# Distributed Self-Adjusting Tree Networks

Bruna Peres
Otavio Souza
Olga Goussevskaia

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Chen Avin

Ben-Gurion University of the Negev

Stefan Schmid

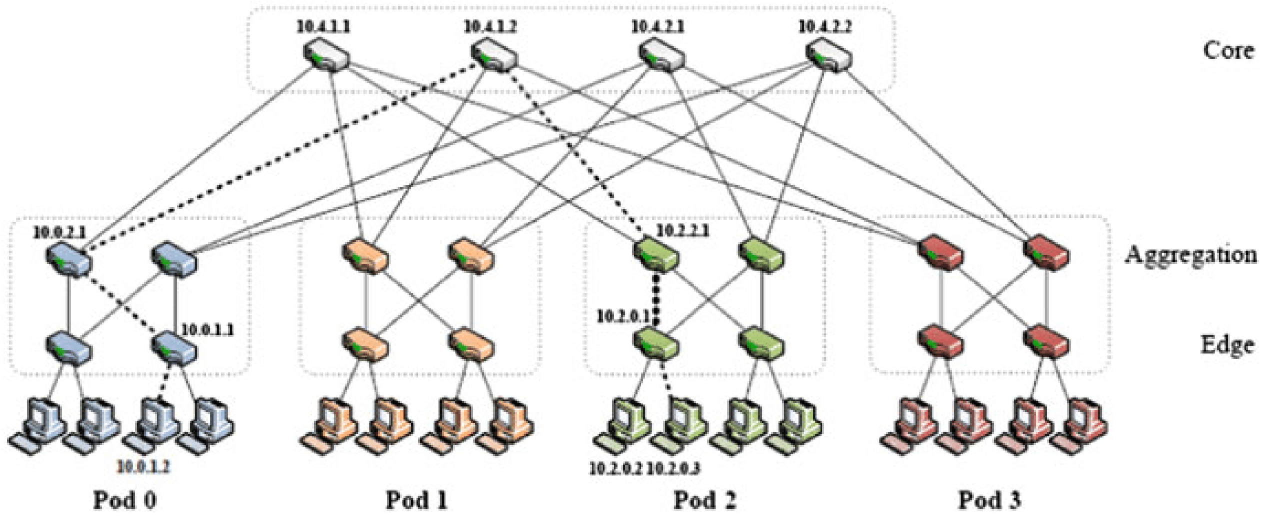
universität
wien

# Traditional Networks

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

Demand-Oblivious

Fixed



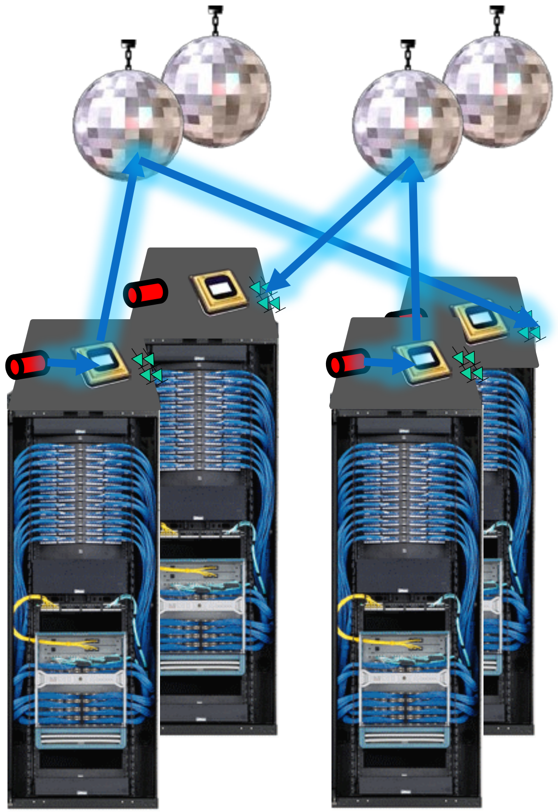
# Technology Enables Dynamic Reconfigurable Networks

# Dynamic Reconfigurable Networks

- Dynamically optimized toward the (time-varying) demand

Demand-Aware

Reconfigurable



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

# Motivation for Reconfigurability

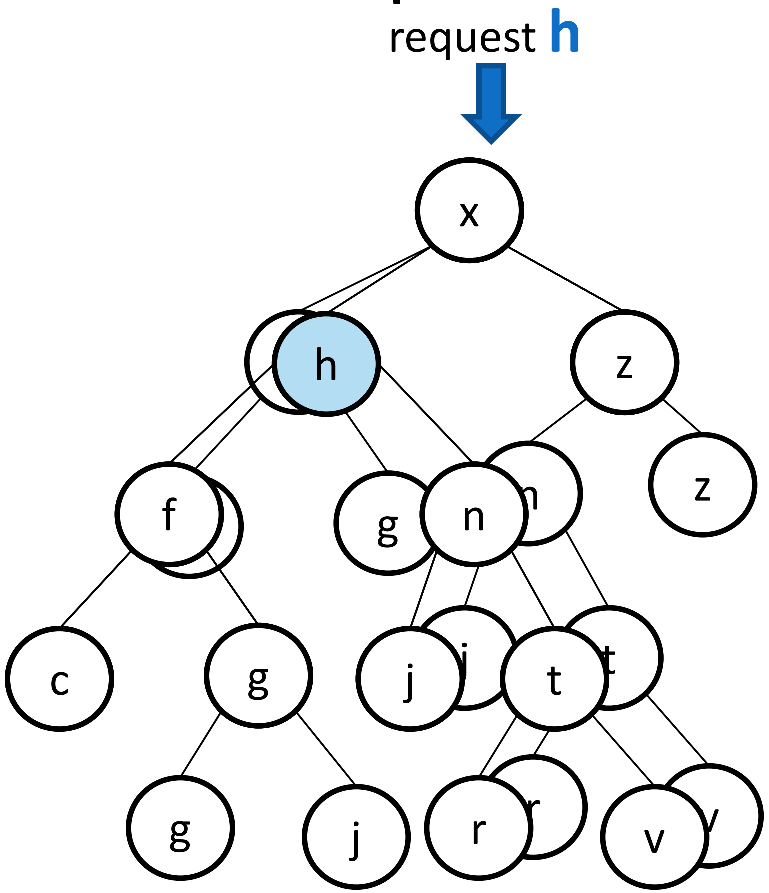
- Sparsity of communication matrix
- The difficulty of estimating traffic matrices ahead of time and predicting the future demand

Dynamic self-adjusting networks can adjust to and leverage these patterns!

# Self-Adjusting Data Structures

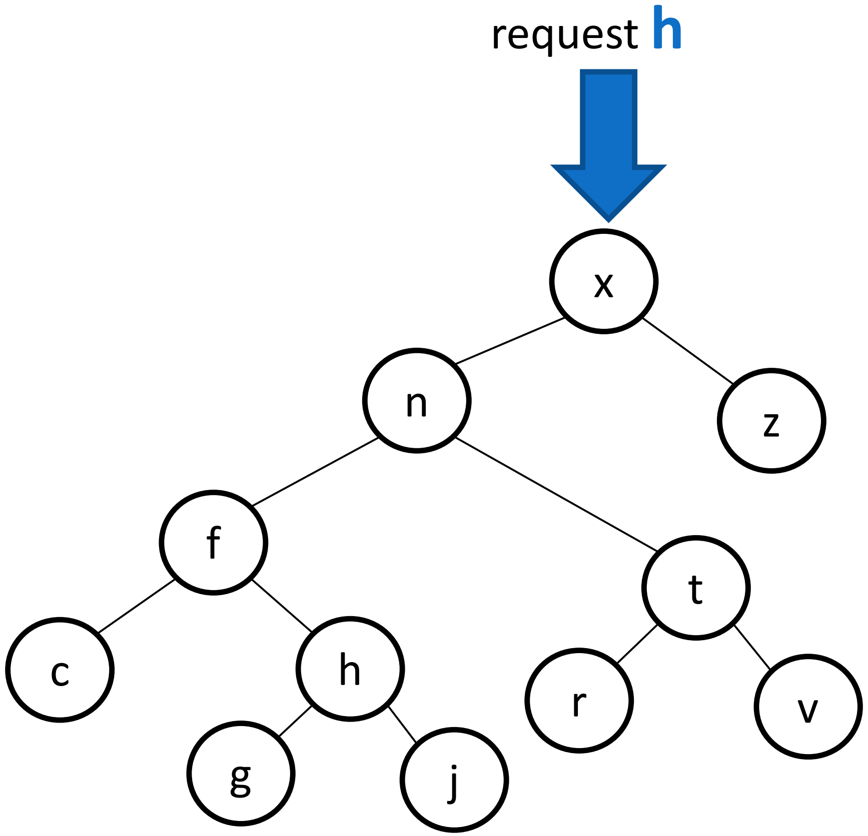
- The vision of self-adjusting networks is similar in spirit to the vision of self-adjusting datastructures

- Splay Trees

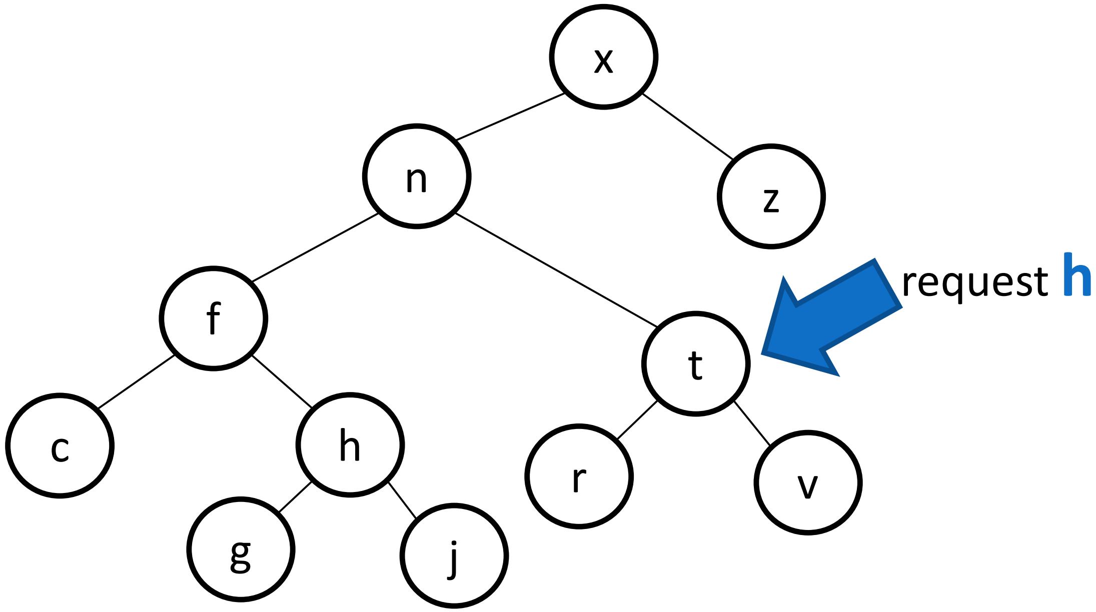



D. Sleator and R. Tarjan, Self-adjusting binary search trees. Journal of the ACM (JACM), vol. 32, no. 3, pp. 652–686, 1985.

# Self-Adjusting Networks




Datastructure




Network

# 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

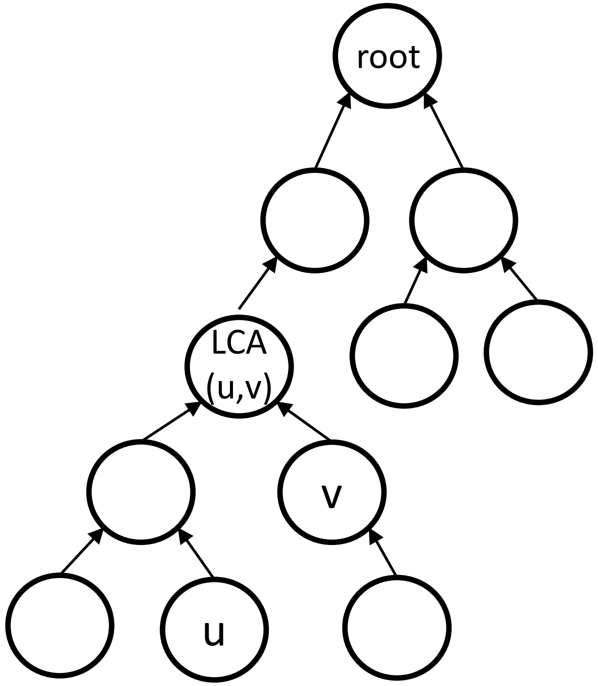
S. Schmid, et al., SplayNet: Towards Locally Self-Adjusting Networks. *IEEE/ACM Transactions on Networking*, 2016.

# 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!




S. Schmid, et al., SplayNet: Towards Locally Self-Adjusting Networks. *IEEE/ACM Transactions on Networking*, 2016.

# 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

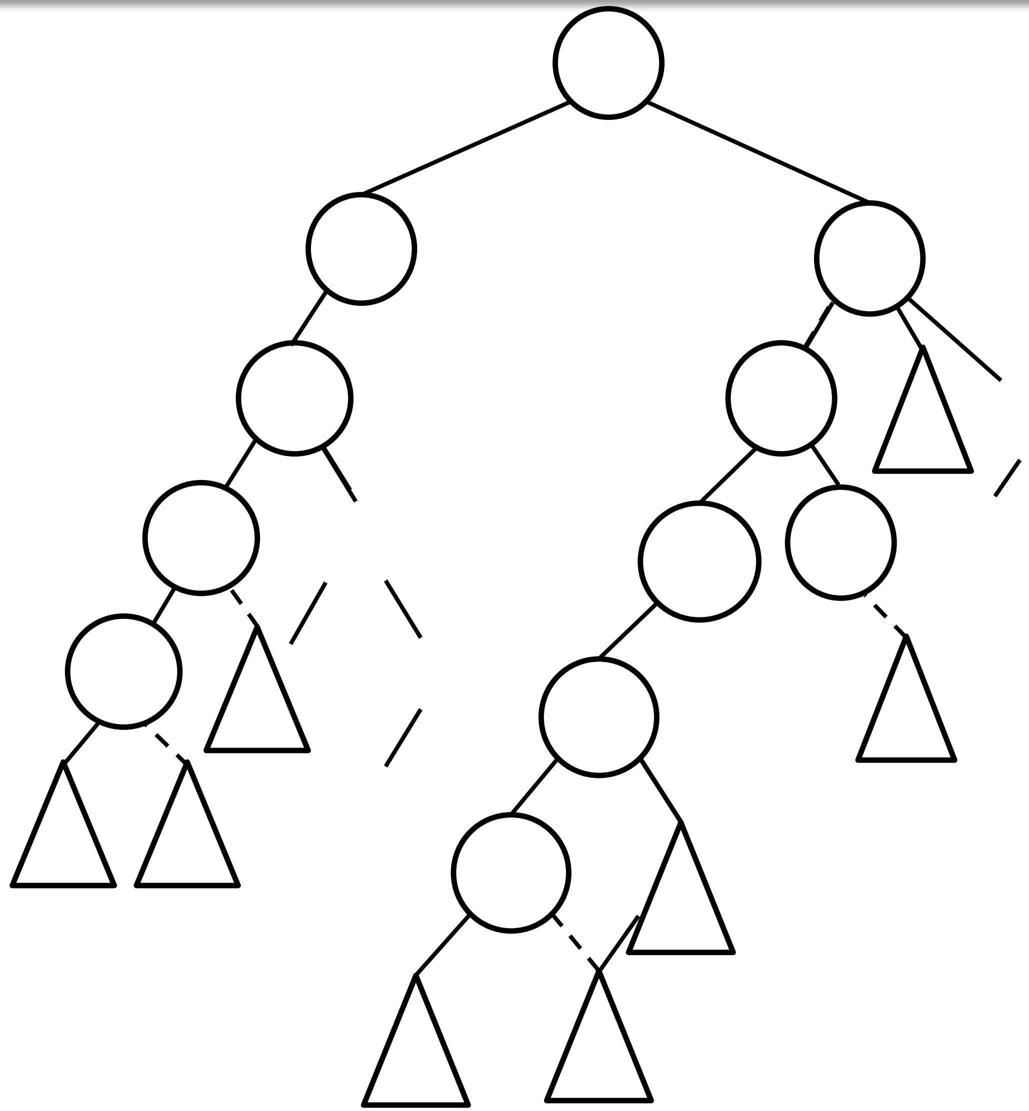
# Model

- 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 $\boldsymbol{b_i}$ and ends at $\boldsymbol{e_i}$

- 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

# DiSplayNets

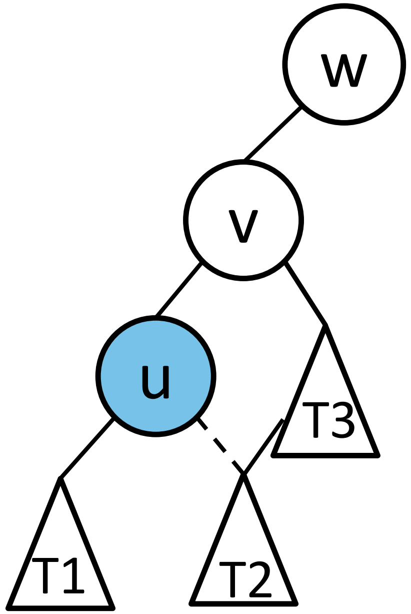


- Local Reconfigurations: decentralized and concurrent topological adjustments
- Independent clustering: nodes in a cluster updated their links in parallel without interference
- Prioritization: for nodes to achieve a consensus

# Local reconfigurations

- Step: $step_t\ (u)$

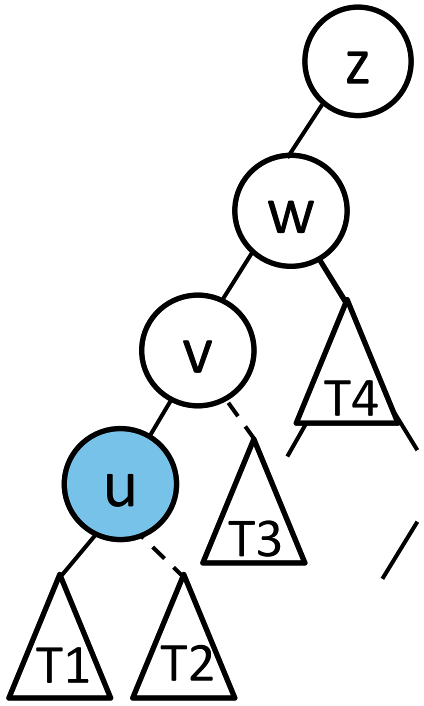
zig



# Local reconfigurations

- Step: $step_t\ (u)$

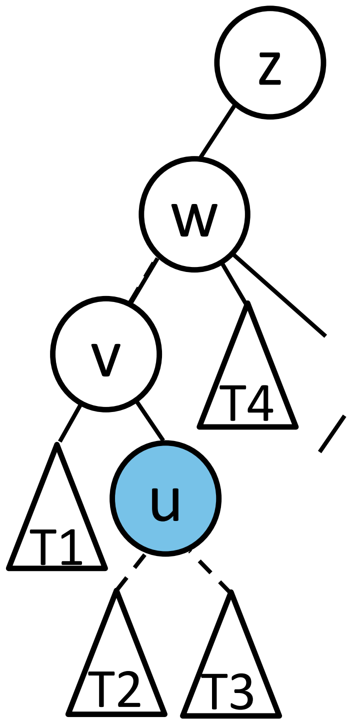
zig-zig



# Local reconfigurations

- Step: $step_t\ (u)$

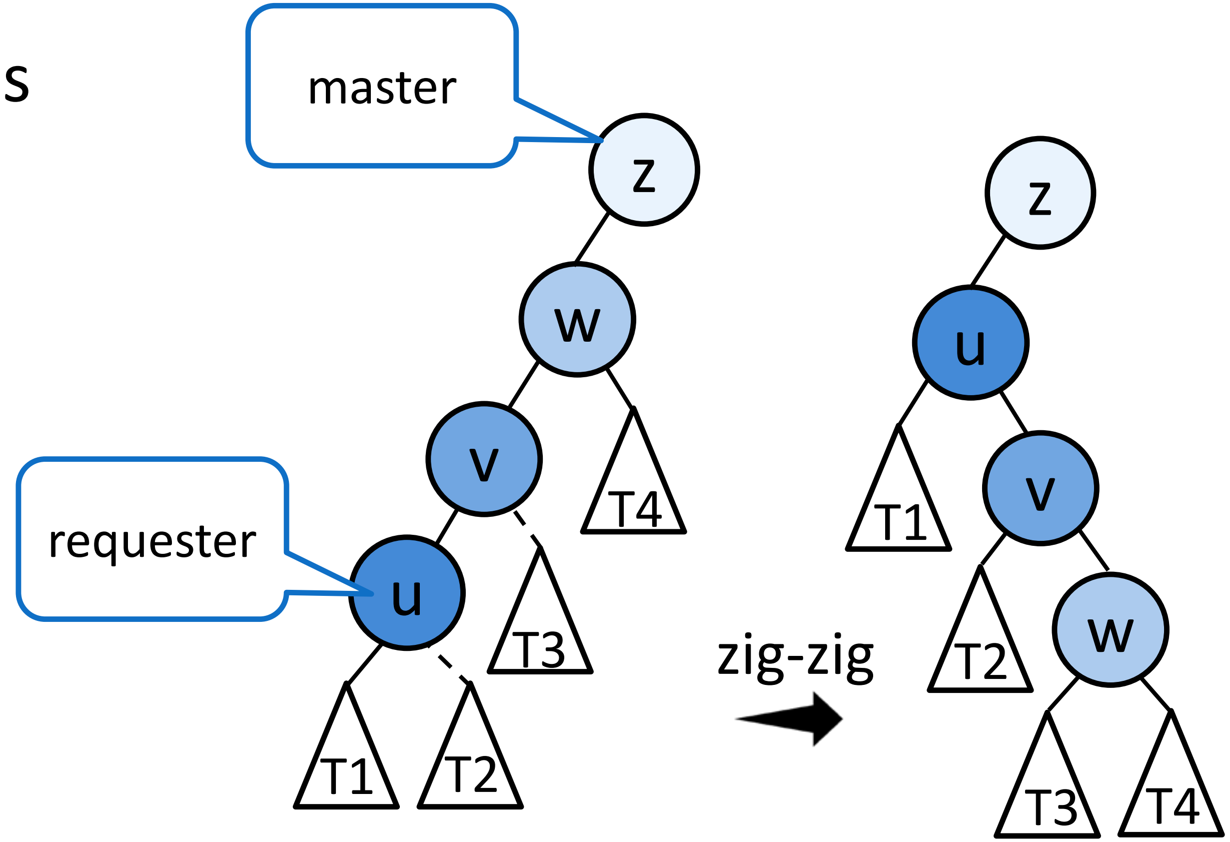
zig-zag



# Independent clustering

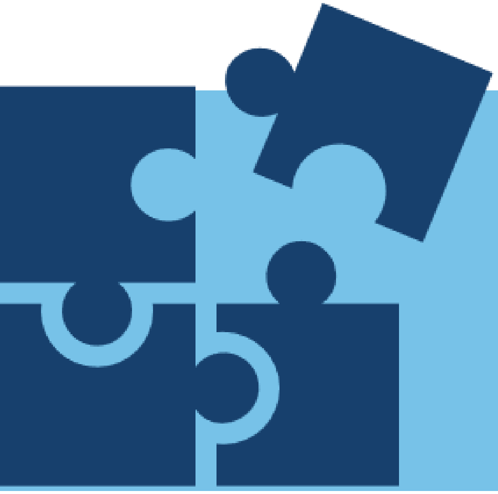
- Cluster: $C_t(u)$
  - *Requester* and *master* nodes

$C_t(u)$



# Prioritization

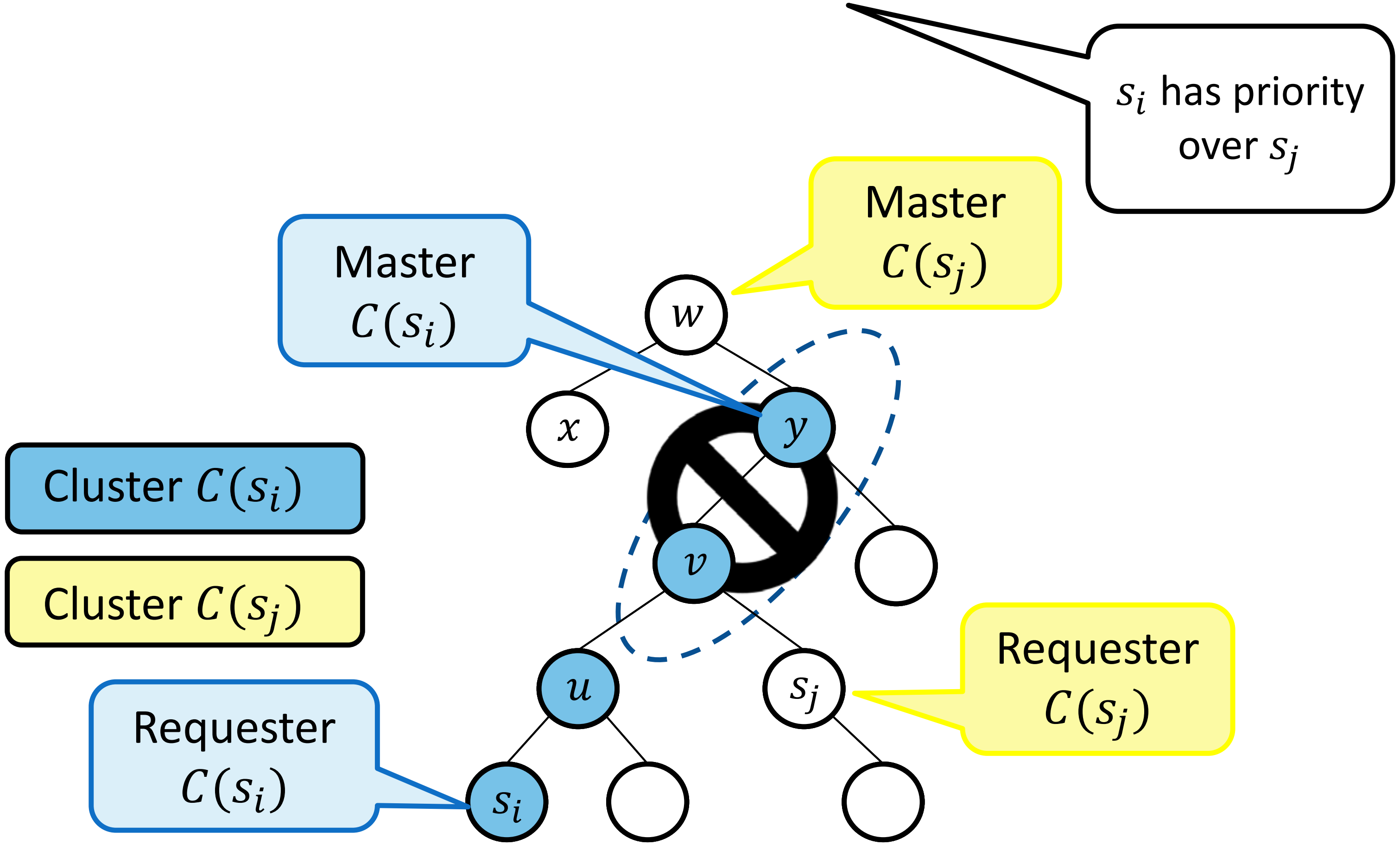
- In order to ensure deadlock and starvation freedom, concurrent operations are performed according to a priority
- Given two requests $\sigma_i(s_i, d_i)$ and $\sigma_j(s_j, d_j)$, such that $b_i < b_j$, $\sigma_i$ has priority over $\sigma_j$



Nodes form an **independent cluster** to perform a **local reconfiguration** given the **priority** of the request

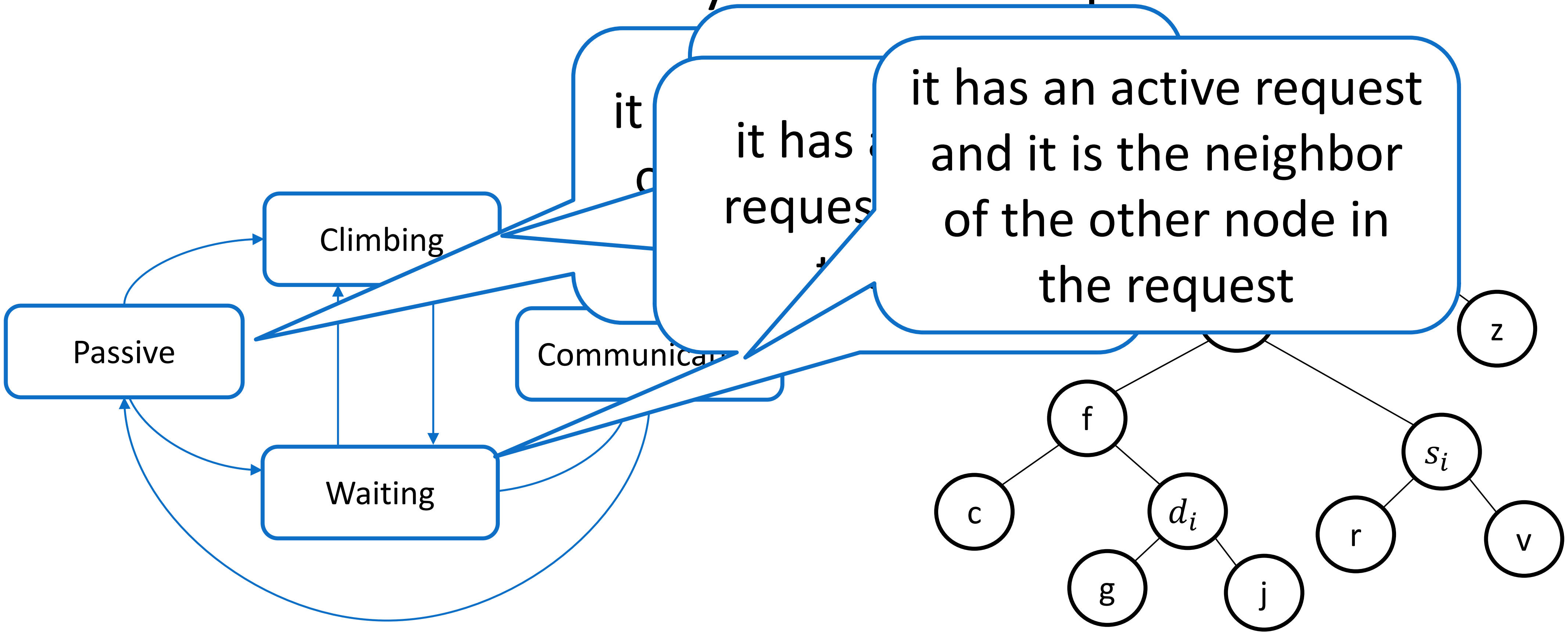
# DiSplayNet

- Given $\sigma_i(s_i, d_i)$ and $\sigma_j(s_j, d_j)$, such that $b_i < b_j$:



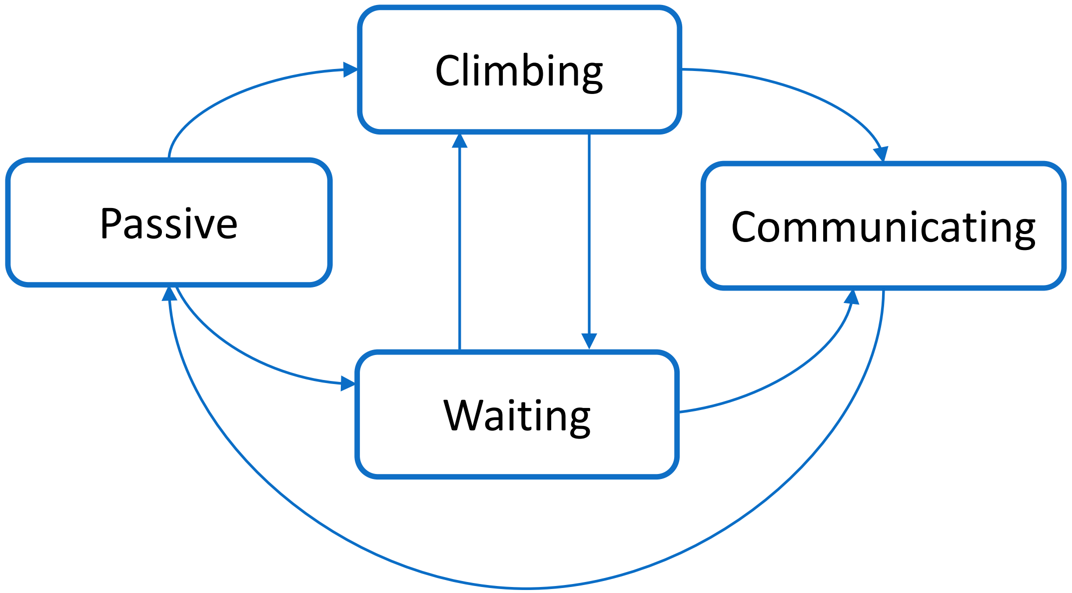
# DiSplayNet

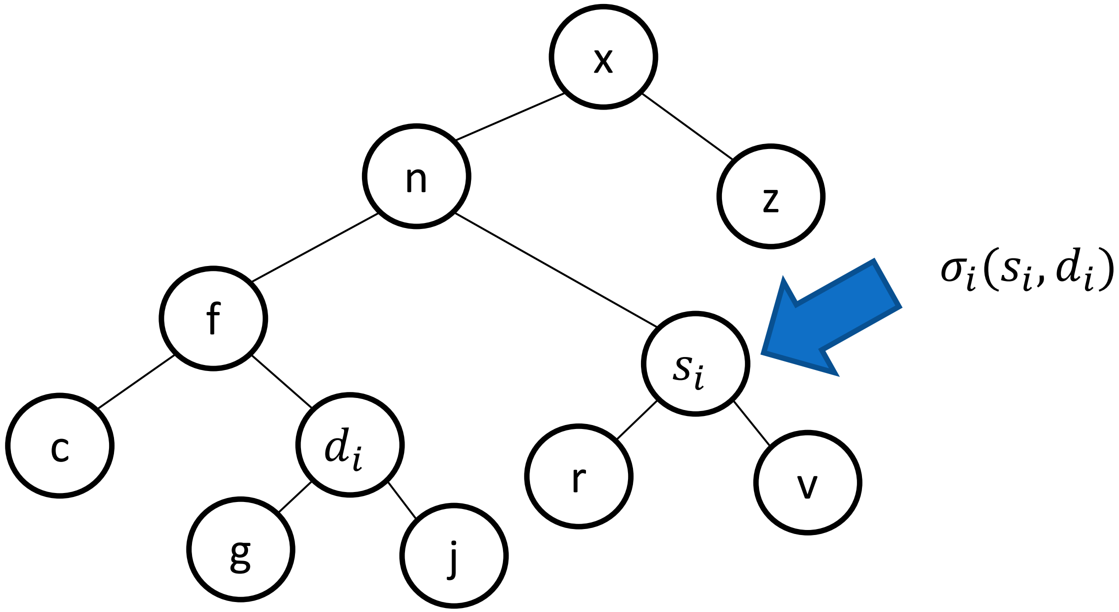
- State machine executed by each node in parallel



# DiSplayNet

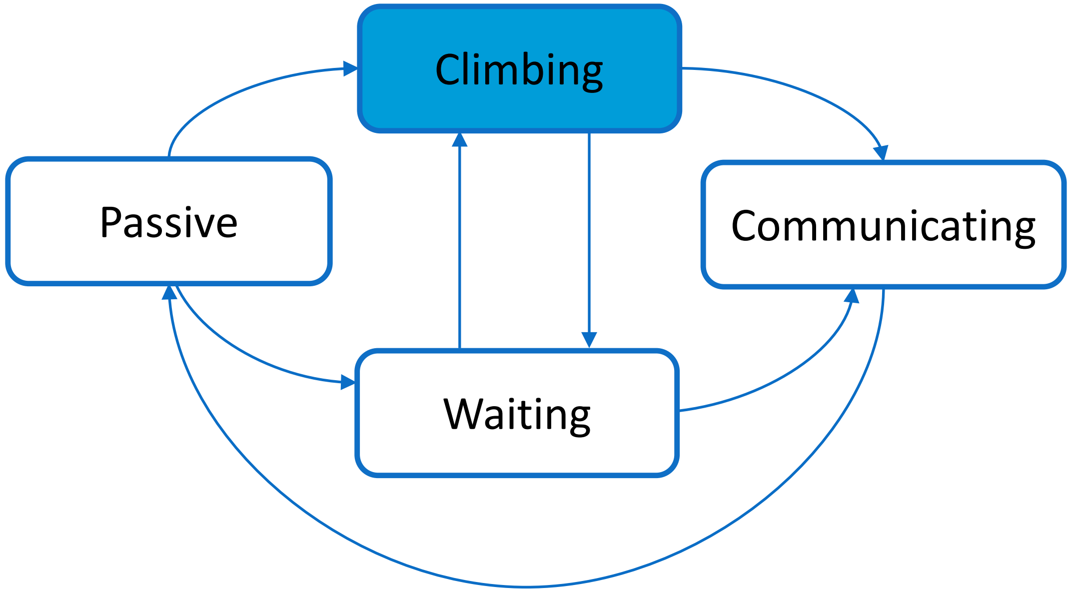
- State machine executed by each node in parallel

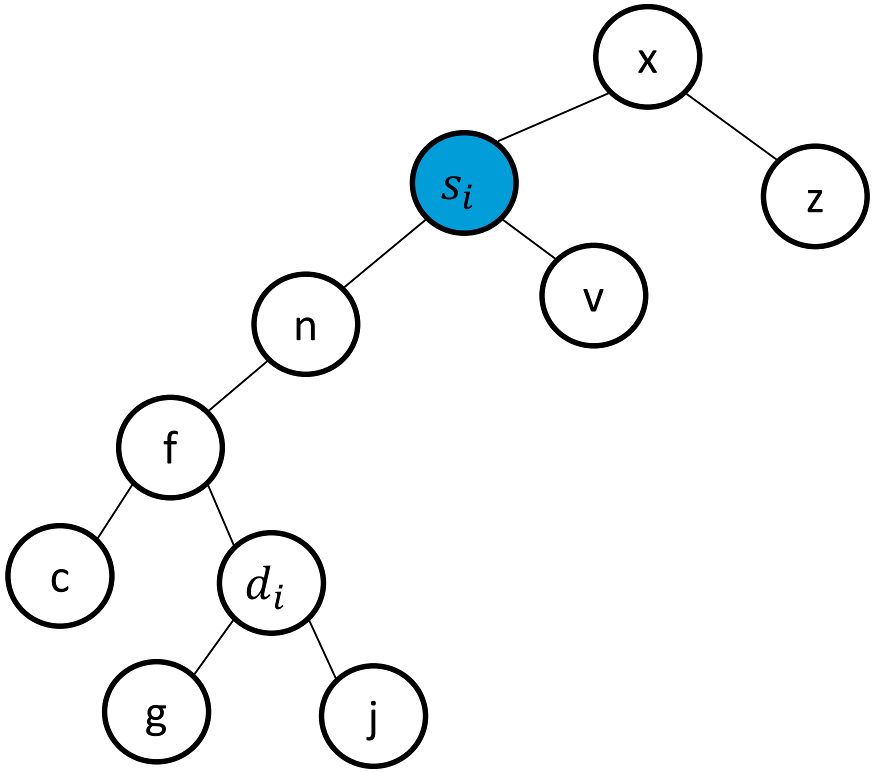


# DiSplayNet

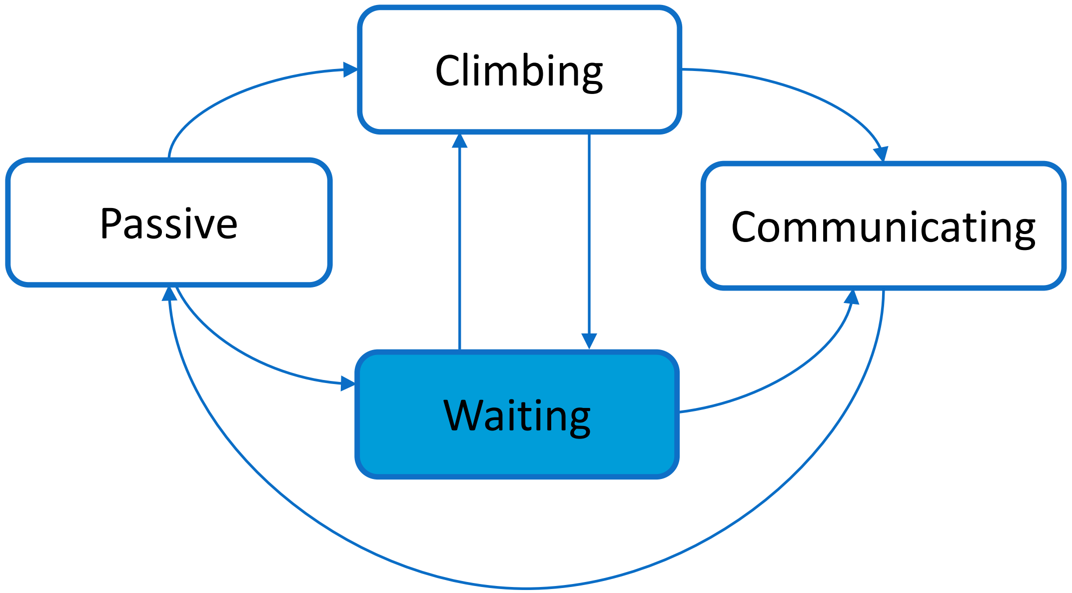
- State machine executed by each node in parallel

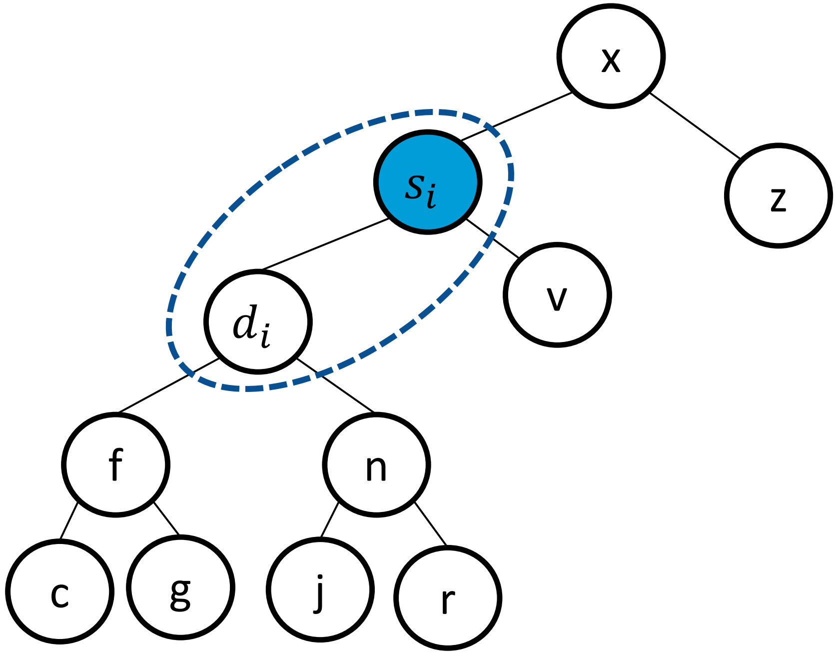


# DiSplayNet

- State machine executed by each node in parallel



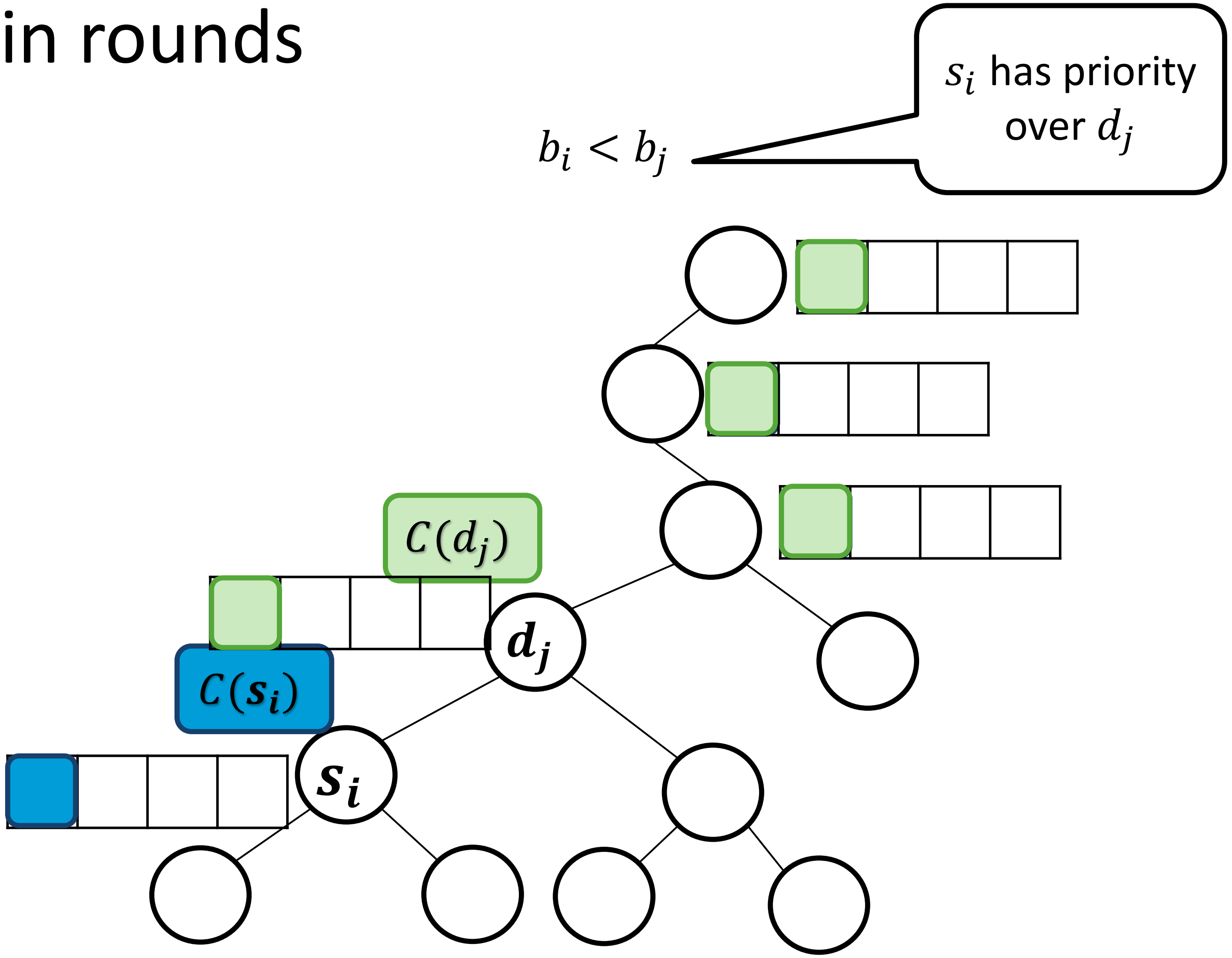
# Algorithm

- 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

# Algorithm

- The algorithm is executed in rounds

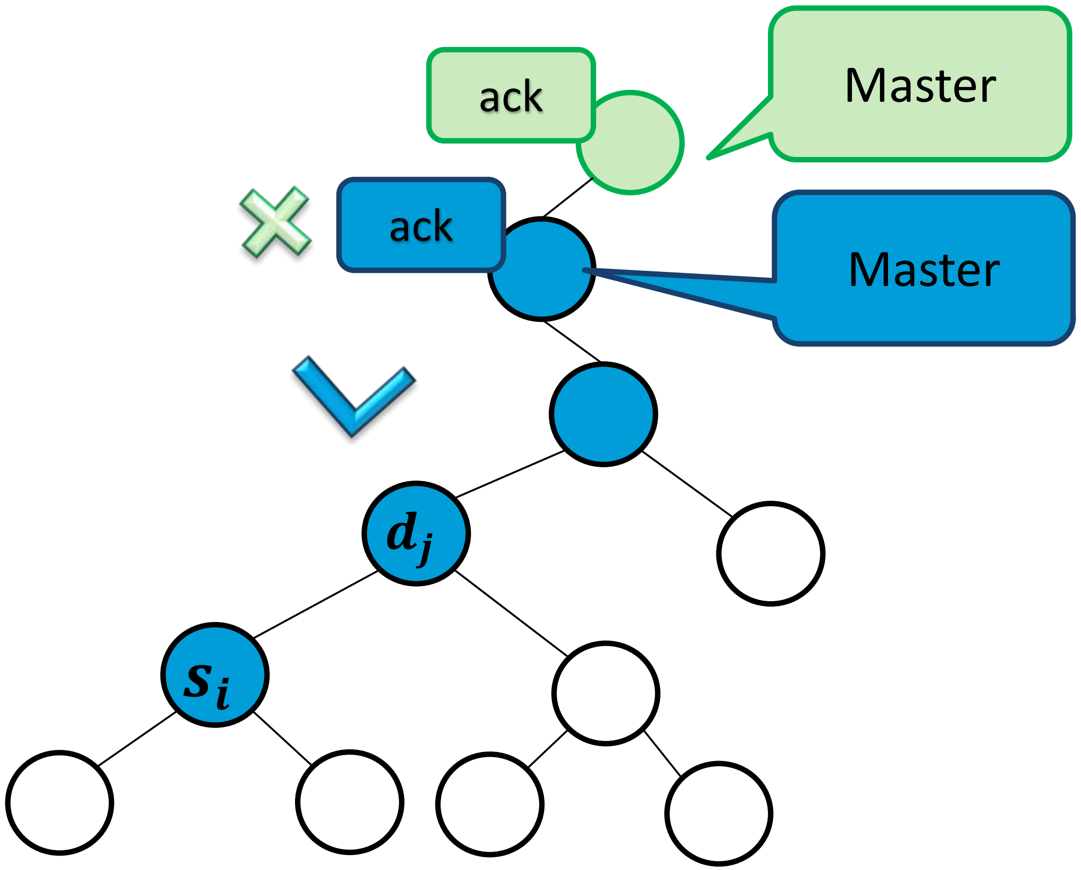
Phase 1
Cluster Requests



# Algorithm

- The algorithm is executed in rounds

Phase 2
Top-down Acks

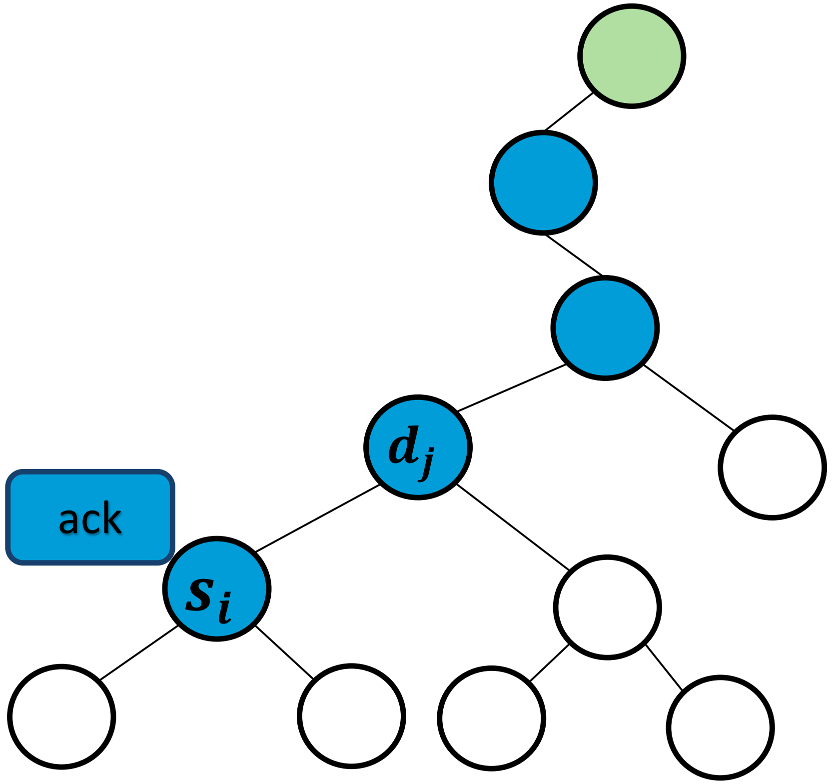


# Algorithm

- The algorithm is executed in rounds

Phase 3

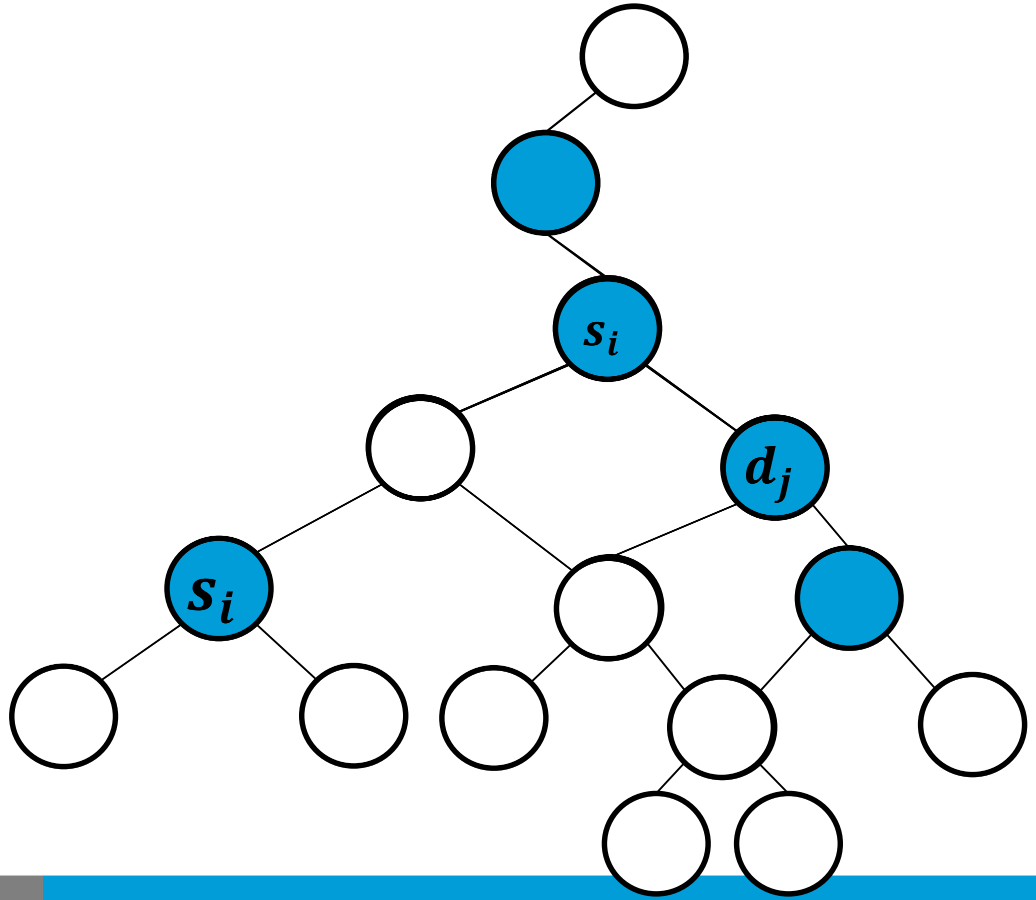
Bottom-up Acks



# Algorithm

- The algorithm is executed in rounds

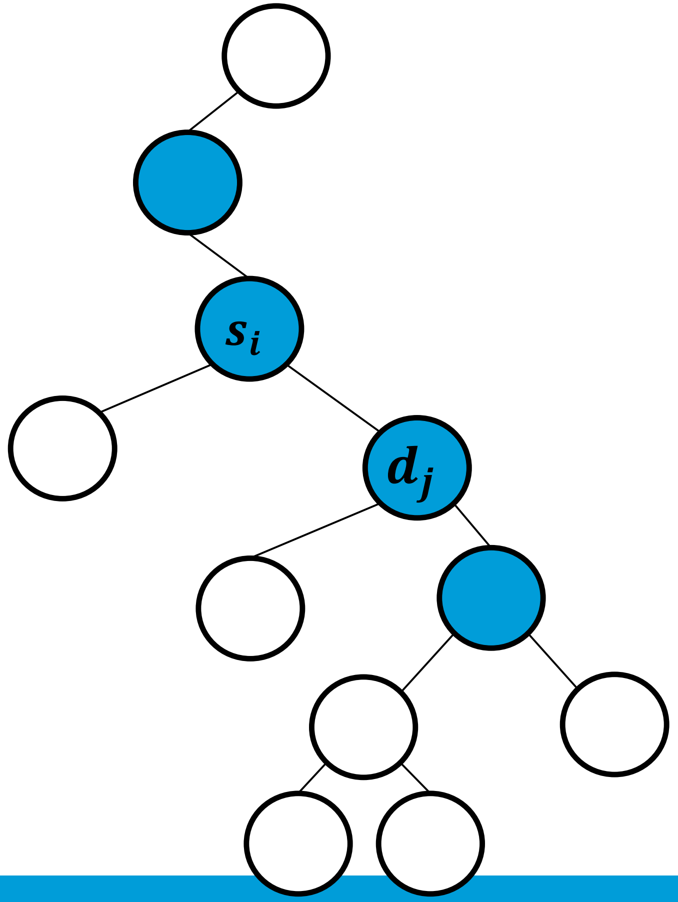
Phase 4
Link Updates



# Algorithm

- The algorithm is executed in rounds

Phase 5
State Updates



# Analysis

- Work cost: $W(DiSplayNet, T_0, \sigma) = \sum_{\sigma_i \,\epsilon\, \sigma} w(\sigma_i)$

number of steps to fulfill all requests

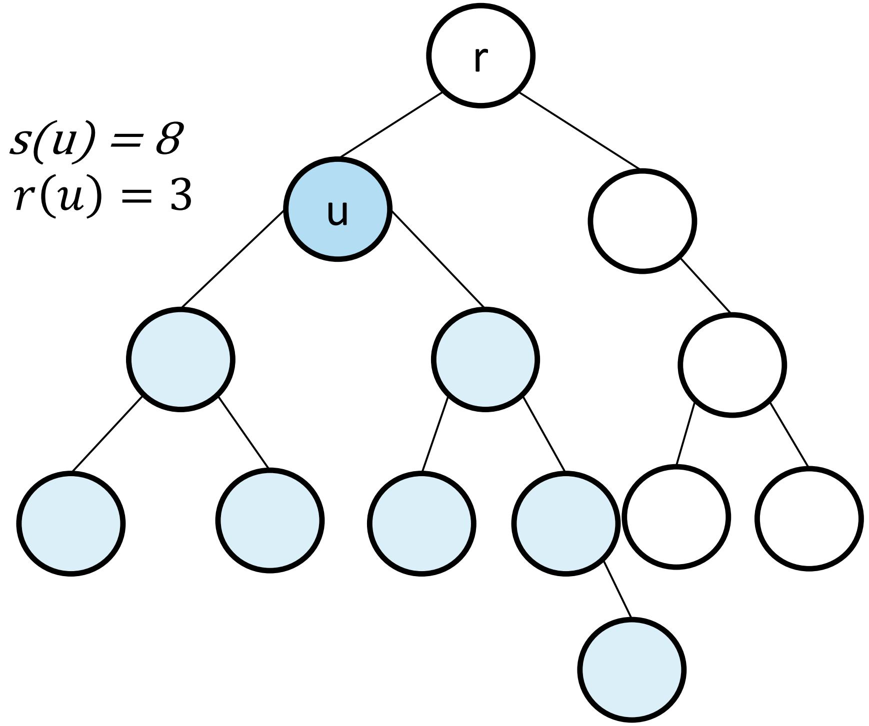
- Time cost:
  - Request time: $t(\sigma_i) = e_i - b_i$
  - Makespan: $T(T_0, \sigma) = \max_{\sigma_i \,\epsilon\, \sigma} e_j - \min_{\sigma_i \,\epsilon\, \sigma} b_j$

rounds to fulfill all requests

# Analysis

- 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

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

✓ progress
× pause
- inactive

| | 1 | 2 | 3 | … | t | t+1 | t+2 | t+3 | t+4 | t+5 | t+6 | … | t’ | … | t’’ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $s_1$ | ✓ | ✓ | ✓ | … | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | - | … | - | … | - |
| $d_1$ | ✓ | ✓ | ✓ | … | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | - | … | - | … | - |
| $s_2$ | - | ✓ | ✓ | … | ✓ | ✓ | ✓ | ✓ | - | - | - | … | - | … | - |
| $d_2$ | - | ✓ | ✓ | … | ✓ | ✓ | × | ✓ | - | - | - | … | - | … | - |
| $s_3$ | - | - | ✓ | … | × | × | × | × | ✓ | × | ✓ | … | ✓ | … | - |
| $d_3$ | - | - | ✓ | … | × | × | × | × | × | × | ✓ | … | ✓ | … | - |

# Progress Matrix

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

✓ progress
× pause
- inactive

| | 1 | 2 | 3 | … | t | t+1 | t+2 | t+3 | t+4 | t+5 | t+6 | … | t’ | … | t’’ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $s_1$ | ✓ | ✓ | ✓ | … | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | - | … | - | … | - |
| $d_1$ | ✓ | ✓ | ✓ | … | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | - | … | - | … | - |
| $s_2$ | - | ✓ | ✓ | … | ✓ | ✓ | ✓ | ✓ | - | - | - | … | - | … | - |
| $d_2$ | - | ✓ | ✓ | … | ✓ | ✓ | × | ✓ | - | - | - | … | - | … | - |
| $s_3$ | - | - | ✓ | … | × | × | × | × | ✓ | × | ✓ | … | ✓ | … | - |
| $d_3$ | - | - | ✓ | … | × | × | × | × | × | × | ✓ | … | ✓ | … | - |

$O(\log n)$ $O(\log n)$ … $O(m) times$

# Analysis

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

1: M. Ghobadi, et al., “Projector: Agile reconfigurable data center interconnect,” in Proceedings of the 2016 ACM SIGCOMM, ser. SIGCOMM ’16. New York, NY, USA: ACM, 2016, pp. 216–229.

# Simulations

- 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

2: A. Roy, et al., “Inside the social network’s (datacenter) network,” in Proceedings of the 2015 ACM SIGCOMM, ser. SIGCOMM ’15. New York, NY, USA: ACM, 2015, pp. 123–137.

# 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

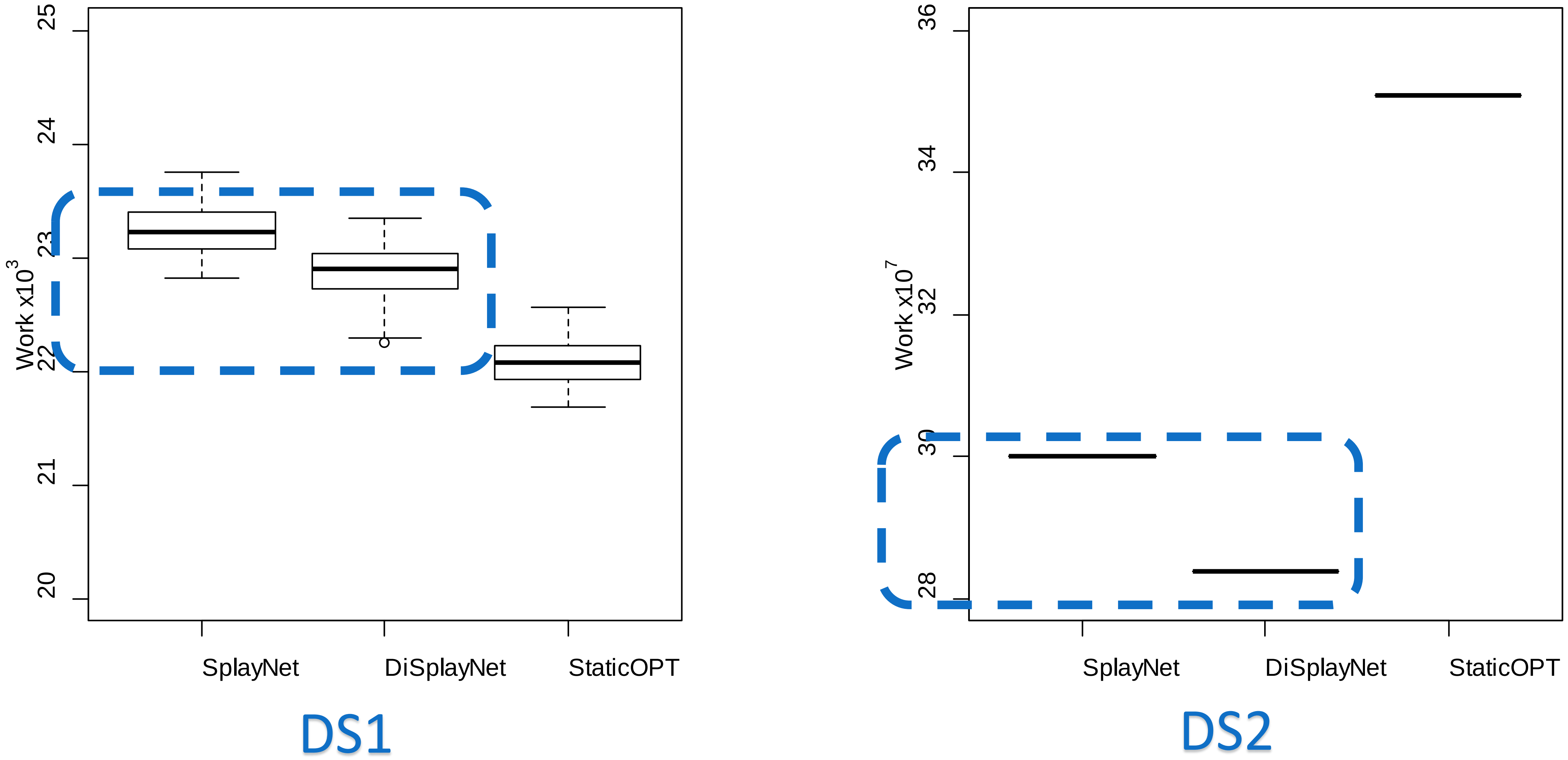
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $(s_1, d_1)$ | $(s_1, d_1)$ | $(s_1, d_1)$ | $(s_1, d_1)$ | $(s_2, d_2)$ | $(s_2, d_2)$ | $(s_3, d_3)$ | $(s_4, d_4)$ |

# Simulations

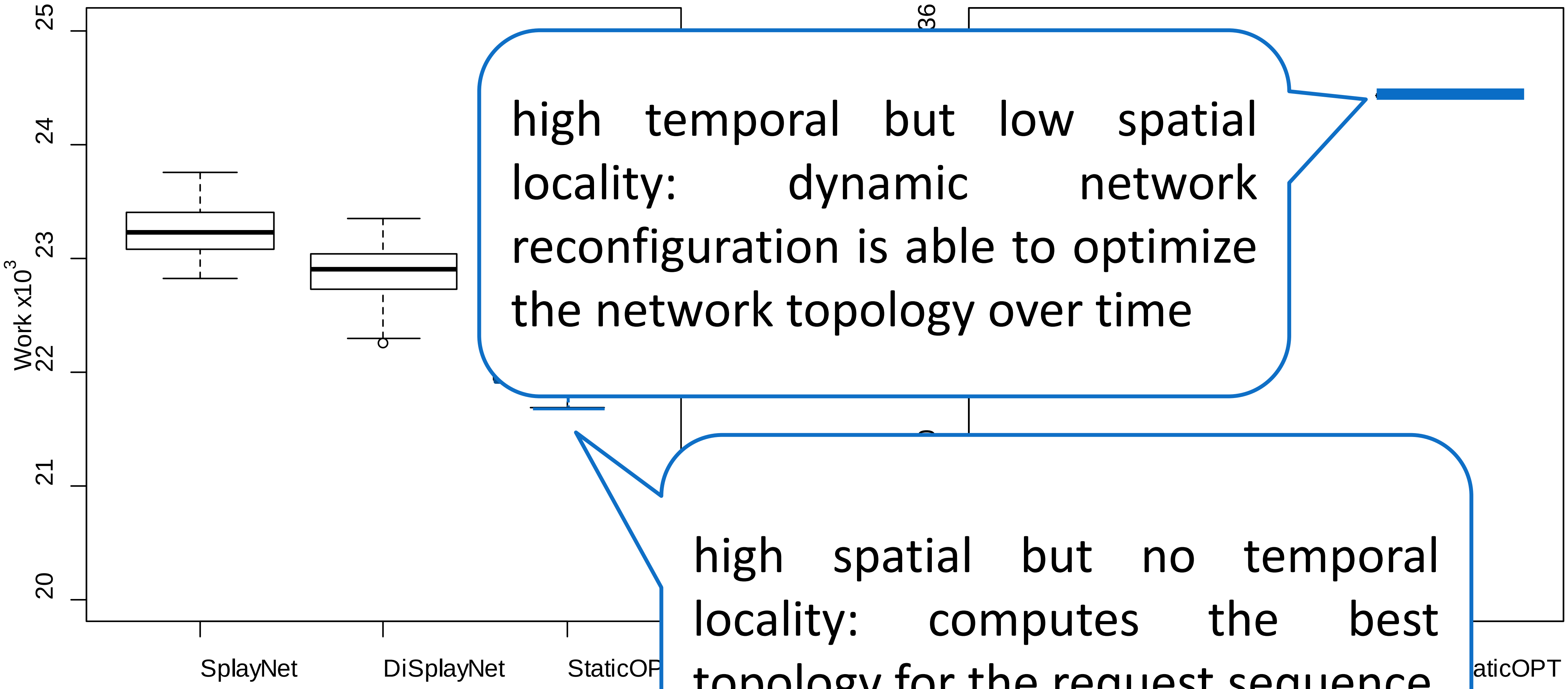
- Baseline:
  - Statically optimum algorithm
    - Dynamic program
    - Demand-aware Static Binary Search Tree
    - Optimized towards the request frequency distribution

  - SplayNet:
    - Sequential self-adjusting network

# Work: A Price of Decentralization?



# Work: A Price of Decentralization?

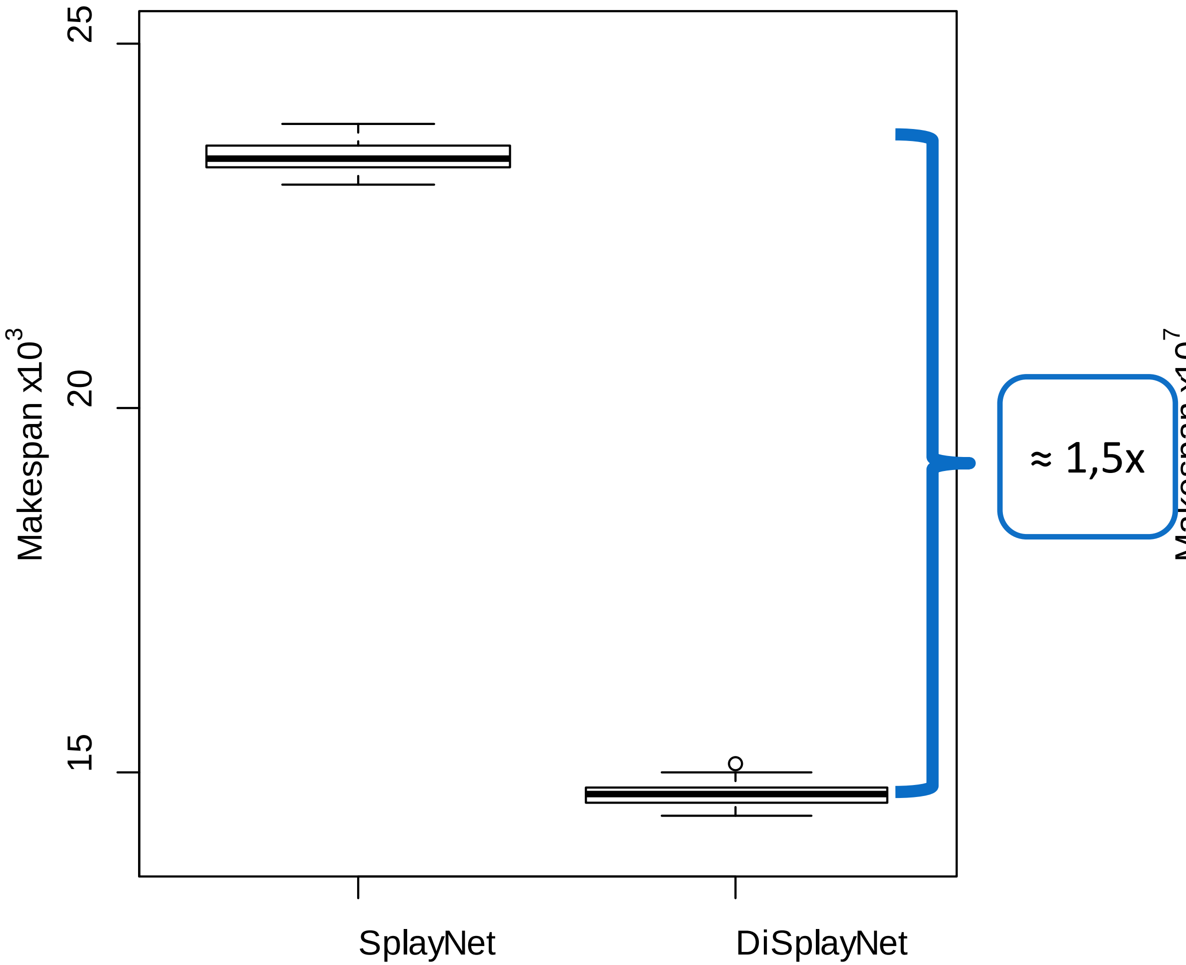



DS1

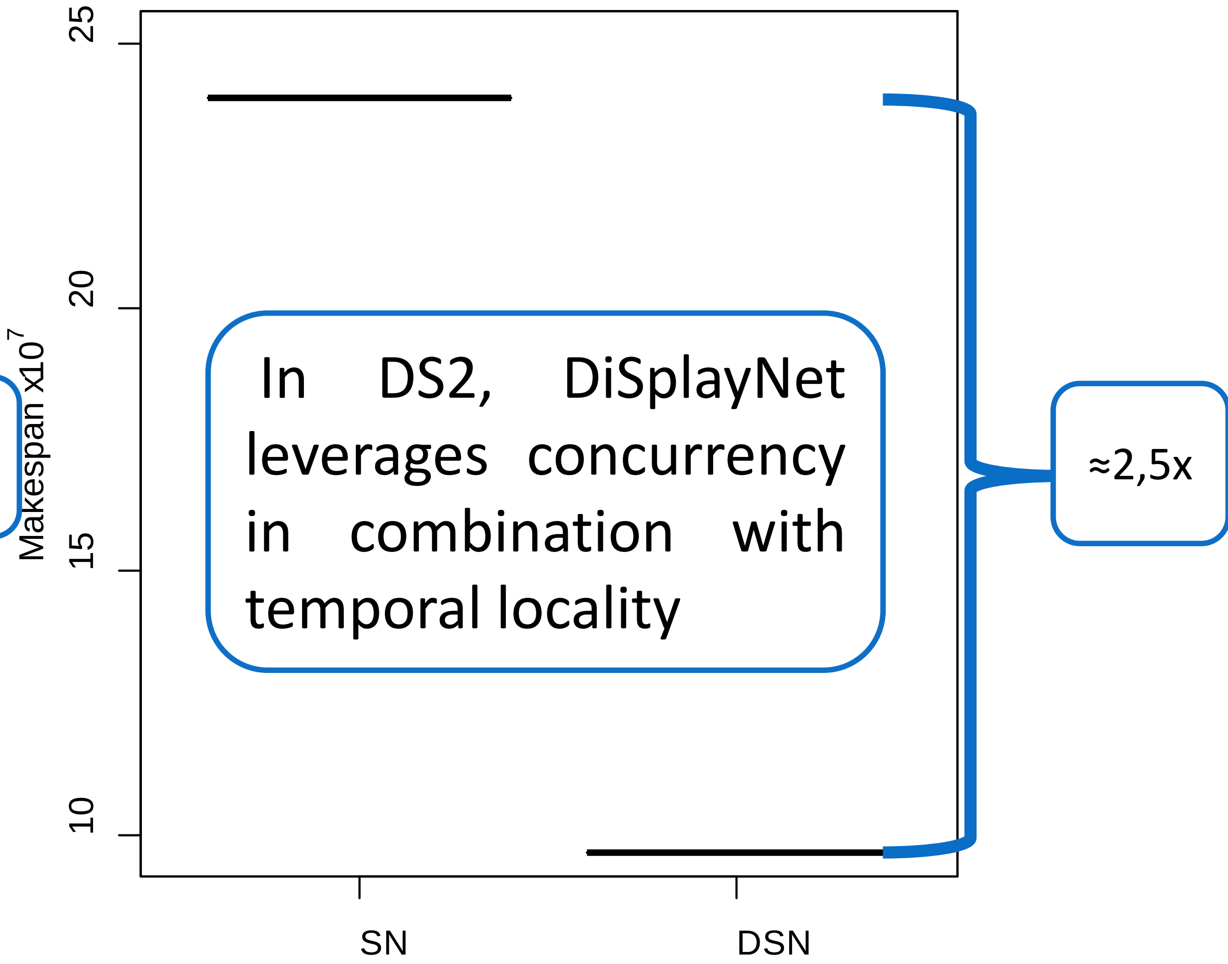
high temporal but low spatial locality: dynamic network reconfiguration is able to optimize the network topology over time

high spatial but no temporal locality: computes the best topology for the request sequence

# Makespan: benefits of concurrency




DS1




DS2

# 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