The Forgiving Graph: A Low-Stretch Distributed Data Structure

Tom Hayes
Jared Saia
Amitabh Trehan

University of New Mexico
Epic Fail

Twitter, August 6, 2009
Facebook, August 6, 2009
Skype, August 15, 2007
Self-Healing

Brain: component fails, brain rewires and does without it

Computer networks: components fail, network fails until components fixed.
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.
Model

• Start: a network $G$.

• An adversary inserts or deletes nodes.

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

G: present state of network

G’: graph of only insertions and original nodes
Goals

• Ensure connectivity.

• Healing should be very fast.

• If vertex v starts with degree d, then its degree should never be much more than d.

• Distance between any two nodes shouldn’t increase by too much.
A series of unfortunate events
Main Result

• A distributed algorithm, Forgiving Graph such that:

  • Degree increase: Degree of node in G ≤ 3 times degree in G'
Main Result (Contd..)

• *Stretch*: Distance between any two nodes in $G \leq \log n$ times their distance in $G'$

\[ d(u, v) = 5 \]

\[ d(u, v) = 3 \]
Main Result (Contd..)

• **Cost**: Repair of node of degree \(d\) requires at most \(O(d \log n)\) messages of length \(O(\log^2 n)\) and time \(O(\log d \log n)\)
Main Result

• A distributed algorithm, Forgiving Graph such that:
  • Degree of node in $G \leq 3$ times degree in $G'$
  • Distance between any two nodes in $G \leq \log n$ times their distance in $G'$
  • Cost: Repair of node of degree $d$ requires at most $O(d \log n)$ messages of length $O(\log^2 n)$ and time $O(\log d \log n)$
FG extends Forgiving Tree

[PODC '08]

• Requires no initialization (saves $O(|E| \log n)$ messages)
• Handles insertions
• Keeps stretch small, not just diameter
• Introduces new techniques e.g. hafts
The FG algorithm: Outline

• Node inserted without restrictions.
• When a node is deleted, replace it by a half-full tree (described later) of “virtual nodes”.
• If two half-full trees become neighbors, ‘merge’ them to form a new half-full tree.
• Somehow the surviving real nodes simulate the virtual nodes
Replacing \( v \) by a Reconstruction Tree (RT) of virtual nodes (in oval). The ‘real’ neighbors are the leaves of the tree.

Merging two reconstruction trees on deletion of \( x \)
Virtual Nodes

• A virtual node has degree at most 3, since internal node of a binary tree.

• Each real node will simulate at most one virtual node per neighbor.

• After any sequence of deletions, the distance between two nodes can only increase by a factor of the longest path in the largest RT i.e. $\log n$. 
Half-Full Trees (hafts)

- A rooted binary tree in which every non-leaf node $v$ has the following properties:
  - $v$ has exactly two children.
  - The left child of $v$ is the root of a complete binary subtree containing at least half of $v$’s children.
Seven Samurai: the first seven hafts
Hafts in binary

- Let $i$ be an integer. There is a unique haft $T$ having $i$ leaves.
- Let $h$ be the number of ones in binary representation of $i$. $T$ has $h-i$ spine nodes and $h$ complete binary trees.
Operations on hafts

- **Strip**: return complete trees on deletion of a node (and its virtual nodes).
Operations on hafts

- **Merge**: Recombine hafts to make new haft. Analogous to binary addition.

- Strip to get forest of complete trees.

- Join adjacent trees with a new node as root, larger tree as left child.

\[
\begin{array}{cccc}
0101 & + & 0010 & + & 0001 \\
\end{array}
\]

\[
= 1000
\]
FG in action

Node v deleted ...
replaced by RT(v)
Node y deleted...
replaced by $RT(y)$
Node w deleted...
RT(v), RT(w) and u merge.
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 homomorphically image (right)
Technical issues

• Implementing Merge: Binary Tree (BT) of post deletion fragments and anchor nodes

• Finding primary roots: probe messages through anchors

• Am I a primary root? maintain and use height, number of descendant information

• Merging hafts: representative mechanism
Summary

• Forgiving graph ensures degree increase is a multiplicative constant. Stretch is at most $\log n$.

• These parameters are essentially optimal.

• Forgiving graph is fully distributed, has $O((\log d \log n)$ latency and $O(d \log n)$ messages exchanged per round, for deletion of node of degree $d$. 

Future Directions

• Extend model and algorithms to apply to sensor networks.

• Functional self-healing: Can we perform robust computation in face of component failures e.g. in circuits.

• Find connections between our work and self-healing in biological and social networks.
Thank You