Distributed computing of efficient routing schemes

Nicolas Nisse[1] Iván Rapaport[2] Karol Suchan[3]

[1] MASCOTTE, INRIA, I3S, CNRS, UNS, Sophia Antipolis, France
[2] DIM, Universidad de Chile, Santiago, Chile
[3] Universidad Adolfo Ibáñez, Santiago, Chile

Working group June 15th 2009,
Alcatel Lucent Belgique/MASCOTTE/LaBRI

The Routing Problem

Problem

- `input`: a network G
- `output`: a routing scheme for G

Routing Scheme: protocol that directs the traffic in a network

Any source must be able to route a message to any destination, given the destination's ID.

name-based: IDs are chosen by the designer of the scheme

Complexity Measures

Stretch

- **Multiplicative stretch**: *ratio* between the length of the computed route and the distance.
  $|route(x,y)| \leq mult\text{-}stretch \cdot d(x,y)$.
- **Additive stretch**: *difference* between the length of the computed route and the distance.
  $|route(x,y)| \leq add\text{-}stretch + d(x,y)$.

Routing tables' size

Space necessary to store local routing table (per node)

Time complexity

Distributed protocol to setup data structures

Example: Interval Routing [Santoro & Khatib, 82]

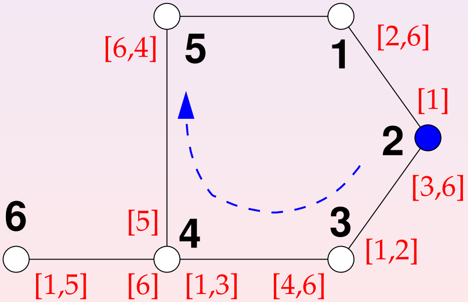
- Nodes labeled using integers
- Outgoing arc labeled with an interval of the name range

Message sent through the arc containing the destination

mult-stretch:
$\frac{route(1,5)}{d(1,5)} = 4$

add-stretch:
$route(1,5)$-$d(1,5) = 3$

space per node:
$O(\Delta \log n)$



Related Works: name-based

Labels are of polylogarithmic size

network	mult-stretch	table	
arbitrary	1	$n \log n$	folklore
$(k \geq 2)$	$4k-5$	$O(n^{1/k})$	Thorup & Zwick
tree	1	$O(1)$	TZ/Fraigniaud & Gavoille
doubling-α	$1+\epsilon$	$\log \Delta$	Talwar/Slivkins
dimension		$O(1)$	Chan et al./Abraham et al.
planar	$1+\epsilon$	$O(1)$	Thorup
H-minor free	$1+\epsilon$	$O(1)$	Abraham & Gavoille

Table: Routing schemes

Related Work: k-chordal Graphs

k-chordal graph: any cycle with length $\geq k$ contains a chord.
chordal graph $\Leftrightarrow$ 3-chordal graph (a.k.a. triangulated graph)

network	stretch	table	computation	
chordal	$+2$	$O(\frac{\log^3 n}{\log \log n})$	$O(m + n \log^2 n)$	Dourisboure Gavoille, 02
k-chordal	$k+1$	$O(\log^2 n)$	$poly(n)$	Dourisboure 04

Table: Routing schemes for k-chordal graphs

Our results: k-chordal Graphs

k-chordal graph: any cycle with length $\geq k$ contains a chord.
chordal graph $\Leftrightarrow$ 3-chordal graph (a.k.a. triangulated graph)

network	stretch	table	computation	
chordal	$+2$	$O(\frac{\log^3 n}{\log\log n})$	$O(m + n\log^2 n)$	Dourisboure Gavoille, 02
	$+1$	$O(\Delta \log n)$	$O(n)$	**this work**
k-chordal	$k+1$	$O(\log^2 n)$	$poly(n)$	Dourisboure 04
	$k-1$	$O(\Delta \log n)$	$O(D)$	**this work**

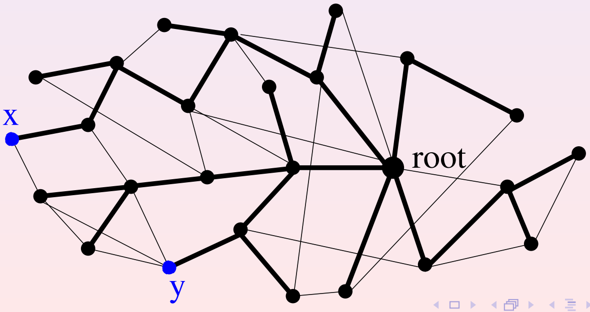
Table: Routing schemes for k-chordal graphs

Routing scheme $RS(G, T)$

G a network and T a routed spanning tree of G

x a source node and y a destination node

> If $x = y$, stop.
> If there is $w \in N_G(x)$, an ancestor of y in T,
>   choose w minimizing $d_T(w, y)$;
> Otherwise, choose the parent of x in T.

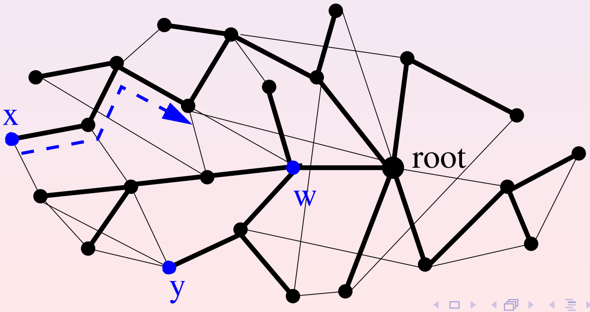


Routing scheme $RS(G, T)$

G a network and T a routed spanning tree of G

x a source node and y a destination node

If $x = y$, stop.
If there is $w \in N_G(x)$, an ancestor of y in T,
choose w minimizing $d_T(w, y)$;
Otherwise, choose the parent of x in T.

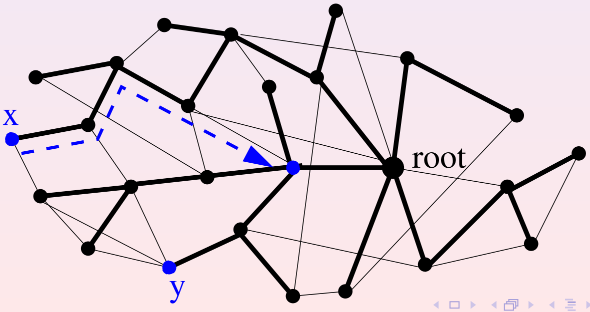


Routing scheme $RS(G, T)$

G a network and T a routed spanning tree of G

x a source node and y a destination node

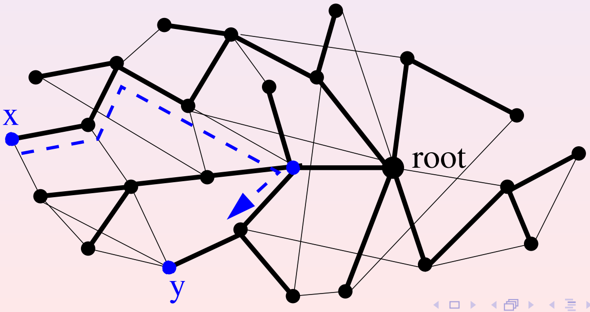
> If $x = y$, stop.
> If there is $w \in N_G(x)$, an ancestor of y in T,
> choose w minimizing $d_T(w, y)$;
> Otherwise, choose the parent of x in T.



Routing scheme $RS(G, T)$

G a network and T a routed spanning tree of G
x a source node and y a destination node

> If $x = y$, stop.
> If there is $w \in N_G(x)$, an ancestor of y in T,
> choose w minimizing $d_T(w, y)$;
> Otherwise, choose the parent of x in T.

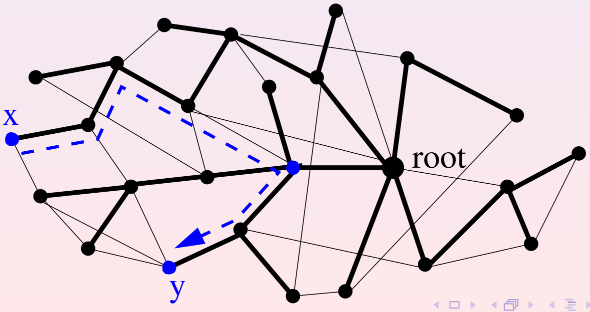


Routing scheme $RS(G, T)$

G a network and T a routed spanning tree of G

x a source node and y a destination node

If $x = y$, stop.
If there is $w \in N_G(x)$, an ancestor of y in T,
  choose w minimizing $d_T(w, y)$;
Otherwise, choose the parent of x in T.



Routing scheme $RS(G, T)$

G a network and T a routed spanning tree of G
x a source node and y a destination node

> If $x = y$, stop.
> If there is $w \in N_G(x)$, an ancestor of y in T,
> choose w minimizing $d_T(w, y)$;
> Otherwise, choose the parent of x in T.

Once T has been chosen
Space: *labeling of nodes*: any rooted subtree $\Leftrightarrow$ interval
routing table: each node knows the interval of its neighbors
$O(\Delta \log n)$ bits per node
Time: easy in time $O(D)$ in synchronous distributed way

Performances

Lemma 1

If T is any BFS-tree of G k-chordal graph, then
Add-stretch of $RS(G, T) = k - 1$

Lemma 2

If T is any MaxBFS-tree of G chordal graph, then
Add-stretch of $RS(G, T) = 1$

Lemma 3: in synchronous distributed way,

a BFS-tree can be computed in time $O(D)$;
a MaxBFS-tree can be computed in time $O(n)$.

BFS orderings and BFS-trees

Let r be an arbitrary node: the root.

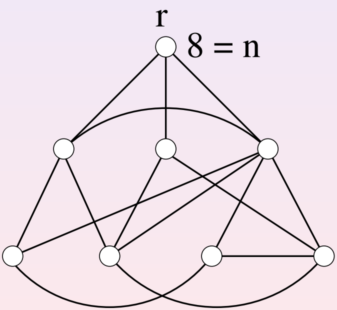
BFS-tree: parent = greatest neighbor

Breadth First Search

Labeled r with n,

While $\exists$ unlabeled vertices

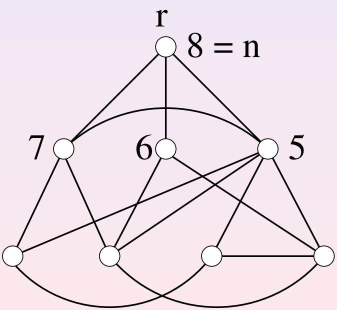
Label a neighbor of greatest v with unlabeled neighbors



BFS orderings and BFS-trees

Let r be an arbitrary node: the root.
BFS-tree: parent = greatest neighbor

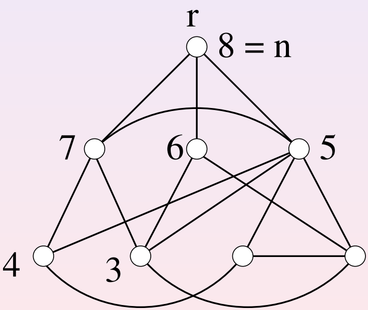
Breadth First Search
Labeled r with n,
While $\exists$ unlabeled vertices
Label a neighbor of greatest v
with unlabeled neighbors



BFS orderings and BFS-trees

Let r be an arbitrary node: the root.
BFS-tree: parent = greatest neighbor

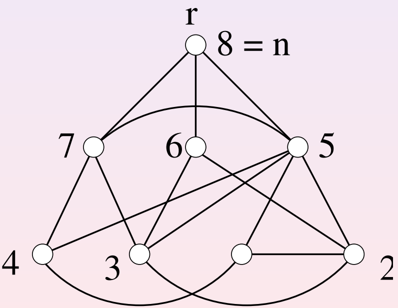
Breadth First Search
Labeled r with n,
While $\exists$ unlabeled vertices
Label a neighbor of greatest v
with unlabeled neighbors



BFS orderings and BFS-trees

Let r be an arbitrary node: the root.
BFS-tree: parent = greatest neighbor

Breadth First Search
Labeled r with n,
While $\exists$ unlabeled vertices
Label a neighbor of greatest v
with unlabeled neighbors

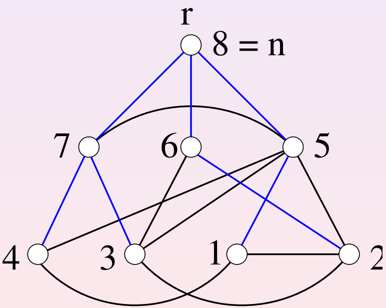


BFS orderings and BFS-trees

Let r be an arbitrary node: the root.

BFS-tree: parent = greatest neighbor

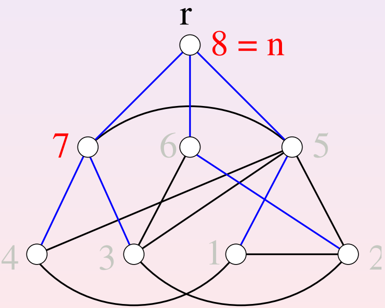
Breadth First Search
Labeled r with n,
While $\exists$ unlabeled vertices
Label a neighbor of greatest v
with unlabeled neighbors



BFS orderings and BFS-trees

Let r be an arbitrary node: the root.
BFS-tree: parent = greatest neighbor

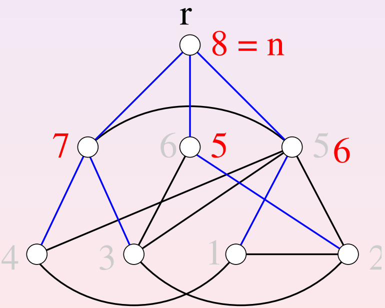
Maximum NeighborhoodBFS
Labeled r with n,
While $\exists$ unlabeled vertices
Label a neighbor of greatest v
with unlabeled neighbors that
has maximum labeled neighbors



BFS orderings and BFS-trees

Let r be an arbitrary node: the root.
BFS-tree: parent = greatest neighbor

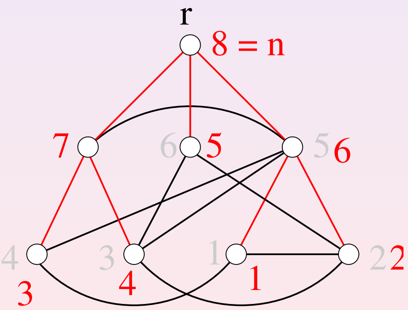
Maximum NeighborhoodBFS
Labeled r with n,
While $\exists$ unlabeled vertices
Label a neighbor of greatest v
with unlabeled neighbors that
has maximum labeled neighbors



BFS orderings and BFS-trees

Let r be an arbitrary node: the root.
BFS-tree: parent = greatest neighbor

Maximum NeighborhoodBFS
Labeled r with n,
While $\exists$ unlabeled vertices
Label a neighbor of greatest v
with unlabeled neighbors that
has maximum labeled neighbors



Lemma 2: Sketch of proof

If T is any MaxBFS-tree of G chordal graph, then
Add-stretch of $RS(G, T) = 1$

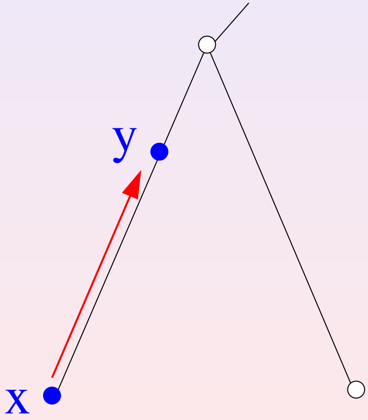
Main tools

- T is a BFS-tree
- G chordal $\Leftrightarrow$ any minimal separator is a clique [Dirac]
- MaxBFS-ordering and G chordal $\Rightarrow$ ($v > w > z$ and $\{z, w\}, \{z, v\} \in E \Rightarrow \{v, w\} \in E$) [BKS 05]

Remainder: **Routing Scheme**: follows T but if one neighbor is an ancestor of the destination.

MaxBFS-tree + chordal graph $\Rightarrow$ add-stretch=1

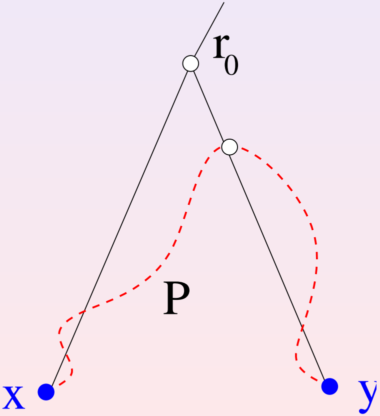
Source x ancestor/descendant of destination y
$\Rightarrow$ add-stretch=0 because T BFS-tree



MaxBFS-tree + chordal graph $\Rightarrow$ add-stretch=1

r_0 closest commun ancestor of x and y
P a shortest path between x and y

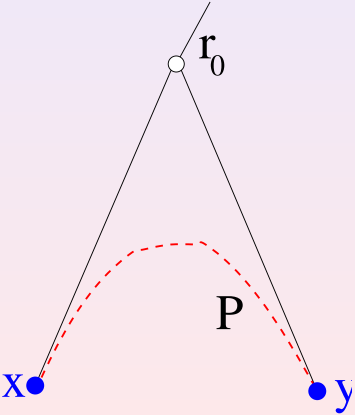
Let us prove that
$|Route(x,y)| \leq |P| + 1$



MaxBFS-tree + chordal graph $\Rightarrow$ add-stretch=1

r_0 closest commun ancestor of x and y
P a shortest path between x and y

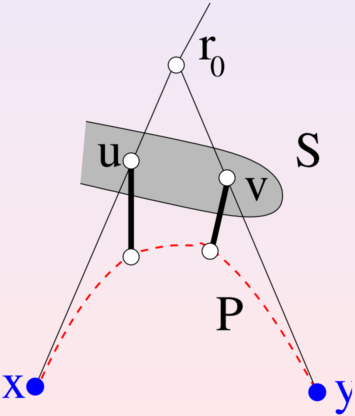
Let us prove that
$|Route(x,y)| \leq |P|+1$



MaxBFS-tree + chordal graph $\Rightarrow$ add-stretch=1

Case $r_0 \notin N(P)$. $\exists S$ minimal r_0, P-separator in $N(P)$.
$u, v \in S$ s.t. $d(u,x) + d(v,y)$ minimum

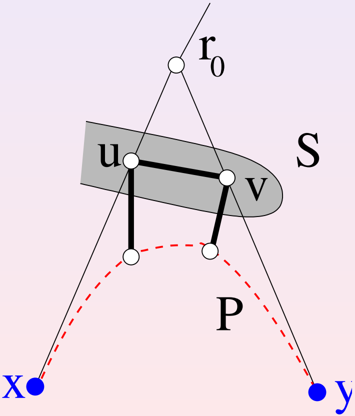
Let us prove that
$|Route(x,y)| \leq |P| + 1$



MaxBFS-tree + chordal graph $\Rightarrow$ add-stretch=1

G chordal $\Rightarrow$ S clique, thus $\{u, v\} \in E(G)$
$d(x,u) + 1 + d(v,y)$ upper bound on $|Route(x,y)|$

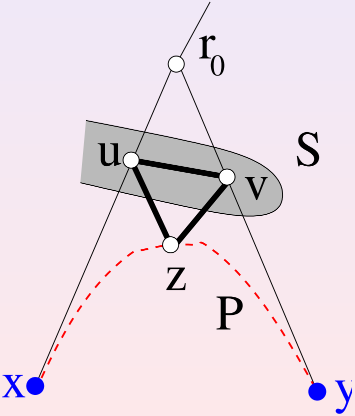
Let us prove that
$|Route(x,y)| \leq |P| + 1$
$d(x,u) + d(v,y) \leq |P|$



MaxBFS-tree + chordal graph $\Rightarrow$ add-stretch=1

$\{u, v\} \in E(G)$ and $u, v \in N(P)$
G chordal $\Rightarrow$ u, v have a commun neighbor z in P

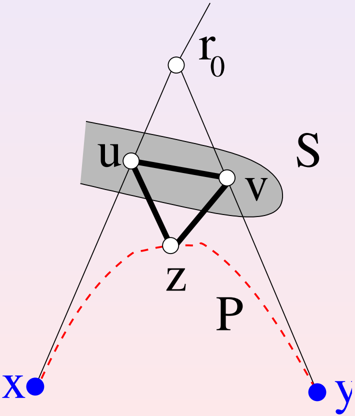
Let us prove that
$|Route(x, y)| \leq |P| + 1$
$d(x, u) + d(v, y) \leq |P|$



MaxBFS-tree + chordal graph $\Rightarrow$ add-stretch=1

T BFS-tree
$d(u,x) \leq d(x,z)+1$ and $d(v,y) \leq d(z,y)+1$

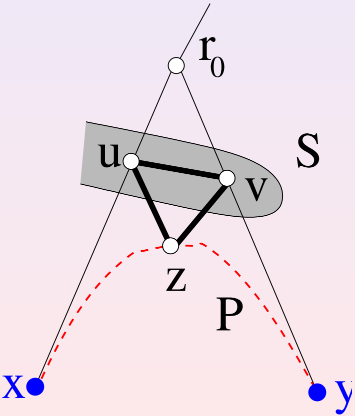
Let us prove that
$|Route(x,y)| \leq |P|+1$
$d(x,u)+d(v,y) \leq |P|$
We know
$d(u,x) \leq d(x,z)+1$
$d(v,y) \leq d(z,y)+1$



MaxBFS-tree + chordal graph $\Rightarrow$ add-stretch=1

W.l.o.g., $u > v$. Recall $d(v,y) \leq d(z,y) + 1$
$d(v,y) \leq d(z,y)$, otherwise $P_{z \to y}$ would belong to T

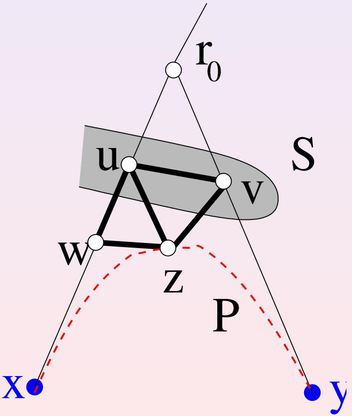
Let us prove that
$|Route(x,y)| \leq |P| + 1$
$d(x,u) + d(v,y) \leq |P|$
We know
$d(u,x) \leq d(x,z) + 1$
$d(v,y) \leq d(z,y)$



MaxBFS-tree + chordal graph $\Rightarrow$ add-stretch=1

Assume $d(u,x) = d(x,z) + 1$. T BFS-tree $\Rightarrow$ $v > w > z$.
G chordal $\Rightarrow \{w,z\} \in E(G)$

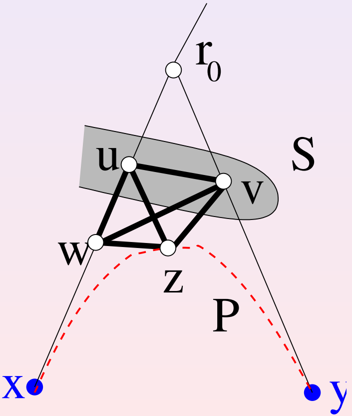
Let us prove that
$|Route(x,y)| \leq |P| + 1$
$d(x,u) + d(v,y) \leq |P|$
We know
$d(u,x) \leq d(x,z) + 1$
$d(v,y) \leq d(z,y)$



MaxBFS-tree + chordal graph $\Rightarrow$ add-stretch=1

$v > w > z$ and $\{z, v\}, \{w, z\} \in E(G) \Rightarrow \{w, v\} \in E(G)$
$|Route(x, y)| \leq d(x, w) + 1 + d(v, y) \leq |P| + 1$

Let us prove that
$|Route(x, y)| \leq |P| + 1$
$d(x, u) + d(v, y) \leq |P|$
We know
$d(u, x) \leq d(x, z) + 1$
$d(v, y) \leq d(z, y)$
If $d(u, x) = d(x, z) + 1$
then $\{w, v\} \in E(G)$



Current/Further works

Can we improve the size of routing tables?

Other graph classes?

Other BFS-ordering?

Case of k-chordal graphs: can we improve the stretch?