The Forgiving Tree: A Self-Healing Distributed Data Structure

Tom Hayes
Navin Rustagi
Jared Saia
Amitabh Trehan

UNIVERSITY OF NEW MEXICO
Motivation

• Skype network crashes.

• August 15, 2007. Service disrupted to about 200 million users. Outage attributed to “failure of self-healing mechanisms”
Motivation

- Skype network crashes.
Ensuring Robustness

- Want to ensure that our network can recover from a number of node failures.
- Idea: build some redundancy into the network?
- Example: Connectivity
  - Use k-connected graph.
  - Price: degree must be at least k.
Ensuring Robustness

• Want to ensure that our network can recover from a number of node failures.

• Idea: build some redundancy into the network?

• Example: Connectivity
  • Use k-connected graph.
  • Price: degree must be at least k.
Self-Healing

Brain: component fails. Brain rewires and does without it

Computer: component fails. We analyze the whole system, identify the component, replace it with an identical working one.

- Which is more efficient?
Model

• Start: a network $G$.

• Nodes fail in unknown order $v_1, v_2, ..., v_n$

• After each node deletion, we can add and/or drop some edges between pairs of nearby nodes, to “heal” the network
And so on...
Goals

• Ensure connectivity
• Healing should be very fast (constant time)
• If vertex v starts with degree d, then its degree should never be much more than d
• Diameter shouldn’t increase by too much
A series of unfortunate events
Main Result

• A distributed algorithm, Forgiving Tree such that, for any network $G$ with max degree $D$, for an arbitrary sequence of deletions,
  • Graph stays connected
  • Diameter increases by $\leq \log(D)$ factor
  • Degrees increase by $\leq 3$ (additive)
  • Each repair takes constant time and involves $O(D)$ nodes.
Main Result

• A distributed algorithm, Forgiving Tree such that, for any network G with max degree D, for an arbitrary sequence of deletions,

• Graph stays connected

• Diameter increases by $\leq \log(D)$ factor

• Degrees increase by $\leq 3$ (additive)

• Each repair takes constant time and involves $O(D)$ nodes.

Matching lower bound
The Forgiving Tree: motivations

- Trees are the “worst case” for maintaining connectivity. Suppose we are given one.

- Our algorithm is based on maintaining a virtual tree. This helps us keep track of which vertices can afford to have their degrees increased, and also avoid blowing up distances.
FT: first approximation

- Find a spanning tree of $G$.

- Choose some vertex to be the root, and orient all edges toward the root.

- When a node is deleted, replace it by a balanced binary tree of “virtual nodes”

- Short-circuit any redundant virtual nodes

- Somehow the surviving real nodes simulate the virtual nodes
Replacing v by a balanced binary tree of virtual nodes

Short-circuiting a redundant virtual node
Virtual Nodes

- A virtual node starts with degree 3, since internal node of a complete binary tree.
- If a neighbor is a leaf, and is deleted, the virtual node becomes redundant. Then we “short-circuit” it.
- This ensures that there are always more real nodes than virtual nodes. Each real node needs to simulate at most one virtual.
Guarantees

• Suppose v and w originally had distance d in the spanning tree. Then after any sequence of deletions, their distance in the ideal tree is at most d log(max degree)

• Degrees of real nodes only decrease. Degrees of virtual nodes are at most 3.

• Each virtual node will be simulated by one real node.
Algorithm in action

Node v deleted:
Node p deleted:
Node d deleted:

The network after all the deletions and redundant by the deletion of d and so is the leaves of RT(h) are at the same depth.
Caveats

- Each virtual node somehow gets assigned to a surviving real node. (next few slides) The actual healed network is just the homomorphic image of the restored tree under this map.

- We won’t go into full detail about how to implement this in a distributed way.
Homomorphism: Given $G_1 = (V_1, E_1), G_2 = V_2, E_2$

a map such that $\{v, w\} \in E_1 \Rightarrow \{f(v), f(w)\} \in E_2$

A virtual tree (left) and its homomorphomic image (right)
A different labeling of the virtual nodes, and the image
Note that in this case the image is not a tree.
Assigning virtual duties

- At the start of the algorithm, each non-leaf node $v$ writes a “will” specifying one child to simulate each virtual node which will be created when $v$ is deleted.

- The relevant piece of this will is entrusted to each child.

- $v$ may revise the will from time to time.

- When $v$ is deleted, the will is implemented.
Where there’s a will...

x assigns virtual nodes in his will

h is the “heir”

The will gets split into 5 parts--one for each neighbor.
Heirs

- What happens when the node simulating a virtual node is deleted?
  - (a) if the virtual node has become redundant, it is short-circuited as usual.
  - (b) if the simulating node \( w \) has children, then \( w \)'s will designates a child to take over the virtual node. This is \( w \)'s “heir”
  - (c) if the simulating node is a leaf, hang on...
Leaf deletions

• When a leaf is deleted, there are two cases:
  • (a) if the parent is still alive, parent updates its will by reassigning v’s virtual node.
  • (b) if the parent is already gone, v should have left a will with the simulator of the virtual node which will become redundant.
Leaf deletions

z is deleted, making y’s virtual node redundant. Now y takes over z’s virtual node (z had no heir).
Summary

• Forgiving tree ensures degree increase is additive constant. Diameter increase is log of max degree.

• These parameters are essentially optimal.

• Forgiving tree is fully distributed, has $O(1)$ latency and $O(1)$ messages exchanged per round.
Future Directions

• Handle stretch (max increase in $d(u,v)$ -- FT only bounds the increase in diameter)

• Allow node insertions as well as deletions

• Extend model and algorithms to apply to peer-to-peer and sensor networks.
Thank You