Routing in Distributed Systems

- Lecture 1 -

Cyril Gavoille

University of Bordeaux, France

April 6-9, 2009
Valparaíso, Chile

1. Routing with Succinct Tables
2. Constructing Sparse Spanners
3. Routing v.s. Spanners

- Lecture 1 -

Routing with Succinct Tables

An introduction to Compact Routing, and some algorithmic tools for constructing routing schemes

Outline

Outline

Message Routing in the Internet

Message Routing in the Internet

1969. Arpanet connects $n = 4$ nodes.

Message Routing in the Internet

1969. Arpanet connects $n = 4$ nodes.

1977. First paper on the **Compact Routing**, and on limitation of the Arpanet: *Hierarchical Routing for Large Networks; Performance Evaluation and Optimization* by Kleinrock and Kamoun.

Message Routing in the Internet

1969. Arpanet connects $n = 4$ nodes.

1977. First paper on the **Compact Routing**, and on limitation of the Arpanet: *Hierarchical Routing for Large Networks; Performance Evaluation and Optimization* by Kleinrock and Kamoun.

2009. Growth rate for *forwarding information base (FIB)* size is **1.3** a year.

- 175 KB in jan 2006;
- 300 KB in jan 2009;
- 1 MB in 2011 (estimate).

Sub-Linear Size Routing Tables

Main router builder compagnies (Alcatel Lucent Bell, Cisco, Nortel) are studying for new routing data organizations to lower the growth rate size of FIBs.

Today the size of FIBs generated by routing protocols (or routing strategy) is linear to n, the number of autonomous systems (AS).

Outline

Statement of the Compact Routing Problem

Input: a network G (a weighted connected graph)

Ouput: a **routing scheme** for G

A *routing scheme* is a distributed algorithm that allows **any** source node to route messages to **any** destination node, given the destination's network identifier

Statement of the Compact Routing Problem

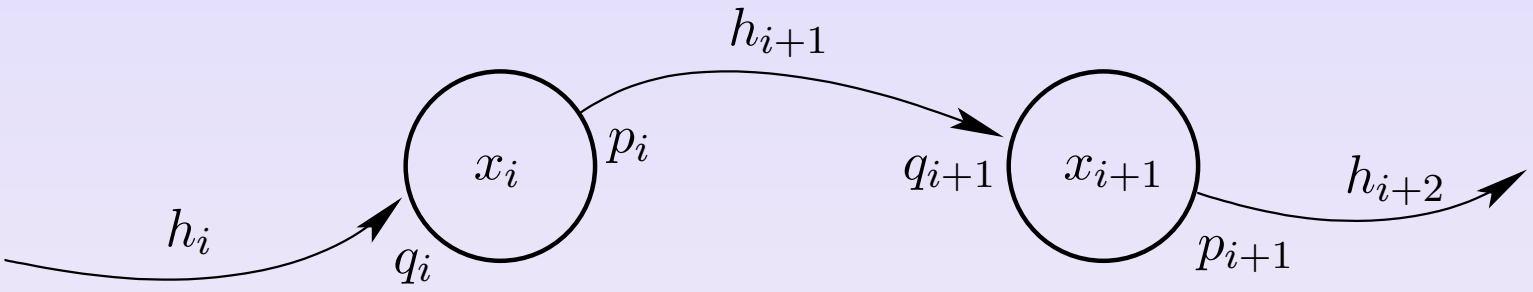
Input: a network G (a weighted connected graph)

Ouput: a **routing scheme** for G

A *routing scheme* is a distributed algorithm that allows **any** source node to route messages to **any** destination node, given the destination's network identifier

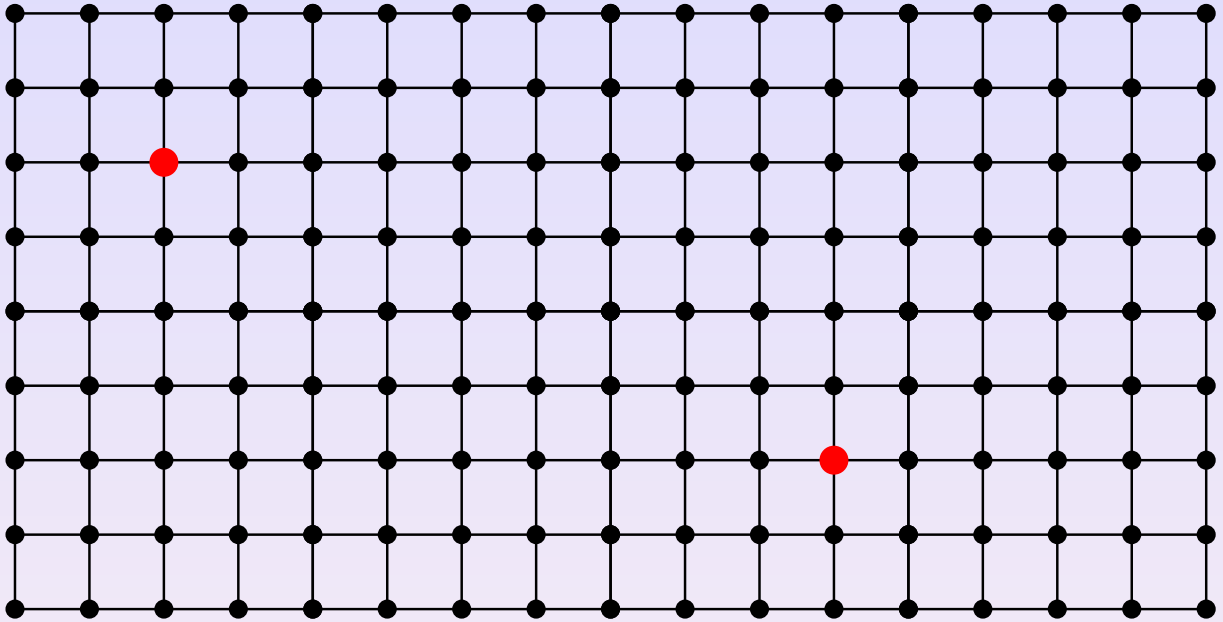
Goal: to minimize the size of the routing tables

Model: $R_{x_i}(h_i, q_i) = (p_i, h_{i+1})$



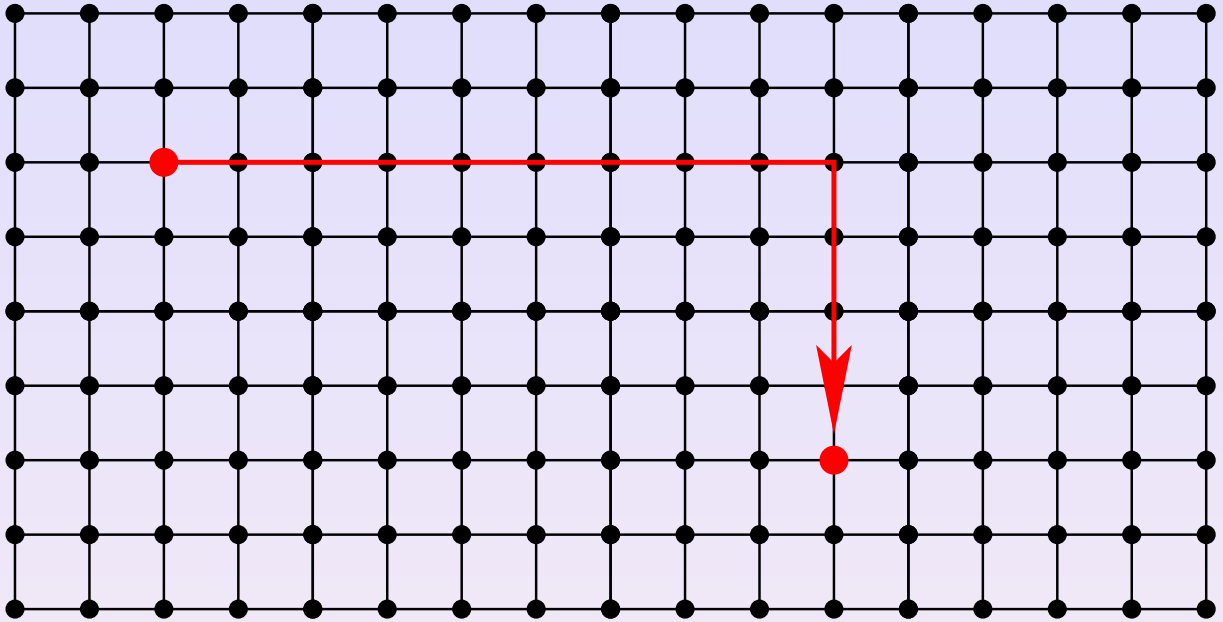
- addresses
- headers
- in/outport numbers

Example: Grid with X,Y-coordinates



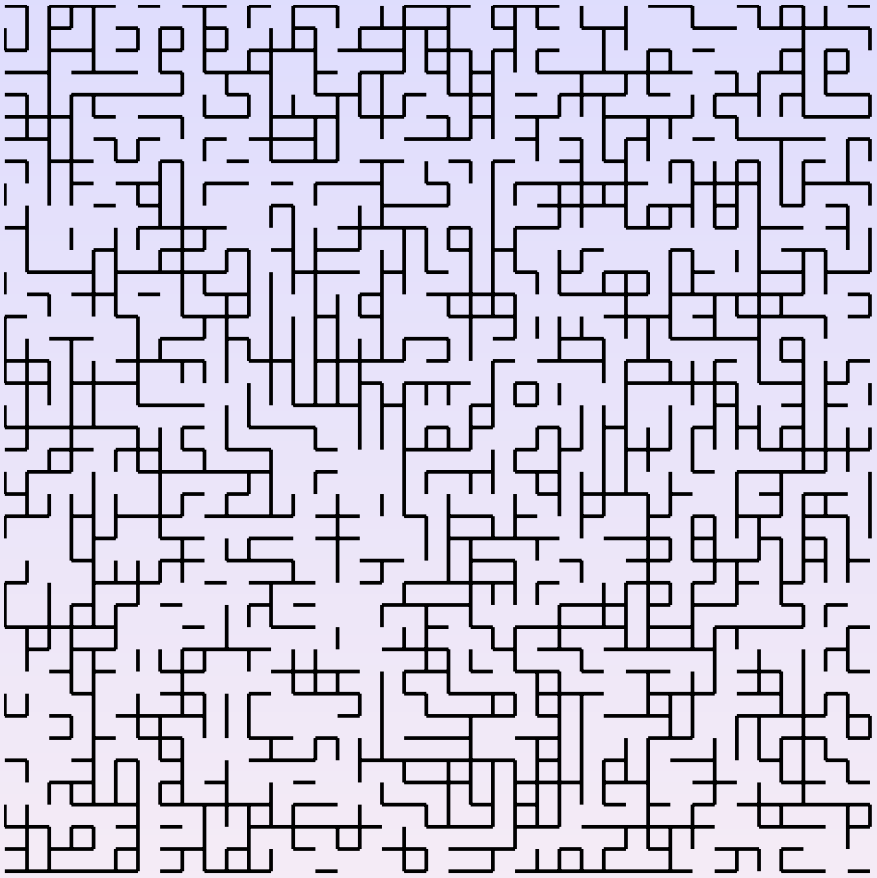
Routing algorithm: X,Y-routing
(local decision)

Example: Grid with X,Y-coordinates



Routing algorithm: X,Y-routing
(local decision)

Another Example ...



Routing scheme: ???

Complexity Measures: Space & Stretch

Space $=$ size of the largest local routing tables

Complexity Measures: Space & Stretch

Space = size of the largest local routing tables
(more precisely, size of the smallest local routing algorithm including all constants and data-structures)

In the grid example: $\text{space} = O(\log n)$ bits

Complexity Measures: Space & Stretch

Space = size of the largest local routing tables
(more precisely, size of the smallest local routing algorithm including all constants and data-structures)

In the grid example: $\text{space} = O(\log n)$ bits

Stretch = ratio between length of the route and distance

$$|\text{route}(x, y)| \leqslant \text{stretch} \cdot \text{dist}(x, y)$$

In the grid example: $\text{stretch} = 1$ (shortest path)

Complexity Measures: Space & Stretch

Space = size of the largest local routing tables
(more precisely, size of the smallest local routing algorithm including all constants and data-structures)

In the grid example: $\text{space} = O(\log n)$ bits

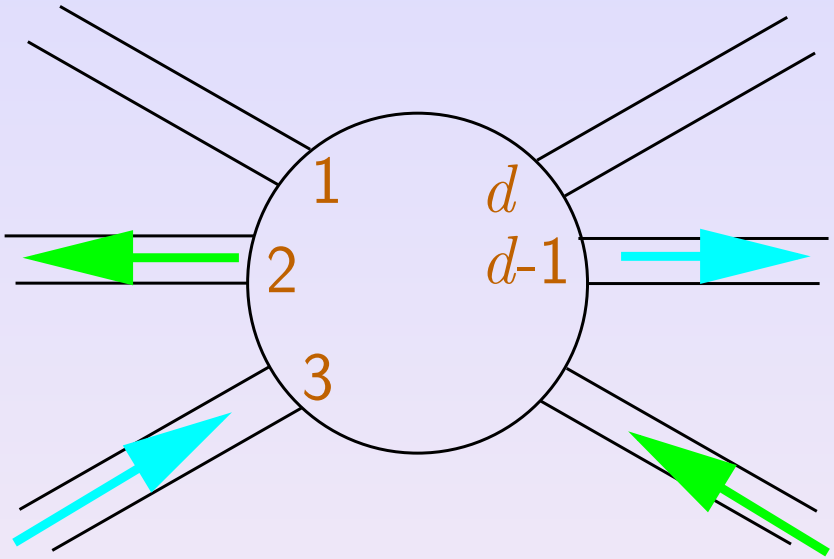
Stretch = ratio between length of the route and distance

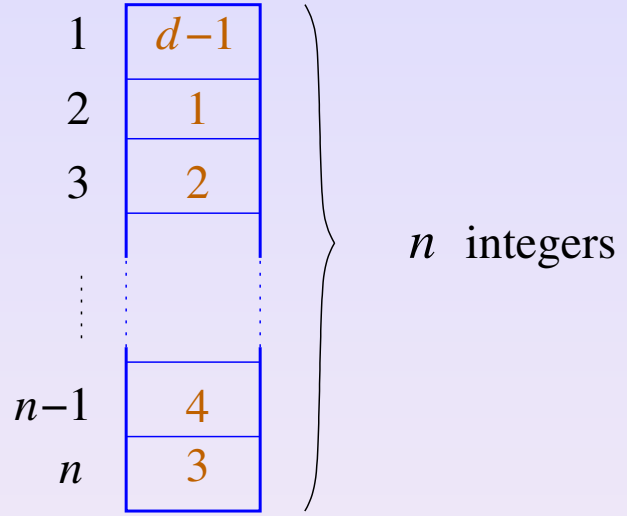
$$|\text{route}(x, y)| \leqslant \text{stretch} \cdot \text{dist}(x, y)$$

In the grid example: $\text{stretch} = 1$ (shortest path)

Question: for a given family of graphs, find the best space-stretch trade-off

Routing Tables: A Universal Routing Strategy





message towards node 3

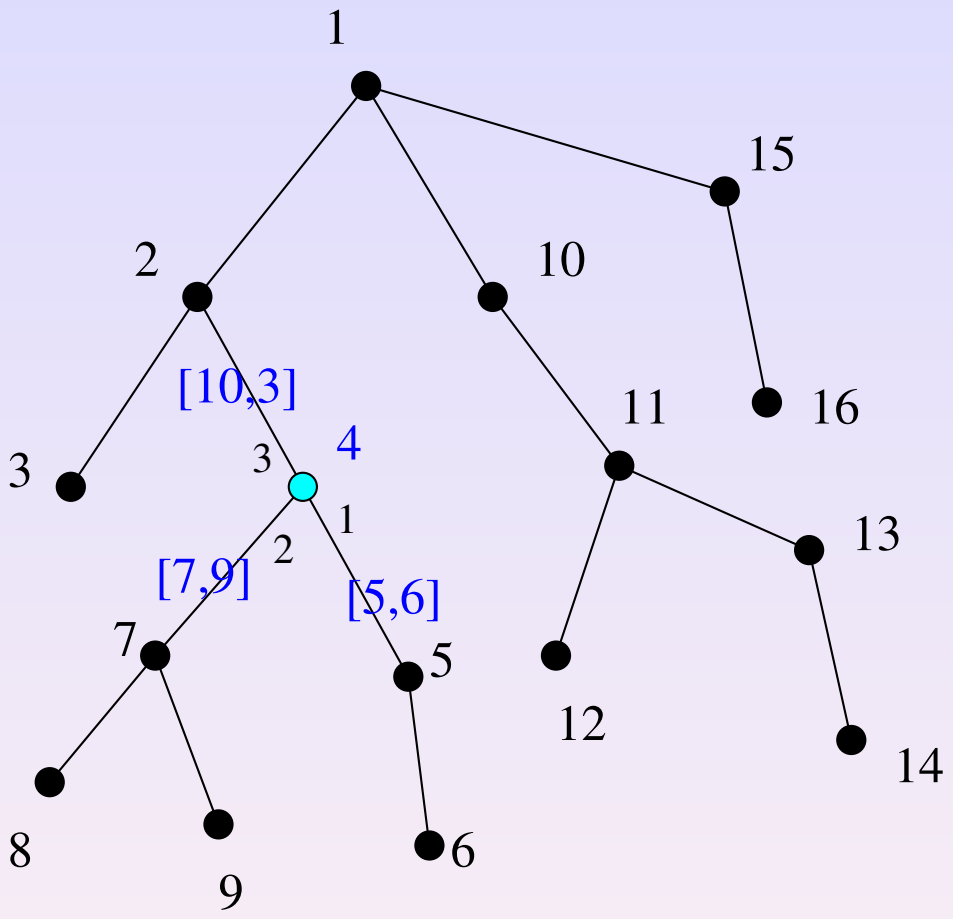
message towards node 1

Space $= O(n \log d)$ bits, $d =$ degree

Stretch $= 1$

Outline

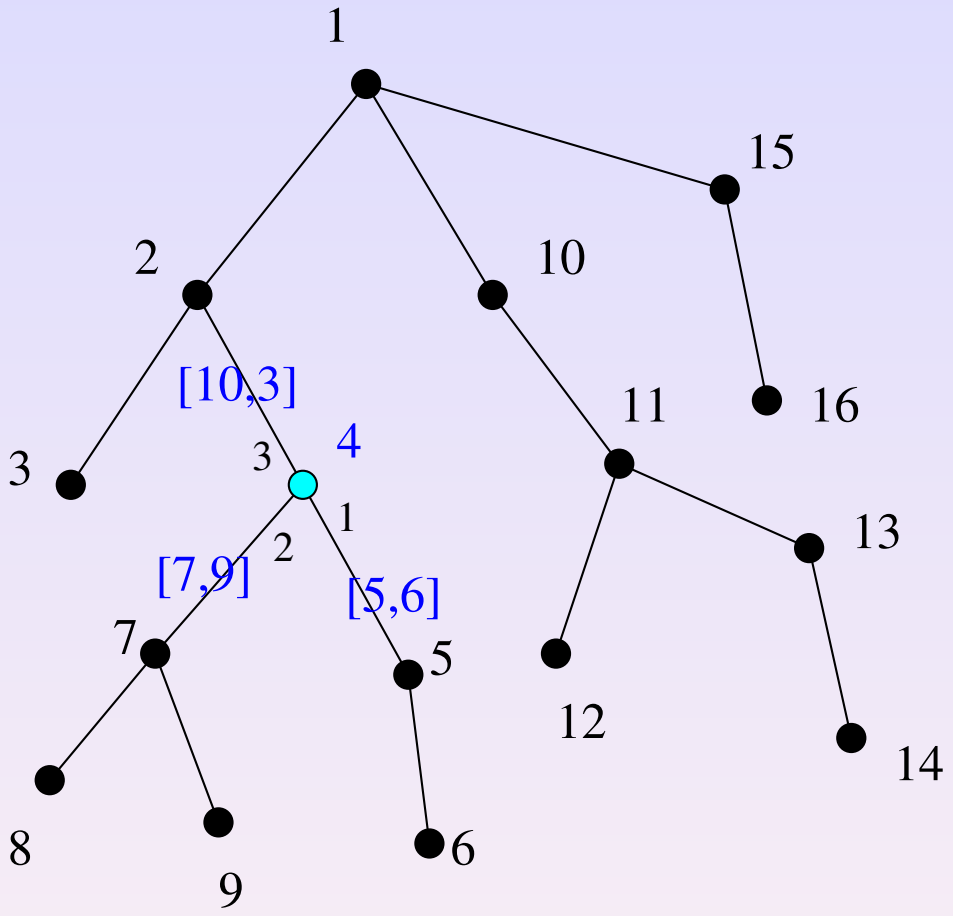
Routing in a Tree (attempt #1)



- DFS numbering
- Interval Routing
- $O(n \log n)$ bits in total!

4: [5,6]:1 [7,9]:2 [10,3]:3

Routing in a Tree (attempt #1)

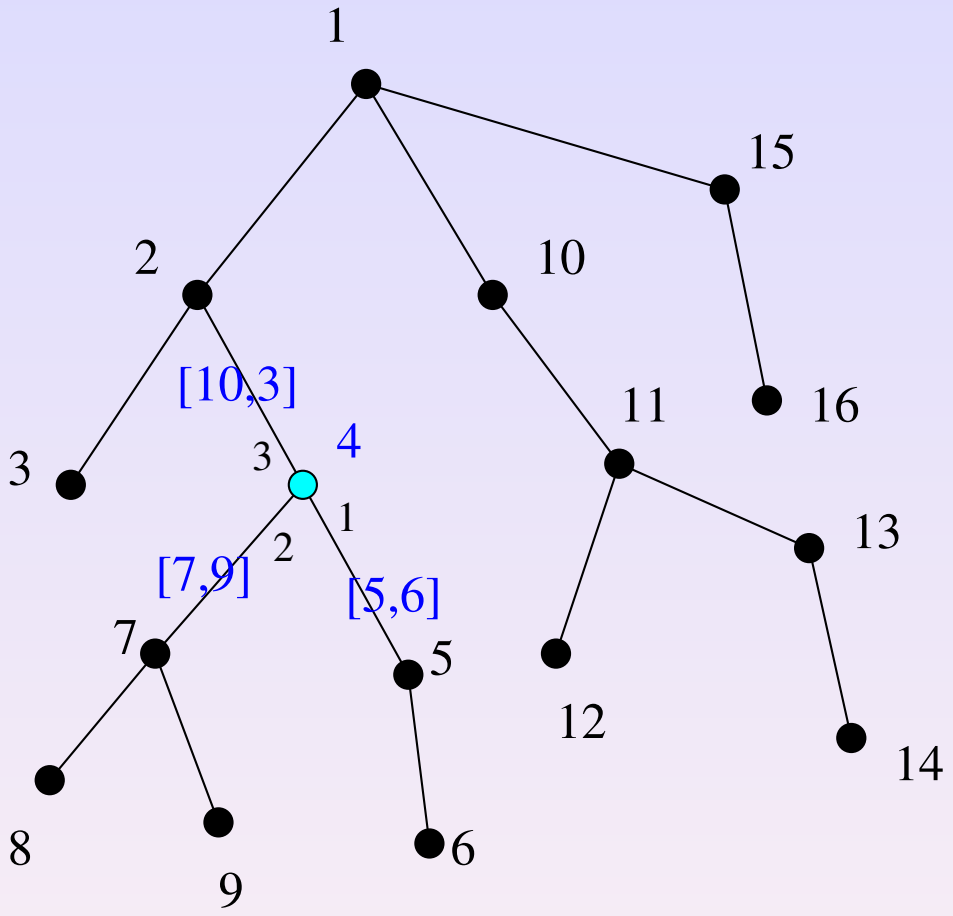


- DFS numbering
- Interval Routing
- $O(n \log n)$ bits in total!

4: [5,6]:1 [7,9]:2 [10,3]:3

Space $= O(d \log n)$ bits, $d =$ degree, Stretch $= 1$

Routing in a Tree (attempt #1)



- DFS numbering
- Interval Routing
- $O(n \log n)$ bits in total!

4: [5,6]:1 [7,9]:2 [10,3]:3

Space $= O(d \log n)$ bits, $d =$ degree, Stretch $= 1$

Problem: Space $\Omega(n)$ whenever $d = \Omega(n/\log n)$

Routing in a Tree (attempt #2)

Idea: enlarge addresses to reduce tables

Routing in a Tree (attempt #2)

Idea: enlarge addresses to reduce tables

- $\mathrm{id}(u)$ is a DFS numbering of u in the tree. The *weight* of a node is the number of its descendants.

Routing in a Tree (attempt #2)

Idea: enlarge addresses to reduce tables

- $\mathrm{id}(u)$ is a DFS numbering of u in the tree. The *weight* of a node is the number of its descendants.
- $\mathrm{table}(u)$ is composed of a partial interval routing of u: only for its parent and its t heaviest children.

Routing in a Tree (attempt #2)

Idea: enlarge addresses to reduce tables

- $\mathrm{id}(u)$ is a DFS numbering of u in the tree. The *weight* of a node is the number of its descendants.
- $\mathrm{table}(u)$ is composed of a partial interval routing of u: only for its parent and its t heaviest children.
- $\mathrm{address}(u) = (\mathrm{id}(u), \mathrm{hpath}(u))$, where $\mathrm{hpath}(u)$ is the sequence of ports encountered when going from the root to u, but we remove ports leading to a child of rank $< t$ (so with a large weight), a parameter fixed later.

The Routing Algorithm: $x \mapsto y$

Given $\mathrm{table}(x)$ and $\mathrm{address}(y)$

- If $\mathrm{id}(x) = \mathrm{id}(y)$, then STOP.
- If $\mathrm{id}(y)$ belongs to some intervals of $\mathrm{table}(x)$, then route with this table.
- Otherwise, route to port $\mathrm{hpath}(y)[1 + |\mathrm{hpath}(x)|]$.

The Routing Algorithm: $x \mapsto y$

Given $\mathrm{table}(x)$ and $\mathrm{address}(y)$

- If $\mathrm{id}(x) = \mathrm{id}(y)$, then STOP.
- If $\mathrm{id}(y)$ belongs to some intervals of $\mathrm{table}(x)$, then route with this table.
- Otherwise, route to port $\mathrm{hpath}(y)[1 + |\mathrm{hpath}(x)|]$.

Hint: If the two first tests fail, $\mathrm{hpath}(x)$ is a proper prefix of $\mathrm{hpath}(y)$.

Space Analysis

- $\mathrm{table}(u)$ is $O(t \log n)$ bits
- $\mathrm{address}(u)$ is $O(|\mathrm{hpath}(u)| \cdot \log n)$ bits

If $\mathrm{port}(a, b)$ is in $\mathrm{hpath}(u)$, then b has rank $\geqslant t$ for a.
So, $w(b) \leqslant w(a)/t$, and thus $|\mathrm{hpath}(u)| \leqslant \log_t n$.

Space Analysis

- $\mathrm{table}(u)$ is $O(t \log n)$ bits
- $\mathrm{address}(u)$ is $O(|\mathrm{hpath}(u)| \cdot \log n)$ bits

If $\mathrm{port}(a, b)$ is in $\mathrm{hpath}(u)$, then b has rank $\geqslant t$ for a.
So, $w(b) \leqslant w(a)/t$, and thus $|\mathrm{hpath}(u)| \leqslant \log_t n$.

$\Rightarrow$ adresses and tables have size $O(\log^2 n / \log\log n)$ with $t = \log n / \log\log n$.

Space Analysis

- $\mathrm{table}(u)$ is $O(t \log n)$ bits
- $\mathrm{address}(u)$ is $O(|\mathrm{hpath}(u)| \cdot \log n)$ bits

If port(a, b) is in $\mathrm{hpath}(u)$, then b has rank $\geqslant t$ for a.
So, $w(b) \leqslant w(a)/t$, and thus $|\mathrm{hpath}(u)| \leqslant \log_t n$.

$\Rightarrow$ adresses and tables have size $O(\log^2 n / \log\log n)$ with $t = \log n / \log\log n$.

Note 1: Any stretch-1 routing scheme on trees requires $|\mathrm{table}(u)| + |\mathrm{address}(u)| = \Omega(\log^2 n / \log\log n)$ bits for some u in some tree.

Space Analysis

- $\mathrm{table}(u)$ is $O(t \log n)$ bits
- $\mathrm{address}(u)$ is $O(|\mathrm{hpath}(u)| \cdot \log n)$ bits

If port(a, b) is in $\mathrm{hpath}(u)$, then b has rank $\geqslant t$ for a.
So, $w(b) \leqslant w(a)/t$, and thus $|\mathrm{hpath}(u)| \leqslant \log_t n$.

$\Rightarrow$ adresses and tables have size $O(\log^2 n / \log\log n)$ with $t = \log n / \log\log n$.

Note 1: Any stretch-1 routing scheme on trees requires $|\mathrm{table}(u)| + |\mathrm{address}(u)| = \Omega(\log^2 n / \log\log n)$ bits for some u in some tree.

Note 2: Any stretch-1 routing scheme on trees requires $|\mathrm{table}(u)| = \Omega(\sqrt{n}\,)$ bits for some u in some graph, if $|\mathrm{address}(u)| = \lceil \log n \rceil$.

Outline

Sub-Linear Routing Table?

Sub-Linear Routing Table?

Bad news: Any stretch-1 routing scheme on general graphs requires $|\mathrm{address}(u)| + |\mathrm{table}(u)| = \Omega(n)$ bits for some u in some graph.

Sub-Linear Routing Table?

Bad news: Any stretch-1 routing scheme on general graphs requires $|\mathrm{address}(u)| + |\mathrm{table}(u)| = \Omega(n)$ bits for some u in some graph.

Idea: Use a stretch $s > 1$ to make wrong the previous statement.

Sub-Linear Routing Table?

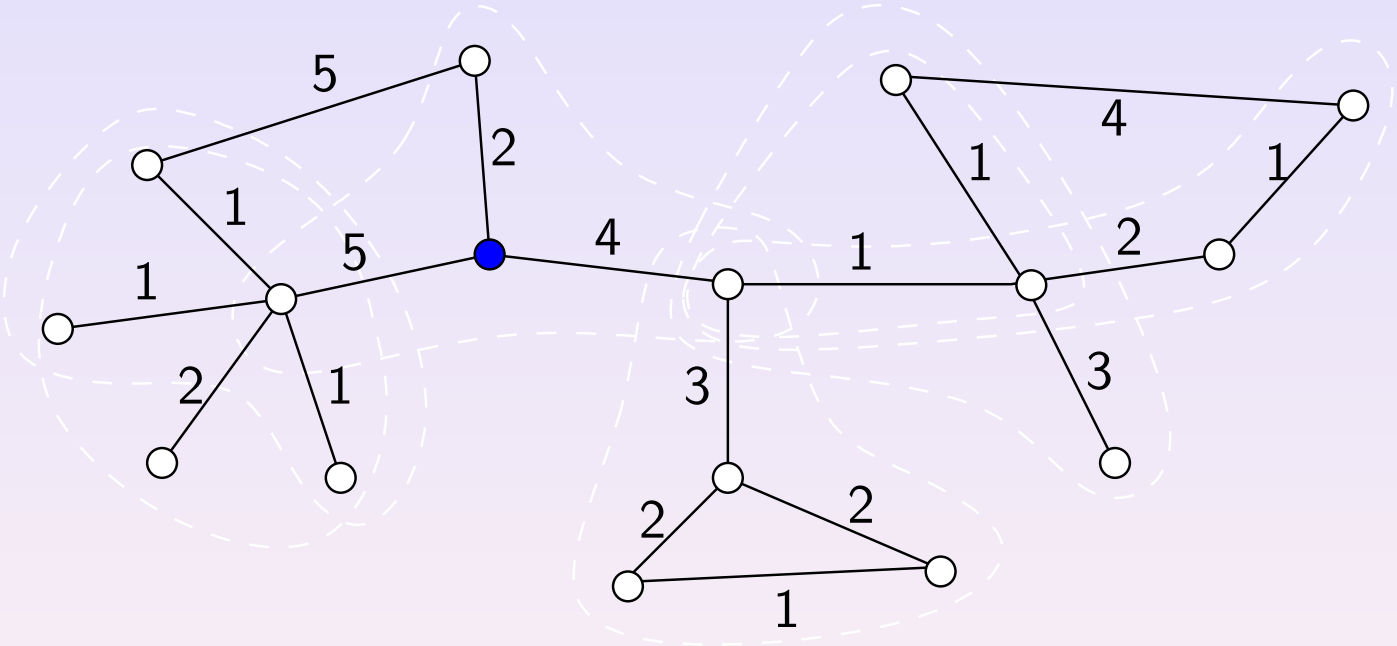
Bad news: Any stretch-1 routing scheme on general graphs requires $|\mathrm{address}(u)| + |\mathrm{table}(u)| = \Omega(n)$ bits for some u in some graph.

Idea: Use a stretch $s > 1$ to make wrong the previous statement.

Hints: Use sparse tree covers, and compact routing in trees.

Step #1: vincinity balls

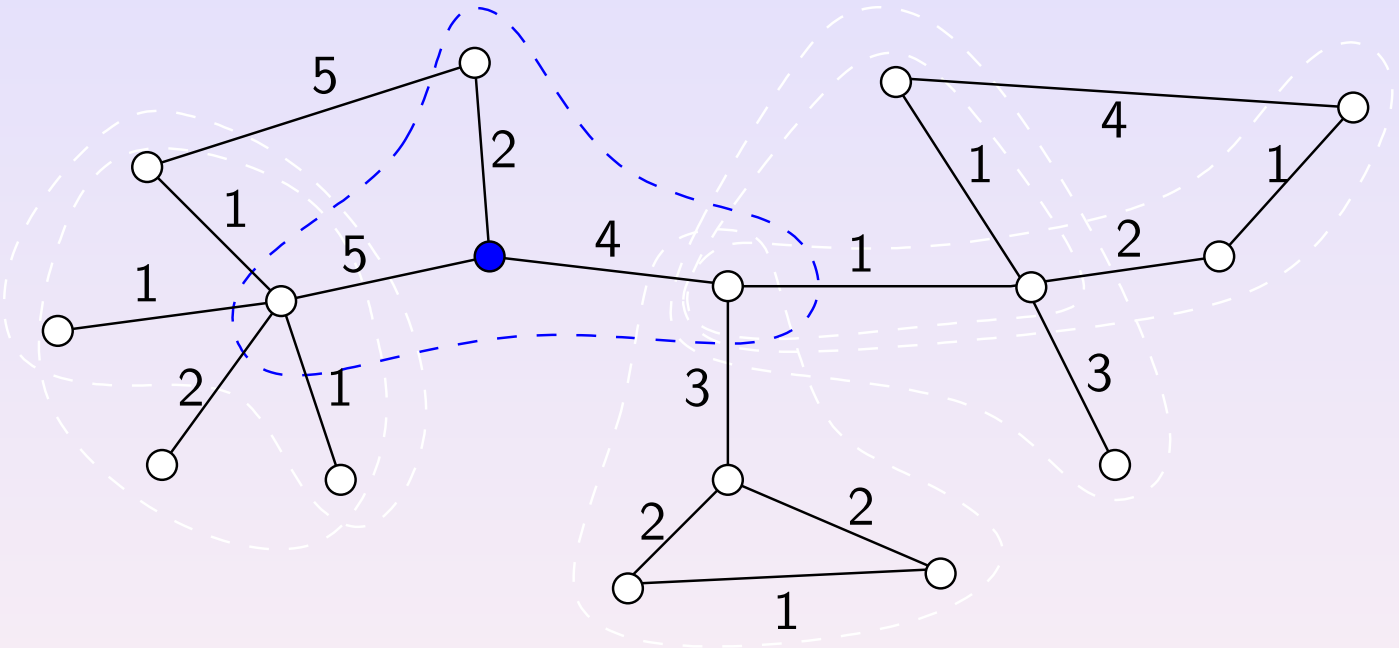
$x \mapsto B(x)$ be the t closest nodes around x, ties are broken consistently.



$n = 16$, $t = 4$

Step #1: vincinity balls

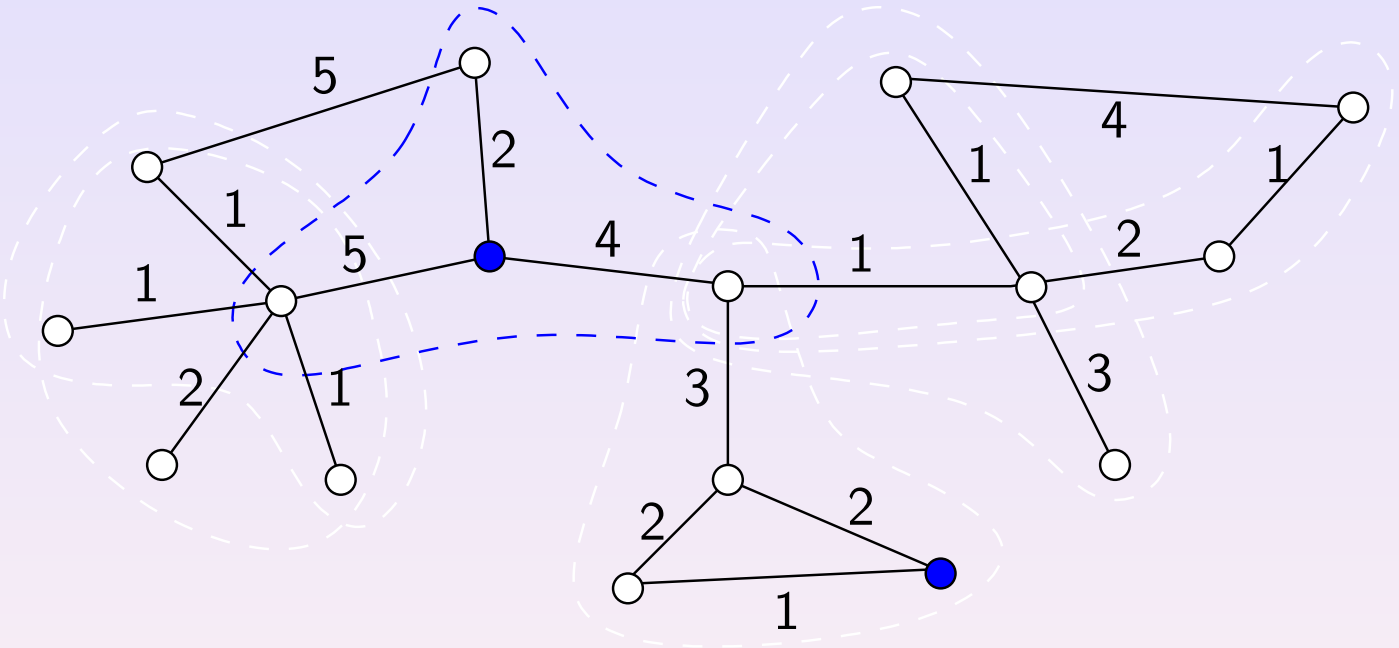
$x \mapsto B(x)$ be the t closest nodes around x, ties are broken consistently.



$n = 16$, $t = 4$

Step #1: vincinity balls

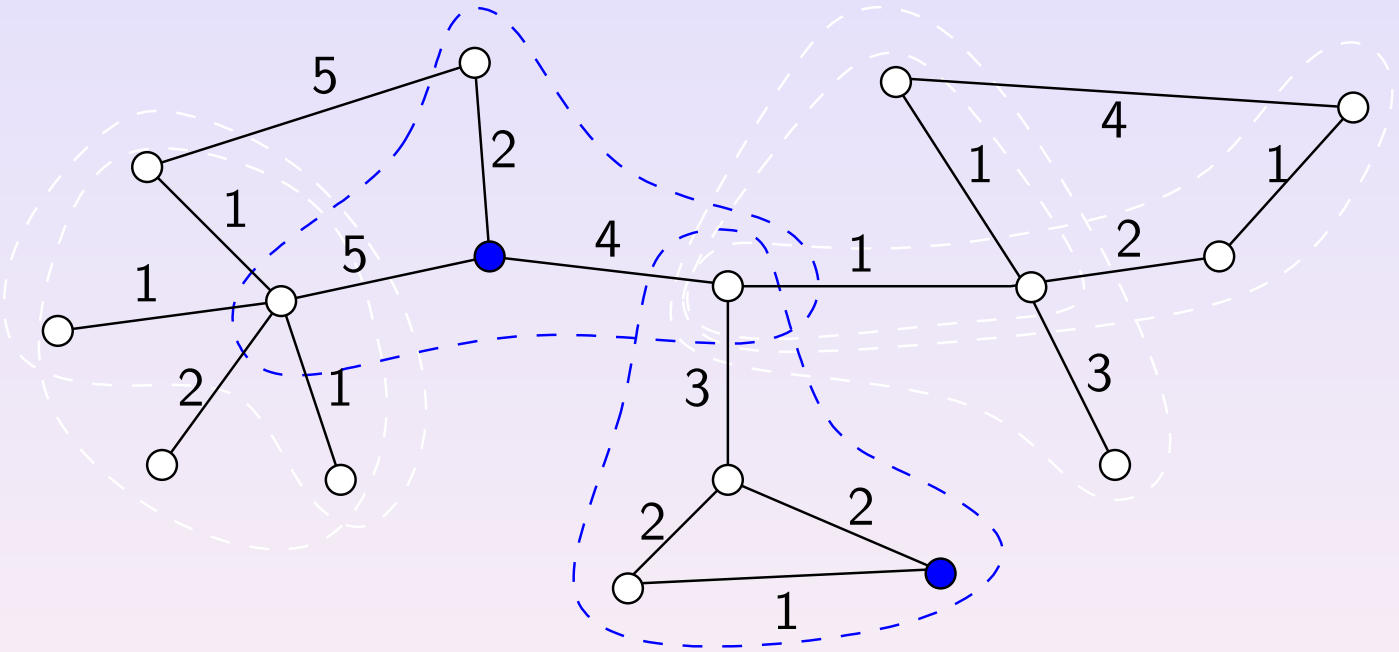
$x \mapsto B(x)$ be the t closest nodes around x, ties are broken consistently.



$n = 16$, $t = 4$

Step #1: vincinity balls

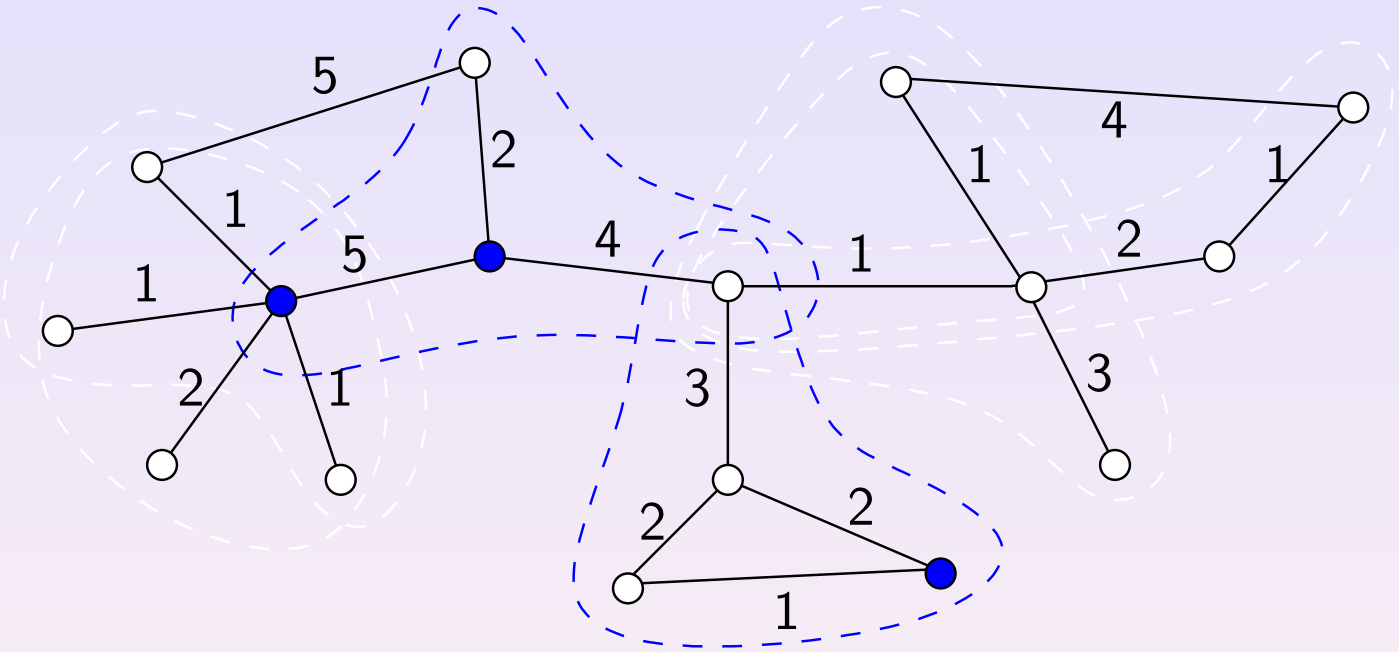
$x \mapsto B(x)$ be the t closest nodes around x, ties are broken consistently.



$n = 16$, $t = 4$

Step #1: vincinity balls

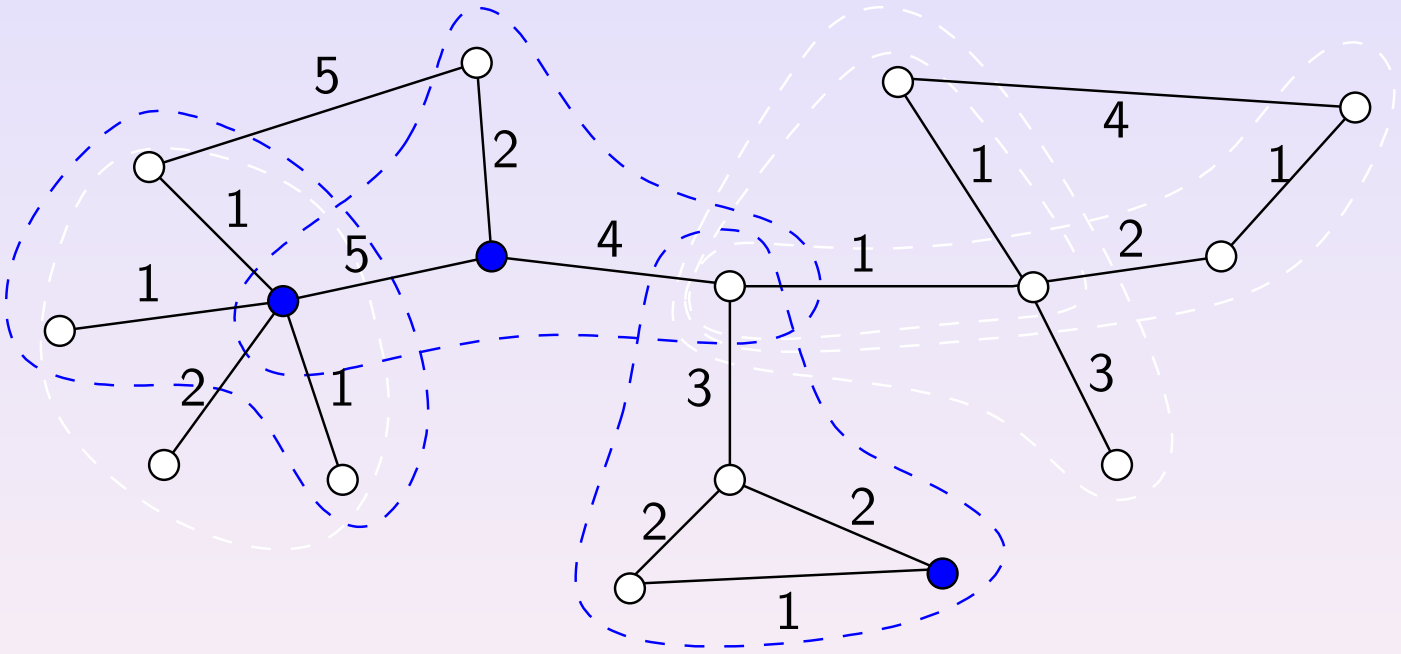
$x \mapsto B(x)$ be the t closest nodes around x, ties are broken consistently.



$n = 16$, $t = 4$

Step #1: vincinity balls

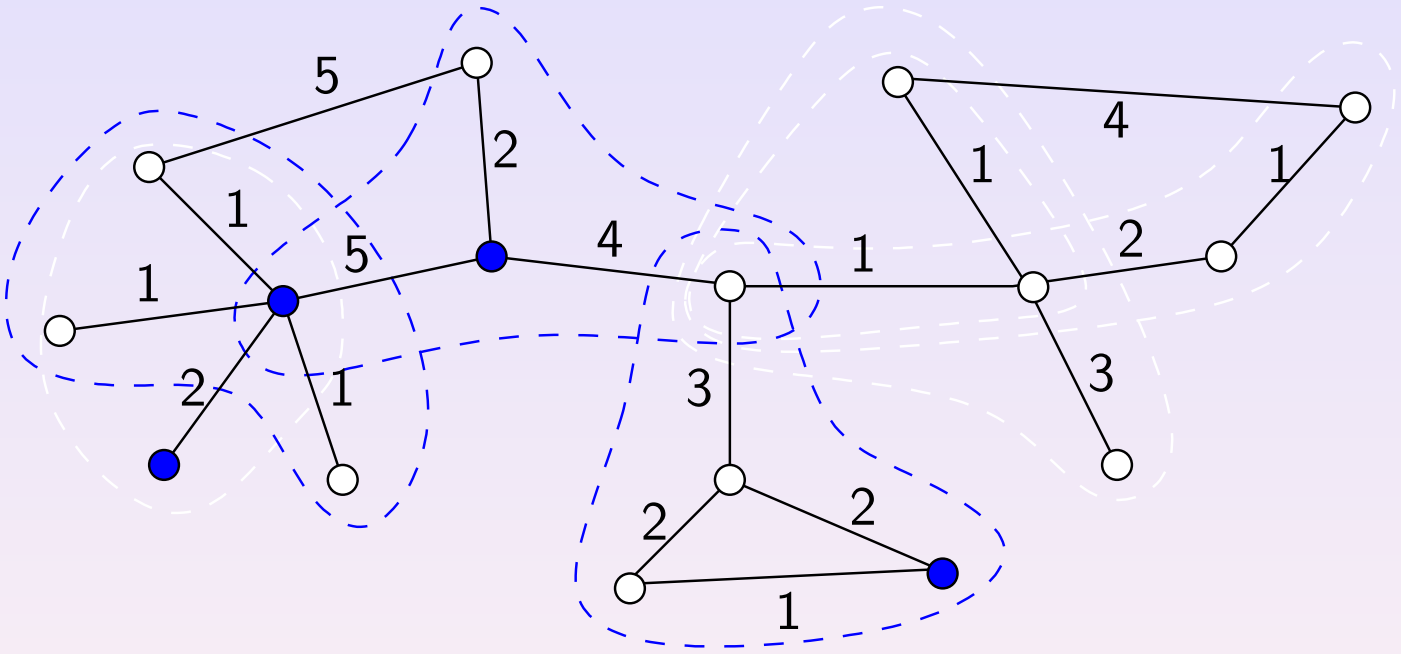
$x \mapsto B(x)$ be the t closest nodes around x, ties are broken consistently.



$n = 16$, $t = 4$

Step #1: vincinity balls

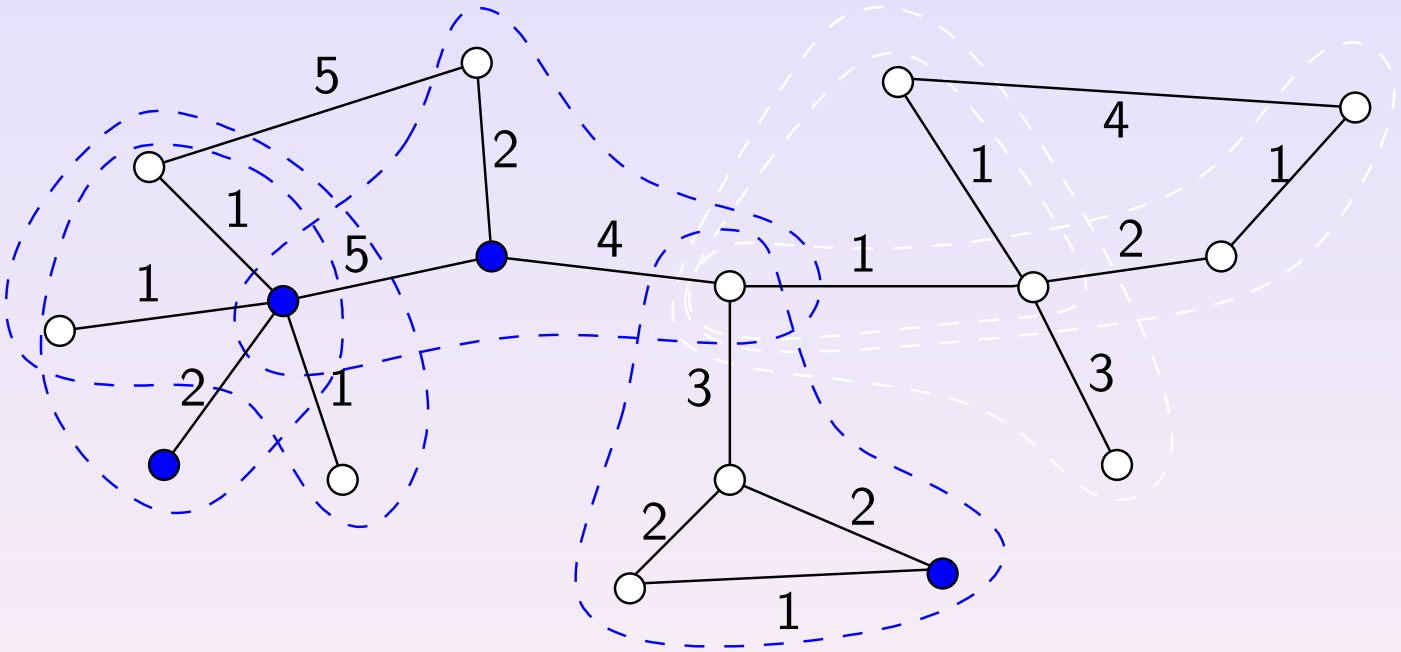
$x \mapsto B(x)$ be the t closest nodes around x, ties are broken consistently.



$n = 16$, $t = 4$

Step #1: vincinity balls

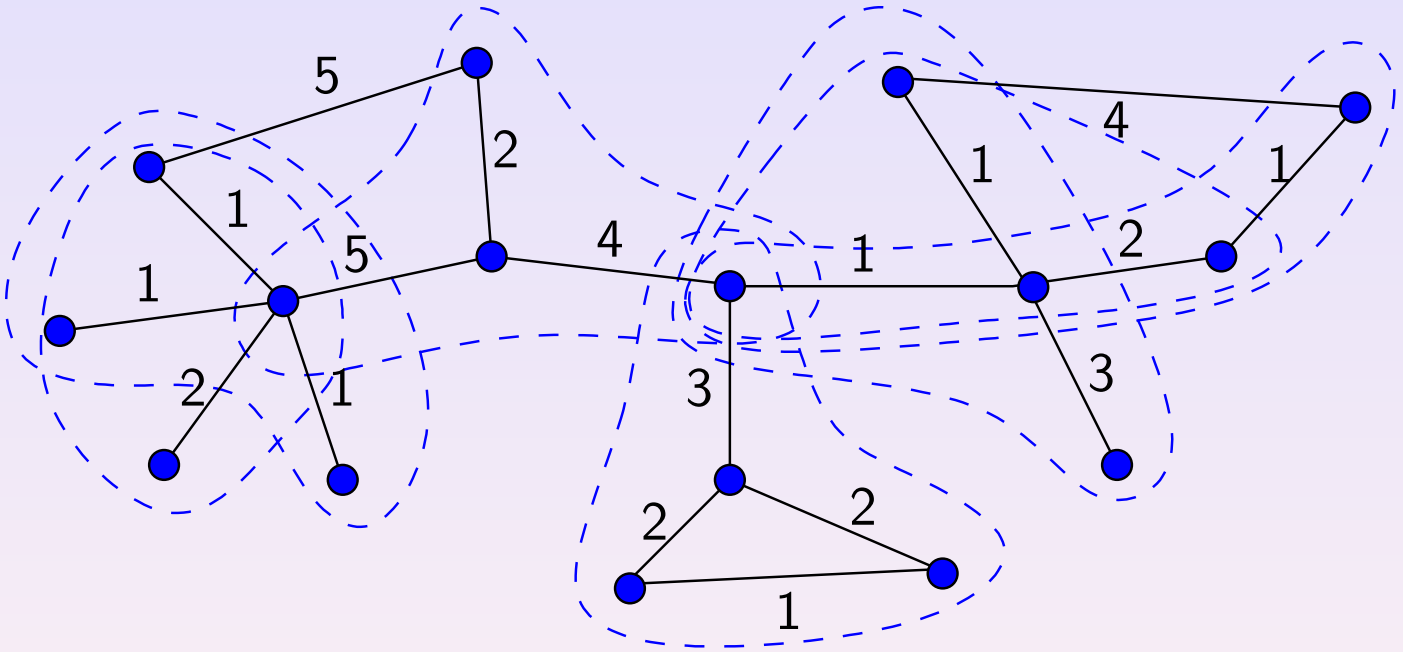
$x \mapsto B(x)$ be the t closest nodes around x, ties are broken consistently.



$n = 16$, $t = 4$

Step #1: vincinity balls

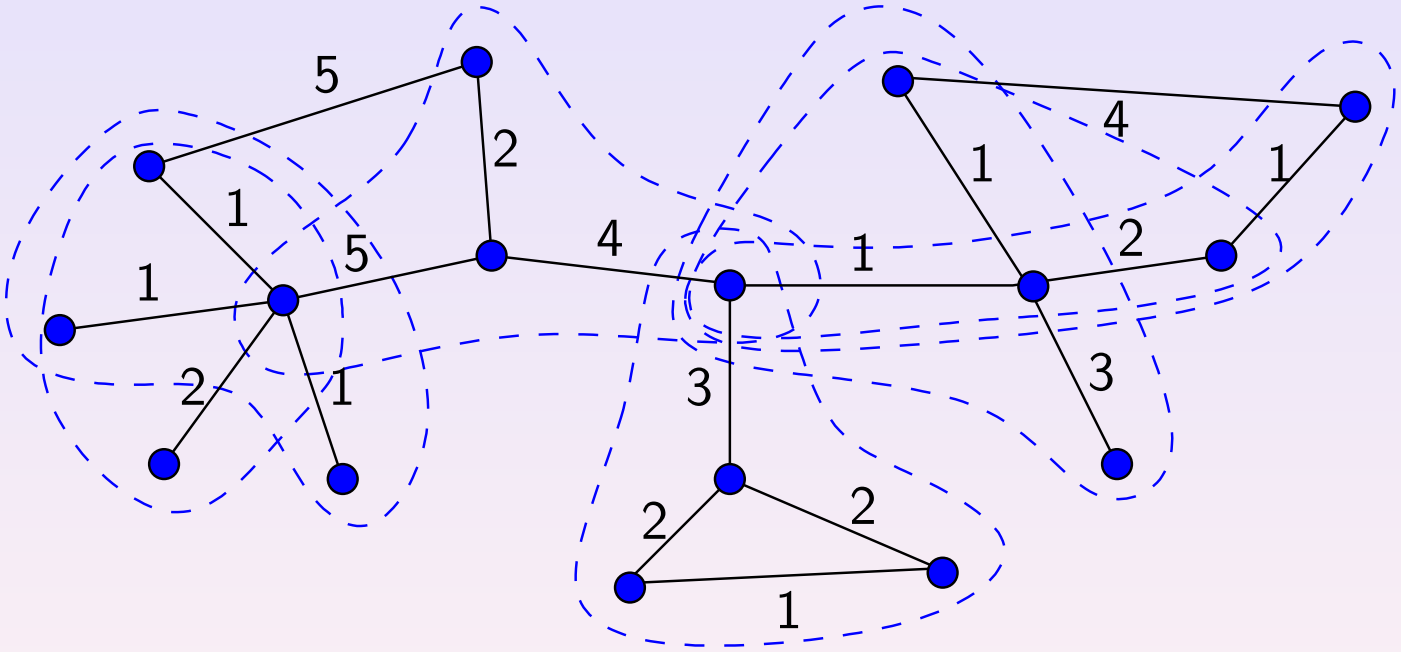
$x \mapsto B(x)$ be the t closest nodes around x, ties are broken consistently.



$n = 16$, $t = 4$

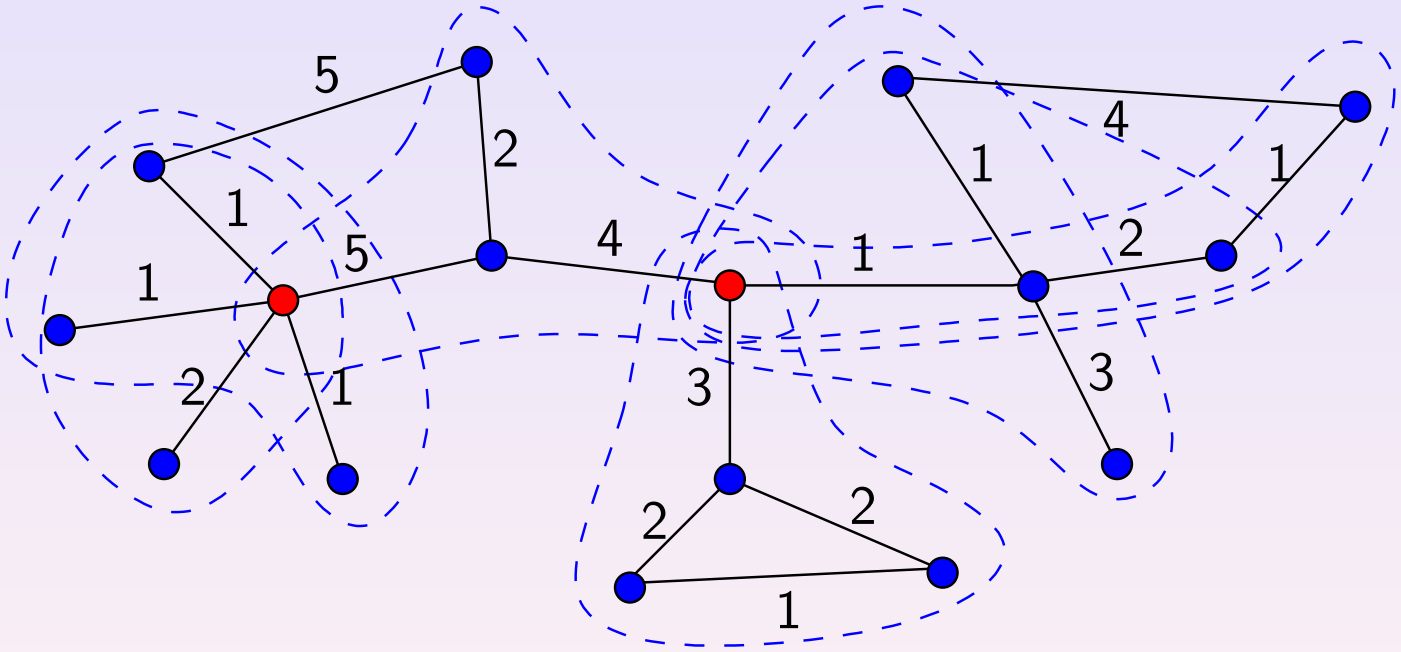
Step #2: Hitting Set

Select a small **hitting set** P for the family $\{B(x)\}_x$ ($P \cap B(x)$ for every $x \in V$). There exists P such that $|P| \leqslant (n \ln n)/t + 1$.



Step #2: Hitting Set

Select a small **hitting set** P for the family $\{B(x)\}_x$ ($P \cap B(x)$ for every $x \in V$). There exists P such that $|P| \leqslant (n \ln n)/t + 1$.



Step #2: Hitting Set (cont'd)

Greedy Algorithm:

1. $P := \varnothing$ and $\mathcal{F} := \{B(x) \mid x \in V\}$.
   Let $F(x) := \{B \in \mathcal{F} \mid x \in B\}$
2. While $\mathcal{F} \neq \varnothing$ do:
   - pick $p \in V$ such that $|F(p)|$ est maximum
   - $P := P \cup \{p\}$, $\mathcal{F} := \mathcal{F} \setminus F(p)$

Step #2: Hitting Set (cont'd)

Greedy Algorithm:

1. $P := \varnothing$ and $\mathcal{F} := \{B(x) \mid x \in V\}$.
   Let $F(x) := \{B \in \mathcal{F} \mid x \in B\}$
2. While $\mathcal{F} \neq \varnothing$ do:
   - pick $p \in V$ such that $|F(p)|$ est maximum
   - $P := P \cup \{p\}$, $\mathcal{F} := \mathcal{F} \setminus F(p)$

Analysis: Define f_i as $|\mathcal{F}|$ at the end of step i.
There must $\exists p \in V$ such that $|F(p)| \geqslant f_i \cdot t/n$. Indeed,

$$\sum_{v \in V} |F(v)| \;=\; \sum_{B \in \mathcal{F}} |B| \;\geqslant\; f_i \cdot t \;.$$

Thus, $\exists p \in V$ whose $F(p)$ has size larger than the average,

i.e., $f_i \cdot t/n$.

Step #2: Hitting Set (cont'd)

$$\begin{aligned} f_1 &= f_0 - f_0 \cdot t/n = f_0 \cdot (1 - t/n) \\ f_2 &= f_1 - f_1 \cdot t/n = f_1 \cdot (1 - t/n) = f_0 \cdot (1 - t/n)^2 \\ \ldots &= \ldots \\ f_i &= f_0 \cdot (1 - t/n)^i \quad \forall i \geqslant 0 \end{aligned}$$

Algorithm stops at the first i_0 such that $f_{i_0} < 1$, and then $|P| = i_0$. For all $0 < \alpha \leqslant 1/2$ and $\beta > 0$, $(1 - \alpha)^\beta < e^{-\alpha\beta}$:

$$(1 - t/n)^{i_0} < e^{-t \cdot i_0/n} \;\Rightarrow\; f_0 \cdot (1 - t/n)^{i_0} < f_0/e^{t \cdot i_0/n}$$

So $f_{i_0} < 1$ implies $f_0/e^{t \cdot i_0/n} < 1$ implies $i_0 > (n \ln f_0)/t$.
Algorithm has finished at the end of step $i_0 = \lfloor (n \ln n)/t \rfloor + 1$.

Step #3: Tree Cover

Small trees: Associate with every $x \in V \setminus P$:
a shortest-path tree T_x spanning $B(x)$

Big trees: Associate with every $p \in P$:
a shortest-path tree T_p spanning G

Denote by $\mathcal{C}$ the family of these trees.

Step #3: Tree Cover

Small trees: Associate with every $x \in V \setminus P$:
a shortest-path tree T_x spanning $B(x)$

Big trees: Associate with every $p \in P$:
a shortest-path tree T_p spanning G

Denote by $\mathcal{C}$ the family of these trees.

Claim: For every x, y, $\exists T \mathcal{C}$ such that $d_T(x, y) \leqslant 3d(x, y)$.

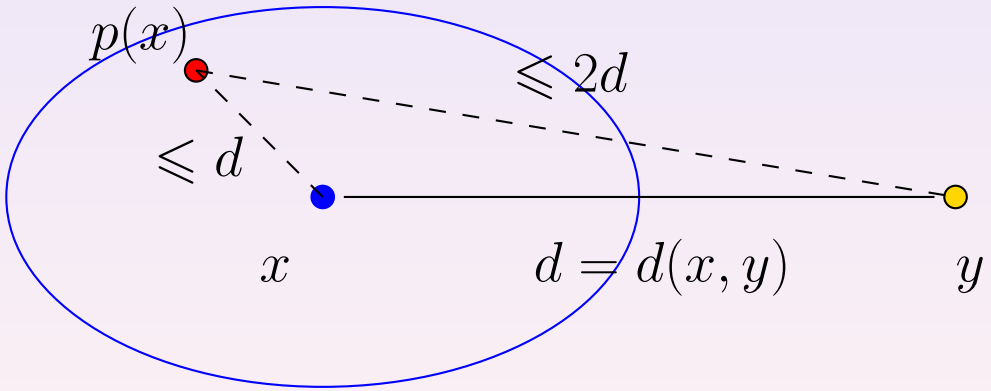
Step #3: Tree Cover

Small trees: Associate with every $x \in V \setminus P$:
a shortest-path tree T_x spanning $B(x)$

Big trees: Associate with every $p \in P$:
a shortest-path tree T_p spanning G

Denote by $\mathcal{C}$ the family of these trees.

Claim: For every x, y, $\exists T \mathcal{C}$ such that $d_T(x, y) \leqslant 3d(x, y)$.



Step #3: Tree Cover (cont'd)

Choose $t := \sqrt{n \ln n}$, so that:
$|P| \leqslant (n \ln n)/t + 1 = \sqrt{n \ln n} + 1.$

Observation: The graph $H := \bigcup_{T \in \mathcal{C}} T$ is a spanning subgraph of G and its **number of edges** is at most:

$$(n - |P|) \cdot (t - 1) + |P| \cdot (n - 1) \quad < \quad 2n\sqrt{n \ln n} = \tilde{O}(n^{3/2})$$

Step #3: Tree Cover (cont'd)

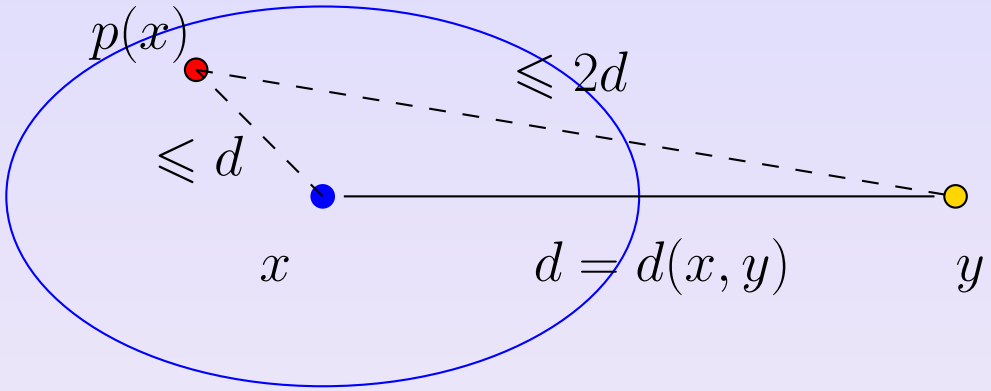
Choose $t := \sqrt{n \ln n}$, so that:
$|P| \leqslant (n \ln n)/t + 1 = \sqrt{n \ln n} + 1$.

Observation: The graph $H := \bigcup_{T \in \mathcal{C}} T$ is a spanning subgraph of G and its **number of edges** is at most:

$$(n - |P|) \cdot (t - 1) + |P| \cdot (n - 1) \quad < \quad 2n\sqrt{n \ln n} = \tilde{O}(n^{3/2})$$

Our Goal: Design a stretch-3 routing scheme with routing tables of $\tilde{O}(n^{1/2})$ bits per node and polylog addresses.

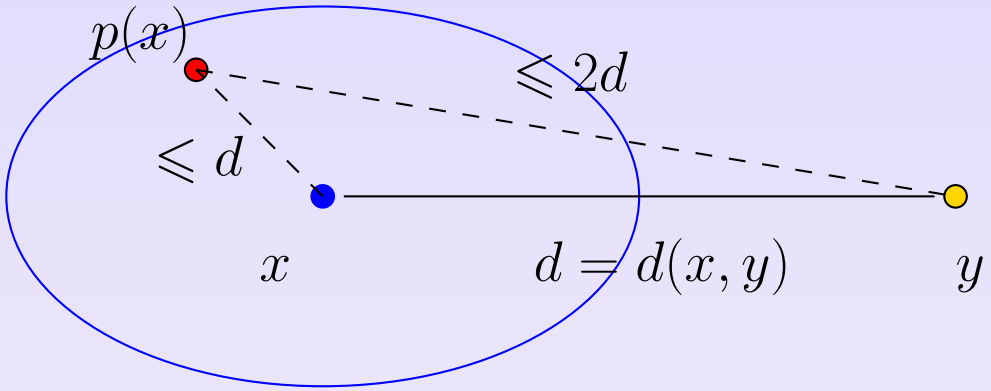
How to Route from x to y?



Easy case: $y \in B(x)$.
Node x can store $B(x)$ and the port to each $v \in B(x)$. This is $\tilde{O}(t) = \tilde{O}(n^{1/2})$ bits of information.

Claim: if w is on a shortest-path from x to v, then $v \in B(w)$.

How to Route from x to y?



Otherwise: $y \notin B(x)$. Then, route to $p(x)$, and then try to reach y along $T_{p(x)}$.

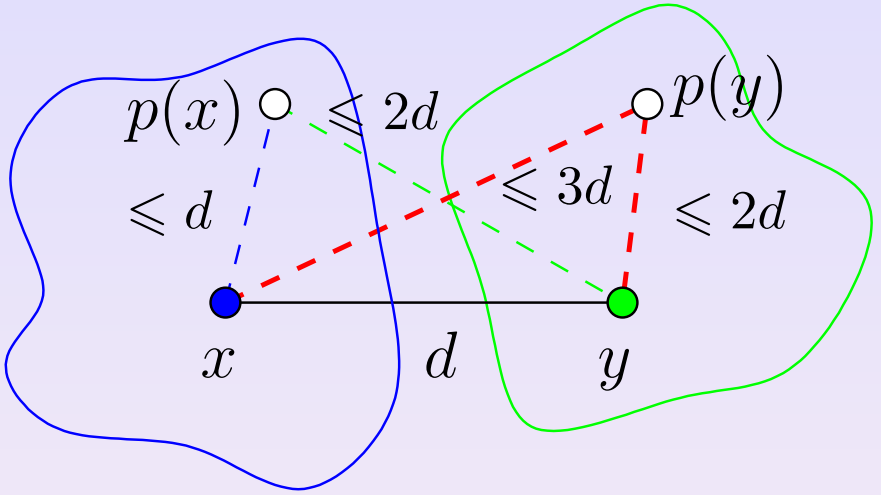
Each node z can store $\mathrm{table}_{T_p}(z)$ for each big tree T_p (there are only $|P|$ such trees). But, neither x neither $p(x)$ can store $\mathrm{address}_{T_{p(x)}}(y)$ for all possible y. If we want polylog(n) addresses, y cannot store $\mathrm{address}_{T_{p(x)}}(y)$ in its address.

Another Solution

Hints: if $y \notin B(x)$, route along $T_{p(y)}$ and store $\mathrm{address}_{T_{p(y)}}(y)$ in the y's address.

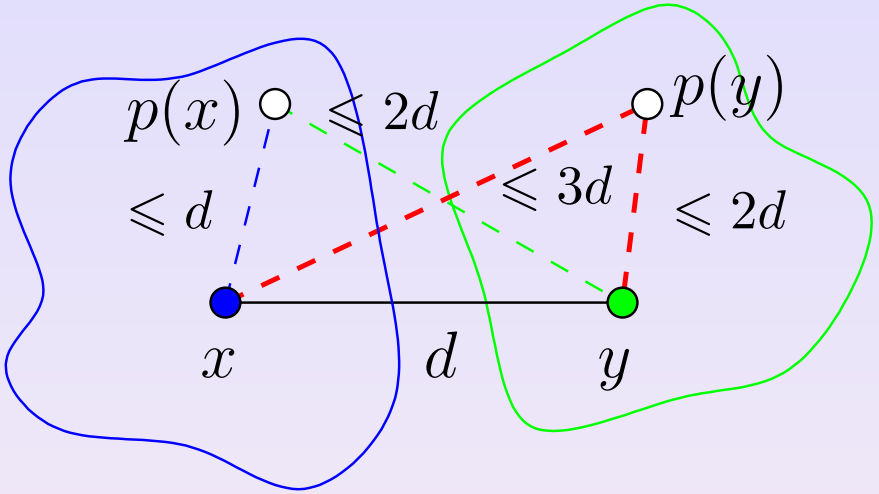
Another Solution

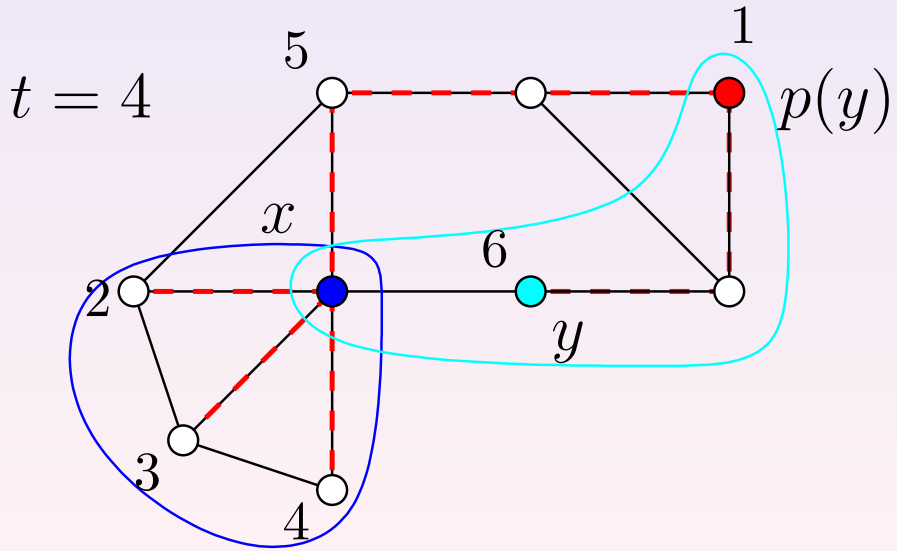
Hints: if $y \notin B(x)$, route along $T_{p(y)}$ and store $\text{address}_{T_{p(y)}}(y)$ in the y's address.



Another Solution

Hints: if $y \notin B(x)$, route along $T_{p(y)}$ and store $\text{address}_{T_{p(y)}}(y)$ in the y's address.





Outline

Best Known Universal Routing Schemes

[addresses are polylog, space is up to polylog]

stretch (LB)	stretch (UB)	space	scheme
1	1	n	name-indep.
3	3	$\sqrt{n}$	name-indep.
5	7	$n^{1/3}$	labeled
...	...	...	...
$1.5k-1$	$4k-5$	$n^{1/k}$	labeled
$2k-1$	$\approx 100k$	$n^{1/k}$	name-indep.

Best Known Universal Routing Schemes

[addresses are polylog, space is up to polylog]

stretch (LB)	stretch (UB)	space	scheme
1	1	n	name-indep.
3	3	$\sqrt{n}$	name-indep.
5	7	$n^{1/3}$	labeled
...	...	...	...
$1.5k-1$	$4k-5$	$n^{1/k}$	labeled
$2k-1$	$\approx 100k$	$n^{1/k}$	name-indep.

Observations: Only the two first lines of the table are known to be optimal. For $n^{1/3}$ space, there is a lower bound of 5 on the stretch. For general k, the conjecture is stretch $2k-1$, but neither the current lower or upper bounds reach this bound when $k > 5$.