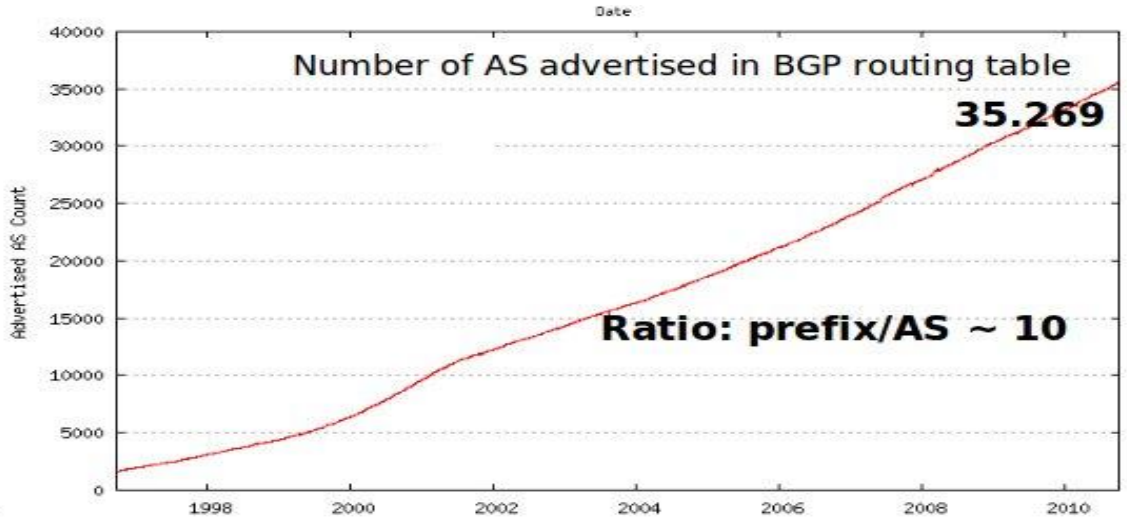
On the Communication Complexity of Distributed Name-Independant Routing Schemes

Cyril Gavoille, Christian Glacet,
Nicolas Hanusse, David Ilcinkas

Bordeaux U. – CNRS (France)

Motivation



Maintenance cost/time ↑

- Routing among the 35 000 Autonomous Systems: BGP !
- Main observations:
  - Network size increases
  - Highly Dynamic: from 1 to 1000 path change per update
  - Engineering techniques progress << Routing Tables increase

Routing & Performance

- Latency of a routing
  - Time to traverse links (stretch)
  - Lookup Time in routing tables (size of routing tables/Memory)
- Maintenance cost: #messages to update tables
- Maintenance time: convergence time

Stretch = Path length of the routing /distance

Our model

- A weighted graph G of weights in [1,W]
  - n nodes, m edges
  - Hop distance hd(u,v) = minimum number of hops of shortest paths between u and v
  - Hop diameter D = max hd(u,v)
- Two distributed models:
  - LOCAL: delay = 1
  - ASYNC: delay < 1
- Name-independant Schemes: relabeling nodes is not allowed

What can be achieved ?

- Lower bounds !
  - Stretch = 1 → $\Omega(n)$ entries per node can be required [G.-Perennes 1996]
  - Stretch < 5 → $\Omega(n^{1/2})$ entries per node can be required [Abraham-Gavoille-Malkhi 2006]
- Upper bounds !
  - Stretch 1, $O(n^2 \log n)$ messages *but* time $\Theta(D \log n)$ [Afek, Ricklin 1993]
  - Stretch 12k+3, $O(k^3 n^{1/k})$ entries *on average for unweighted graphs* [Peleg – Upfal 1989]
  - Stretch 2^k-1 and $O(kn^{1/k})$ entries *on average* [Awerbuch et al. 1990]
  - Stretch k^2 and $O(n^{2/k})$ entries per node [Arias et al. 2006]
  - Stretch $O(k)$ and $O(n^{1/k})$ entries per node [Abraham, Gavoille, Malkhi 2006]
- Distance/path vector has conv. time D and $\Omega(mn)$ messages
- Compact routing schemes: centralized or synchronous !!

Our goal:
How to build tables from scratch ?

Universal:

→ Any topology, asynchronous network

Fast distributed computation:

→ Convergence time O(D).

Light consumption:

→ Memory: $O(n^{1/k})$

→ Messages: $o(n^2 \log n)$ for a synchronous version

Guarantee on routing:

→ Stretch O(k)

Our results

Asynchronous distributed name-independant routing schemes

- ✓ Stretch 7
- ✓ Convergence Time O(D)
- ✓ Memory $O(n^{1/2})$

In a synchronous scenario

- ✓ $O(\xi\,(m\,n^{1/2}) + n^{3/2}\,\mathit{min}(D, n^{1/2}))$ messages

ξ=1+D(1-1/W) for weighted graphs

ξ=1 for unweighted graphs

Our results for « realistic graphs » in a synchronous scenario

Schemes	Stretch	Memory	#Messages	Time	
Dist/Path Vector	1	$\Omega(n)$	$O(n^2)$	O(D)	
ALGO 1	7	$O(n^{1/2})$	$O(n^{3/2})$	O(D)	
ALGO 2	5	$O(n^{2/3})$	$O(n^{5/3})$	O(D)	
Memory LB	< 2k+1	$\Omega((n \log n)^{1/k})$	Any	Any	Abraham-Gavoille-Malkhi 2006
Message LB	1	Any	$\Omega(n^2)$	o(n)	
Time LB	< n/3D	Any	Any	$\Omega(D)$	

« realistic graphs »: O(n log n) edges, hop-diameter O(log n) and unweighted.

Our results for « realistic graphs » in a synchronous scenario

Schemes	Stretch	Memory	#Messages	Time	
Dist/Path Vector	1	$\Omega(n)$	$O(n^2)$	$O(D)$	
ALGO 1	7	$O(n^{1/2})$	$O(n^{3/2})$	$O(D)$	
ALGO 2	5	$O(n^{2/3})$	$O(n^{5/3})$	$O(D)$	
Memory LB	$< 2k+1$	$\Omega((n \log n)^{1/k})$	Any	Any	Abraham-Gavoille-Malkhi 2006
Message LB	1	Any	$\Omega(n^2)$	$o(n)$	
Time LB	$< n/3D$	Any	Any	$\Omega(D)$	

« realistic graphs »: O(n log n) edges, hop-diameter O(log n) and unweighted.

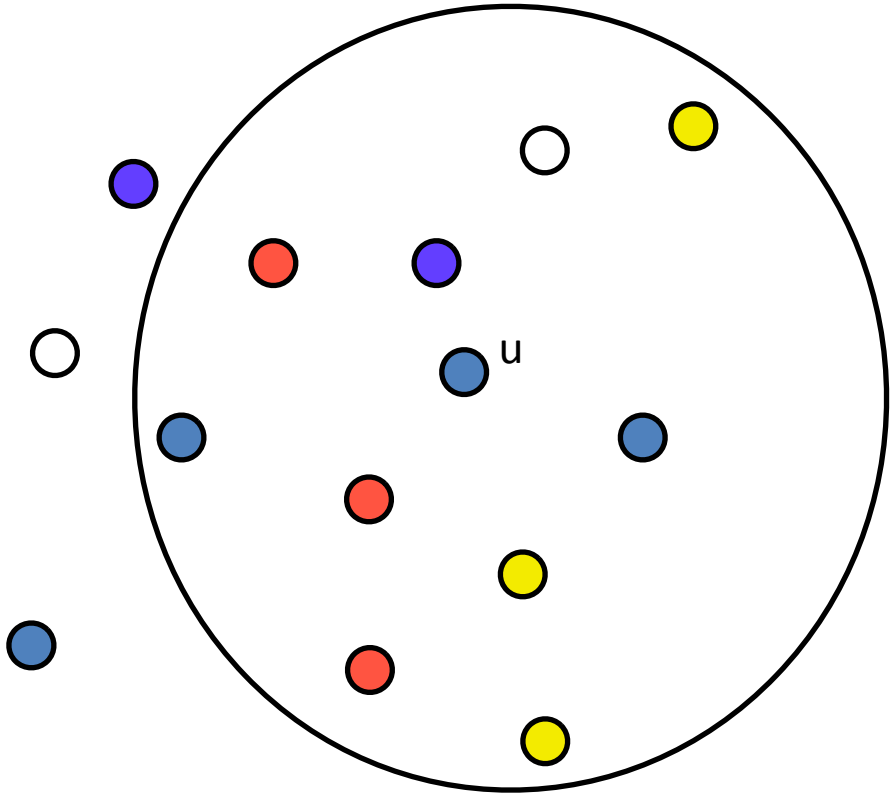
To sum up

inspired by Abraham-Gavoille-Malkhi-Nisan-Thorup 2008

- Each node has a color c(u) among k
- Random hash function h: identifier $\longrightarrow$ [1,k]
- Some nodes are of type **landmark** (can be color 1)
- Complete and minimal **vicinity balls** B(u): closest nodes with at least one node of each color and one landmark L(u).
- **Color Table** C(u): « paths » to v such that h(v)=c(u)
- 2 routing mecanisms:
  - Direct: within balls and toward landmarks
  - Indirect: through intermediate nodes/landmarks

ALGO 1 ingredients

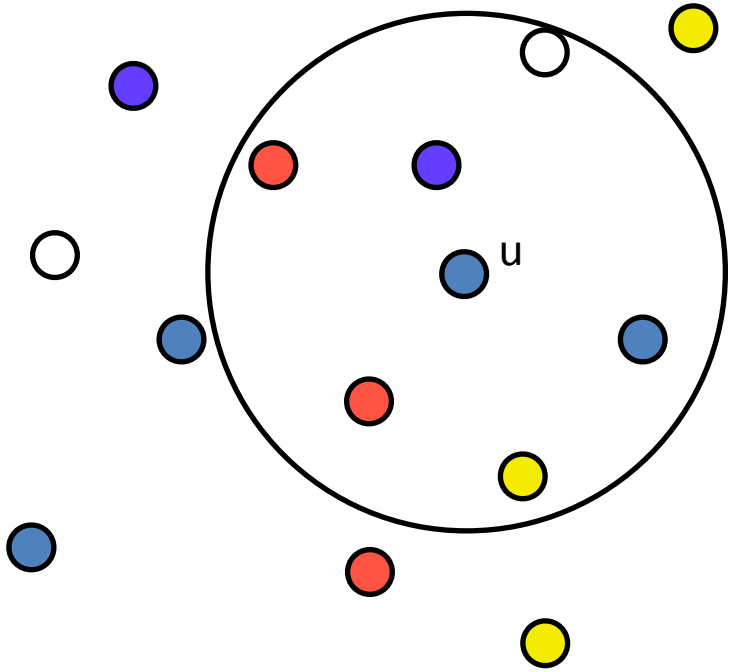
Vicinity Balls B



B(u) - closest nodes from u respecting two properties :

- **complete**, contains at least one of each color,

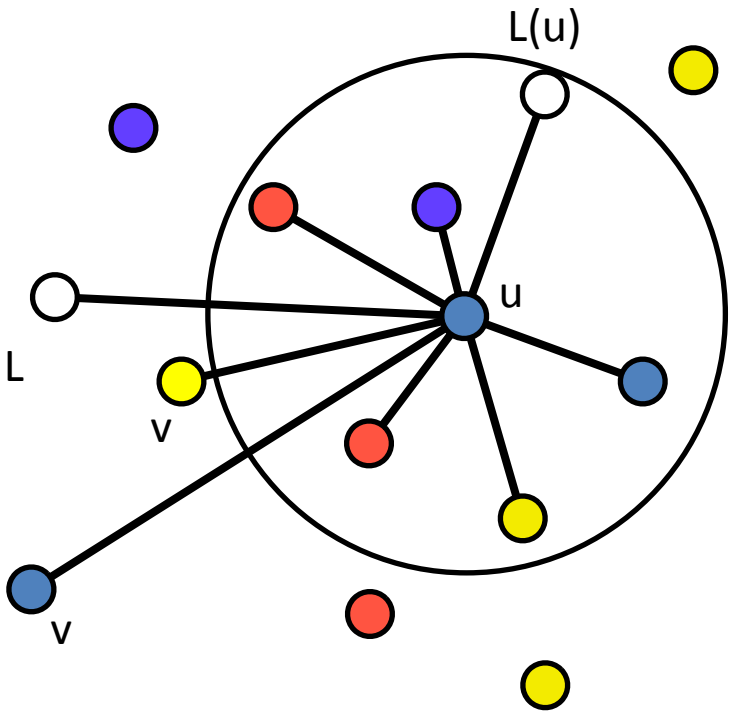
ALGO 1 ingredients



B(u) - closest nodes from u respecting two properties :

- **complete**, contains at least one of each color,
- **minimal** in number of nodes

ALGO 1 routes

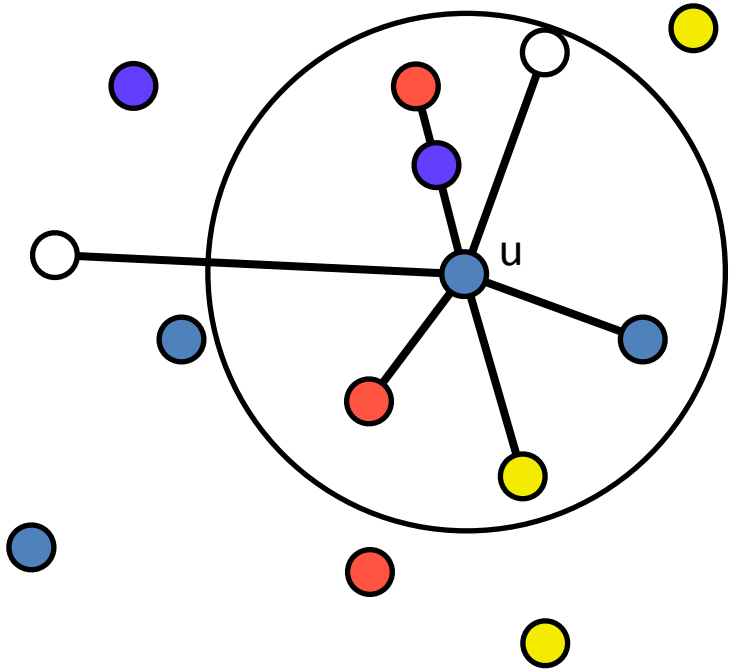


Every node u has to know routes to :

- B(u), close nodes
- L, landmarks
- C(u), nodes v managed by color c (u)=h(v)

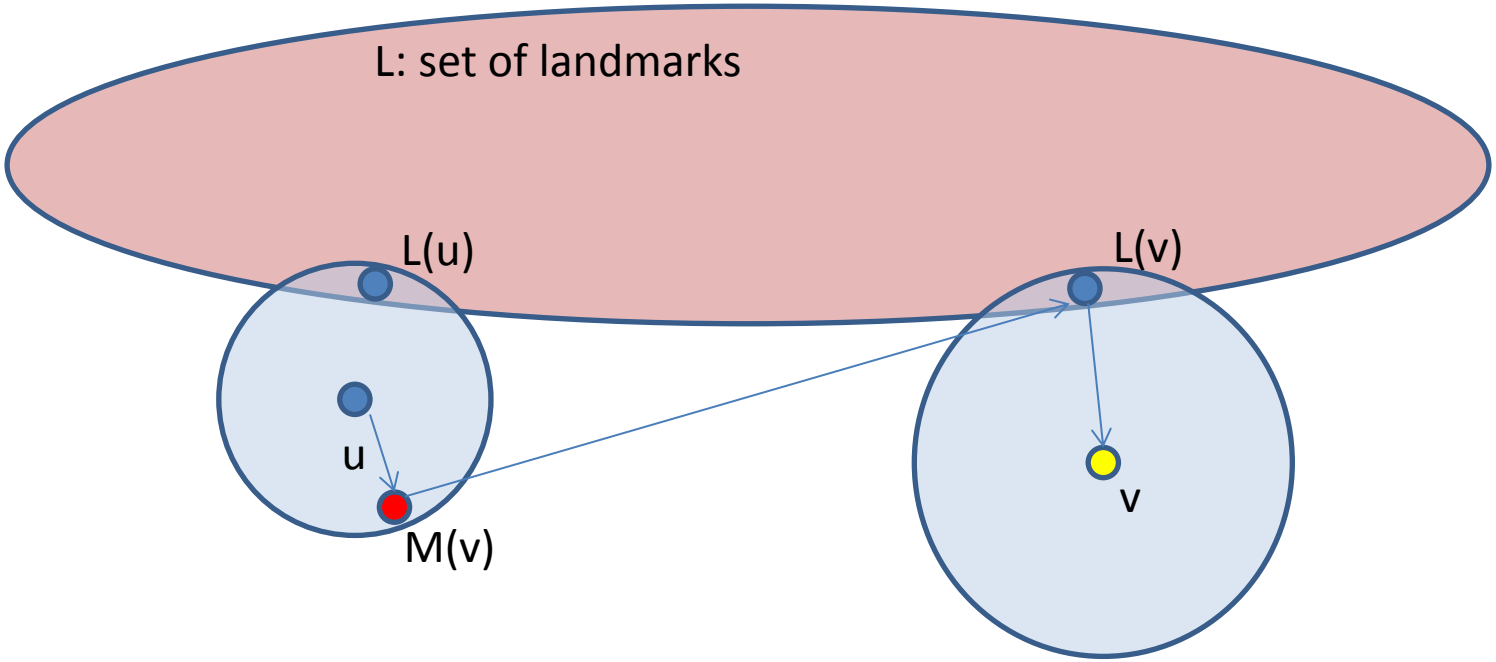
we also define L(u) as the closest landmark of u.

Route within a vicinity ball



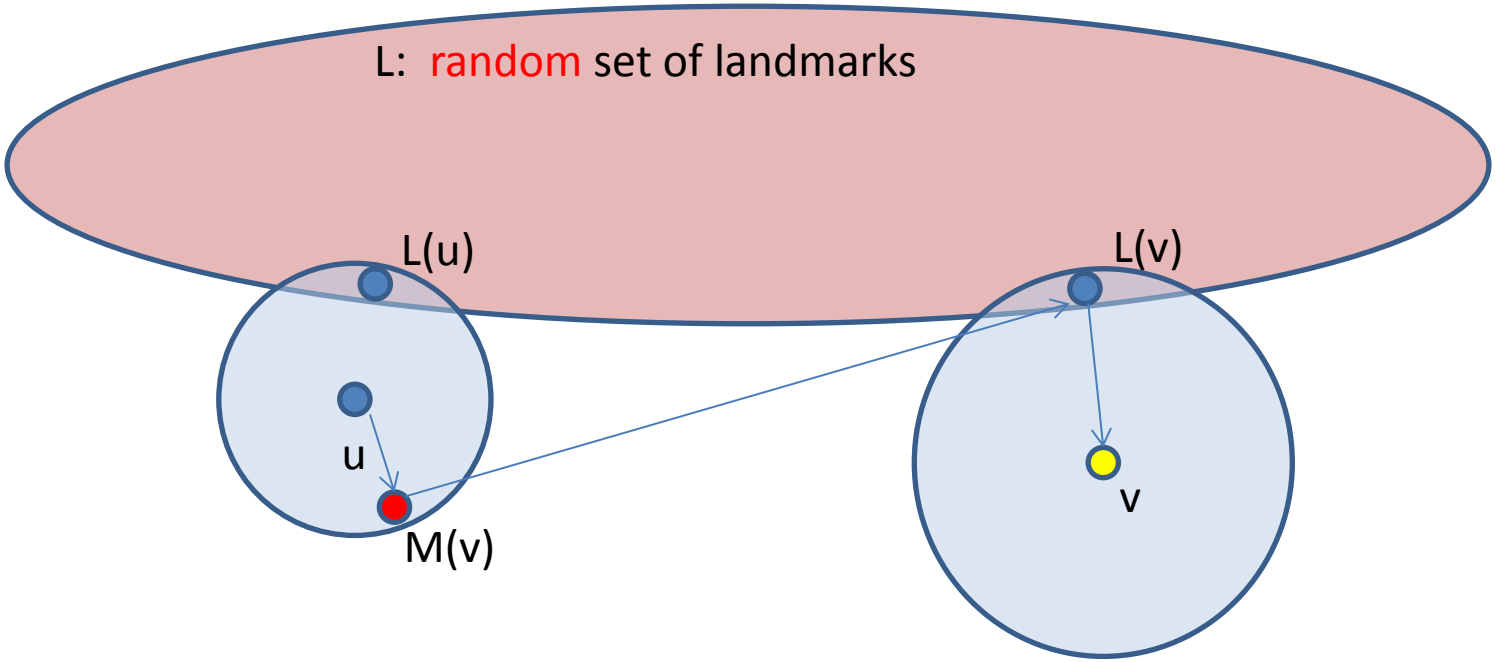
The routing protocol to route to a destination in B(u) or L is a shortest path routing.

How to route further



1. Source u computes h(v)=c
2. Forward message to M(c), its manager of color c in B(u)
3. M(c) knows Path P=L(v) → v and forward message to L(v)
4. Message follows P to reach v

Random coloring



Each node chooses its color uniformly at random

→ between n/2k and 3n/2k nodes per color w.h.p.

→ B has size Θ(k log k) w.h.p.

→ L has size Θ(n/k) w.h.p

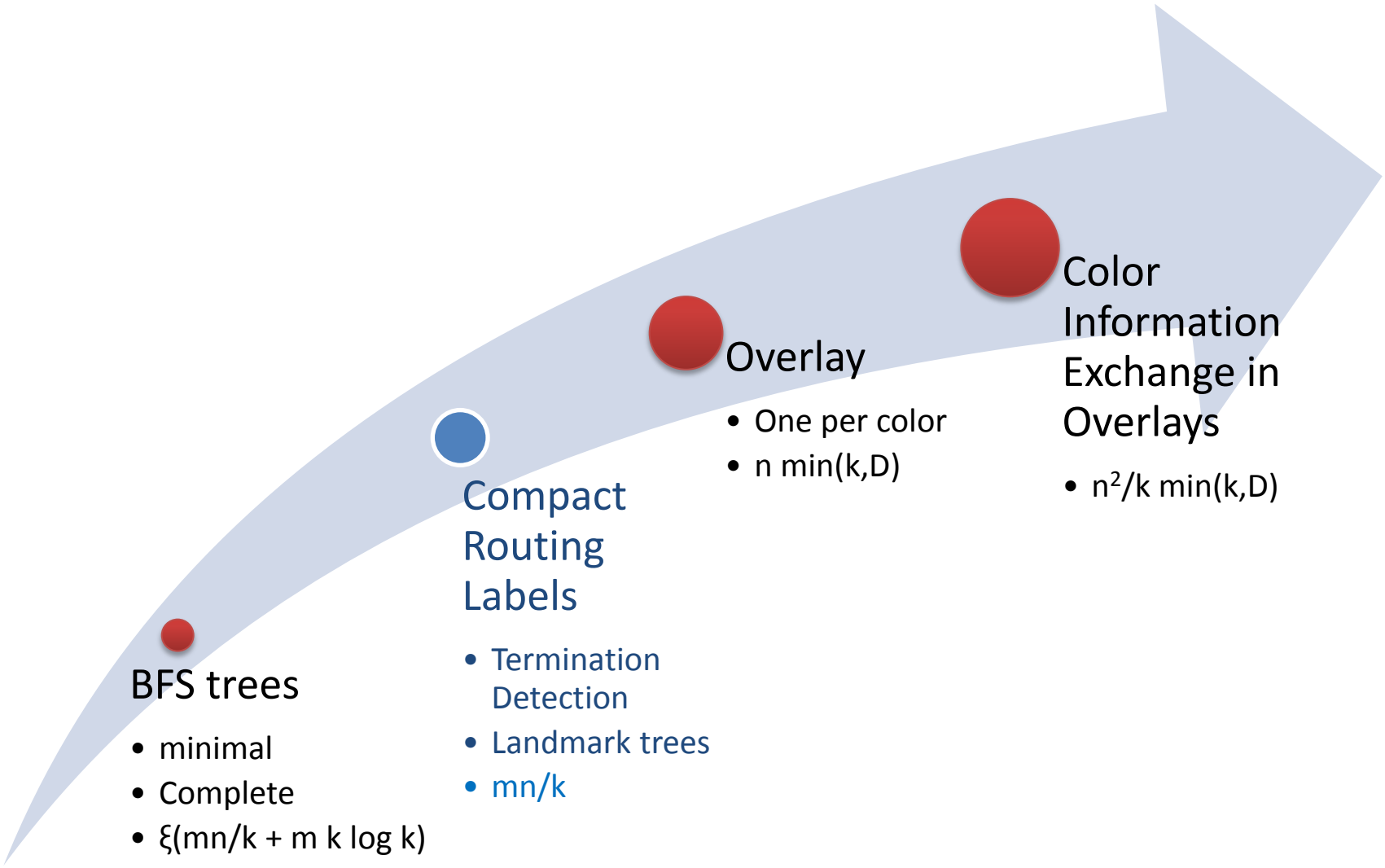
How to build such distributed data structures with a low communication cost ?

Synchronization of few asynchronous phases:

1. Detection of landmarks and election of one leader tree
2. Vicinity Balls and Landmarks trees: bellman-ford based
3. Compact Routing labels assignements
4. Color tables construction using overlay networks (one per color) from the leader tree

Phases 1, 2 are done in parallel before phases 3 and phase 4.

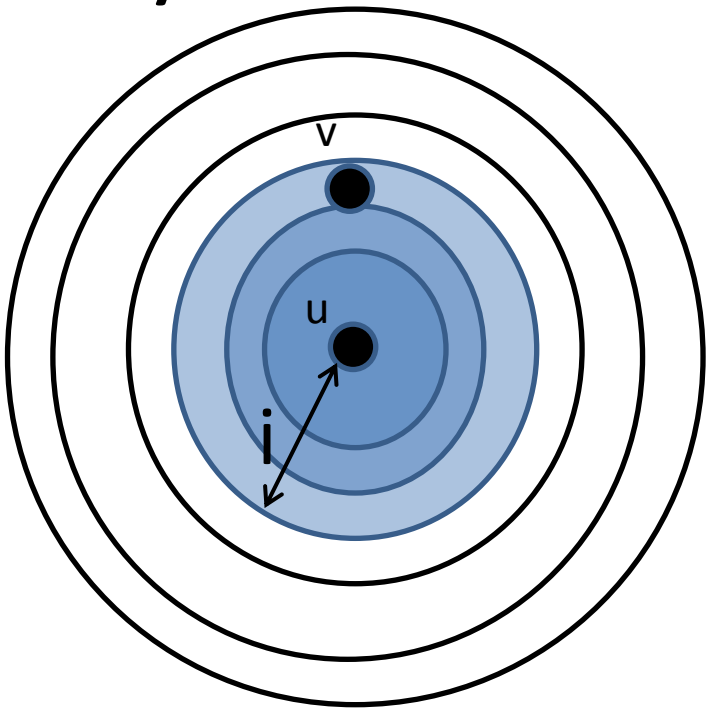
Details on the communication cost



Truncated Bellman-Ford for non landmarks nodes

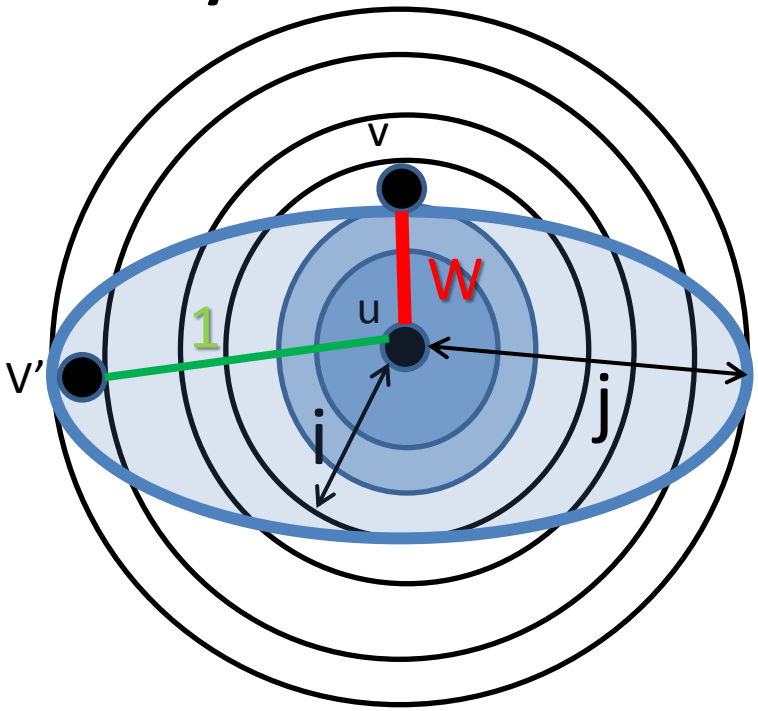
- Entries of B:
  - Id, distance,hop-distance,nexthop
- For a non-landmark node v: tree T_v
  - If u in T_v then v in B(u)
- Add nodes to B(u) until B(u) is complete
- Get better nodes for B(u)
- For any node inserted/removed/updated (distance,hop-distance):
  - Send this event to neighbors

Analysis of synchronous truncated BFS



Step 1 - Insertions: B is complete as soon as |B| reaches k log k at time i (< klog k messages for one node)

Analysis of synchronous truncated BFS

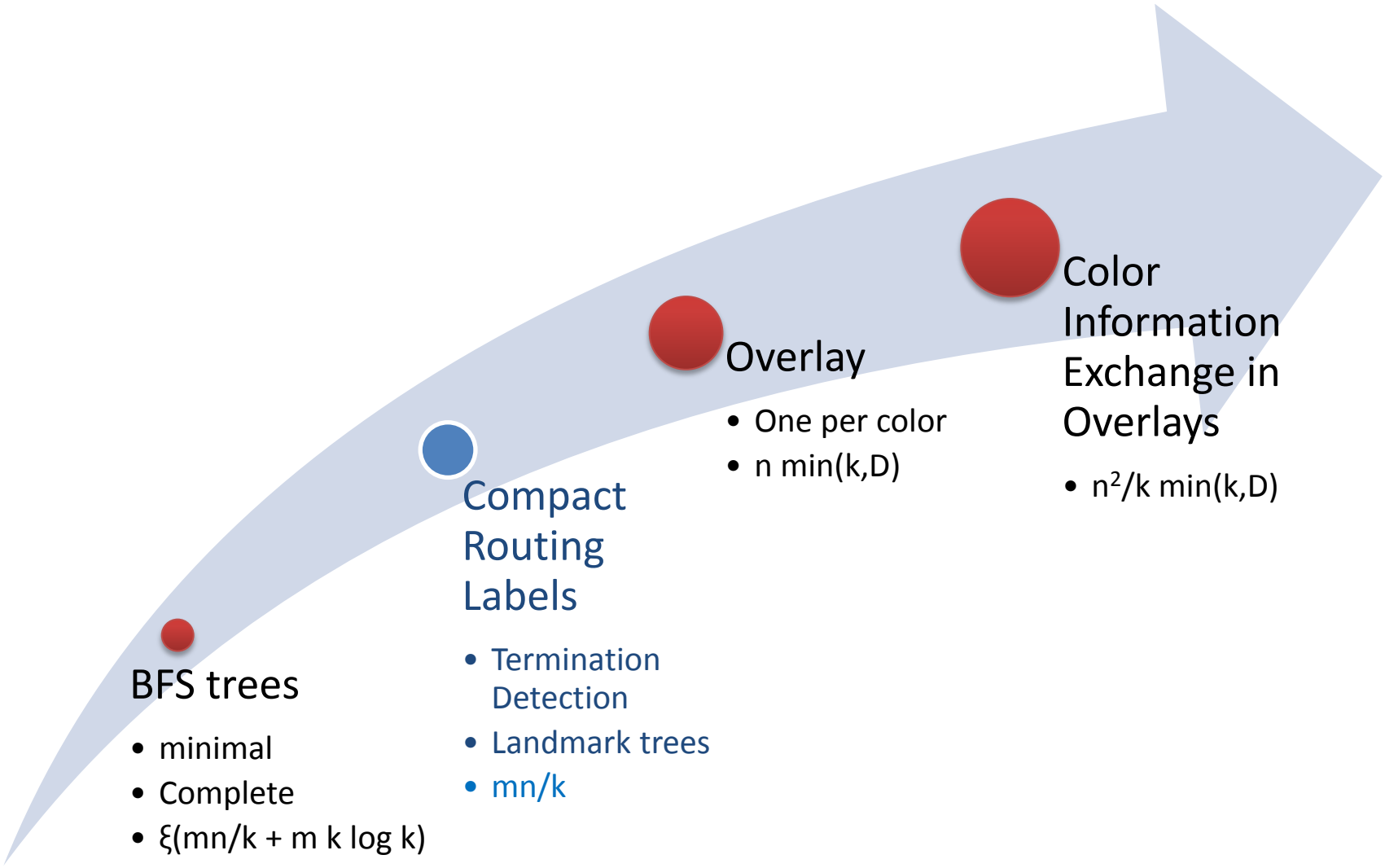


Step 2 – Get closer nodes: Some nodes are removed (added) from B between time i and time j ≤ min(D,iW).

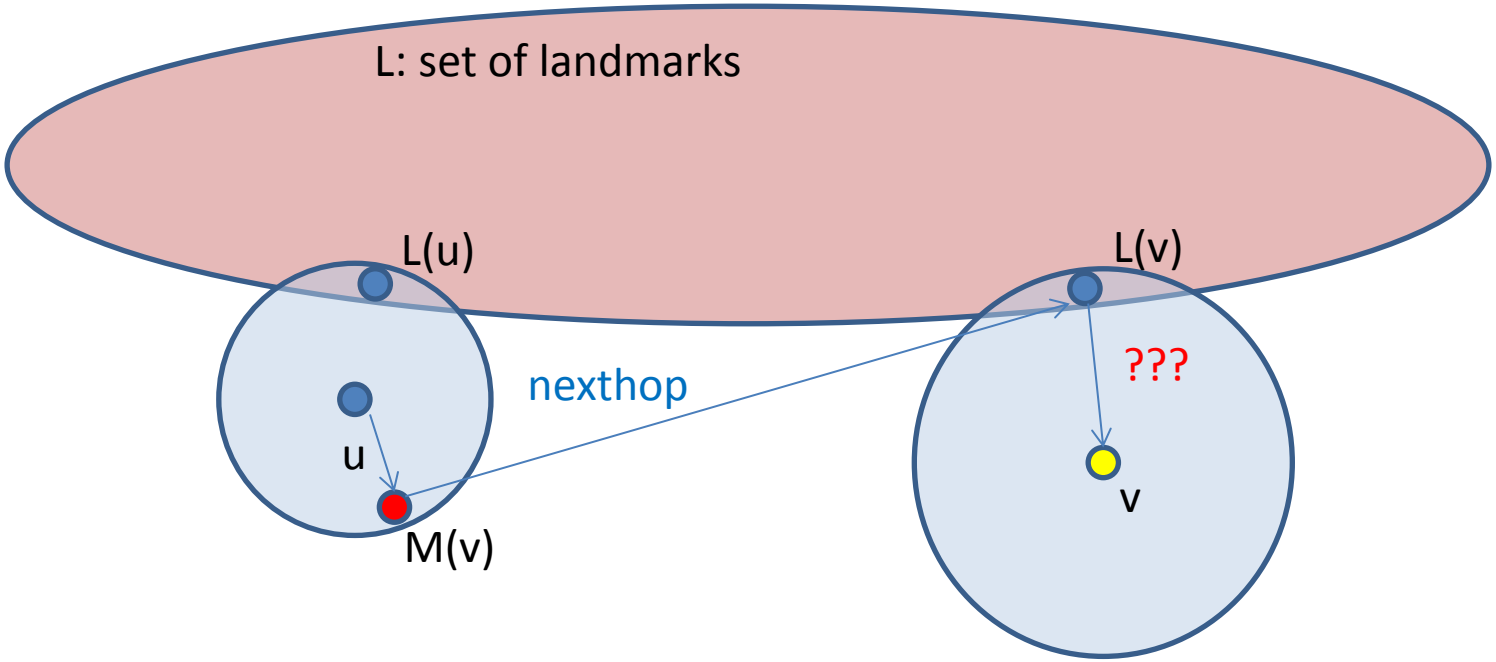
→ j-i ≤ j(1-1/W) ≤ D(1-1/W)=ξ-1

For all nodes: mk log k + (ξ-1) mk log k= ξ mk log k

Details on the communication cost



How to route further



1. Source u computes h(v)=c
2. Forward message to M(c), its manager of color c in B(u)
3. M(c) knows Path P=L(v) → v and forward message to L(v)
4. Message follows P to reach v

Learning C(u) avoiding wild broadcasts

A first idea

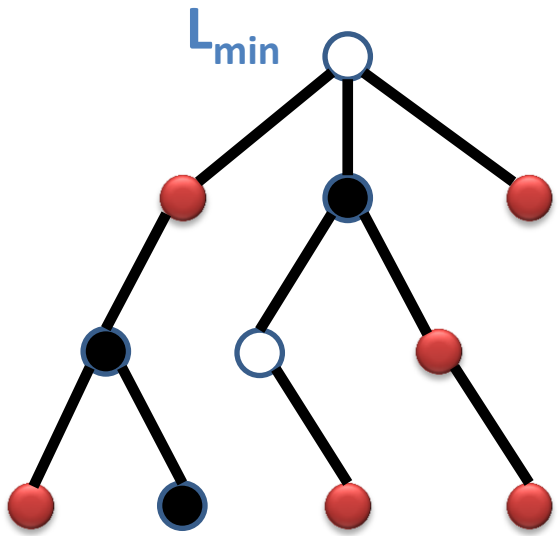
- ✓Every node v broadcasts piece of info: L(v)→v, … using landmark trees
- ✓ nm messages

A better idea

- ✓Nodes of same color shares same interest
- ✓Make them communicate !
- ✓ $n^2 min(k,D)/k$ messages on average

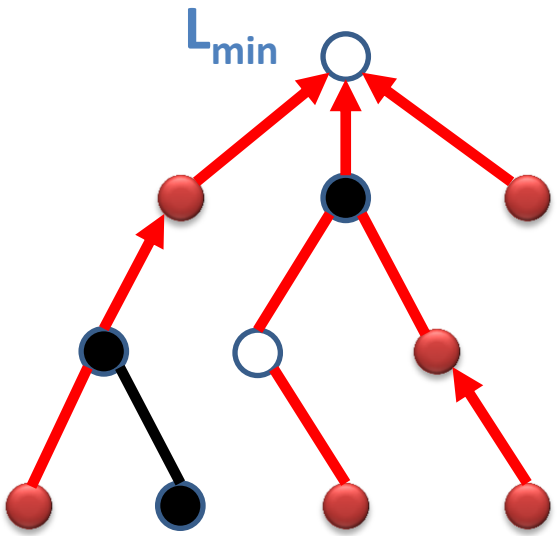
Communication within an overlay network

- A landmark L_{min} (and its tree) is elected
- One overlay per color



Communication within an overlay network

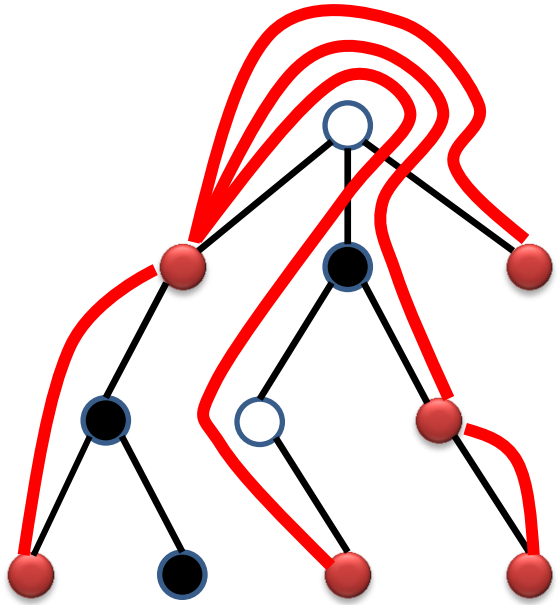
- Overlay Construction



Every node of a given color search for an ancestor of same color

Communication within an overlay network

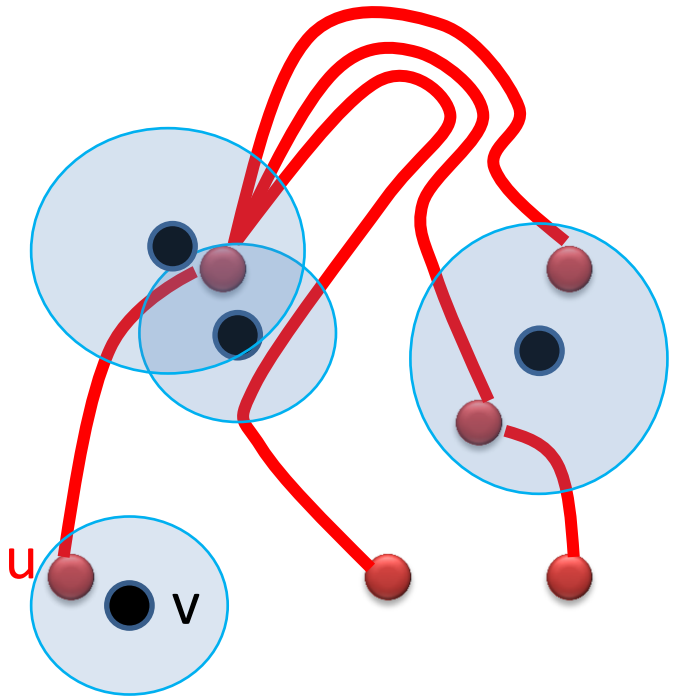
- Overlay Construction



Each node knows the paths to its logical neighbors through L_{min}

Communication within an overlay network

- Color Table construction



1. Each node v chooses a manager node u of color h(v) in its ball
2. Nodes of color h(v) exchange routing information to v's: L(v)→v

Conclusion

- Good step toward dynamic and light compact routing schemes (not far from being self-stabilizing)
- Can we get a smaller stretch with a similar communication cost ?