Traditional Networks

- Usually optimized for the “worst case” (all-to-all communication)
- Lower bounds and hard trade-offs, e.g., degree vs diameter
Technology Enables Dynamic Reconfigurable Networks
Dynamic Reconfigurable Networks

- Dynamically optimized toward the (time-varying) demand

ProjecToR: Agile Reconfigurable Data Center Interconnect. Ghobadi et al., SIGCOMM'16
Motivation for Reconfigurability

- Sparsity of communication matrix
- The difficulty of estimating traffic matrices ahead of time and predicting the future demand

Dynamic self-adjusting networks can adjust to and leverage these patterns!
Self-Adjusting Data Structures

- The vision of self-adjusting networks is similar in spirit to the vision of self-adjusting datastructures

- Splay Trees

INFOCOM 2019
D. Sleator and R. Tarjan, Self-adjusting binary search trees.
Self-Adjusting Networks

Datastructure

Network

request h
SplayNets

• Distributed tree network

• Improves the communication cost between two nodes in a self-adjusting manner

• Nodes communicating more frequently become topologically closer to each other over time

SplayNets

- Move-to-root × Lowest common ancestor (LCA)

- \( LCA(u,v) \): The lowest common ancestor of two nodes \((u,v)\) is the closest node to \(u\) and \(v\) that has both of them as descendants

- Locality is preserved!
Our Contributions

• While SplayNets are intended for distributed applications, so far, only sequential algorithms are known to maintain SplayNets

• We present DiSplayNets, the first distributed and concurrent implementation of SplayNets
Model

• Network model:
  • Binary tree $T$ comprised of a set of $n$ communication nodes

• Sequence of communication requests $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_m)$:
  • $\sigma_i = (s_i, d_i)$: begins at $b_i$ and ends at $e_i$

• Given $\sigma_i(s_i, d_i)$, $s_i$ and $d_i$ rotate in parallel towards the $LCA(s_i, d_i)$:
  • LCA might change over time
- Local Reconfigurations: decentralized and concurrent topological adjustments
- Independent clustering: nodes in a cluster updated their links in parallel without interference
- Prioritization: for nodes to achieve a consensus
Local reconfigurations

- Step: $step_t (u)$
Local reconfigurations

- Step: $step_t (u)$

zig-zig
Local reconfigurations

- Step: $step_t (u)$

zig-zag
Independent clustering

- Cluster: $C_t(u)$
  - Requester and master nodes

$C_t(u)$

INFOCOM 2019  Distributed Self-Adjusting Tree Networks
Prioritization

• In order to ensure deadlock and starvation freedom, concurrent operations are performed according to a priority

• Given two requests $\sigma_i(s_i, d_i)$ and $\sigma_j(s_j, d_j)$, such that $b_i < b_j$, $\sigma_i$ has priority over $\sigma_j$
Nodes form an independent cluster to perform a local reconfiguration given the priority of the request.
• Given $\sigma_i(s_i, d_i)$ and $\sigma_j(s_j, d_j)$, such that $b_i < b_j$: 

$s_i$ has priority over $s_j$
DiSplayNet

- State machine executed by each node in parallel

**Passive**

**Climbing**

**Waiting**

- it is not the source or destination of any request
- it has an active request and it is not the LCA
- it has an active request and it is the LCA
- it has an active request and it is the neighbor of the other node in the request

INFOCOM 2019 Distributed Self-Adjusting Tree Networks
DiSplayNet

- State machine executed by each node in parallel
- State machine executed by each node in parallel
• State machine executed by each node in parallel
The algorithm is executed in rounds:

- Phase 1: Cluster Requests
- Phase 2: Top-down Acks
- Phase 3: Bottom-up Acks
- Phase 4: Link Updates
- Phase 5: State Updates
• The algorithm is executed in rounds

\[ s_i < d_j \]

\[ b_i < b_j \]

Phase 1
Cluster Requests

\[ s_i \text{ has priority over } d_j \]
• The algorithm is executed in rounds
Phase 3
Bottom-up Acks

• The algorithm is executed in rounds
Algorithm

• The algorithm is executed in rounds

Phase 4
Link Updates
• The algorithm is executed in rounds

Phase 5
State Updates
Analysis

- Work cost: $W(Display\text{Net}, T_0, \sigma) = \sum_{\sigma_i \in \sigma} w(\sigma_i)$

- Time cost:
  - Request time: $t(\sigma_i) = e_i - b_i$
  - Makespan: $T(T_0, \sigma) = \max_{\sigma_i \in \sigma} e_j - \min_{\sigma_i \in \sigma} b_j$

number of steps to fulfill all requests

rounds to fulfill all requests
Amortized Analysis: the average cost per request for a given sequence of communication requests

Potential Method
- $size(u)$: number of nodes in u’s subtree
- $rank(u)$: $\log_2 size(u)$
Progress Matrix

- \( \sigma = (\sigma_1(s_1, d_1), \sigma_2(s_2, d_2), \sigma_3(s_3, d_3)) \)
- rounds: 1, 2, 3, ... , t, t+1, ..., t''

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>...</th>
<th>t</th>
<th>t+1</th>
<th>t+2</th>
<th>t+3</th>
<th>t+4</th>
<th>t+5</th>
<th>t+6</th>
<th>...</th>
<th>t'</th>
<th>...</th>
<th>t''</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1 )</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>...</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>...</td>
<td>-</td>
<td>...</td>
<td>-</td>
</tr>
<tr>
<td>( d_1 )</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>...</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>...</td>
<td>-</td>
<td>...</td>
<td>-</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>...</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>...</td>
<td>-</td>
</tr>
<tr>
<td>( d_2 )</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>...</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>...</td>
<td>-</td>
</tr>
<tr>
<td>( s_3 )</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>...</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>...</td>
<td>✓</td>
<td>...</td>
<td>-</td>
</tr>
<tr>
<td>( d_3 )</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>...</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>...</td>
<td>✓</td>
<td>...</td>
<td>-</td>
</tr>
</tbody>
</table>

• progress
• pause
• inactive
Progress Matrix

- $\sigma = (\sigma_1(s_1, d_1), \sigma_2(s_2, d_2), \sigma_3(s_3, d_3))$
- rounds: 1, 2, 3, ..., $t$, $t+1$, ..., $t''$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>...</th>
<th>$t$</th>
<th>$t+1$</th>
<th>$t+2$</th>
<th>$t+3$</th>
<th>$t+4$</th>
<th>$t+5$</th>
<th>$t+6$</th>
<th>...</th>
<th>$t'$</th>
<th>...</th>
<th>$t''$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>...</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>...</td>
<td>-</td>
<td>...</td>
<td>-</td>
</tr>
<tr>
<td>$d_1$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>...</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>...</td>
<td>-</td>
<td>...</td>
<td>-</td>
</tr>
<tr>
<td>$s_2$</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>...</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>...</td>
<td>-</td>
<td>...</td>
<td>-</td>
</tr>
<tr>
<td>$d_2$</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>...</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>...</td>
<td>-</td>
<td>...</td>
<td>-</td>
</tr>
<tr>
<td>$s_3$</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>...</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>...</td>
</tr>
<tr>
<td>$d_3$</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>...</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>...</td>
</tr>
</tbody>
</table>

$O(|\log n|)$ $O(|\log n|)$ ... $O(m)$ times
• Amortized Analysis
  • Work:
    • For $\sigma_i \in \sigma$, $C_A = O(m \log n)$
    • The total work cost to fulfill $\sigma$ is $O(m(m + n) \log n)$
  
• Makespan:
  • The makespan of $\sigma$ is $O(m(m + n) \log n)$
Simulations

• Setup:
  • Dataset DS1 (i.i.d. over ProjecToR)$^1$:
    • $n = 128$ nodes
    • $m = 10,000$
    • 8,367 communication pairs
    • 2 production clusters: MapReduce-type jobs, index builders, and database and storage systems

• Setup:
  • Dataset DS2 (Facebook)$^2$:  
    • n = 159 nodes  
    • m = 48.485.220  
    • per-packet sampling: uniformly distributed with rate 1:30.000  
    • 24-hour time window

Simulations

- Locality of reference
  - DS1: high spatial locality
  - DS2: high temporal locality

Spatial locality

- $(s_1, d_1)$
- $(s_2, d_2)$
- $(s_1, d_1)$
- $(s_3, d_3)$
- $(s_4, d_4)$
- $(s_1, d_1)$
- $(s_2, d_2)$
- $(s_1, d_1)$

Temporal locality

- $(s_1, d_1)$
- $(s_1, d_1)$
- $(s_1, d_1)$
- $(s_1, d_1)$
- $(s_2, d_2)$
- $(s_2, d_2)$
- $(s_3, d_3)$
- $(s_4, d_4)$
Simulations

• Baseline:
  • Statically optimum algorithm
    • Dynamic program
    • Demand-aware Static Binary Search Tree
    • Optimized towards the request frequency distribution

• SplayNet:
  • Sequential self-adjusting network
Work: A Price of Decentralization?

INFOCOM 2019  Distributed Self-Adjusting Tree Networks
Work: A Price of Decentralization?

- **Work x 10^3**
  - SplayNet
  - DiSplayNet
  - StaticOPT

**DS1**

- High spatial but no temporal locality: computes the best topology for the request sequence.
- High temporal but low spatial locality: dynamic network reconfiguration is able to optimize the network topology over time.
Makespan: benefits of concurrency

In DS2, DiSplayNet leverages concurrency in combination with temporal locality.
Conclusion and Future directions

• We understand our work as a first step

• Lower bounds for our algorithm and the problem in general

• Integration and use of self-adjusting links with links that are not self-adjusting
Thank you

Bruna Peres
bperes@dcc.ufmg.br