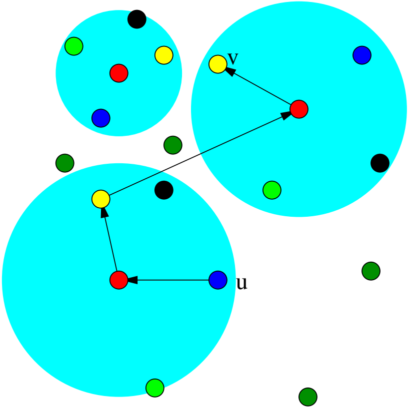
- For each $x \in N$, assign each node in $N_\epsilon(x))$ a different "color" in $[\epsilon n]$
- $h : V \to [\epsilon n]$ any easy to compute balanced hash function (e.g. ID(v) mod ϵn)

Distributing Labels (cont'd)



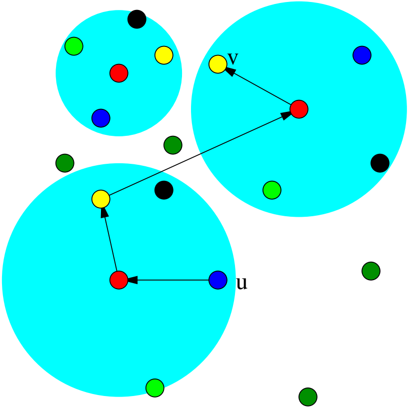
- For each $x \in N$, assign each node in $N_\epsilon(x)$) a different "color" in $[\epsilon n]$
- $h : V \longrightarrow [\epsilon n]$ any easy to compute balanced hash function (e.g. ID(v) mod ϵn)
- Node x stores the labels of all nodes y such that $h(y) = color(x)$

Distributing Labels (cont'd)



- For each $x \in N$, assign each node in $N_\epsilon(x))$ a different "color" in $[\epsilon n]$
- $h : V \rightarrow [\epsilon n]$ any easy to compute balanced hash function (e.g. ID(v) mod ϵn)
- Node x stores the labels of all nodes y such that $h(y) = color(x)$

Distributing Labels (cont'd)



- For each $x \in N$, assign each node in $N_\epsilon(x))$ a different "color" in $[\epsilon n]$
- $h : V \rightarrow [\epsilon n]$ any easy to compute balanced hash function (e.g. ID(v) mod ϵn)
- Node x stores the labels of all nodes y such that $h(y) = color(x)$
- New problem: how to route to node x with the right color

Tree Routing

- In name-independent model, tree routing is basically no easier than general routing

Tree Routing

- In name-independent model, tree routing is basically no easier than general routing
- AGM '04: Name-independent, designer-port tree routing scheme on unweighted trees with additive distortion twice the depth of the tree

Tree Routing

- In name-independent model, tree routing is basically no easier than general routing
- AGM '04: Name-independent, designer-port tree routing scheme on unweighted trees with additive distortion twice the depth of the tree
- Reason for depth: travels to an intermediate node first (along shortest path)

Tree Routing

- In name-independent model, tree routing is basically no easier than general routing
- AGM '04: Name-independent, designer-port tree routing scheme on unweighted trees with additive distortion twice the depth of the tree
- Reason for depth: travels to an intermediate node first (along shortest path)
- But by properties of density net, depth of the tree isn't too large compared to $d(u, v)$!
- So use this scheme to find node of the right color

Fixed-Port Lower Bound

- AGM '06: Any name-independent fixed-port routing scheme with constant average stretch requires polynomial space
- Rules out good gracefully degrading schemes, but not slack schemes

Fixed-Port Lower Bound

- AGM '06: Any name-independent fixed-port routing scheme with constant average stretch requires polynomial space
- Rules out good gracefully degrading schemes, but not slack schemes
- Easy extension:

Theorem

There exists a graph such that for constant $0 < \epsilon < 1/2$, every ϵ-uniform slack scheme with stretch $2k - 1$ uses at least $\Omega(n^{1/k})$ space

Conclusions and Open Questions

- Density nets make labeled routing with slack straightforward (and spanners, and distance oracles, and ...)
- In name-independent model, large difference between designer-port and fixed-port schemes

Conclusions and Open Questions

- Density nets make labeled routing with slack straightforward (and spanners, and distance oracles, and ...)
- In name-independent model, large difference between designer-port and fixed-port schemes
- Open questions:
  - Gracefully degrading name-independent designer-port scheme?
  - Lower bounds for for ϵ-slack instead of ϵ-uniform slack in name-independent fixed-port model?

Thank you!