Scalable Routing Easy as PIE: a Practical Isometric Embedding Protocol

Julien Herzen (EPFL)

joint work with

Cedric Westphal (Huawei Innovations)
Patrick Thiran (EPFL)

October 18th, 2011
Internet routing has a scalability problem

- Costly recomputation of tables
- Instabilities
- Costly lookups in huge tables
- Energy hungry
- Heavily relies on Moore’s law to keep up
- Could get much worse with IPv6...
**Fundamental limit**

- **Stretch**: Length of a path found by a routing algorithm, divided by the shortest possible path length

---

[Gavoille et al. ’97]

For a network of $n$ nodes, **guaranteeing a stretch strictly below 3 requires routing tables of size $O(n)$**

⇒ Consider schemes that *may* inflate path length to achieve sub-linear scalability
Geometric routing

Each node needs to know only the coordinates of its neighbors

**Forwarding:** pick the neighbor closest to the destination

**Problem:** The packets can meet a dead end!
The Internet has a hierarchical structure
Tree routing

- Trees are easy to build distributively
- They can ensure 100% routing success (exactly one path between any two nodes)
Tree routing

- Trees are easy to build distributively
- They can ensure 100% routing success (exactly one path between any two nodes)

\[ \text{stretch} = 1.5 \]

Tree routing is not efficient...
PIE embeds trees into metric spaces

- Root has coordinate 0
- Binary representation of each child

![Diagram of tree with binary coordinates and stretch metric]
PIE embeds trees into metric spaces

- Then recursively, each parent:
  - Send its coordinates to its children. The children keep the signs, but increase absolute values of these coordinates by link cost to parent
  - If more than one child: the parent also sends the binary representation of each child, that is appended to the coordinates
PIE embeds trees into metric spaces

- Then recursively, each parent:
  - Send its coordinates to its children. The children keep the signs, but increase absolute values of these coordinates by link cost to parent
  - If more than one child: the parent also sends the binary representation of each child, that is appended to the coordinates
PIE embeds trees into metric spaces

- Then recursively, each parent:
  - Send its coordinates to its children. The children keep the signs, but increase absolute values of these coordinates by link cost to parent.
  - If more than one child: the parent also sends the binary representation of each child, that is appended to the coordinates.
Routing using the embedding

Distance computation: $l_{\infty}$-norm on the common coordinates
Routing using the embedding

Distance computation:
\( l_\infty \)-norm on the common coordinates
Routing using the embedding

Distance computation:

\[ l_\infty \text{-norm on the common coordinates} \]
Routing using the embedding

Distance computation:
\( l_\infty \)-norm on the common coordinates
Routing using the embedding

Distance computation:
\( l_\infty \)-norm on the **common coordinates**

![Graph diagram showing node coordinates and distances]

- \( s \rightarrow -2, -2, 2 \)
- \( -2, -2, 2 \rightarrow -2, 2, 2 \)
- \( -2, 2, 2 \rightarrow -2, 2, 2 \)
- \( -2, 2, 2 \rightarrow -1, 1, 1 \)
- \( -1, 1, 1 \rightarrow -1, -1, -1 \)
- \( -1, -1, -1 \rightarrow 0 \)
- \( 0 \rightarrow 2, -2, -2, -1, -1 \)
- \( 2, -2, -2, -1, -1 \rightarrow 4, -4, -4, -3, 3 \)
- \( 4, -4, -4, -3, 3 \rightarrow 3, -3, -3, -2, 2 \)
- \( 3, -3, -3, -2, 2 \rightarrow 2, -2, -2, -1, 1 \)
- \( 2, -2, -2, -1, 1 \rightarrow 2, -2, -2, 1, -1 \)
- \( 2, -2, -2, 1, -1 \rightarrow -2, -2, -2, 1, -1 \)
Routing using the embedding

Distance computation:
\( l_\infty \)-norm on the common coordinates

![Diagram showing distance computation and stretch = 1]
PIE embeds trees into metric spaces

- This approach still guarantees 100% routing success
- It is better than tree routing
- But still lacks some topological information in some situations...
PIE embeds trees into metric spaces

- This approach still guarantees 100% routing success
- It is better than tree routing
- But still lacks some topological information in some situations...
PIE embeds trees into metric spaces

- This approach still guarantees 100% routing success
- It is better than tree routing
- But still lacks some topological information in some situations...
**Solution:** build several smaller trees

- Easy to build distributively (random self-elected roots)
- Still scalable if each node belongs to $O(\log n)$ trees
Solution: build several smaller trees

- Easy to build distributively (random self-elected roots)
- Still scalable if each node belongs to $O(\log n)$ trees
**Solution:** build several smaller trees

- Easy to build distributively (random self-elected roots)
- Still scalable if each node belongs to $O(\log n)$ trees
Trees covering several **levels**

- **Forwarding:** use common tree that provides smallest distance
- **Big trees:** good for long paths
- **Small trees:** good for short paths
- Match well the **self-similar** structure of the Internet
- \( O(\log n) \) **levels** \( \rightarrow \) only \( O(\log n) \) set of coordinates per node
Trees covering several **levels**

- Level 1
- Level 2
- Level 3
Trees covering several levels

Level 1

Level 2

Level 3
Trees covering several **levels**

- Level 1
- Level 2
- Level 3
Trees covering several **levels**
Trees covering several **levels**
Wrapping up

**Theorem 1**
The number of coordinates is $O(\log^3 n)$ w.p. 1 for random power-law graphs

Proof uses recent results on the diameter of such graphs

**Theorem 2**
The embedding produced by PIE ensures 100% routing success

The embedding is *greedy*

- **Distributed**
  - Embedding procedure goes from root to leaves
  - Self-elected roots

- **Local and fast forwarding decisions**
  - Only compute a few distances
Performance

- Internet AS level\(^1\)
- \(m\): Number of levels
- Link weights \(\sim\) Unif\([1, 10]\)

**Stretch CDF:**

Average stretch \(<1.03\) for 7 levels and more

\(^1\): DIMES [Shavitt et al. ’05], dataset of March 2010
Performance

• Synthetic graphs\[^1\], with power-law exponent $\lambda$
• Number of levels $m \in O(\log n)$

**Average stretch:**

![Graph showing average stretch](image)

- Low stretch scales with the size of the network

[1]: GLP [Bu et al. '02]
Scalability

- Number of levels $m \in O(\log n)$

**Total number of coordinates per node (min, max, average):**

Routing tables of size $O(\log^3 n)$
Resilience to network failures

Geometric coordinates provide route diversity for free

Routing success after failures:

For a given success ratio, PIE needs to re-compute its state less often
Conclusion

• **Distributed** construction of the coordinates

• **Scalable:** routing tables of size $O(\log^3 n)$ with probability 1

• **Efficient paths**
  ▶ Can maintain average stretch $< 1.03$
  ▶ Adapts well to weighted graphs

• **Guaranteed routing success** on any connected graph

• **Other applications:** overlay, peer-to-peer, distance estimation, etc...

• **Future work:**
  ▶ Policy routing, traffic engineering, etc...
  ▶ Economic considerations (who is the root?)
The congestion induced is the same than for shortest path routing.
Some related work

- Geographic/geometric routing for ad-hoc networks
  - Euclidean embeddings, not well suited for the Internet, local minima
- Compact routing [Thorup et al. '01] (TZ)
  - Scalability $O(n^{1/2}) \rightarrow$ still a fractional power of $n$
- Hyperbolic embeddings of Internet topology [Papadopoulos et al. 2010] and [Boguna et al. 2010]
  - Presence of local minima, routing success not guaranteed
- Quasi-greedy embedding in Euclidean spaces [Westphal et al. ’09]
  - Produces local minima and requires a recovery mechanism
- Geometric routing with bounded stretch [Flury et al. ’09]
  - Not distributed
- Compact routing for power-law graphs [Brady et al. ’06] (BC)
  - Not distributed
Comparison with TZ, BC and TZ+BC

- Power-law random graphs with exponent $\lambda$
- Graphs and results for TZ, BC and TZ+BC come from [Brady et al. ’06]

**Average stretch:**

![Graph showing average stretch](image)