Compact Routing with Name Independence

Henri Dubois-Ferrière

Distributed Computing Seminar

ETHZ 20/1/2004

1. Introduction and Background

About the Paper and Topic

- Paper: “Compact Routing with Name Independence”
  - M. Arias, L. Cowen, K. Laing, R. Rajaraman, O. Taka, SPAA 2003
  - Several existing proposals.
  - This one offers best bounds to date (in a particular setting).
- Topic: Compact Routing
  - Reduce size of routing table size, at the cost of suboptimal route lengths.
  - Trade off route lengths for space
    - As opposed to approximate all-pairs shortest paths, which trades off route length for time.
  - Several existing proposals

Historical Context

- Early work on compact routing (~ 1985)
  - Network specific schemes
    - i.e., ring, tree, grid considered in isolation.
- Universal schemes (~ 1989)
  - Worked on general graphs
  - Bounds on *average* RT size
- More recent work
  - Bounds on *maximal* RT size
  - Name-independent routing

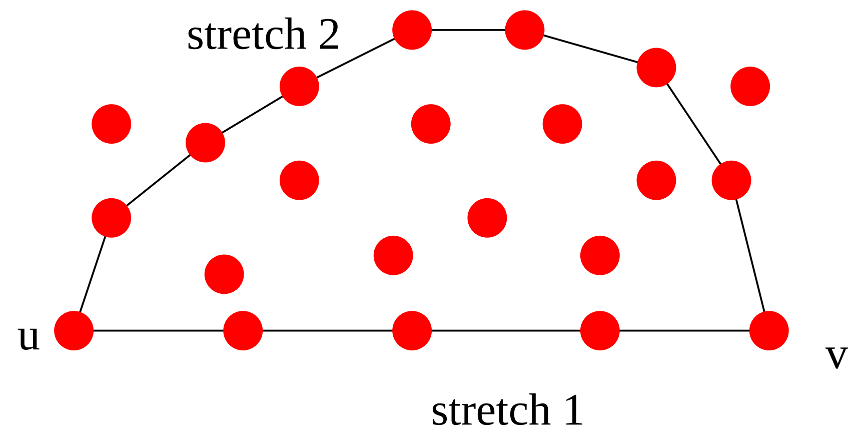
Compact Routing Taxonomy

- Node naming:
  - Name Independent *(harder):* nodes have arbitrary, fixed *names* with no topological information.
  - Topology Dependent *(easier):* Nodes can be assigned topologically relevant *addresses* (i.e., internet).
- Link naming:
  - Fixed-port *(harder)*: outgoing links (ports) at each node have arbitrary, non-topological names.
  - Designer-port *(easier)*: ports can be named by the algorithm (i.e., label each port with the name of node on other end)
- Re-writable vs. fixed packet headers
  - Notion of read-only packet seems somewhat esoteric…
- This work is concerned with ***name independent, fixed-port compact routing with rewritable headers***. (the hardest setup)

Quantities of Interest

- CR scheme is characterized by 3 quantities:
  - *Stretch:* $\frac{|p(u,v)|}{d(u,v)}$
  - *Storage:* size of routing tables
  - *Packet header size*
- Example:

  Shortest-path routing:
  - Stretch 1
  - O(nlogn) routing tables.
  - O(logn) headers



- Note: Graph considered is weighted and undirected.

Performance of Name-Indep. Schemes

	Table Size	Header Size	Stretch	
[1]	$\tilde{O}\left(n^{1/2}\right)$	$O(\log n)$	2592	
[1]	$\tilde{O}\left(n^{2/3}\right)$	$O(\log n)$	486	
[3]	$\tilde{O}\left(n^{1/2}\right)$	$O(\log n)$	1088	
[3]	$\tilde{O}\left(n^{2/3}\right)$	$O(\log n)$	624	
This paper	$\tilde{O}\left(n^{1/2}\right)$	$O(\log^2 n)$	**5**	Covered here
This paper	$\tilde{O}\left(n^{1/2}\right)$	$O(\log n)$	**7**	
This paper	$\tilde{O}\left(n^{2/3}\right)$	$O(\log n)$	**5**	
Lower Bound [9]	$o(n)$	$\log_2 n$	3	

- [1] : Awerbuch et al, 1989
- [3] : Awerbuch et al, 1990
- [9] : Gavoille et al, 1997 (any routing scheme using sublinear space has stretch ≥ 3)

2. Name-Independent Compact Routing with Stretch 5

High-level view

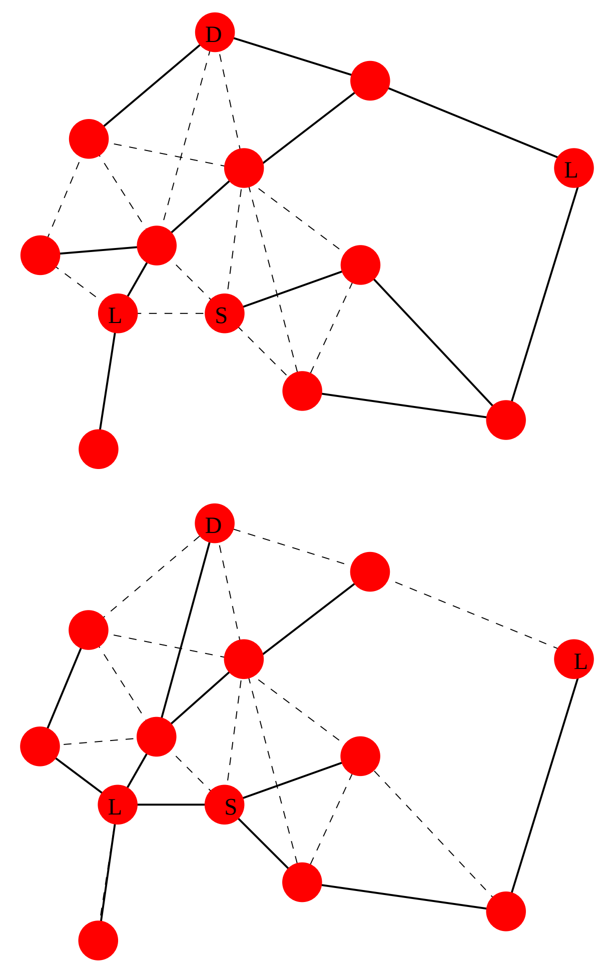
- Select “a few” landmark nodes.
- Keep name-independent shortest-path routes to:
  - Subset of “close” nodes
  - All landmarks
- Use topology-dependent shortest-path spanning trees rooted at each landmark, for which there exist small routing tables
- Reuse parts of prior work
  - Result on topology-dep. routing over trees with O(1) tables
  - Result on size of a “well-distributed” landmark set
  - Result on distribution of nodes for lookup

Topology-dependent CR on a Tree

- For any tree T, there is a routing scheme that provides optimal (stretch 1) routes, with:
  - Õ(1) storage
  - $O(\log^2 n)$ headers
- Note: If we require $\tilde{O}(n^{½})$ storage, then we can afford up to $\tilde{O}(n^{½})$ such trees in our scheme.
- Prior result from
  - Fraigniaud et al, 2001
  - Thorup et al, 2001

High-level example

- S must route to D
- We have two landmarks
- Nodes have optimal route to each landmark
- Landmarks have optimal route to each node
- The hard part is figuring out:
  - Which landmark to route through
  - What is the topology-dep. address of D in chosen landmark's tree.



The Landmark Set

- How many?
  - If “too many” (e.g. O(n)), storage requirements grow too large (remember each node stores one $\tilde{O}(1)$ table per tree).
  - If “too few”, (e.g. O(1)), then avg distance to landmark grows with network size and we will not have constant stretch
  - Therefore we must have at most $\tilde{O}(n^{1/2})$ landmarks.

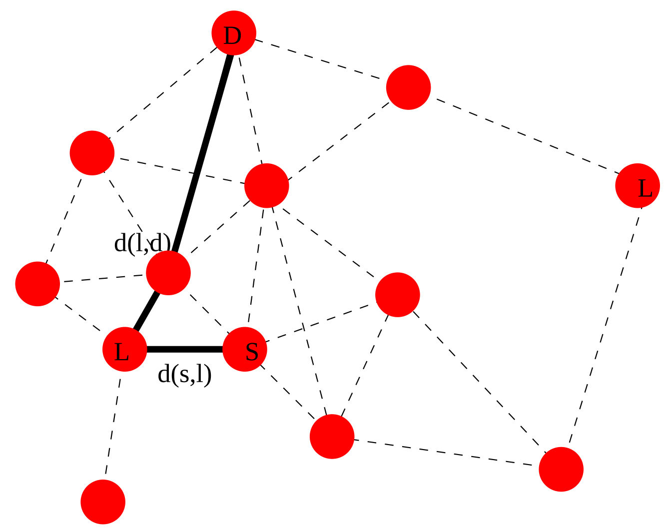
- Where?
  - Should be spread out “uniformly” – so that every node pair has a landmark which is “close” to their optimal route.

Landmark Set as a Hitting Set

- G = (V, E) : undirected graph of size n
- N(v) : set of v's $n^{½}$ closest nodes *("neighborhood ball")*
- Thm. *(hitting set)* : [Lovasz, 1975]
  - There exists a set L s.t.
    - $\forall v \in V, L \cap N(v) \neq \varnothing$ (all nodes have nearby landmark)
    - $|L| = \tilde{O}(n^{1/2})$ (sublinear size)
  - Exists an algorithm to compute L in polynomial time
- Our CR scheme makes use of any set of landmarks satisfying this theorem.
- Note: If there are $\tilde{O}(n^{½})$ landmarks, then we can afford to maintain optimal route entries to each of them

Which landmark to route through?

- So far:
  - $\tilde{O}(n^{1/2})$ landmarks
  - Nodes have optimal routes to each landmark
  - Nodes have optimal routes to nodes in neighborhood ball
  - Most routes will go through a landmark



- Pick landmark which minimizes

  d(s,l) + d(l,d) ("best" landmark)

- Remark: Can only store "best" landmark for $\tilde{O}(n^{1/2})$ destinations!
- So we need some assignment of which $\tilde{O}(n^{1/2})$ subset of destinations each node knows about

Block Set

- Lemma:
  - Given G = (V, E), |G| = n
  - N(v): set of v's $n^{½}$ closest nodes *("neighborhood ball")*
  - **Blocks**: Namespace partitioned into $n^{½}$ blocks, each of size $n^{½}$.

B_1	B_2	…	B_k

  - There exists an assignment of sets of blocks S_v to each node v such that:
    - $\forall v \in G, \forall B_i (0 \le i < n^{1/2}), \exists j \in N(v) : B_i \in S_j$
    - $\forall v \in G, |S_v| = O(\log n)$

- Each node v keeps track of the "best" landmark to reach all nodes in S_v. This takes $\tilde{O}(n^{1/2})$ space.

Storage Recap & Analysis

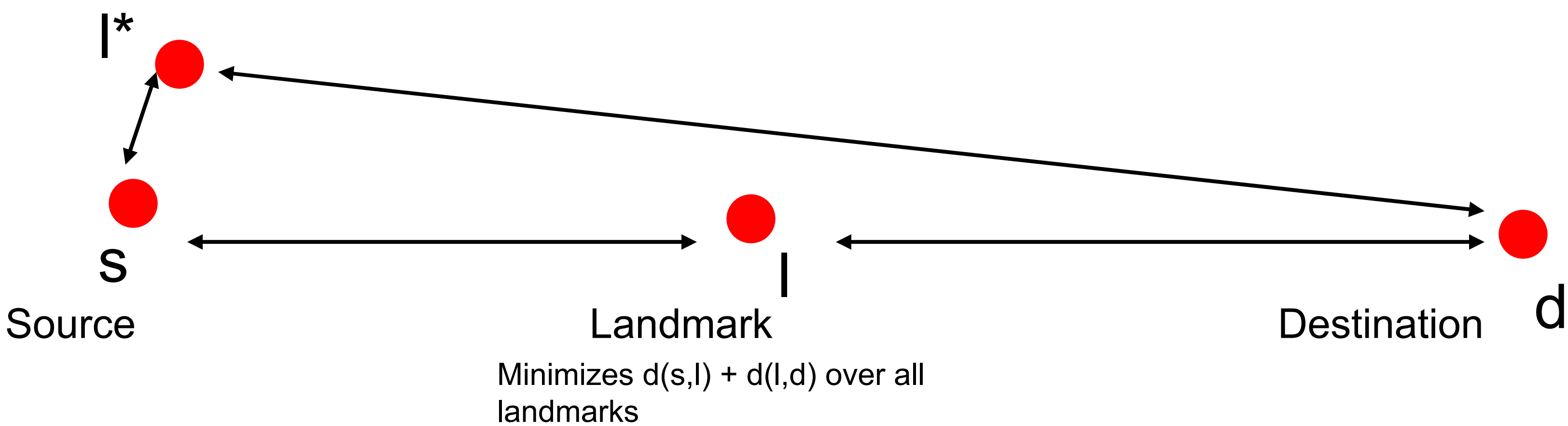
Data (at node u)	*Space*
Next-hop entries (shortest-path) to all nodes in N(u)	$O(n^{1/2})$ (Because N(u) contains by construction $n^{1/2}$ closest nodes)
Next-hop entries (shortest-path) to all landmark nodes.	$\tilde{O}(n^{1/2})$ (Because L contains by the hitting set thm $\tilde{O}(n^{1/2})$ nodes)
For each node j in S_u, the triple (j, l, addr(j,l)) where: •l minimizes d(u,l) + d(l,j) over all landmarks • addr(j,l) is the address of j in tree rooted at l	$\tilde{O}(n^{1/2})$ (S_u contains O(logn) blocks, each of size $O(n^{1/2})$)
For every landmark l, the routing table Tab(u) for the tree T_l	$\tilde{O}(n^{1/2})$ (There are $\tilde{O}(n^{1/2})$ landmarks, each routing tables is $\tilde{O}(1)$)

Routing Algorithm I

- Case $d \in N(s)$
- Easy: s can route along stretch-1 path to d (remember that we keep routing entries for all nodes in neighborhood)

Routing Algorithm II

- Case $d \notin N(s), d \in S_s$ (s knows which landmark to choose)



- Stretch is 3:

Call l^* the landmark closest to s.

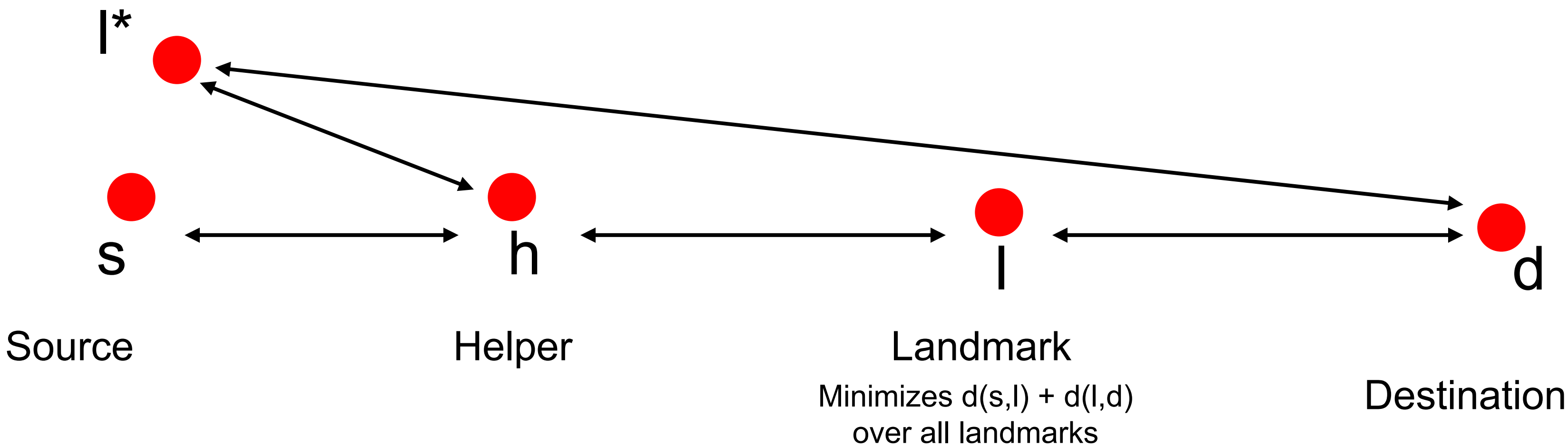
Then $d(s,l^*) \leq d(s,d)$ (because $l^* \in N(s)$, and by assumption $d \notin N(s)$)

$d(s,l) + d(l,d) \leq d(s,l^*) + d(l^*,d)$ (by construction)

$d(l^*,d) \leq d(s,l^*) + d(s,d) \leq 2d(s,d)$

Routing Algorithm II

- Case $d \notin N(s), d \notin S_s$ (s knows which landmark to choose)



- Stretch is 5:

Call l^* the landmark closest to s.

Then $d(s,h) \le d(s,d)$ (because $h \in N(s)$, and by assumption $d \notin N(s)$)

$$d(h,l^*) \le d(h,s) + d(s,l^*) \text{ (tri. inequality)}$$
$$\le 2d(s,d)$$
$$d(l^*,d) \le d(l^*,s) + d(s,d) \text{ (tri. inequality)}$$
$$\le 2d(s,d)$$
$$d(s,h) + d(h,l) + d(l,d) \le 5d(s,d)$$

3. Remaining bits, comments, and conclusion

Bits not covered

- Stretch 7 and other stretch 5 schemes
  - Similar flavor to this one
- Above schemes generalized to provide schemes with different stretch/space tradeoffs
  - $\tilde{O}(k^2 n^{2/k})$ tables
  - $\tilde{O}(\log^2 n)$ headers
  - $\min\{1 + (k - 1)(2^{k/2} - 2), 16k^2 + 4k\}$
- Method to apply these schemes when node names are picked from an arbitrary namespace (of size larger than n)

Comments and Questions

- Open questions from conclusions
  - Bridge gap to lower bound (stretch 3)
  - Study problem in dynamic context
- Comments:
  - Scheme is flat (non-hierarchical) in terms of storage, but not in terms of load (landmarks get more traffic)
  - After first lookup, can we take shorter route?
  - Maybe node names could be considered as data ids, in which case this problem (and solution) could be cast in a p2p setting?
  - Would this work if the name-space is much larger than |G|, ie each node has many labels attached to it? (we would then be close to the p2p setup)