Highly Dynamic and Fully Distributed Data Structures

John Augustine [1] Antonio Cruciani [2] Iqra Altaf Gillani[3]

[1]IIT Madras, Chennai, India, [2]Gran Sasso Science Institute, L'Aquila, Italy, [3]National Institute of Technology Srinagar, Srinagar, India

Check out our paper!

Introduction

- Real-world networks are dynamic
- Fast evolution
- Nodes continuously join and leave the network (churn)
- The churn can be arbitrarily bad

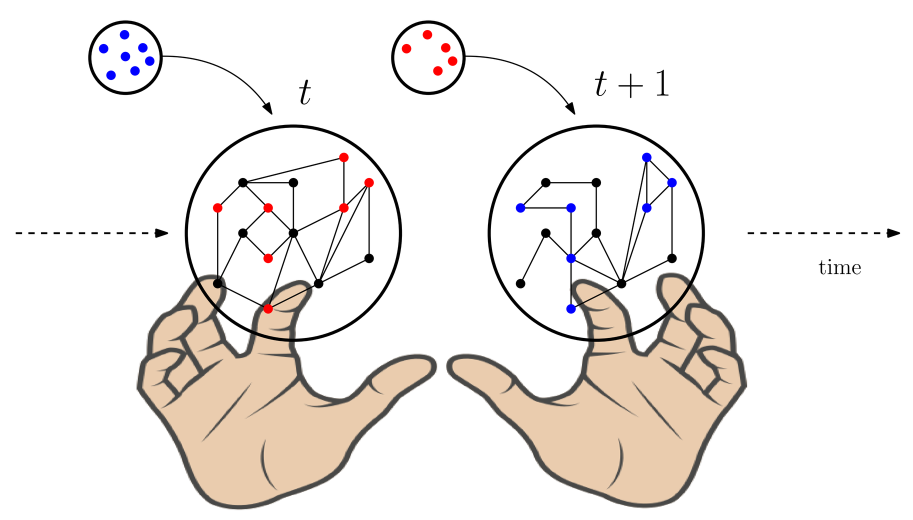


Figure 1. Example of a dynamic graph in which nodes are replaced at each round by an adversary.

- What about maintaining a dynamic distributed data structure?

Challenges

- Classic distributed protocols do not work!
- Powerful (**oblivous**) adversary
- High churn rate (almost linear)
- We need to be **very fast** in updating the data structure!
- We must spend as "little effort as possible" to maintain the data structure!

Some related work

- The Dynamic Network with churn model (survey by Augustine et al. [1]).
- Skip List-like data structures [4, 3, 2].

Problem Definition

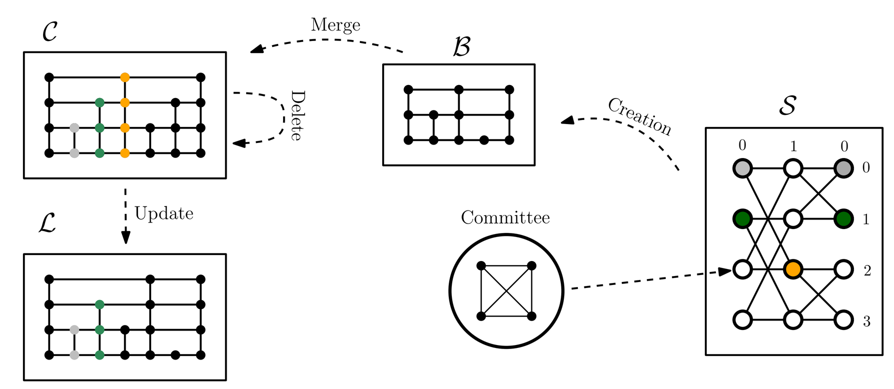
We want to maintain a distributed skip list despite an adversarial churn of $\mathcal{O}(n/\log n)$ nodes per rounds where n is the stable network's size. We must:

- Build and maintain a data structure of all the items/nodes in the network
- Ensure that the data structure can be queried at any time
- Update the data structure as fast a possible
- Be *dynamic resource competitive*, i.e., effort paid to maintain the data structure must be proportional to the overall experienced churn.

Centre for Cybersecurity, Trust and Reliability
Indian Institute of Technology Madras

Our Idea

Four networks approach: (1) A churn resilient overlay network $\mathcal{S}$; (2) A **live** queryable network $\mathcal{L}$; (3) A **buffer** network $\mathcal{B}$ of newly added elements; and (4) a **clean** network $\mathcal{C}$ with all the updates.



We propose an $\mathcal{O}(\log n)$ rounds continuous maintenance cycle that preserves the data distributed data structure despite high adversarial churn.

Dealing with removed nodes

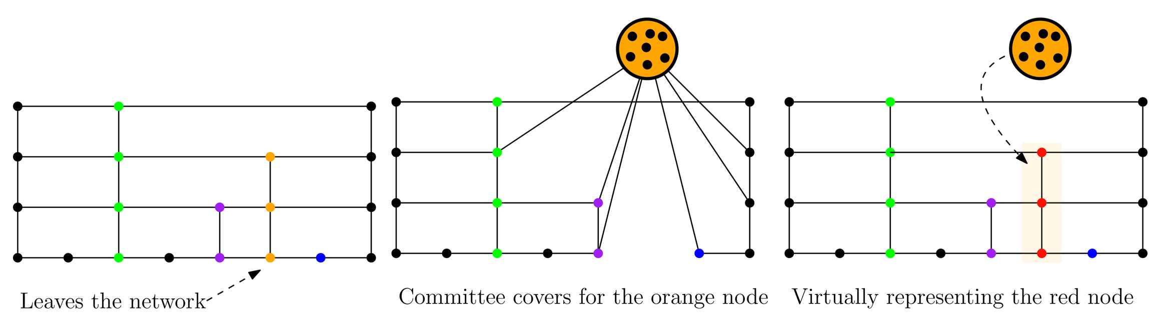


Figure 2. Nodes leave the network are temporarily replaced in $\mathcal{O}(1)$ rounds by committees in the overlay network $\mathcal{S}$.

Deleting virtual nodes in $\mathcal{O}(\log n)$ rounds

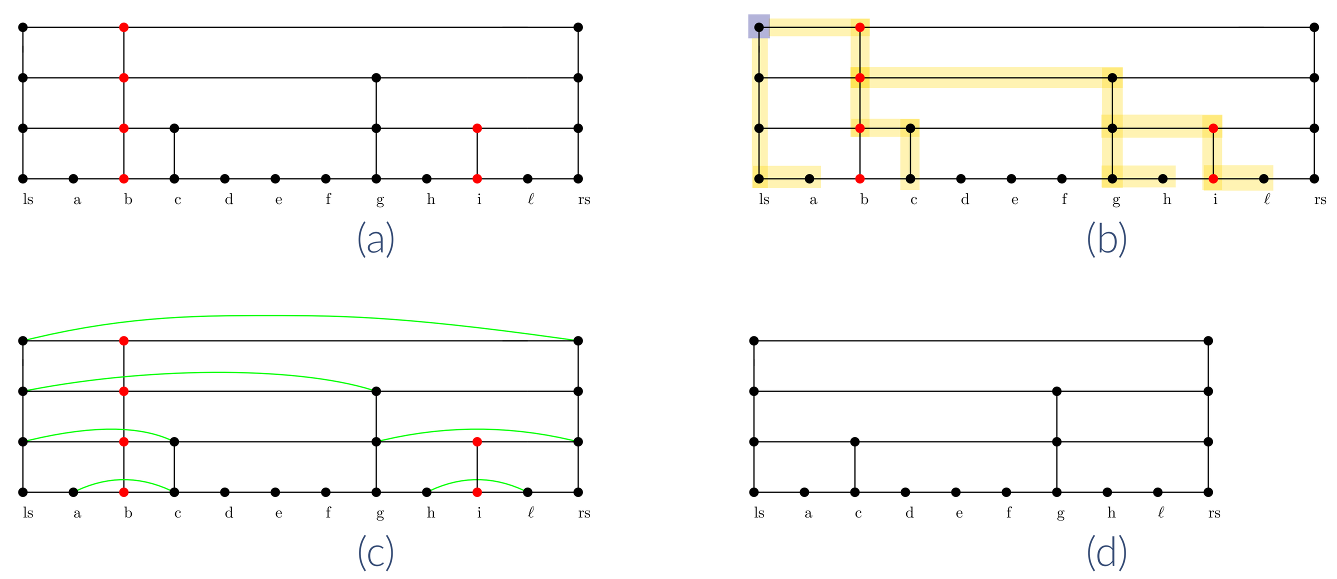


Figure 3. Illustration of our novel distributed and parallel skip list partition algorithm that removes a batch of nodes from a skip list in $\mathcal{O}(\log n)$ rounds with high probability.

Creating the Buffer Network in $\mathcal{O}(\log n)$ rounds

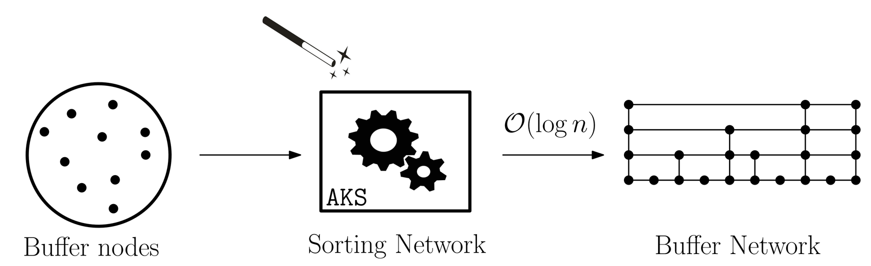


Figure 4. We build a temporary churn resilient sorting network on top of $\mathcal{S}$ and we build the Buffer network using the nodes that have joined the network until this time.

WAVE: An $\mathcal{O}(\log n)$ merge procedure

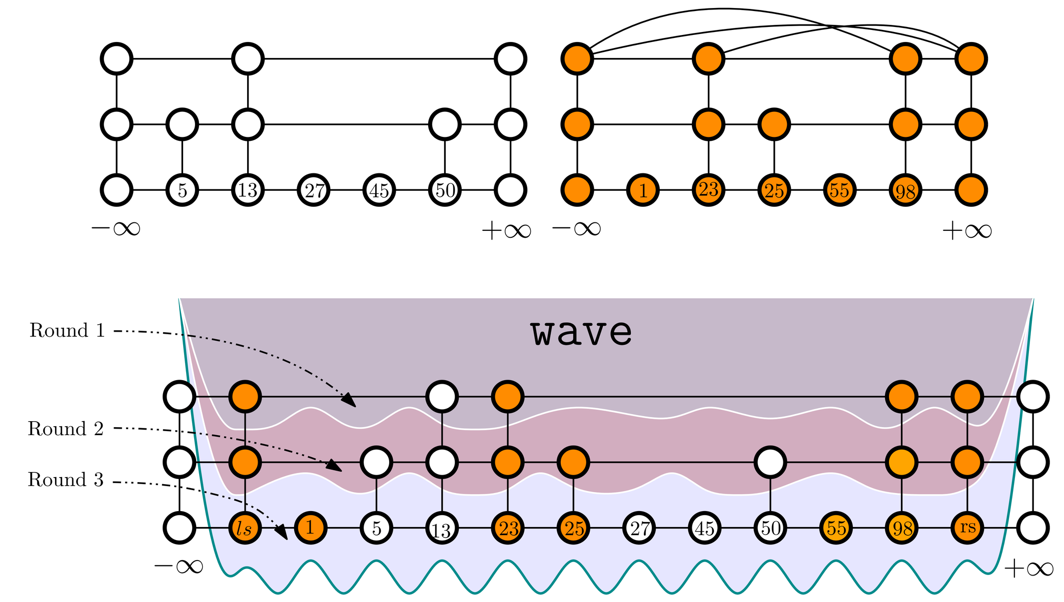
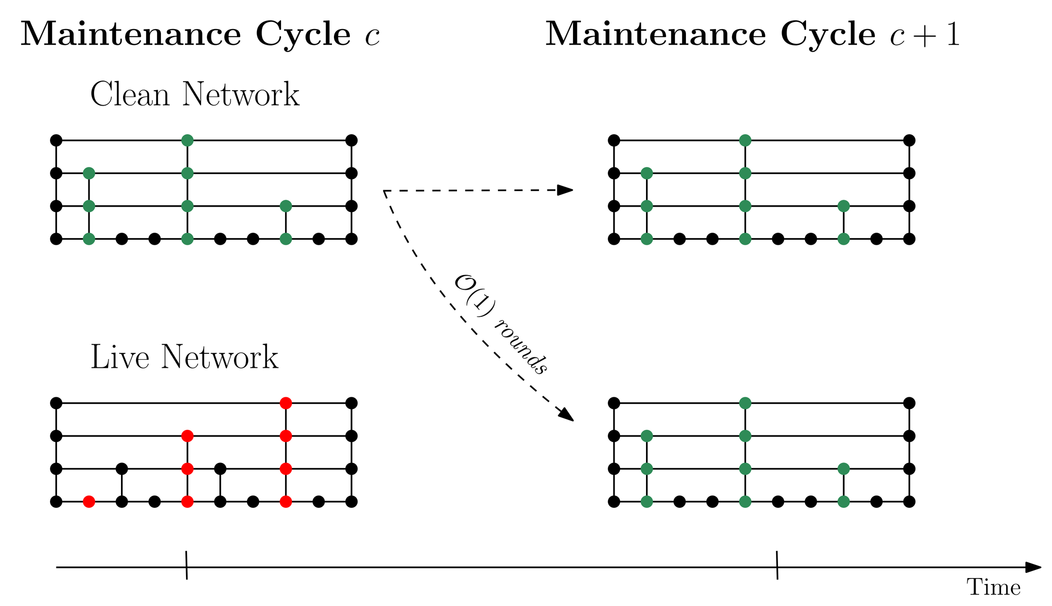


Figure 5. Illustration of our novel merge algorithm that merges two skip list of n elements in $\mathcal{O}(\log n)$ rounds with high probability.

Using the Clean network as the new Live network



Our main result

In this work:

- $\mathcal{O}(\log n)$ round algorithm that builds and maintains a distributed skip list despite an $\mathcal{O}(n/\log n)$ churn rate per round.
- Maintenance protocol the guarantees $\mathcal{O}(\log n)$ rounds insertion/deletion (of a batch of nodes) and query time on the skip list.
- $\mathcal{O}(\text{polylog}(n))$ bits messages and each node sends/receives $\mathcal{O}(\text{polylog}(n))$ messages per round.
- Workload proportional to the churn.
- The skip list is maintained for at least $\text{poly}(n)$ rounds w.h.p.

Our algorithm: (1) can be used to maintain **any** skip list-like data structure (e.g. skip graphs and skip$^+$ [3, 2]); (2) works with any number of data structure's keys per node; and, (3) can be adapted to maintain any distributed pointer-based data structure (e.g. graphs).

[1] J. Augustine, G. Pandurangan, and P. Robinson. Distributed algorithmic foundations of dynamic networks. *SIGACT News*, 2016.

[2] R. Jacob, A. W. Richa, C. Scheideler, S. Schmid, and H. Täubig. Skip$^+$: A self-stabilizing skip graph. *J. ACM*, 2014.

[3] A. James and S. Gauri. Skip graphs. *ACM Trans. Algorithms*, 2007.

[4] P. William. Concurrent maintenance of skip lists. In *Technical Report*, 1998.