OBST: A Self-Adjusting Peer-to-Peer Overlay Based on Multiple BSTs

Chen Avin (BGU)
Michael Borokhovich (BGU)
Stefan Schmid (T-Labs)
From “Optimal” Networks to Self-Adjusting Networks

- Networks become more and more dynamic (e.g., flexible SDN control)
- Vision: go beyond classic “optimal” static networks
- Example (of this paper): Peer-to-peer

**Chord, Pastry, SHELL**
- Hypercubic
- Log diameter
- Log degree
- Log routing

**Koorde, ...**
- Constant degree
- Log routing

**Pancake**
- Log/loglog degree and log/loglog routing
From “Optimal” Networks to Self-Adjusting Networks

- Networks become more and more dynamic (e.g., flexible SDN control)
- Vision: go beyond classic “optimal” static networks
- Example: Peer-to-peer Chord, Pastry, SHELL, Koorde, Pancake
- Hypercubic
- Log diameter
- Log degree
- Log routing
- Constant degree
- Log/loglog degree and log/loglog routing

What if networks could self-adjust depending on communication pattern?

Stefan Schmid (T-Labs)
An Old Concept: Move-to-front, Splay Trees, ...

- Classic data structures: lists, trees
- Linked list: move frequently accessed elements to front!
- Trees: move frequently accessed elements closer to root
An Old Concept: Move-to-front, Splay Trees, ...

- Classic data structures: lists, trees
- Linked list: move frequently accessed elements to front!
- Trees: move frequently accessed elements closer to root
An Old Concept: Move-to-front, Splay Trees, ...

- Classic data structures: lists, trees
- Linked list: move frequently accessed elements to front!
- Trees: move frequently accessed elements closer to root!

Splay Trees!
The Vision: Splay Networks ("Distributed Splay Trees")

- Most simple self-adjusting tree network: Binary Search Tree (BST)
The Vision: Splay Networks (“Distributed Splay Trees”)

- Most simple self-adjusting tree network: Binary Search Tree (BST)
The Vision: Splay Networks ("Distributed Splay Trees")

- Most simple self-adjusting tree network: Binary Search Tree (BST)

Communication between peer pairs!
(Not only lookups from root…)

Stefan Schmid (T-Labs)
The Vision: Splay Networks ("Distributed Splay Trees")

- Most simple self-adjusting tree network: Binary Search Tree (BST)

Why BST?!
- Most simple generalization of classic data structure
- Allows for local routing!
- Allows for algebraic gossip
Model: Self-Adjusting SplayNets

Input:
- communication pattern: (static or dynamic) graph

Output:
- sequence of network adjustments

Cost metric:
- expected path length
- # (local) network updates

Stefan Schmid (T-Labs)
Some Facts: Optimal Algorithm and Amortized Cost

**Optimal Static Solution**
- Dynamic program: decouple left from right!
- Polynomial time (unlike MLA!)
- So: solved M”BST”A

**Dynamic Solution**
- There exists self-adjusting algorithm
- Inspired by Splay trees
- E.g., optimal under product distribution: $P[(u,v)] = P(u)P(v)$
- E.g., optimal under directed BST, non-crossing matching, …
- Lower bounds…

**Upper Bound**

$A$-Cost $< H(X) + H(Y)$

where $H(X)$ and $H(Y)$ are empirical entropies of sources resp. destinations

Adaption of Tarjan&Sleator

**Lower Bound**

$A$-Cost $> H(X | Y) + H(Y | X)$

where $H( | )$ are conditional entropies.

Assuming that each node is the root for “its tree”
From One to Multiple BSTs

Research Question:
- What is the benefit of multiple BSTs?
- Focus on amortized communication cost

Two Models:
- **Lookup Model**: Classic datastructure where requests originate at root
- **Routing Model**: Peer-to-peer communication
Our Contribution

Routing:
- Static OBST:
  - A single additional BST can reduce costs from $O(\log n)$ to $O(1)$!
  - Entropy-based upper bounds on amortized communication costs
- Dynamic OBST:
  - Self-adjusting splay trees
  - Simulations
- Simulations

Lookup:
- Static OBST:
  - OBST($k$) can only improve by additive $-\log(k)$ compared to OBST(1)
Routing: OBST(2) vs OBST(1)

- Easy to embed in two BSTs: one for each (cost $O(1)$)
- Hard to embed in one BST: because large interval cut ("crossing-matching")

Laminated scenario:
Self-Adjusting OBST

**Algorithm 1** Dynamic OBST($k$)

1: (* upon request $(u, v)$ *)
2: find BST $T \in$ OBST where $u$ and $v$ are closest;
3: $w := \text{LCA}_T(u, v)$;
4: $T' := \text{splay } u \text{ to root of } T(w)$;
5: \text{splay } $v \text{ to the child of } T'(u)$;

- Splay to Least Common Ancestor (more local!)…
- … in best tree!
Initially: $k$ independent, random BSTs
- Communication models: matching and random walk
- (a) More trees help
- (b) On Random Graph relatively stable, almost perfect convergence
- (c) OBST(2) convergence to perfect tree for “bad example”
- Routing cost under $\lambda$ joins/leaves between a lookup operation
- More plots in full paper…
Conclusion

- Vision: self-adjusting networks
- Interesting generalization of Splay trees
  - SplayNets
    - Formal analysis reveals nice properties
    - Amortized costs good: but tight?
    - Competitive ratio remains open
- OBST: lookup vs routing
Algorithm 2 Double Splay Algorithm DS

1: (* upon request \((u, v)\) in \(T\) *)
2: \(w := \alpha_T(u, v)\)
3: \(T' := \text{splay} \ u\) to root of \(T(w)\)
4: \(\text{splay} \ v\) to the child of \(T'(u)\)

"Host Graph"

"Guest Graph"
Peer-to-Peer and the Cloud!
Keynote by Rick McGeer, papers by Jen Rexford, Holger Karl, Christof Fetzer, Pietro Michardi, etc.
Dynamic program
- Binary search: decouple left from right!
- Polynomial time (unlike MLA!)
- So: solved M”BST”A

See also:
- Related problem of phylogenetic trees