THE TEMPORAL EVOLUTION OF SELF-HEALING

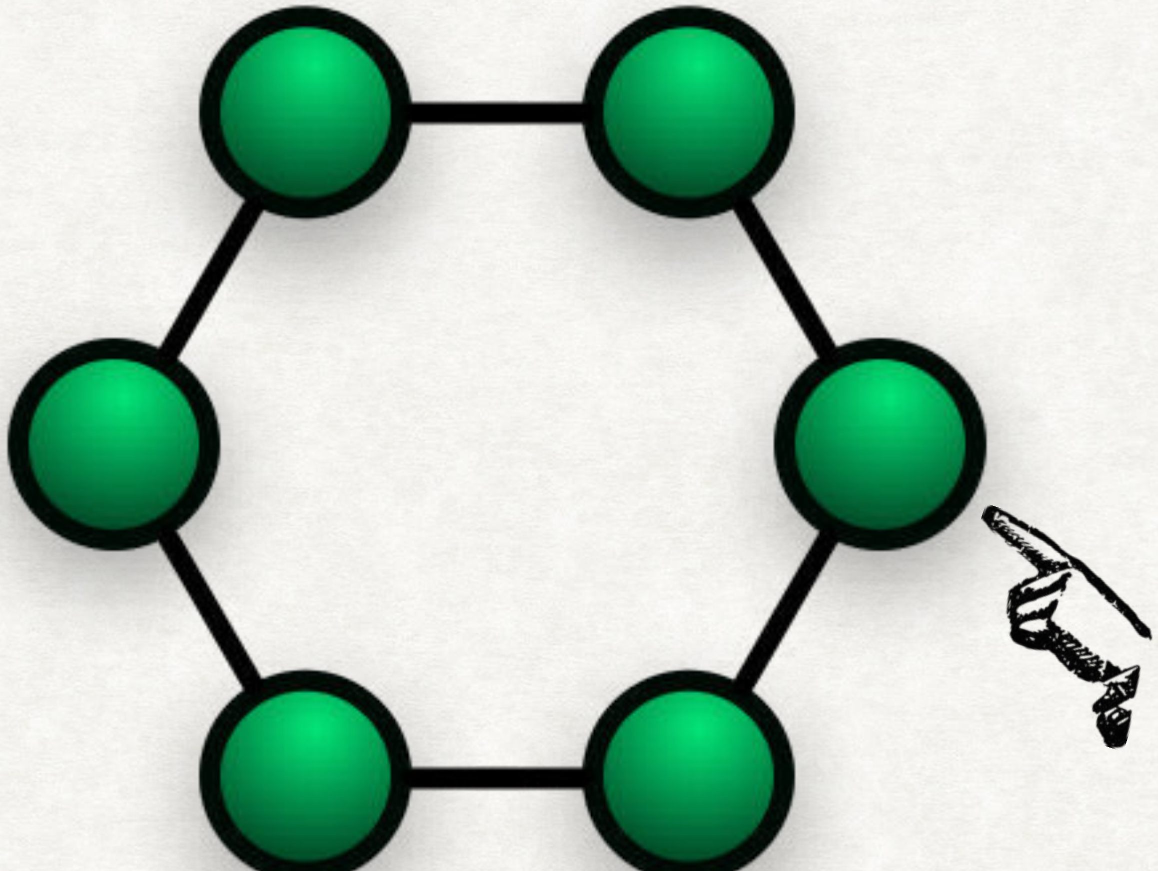
AMITABH TREHAN
LOUGHBOROUGH UNIVERSITY

(Joint work with
Armando Castanader, Danny Dolev, Gopal Pandurangan, Peter Robinson, Danupon Nanangkoi, Jared Saia, Tom Hayes, … and anybody else who cared to listen!)

www.amitabhtrehan.net
www.huntforthetowel.wordpress.com

CENTRALISED: WHO GETS TO PRINT?

A Network



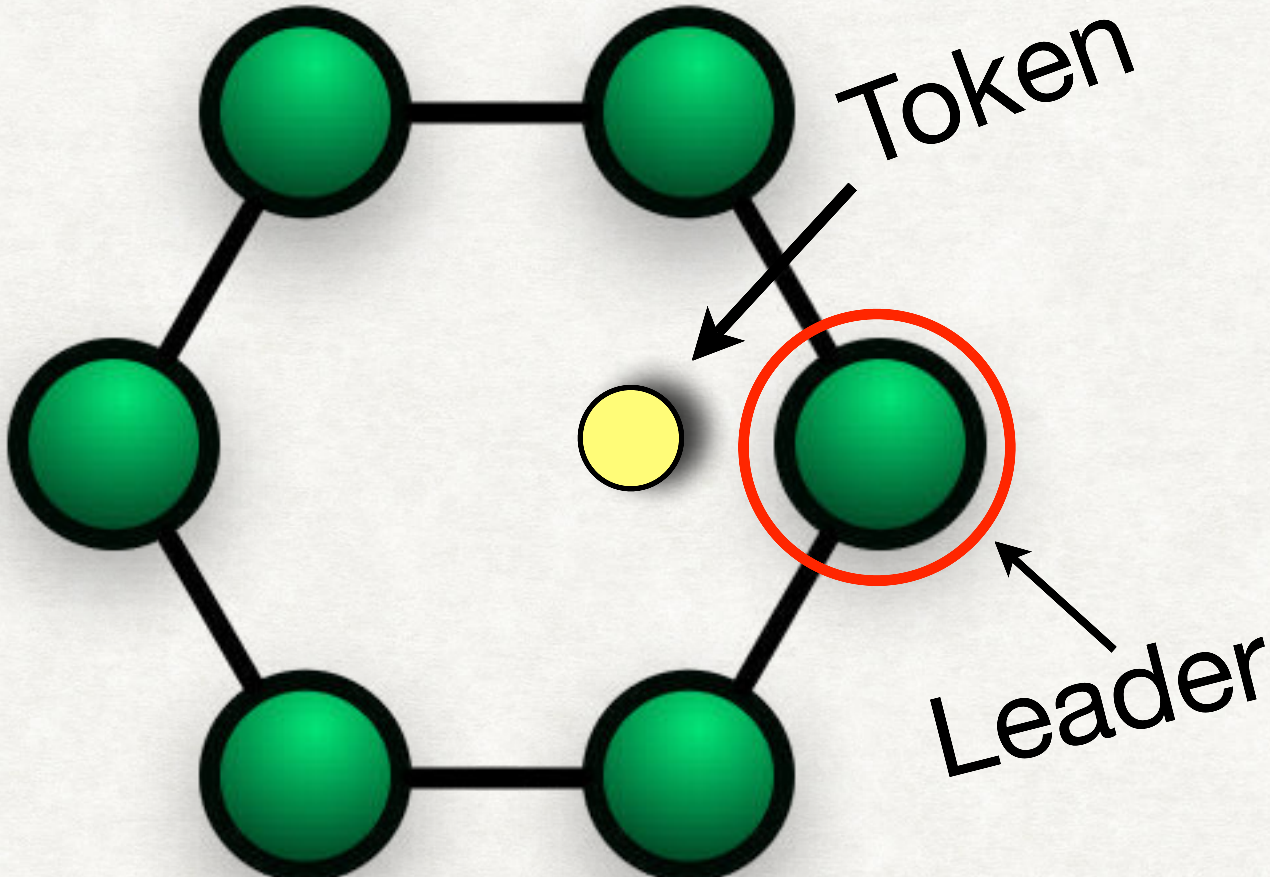
A Printer

<u>Centralised Algorithms:</u>
Single computer with the whole problem instance/data available.

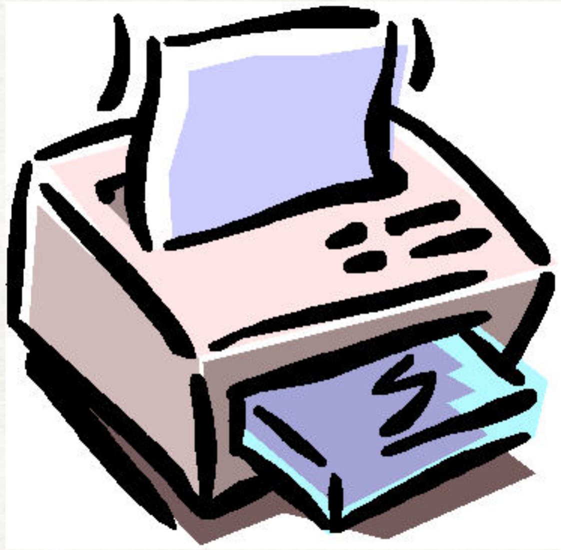
Q:
which one of <u>them</u> will get the printer?

DISTRIBUTED: WHO GETS TO PRINT

A Network



A Printer

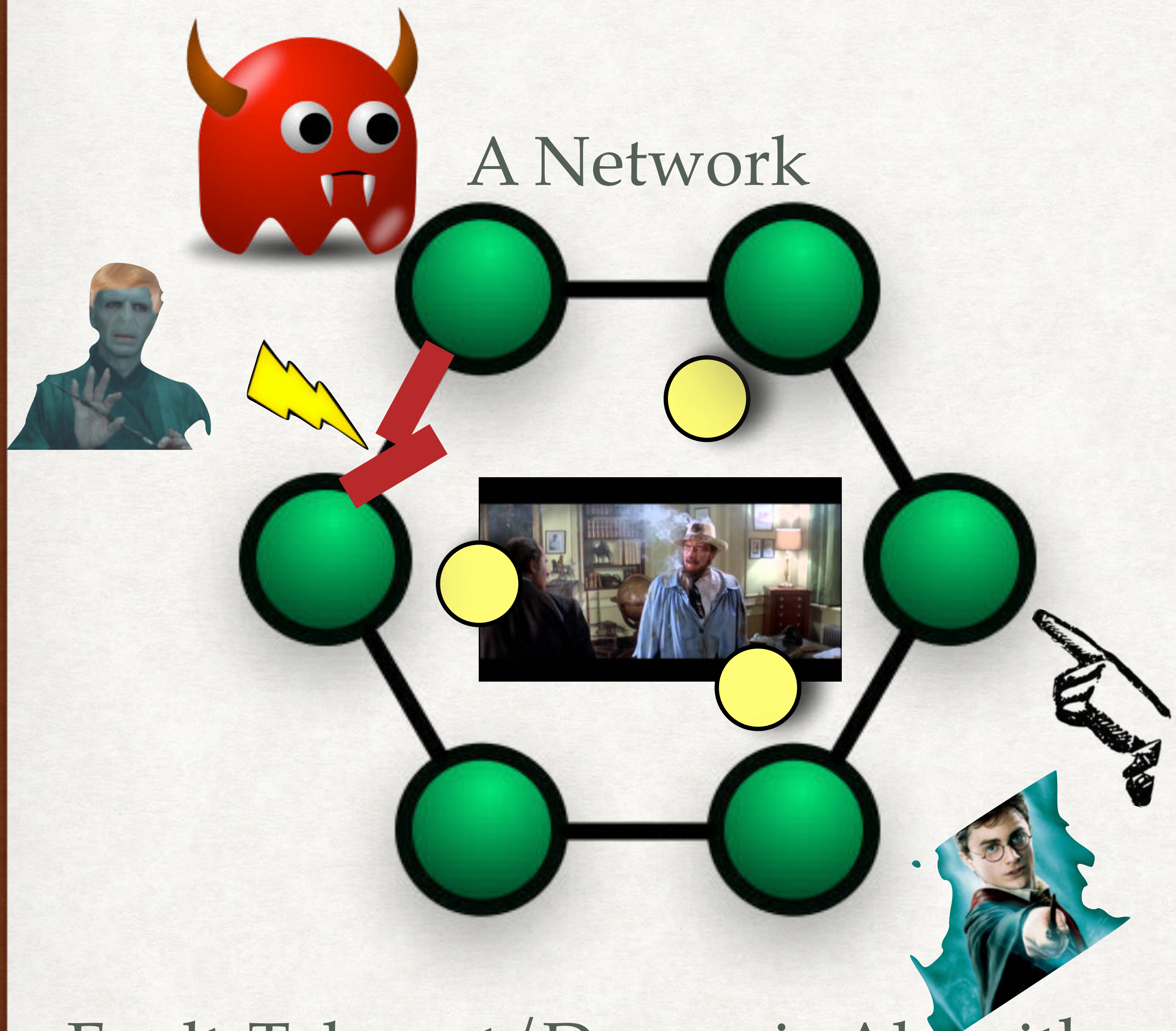


<u>DeCentralised/</u>
<u>Distributed Algorithm:</u>
Multiple `computers' each with it's own local view/data.

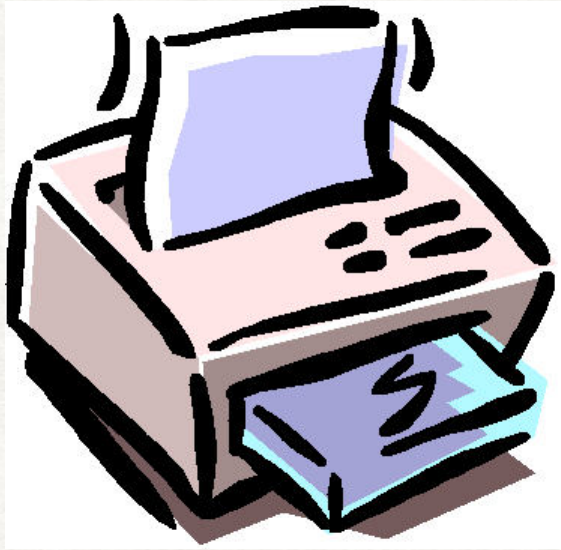
Q:
which one of <u>us</u> will get the printer?

DISTRIBUTED IN A DYNAMIC/FAULTY ENVIRONMENT: AY YE PRINTER!

A Network



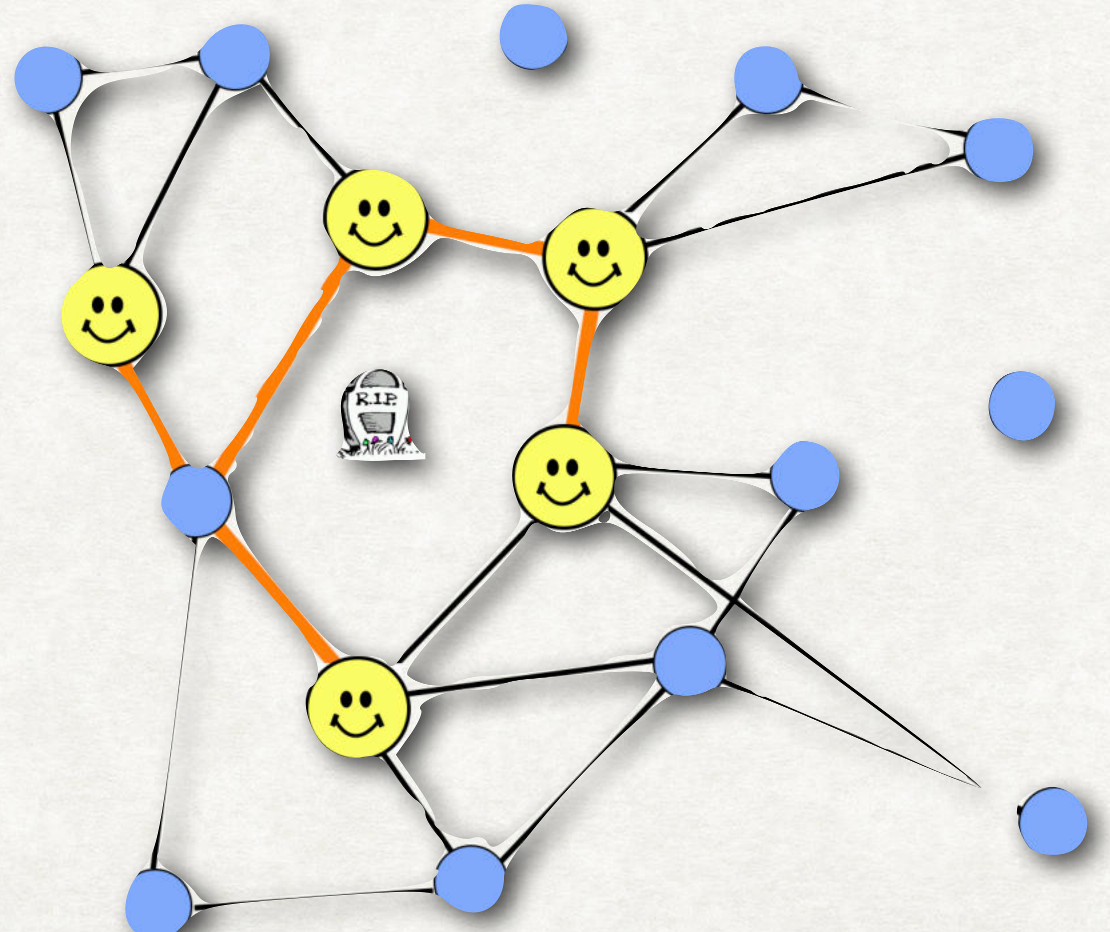
A Printer



Q:
which one of <u>us</u> will get the printer despite failures or changes?

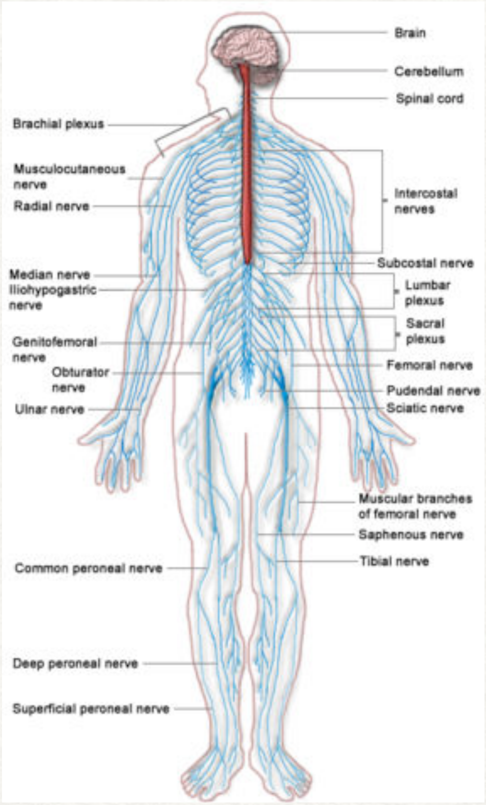
<u>Fault-Tolerant/Dynamic Algorithms:</u>
In faulty/dynamic environments

GRAPH RECONSTRUCTION (SELF-HEALING) GAME!



MOTIVATIONS

- Responsive Repair: As in the human brain! (rewire, don't reboot!)
- Autonomic systems:



- **Churn** in P2P/Reconfigurable networks: Nodes come and go!

SELF-HEALING (ON NETWORKS)

- Start: a distributed network G

Run forever or till possible

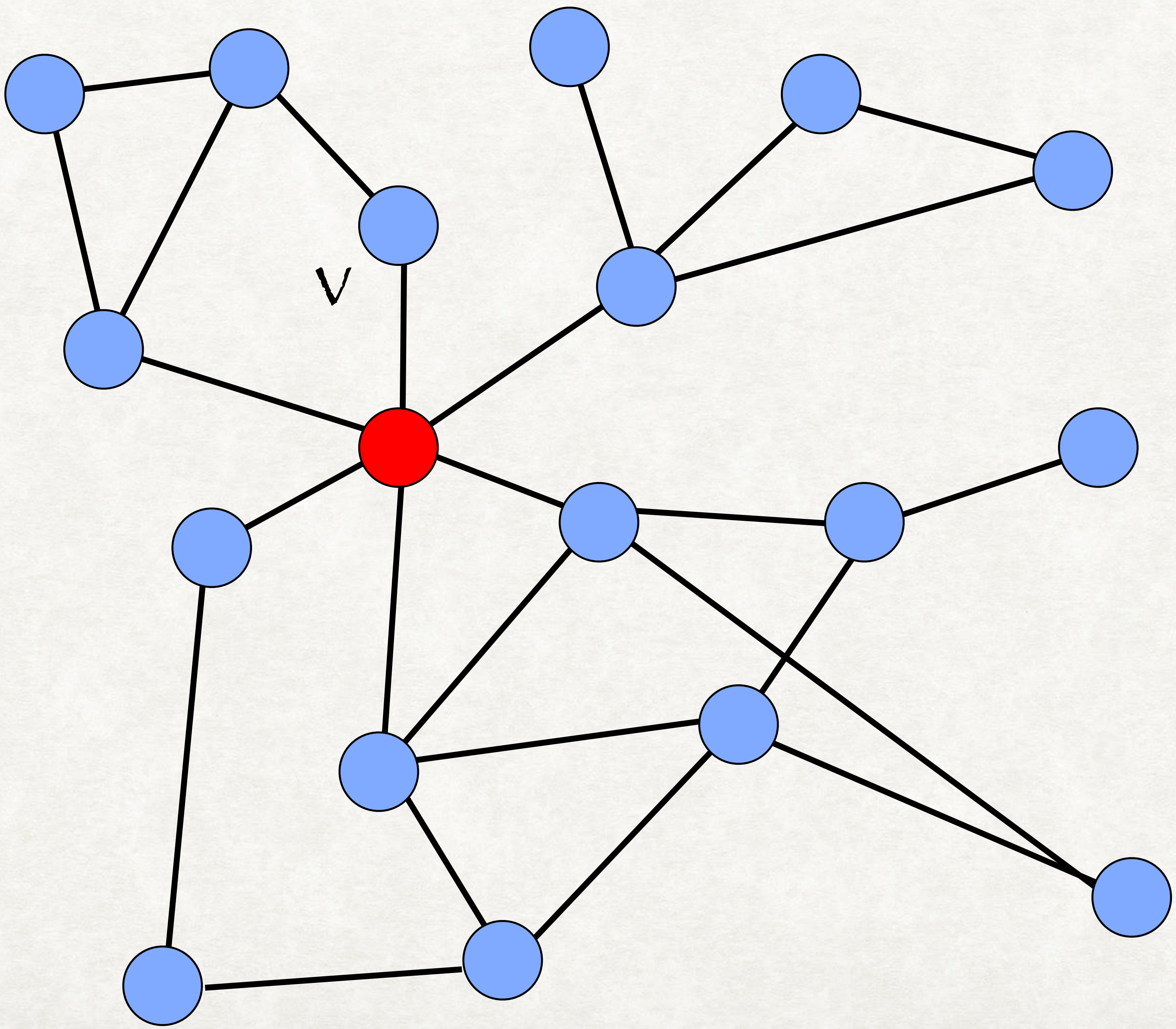
{

- Attack: An adversary inserts or deletes one node per round
- Healing: After each adversary action, we add and/or drop some edges between pairs of nearby nodes, to "heal" the network

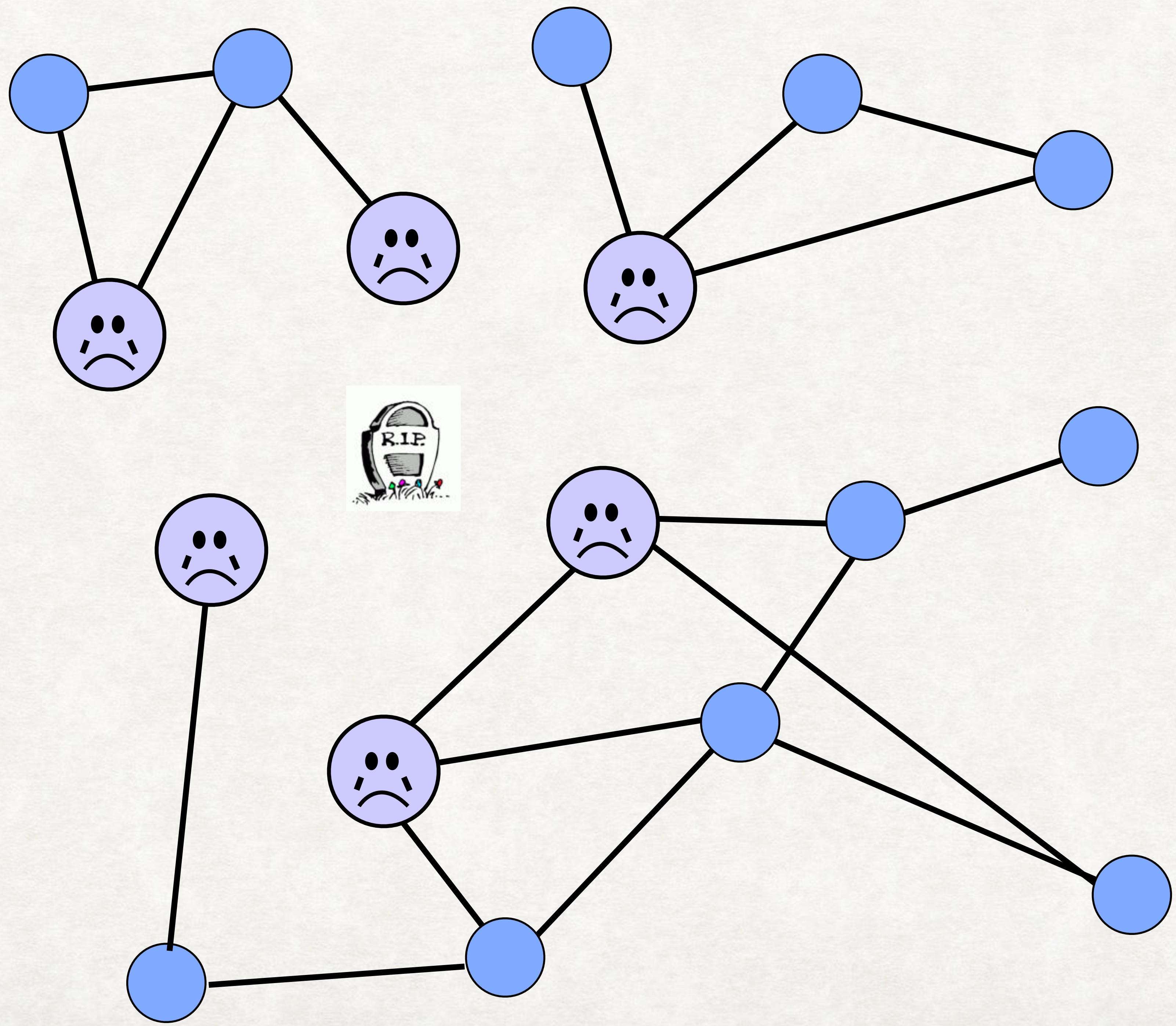
}

- Node dynamic as opposed to edge dynamic!

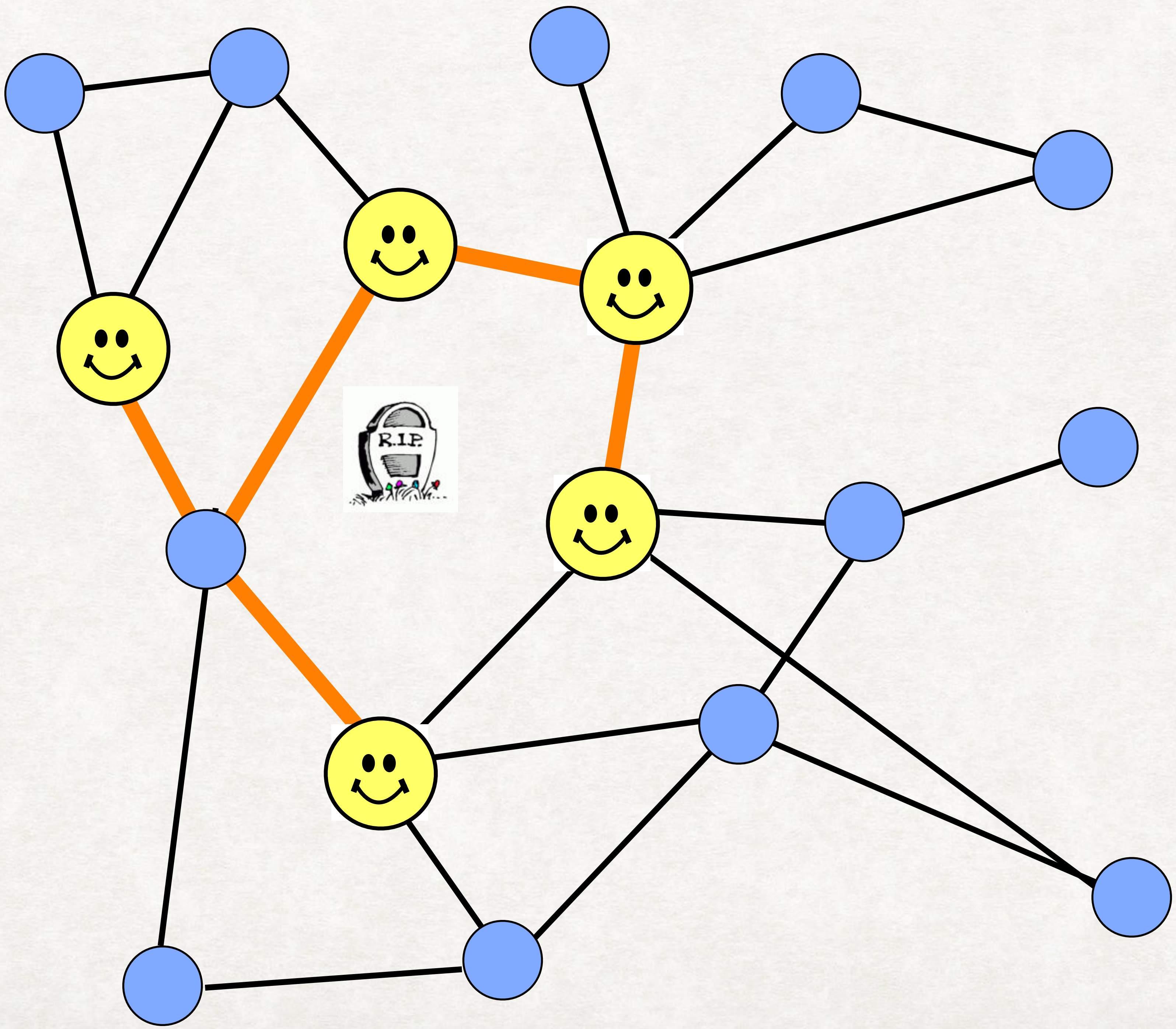
SELF-HEALING ILLUSTRATION



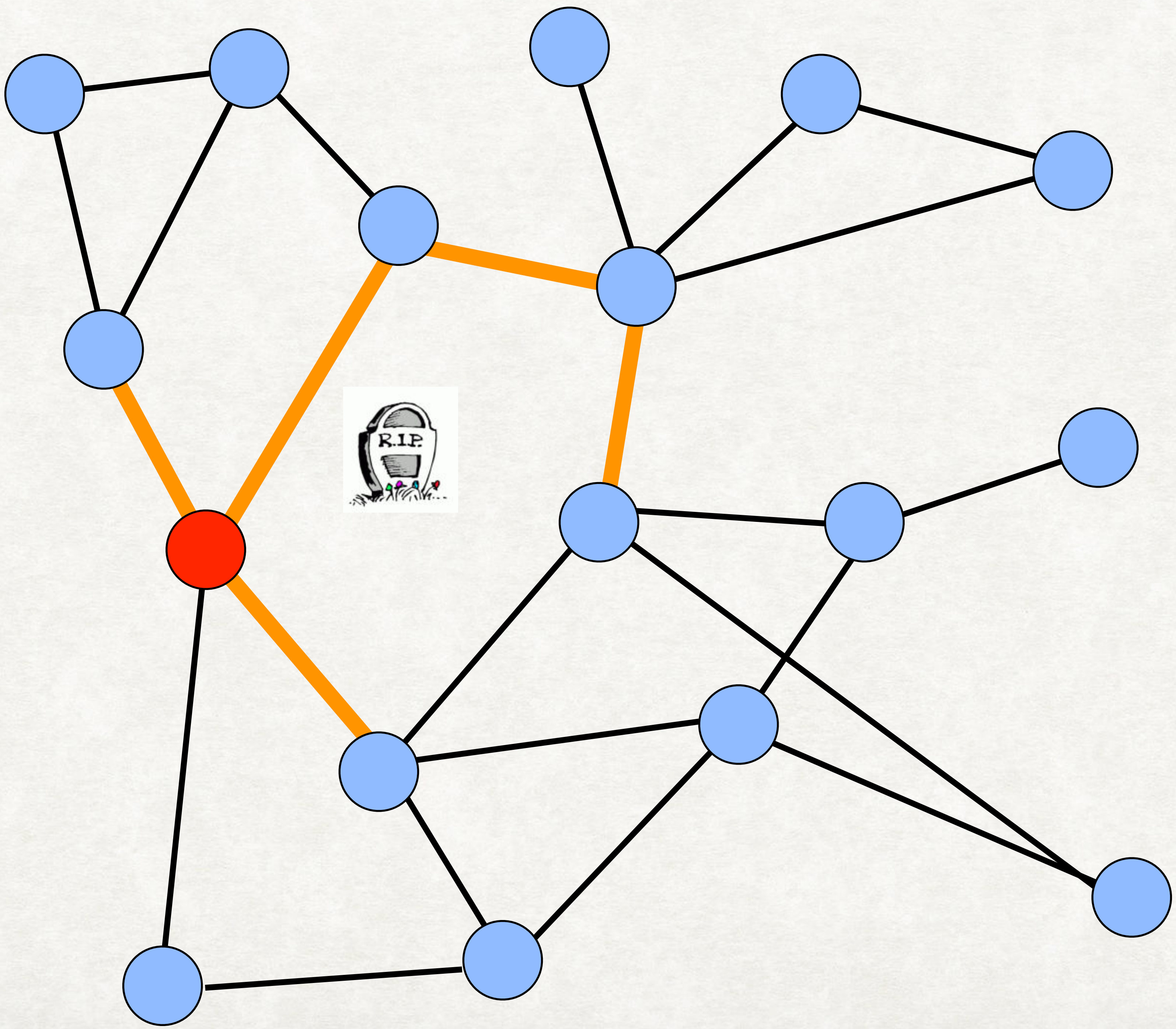
SELF-HEALING ILLUSTRATION



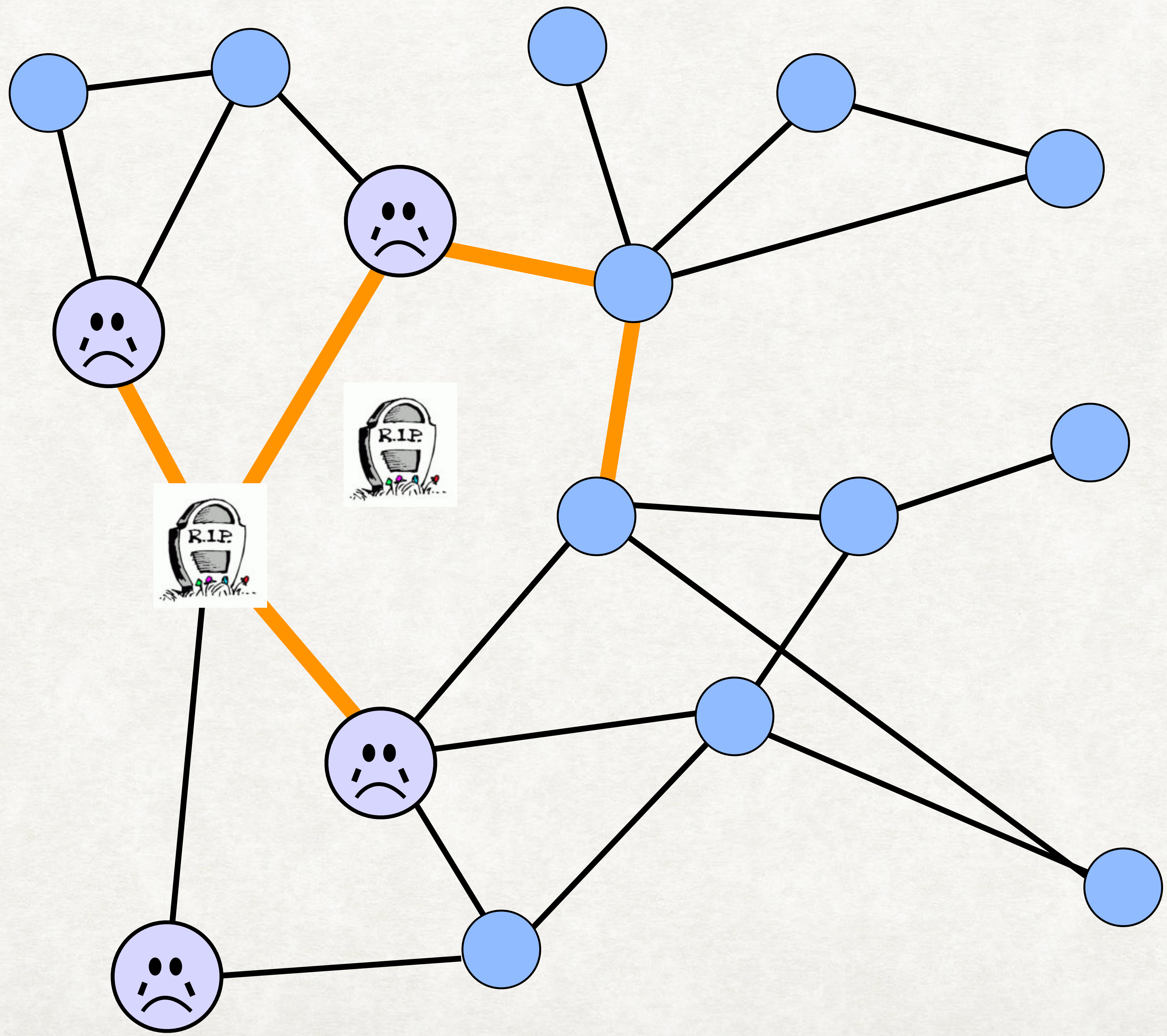
SELF-HEALING ILLUSTRATION



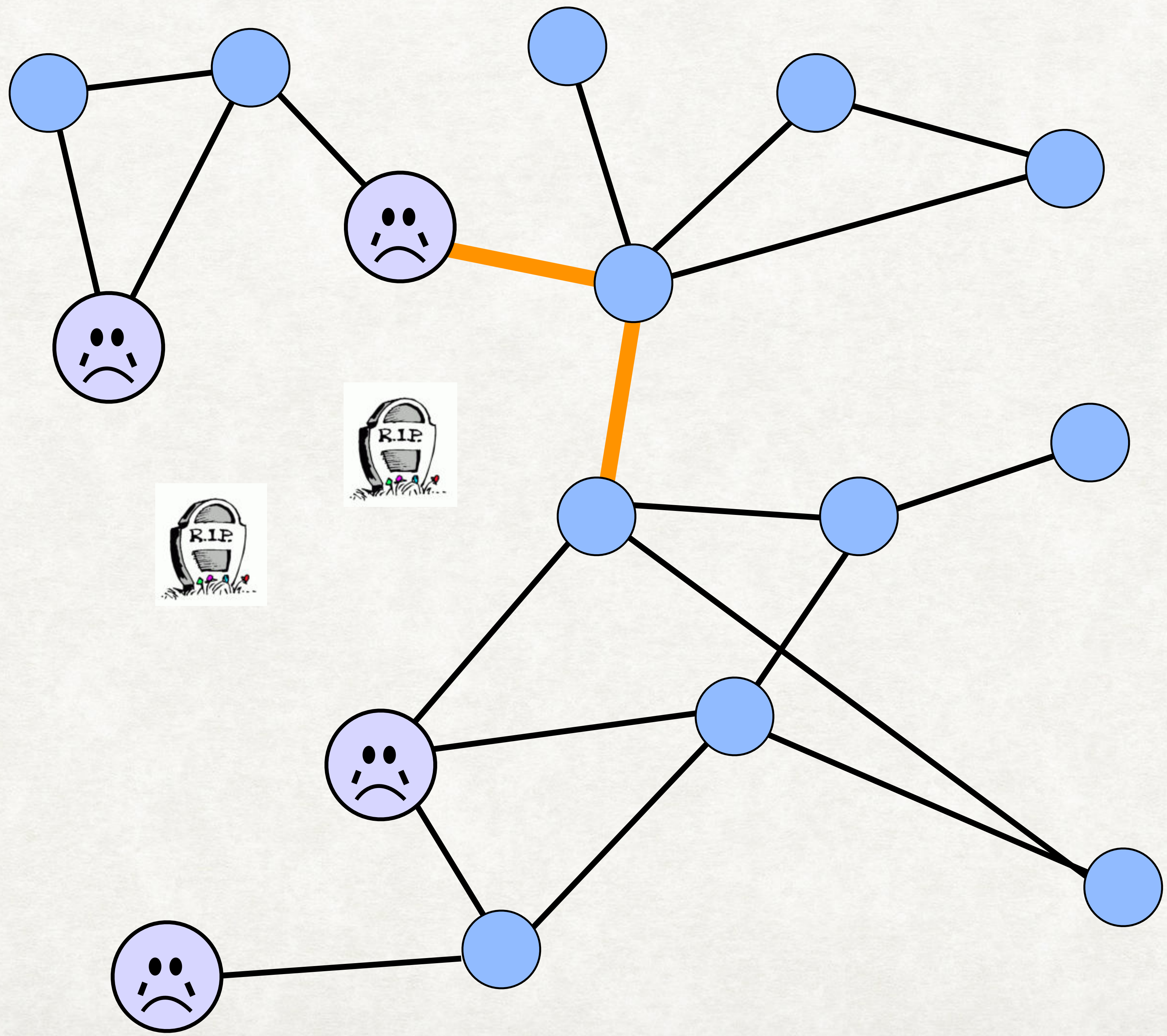
SELF-HEALING ILLUSTRATION



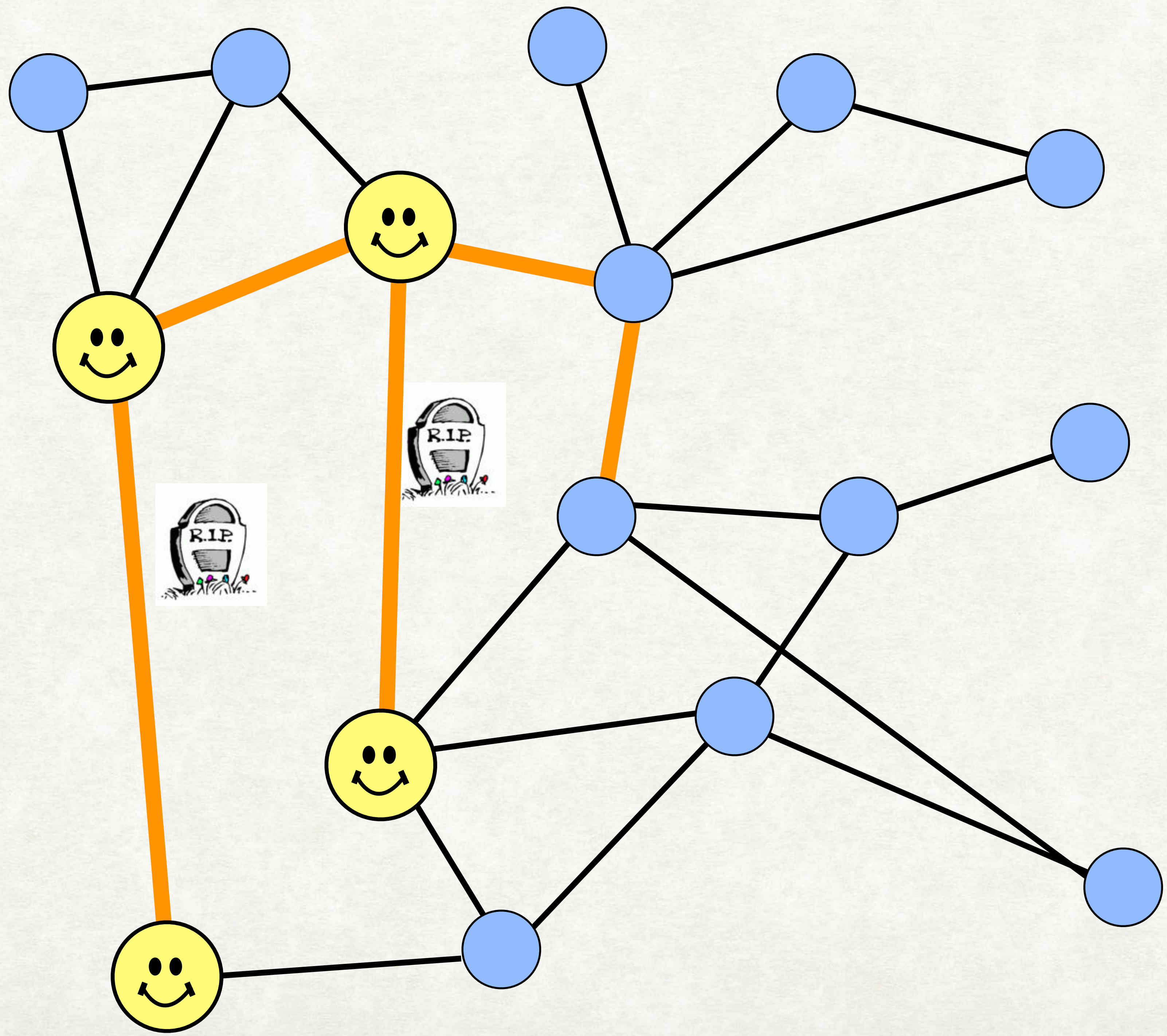
SELF-HEALING ILLUSTRATION



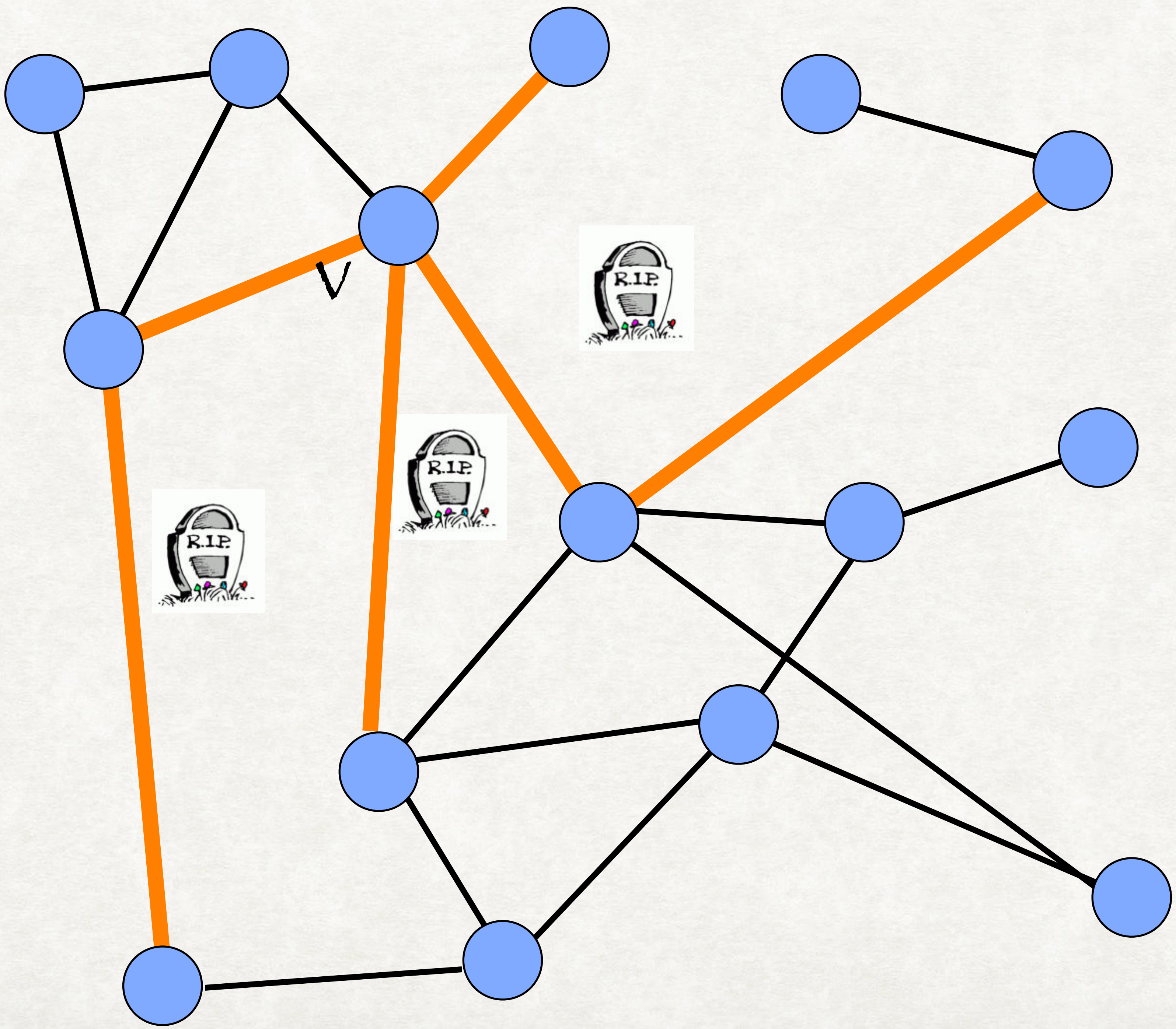
SELF-HEALING ILLUSTRATION



SELF-HEALING ILLUSTRATION

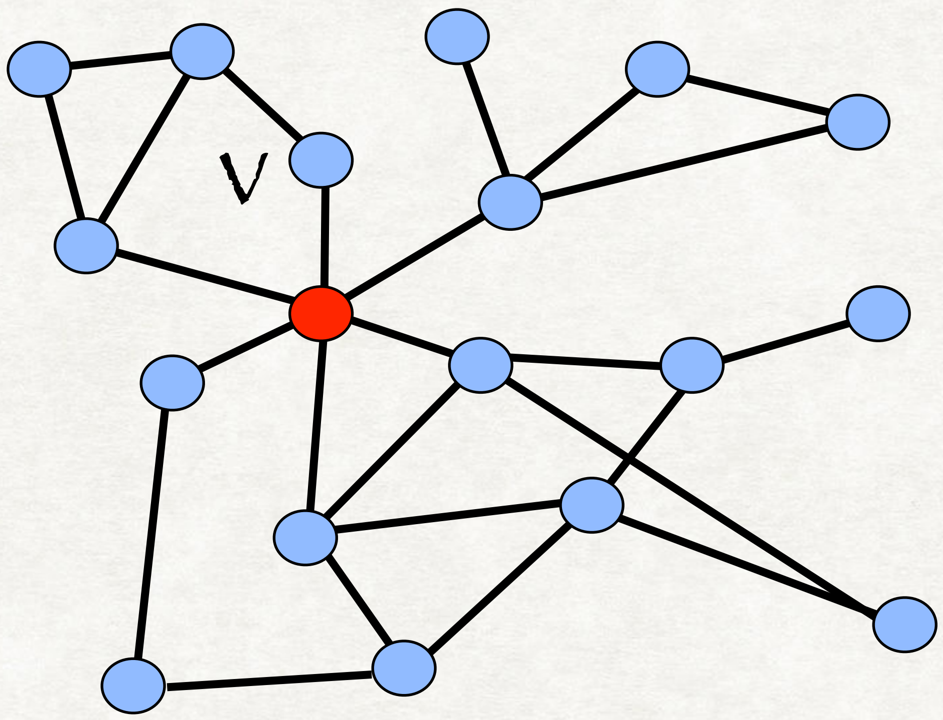


SELF-HEALING ILLUSTRATION

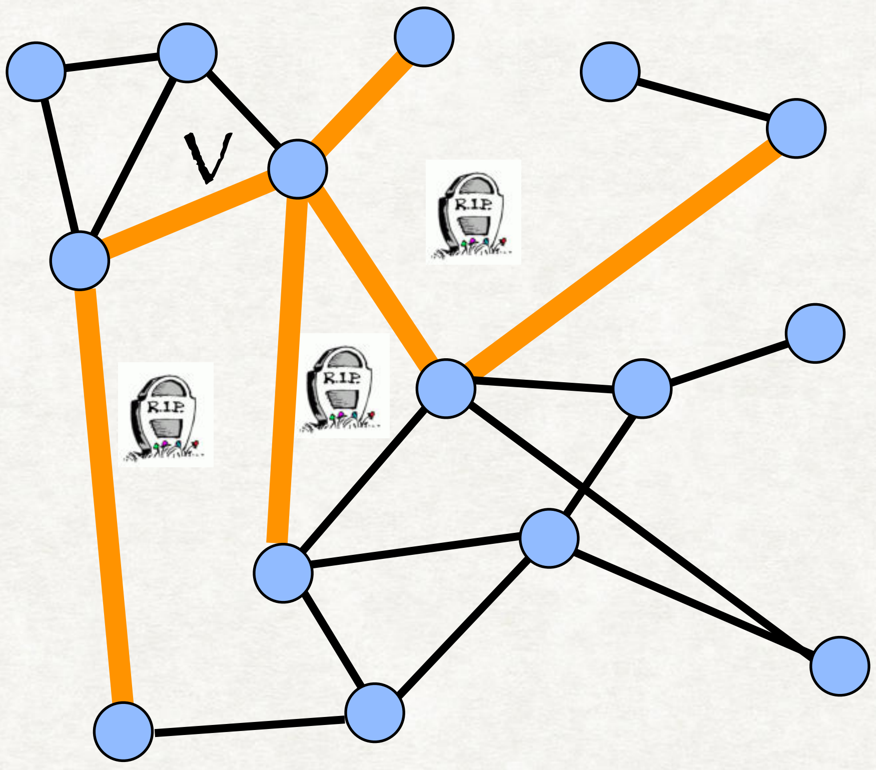


and so on ....

PROBLEM



G_0



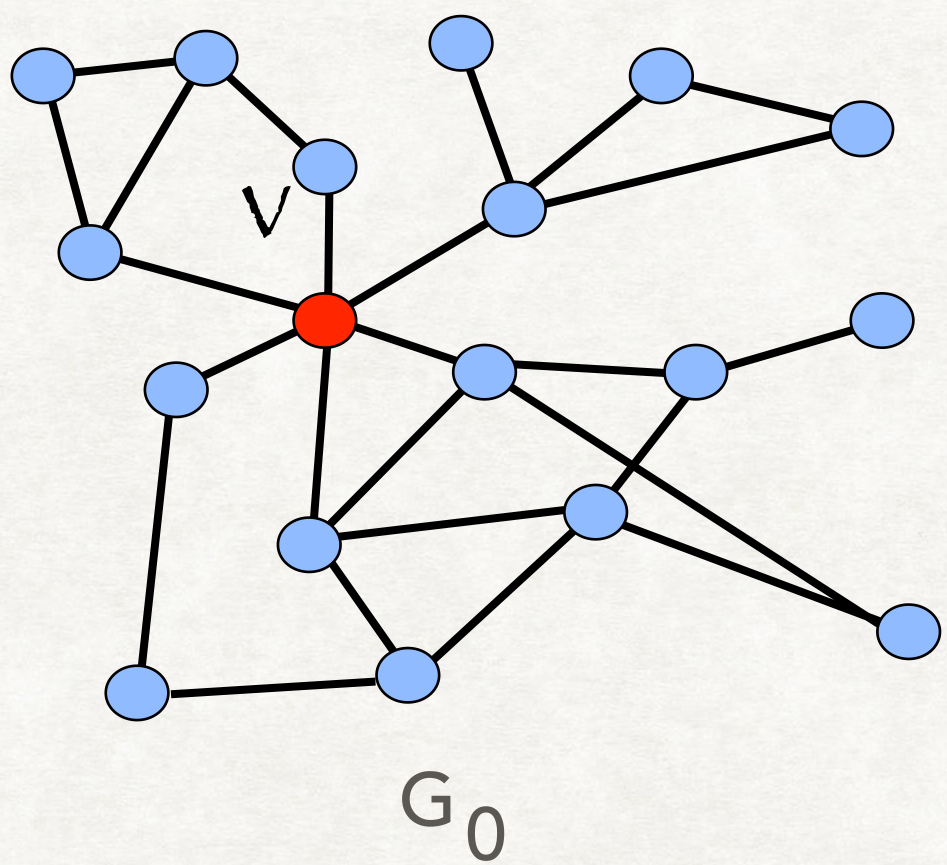
G_3

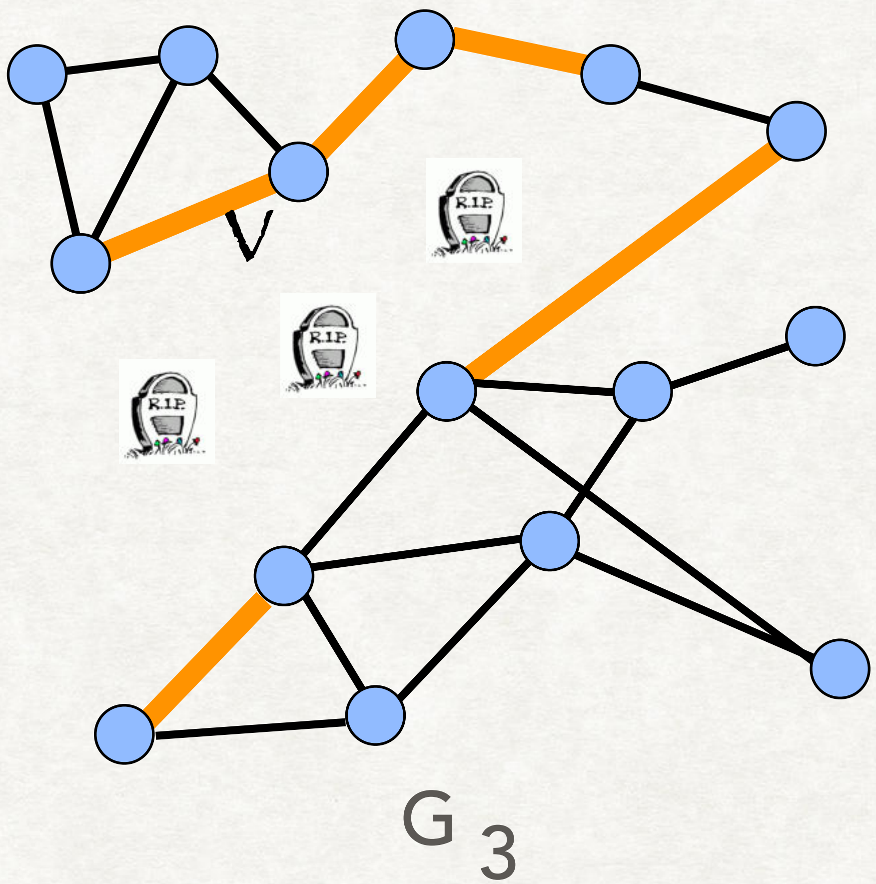
Degree(v,G_0) = 2

Degree(v,G_3) = 5

POSSIBLE HEALING TOPOLOGIES:

LINE GRAPH

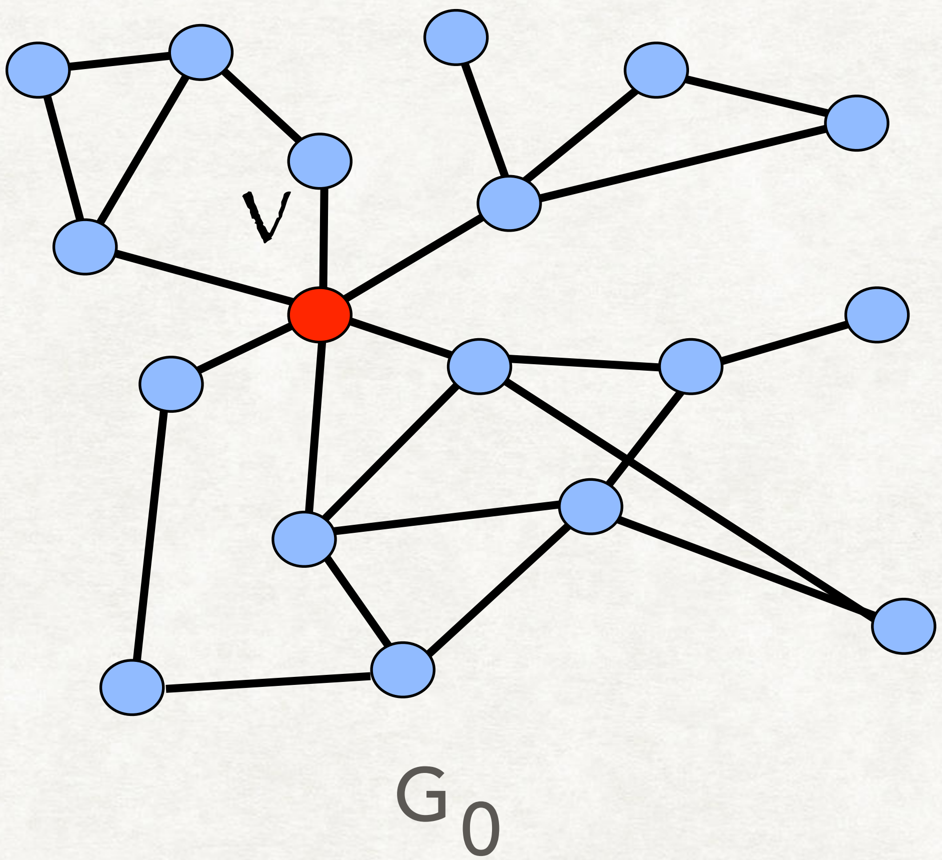


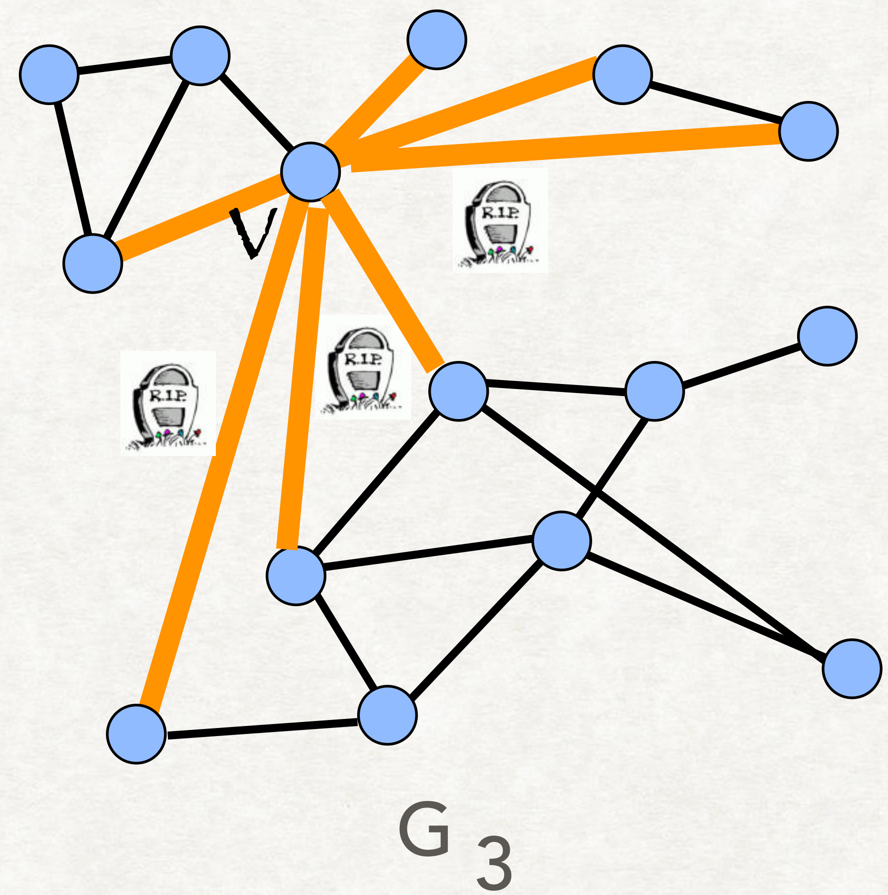


Low degree increase but diameter/ distances blow up

POSSIBLE HEALING TOPOLOGIES:

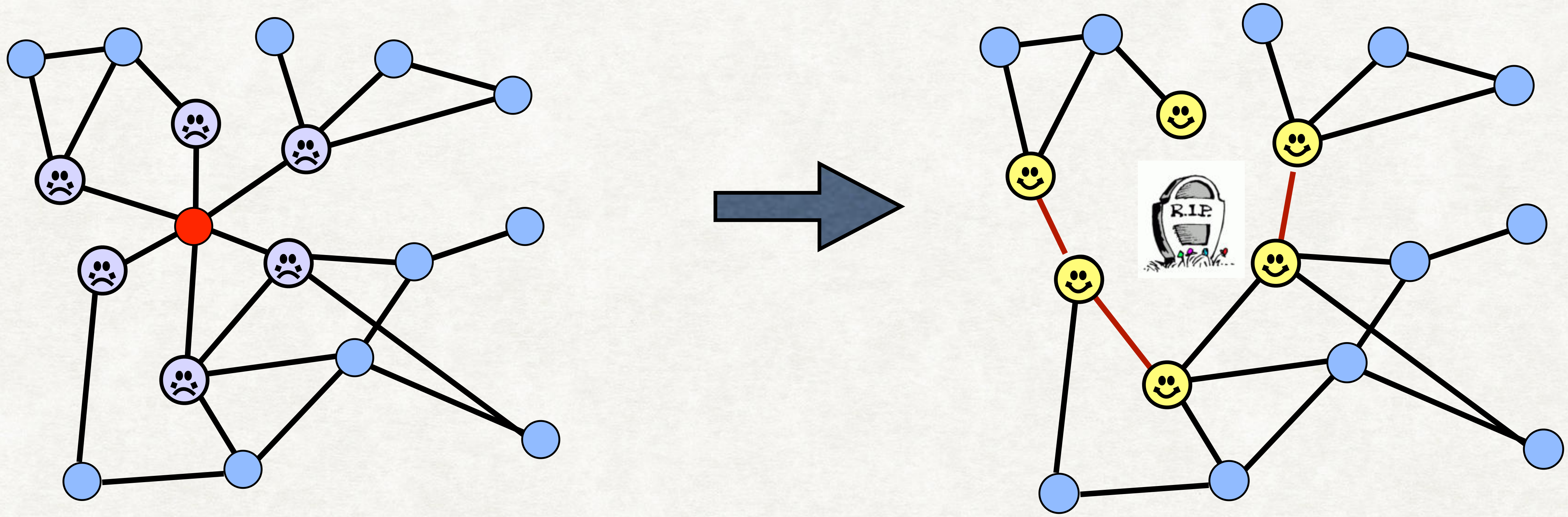
STAR GRAPH





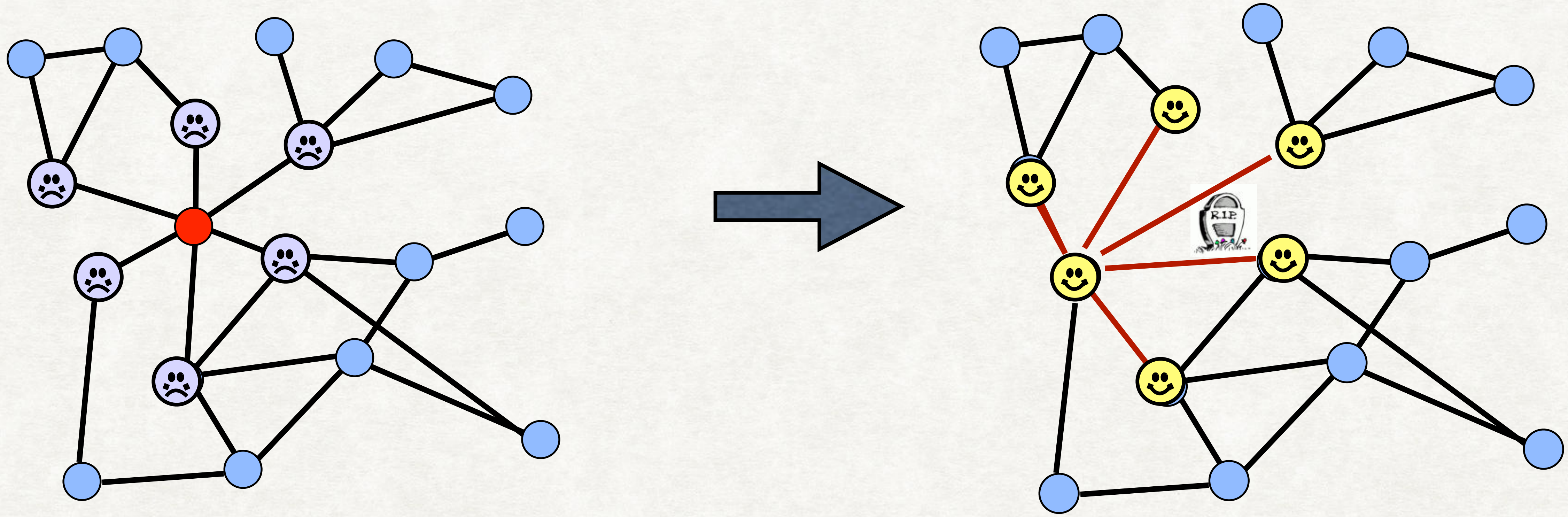
Low distances but degree blows up

CHALLENGE 1: PROPERTIES CONFLICT



- Low degree increase => high diameter/stretch/ poorer expansion?
- Low diameter => high degree increase?

CHALLENGE 2: LOCAL FIXING OF GLOBAL PROPERTIES

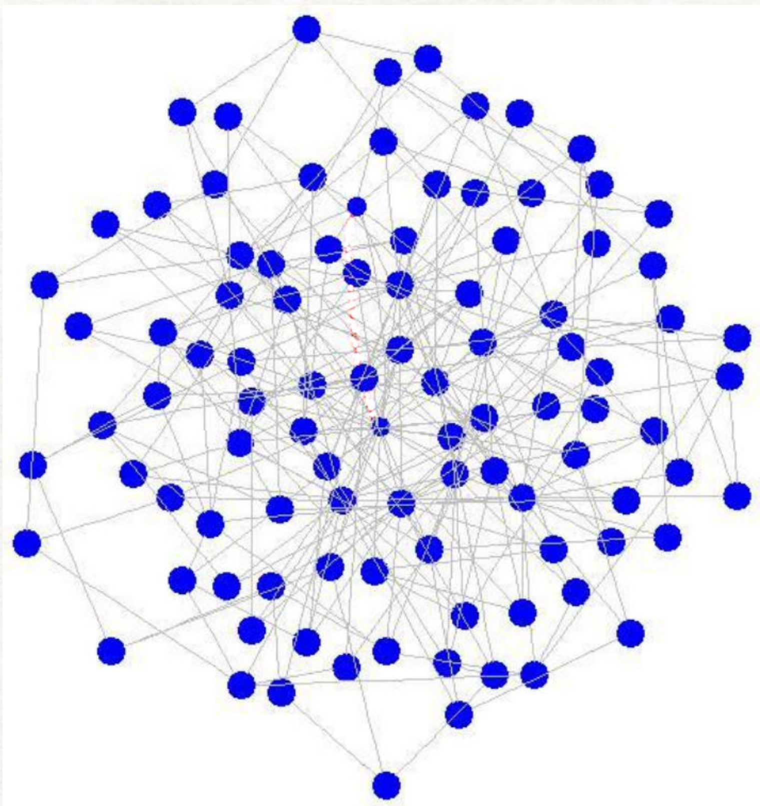


- Limited global Information with nodes
- Limited resources and time constraints

SELF-HEALING(TOPOLOGICAL) GOALS

- Healing should be fast.
- Certain (topological) properties should be maintained within bounds:
  - Connectivity
  - Degree (quantifies the work done by algorithm)
  - Diameter/ Stretch
  - Expansion/ Spectral properties

DASH: DEGREE ASSISTED SELF-HEALING*



Original Graph, n = 100; t =0

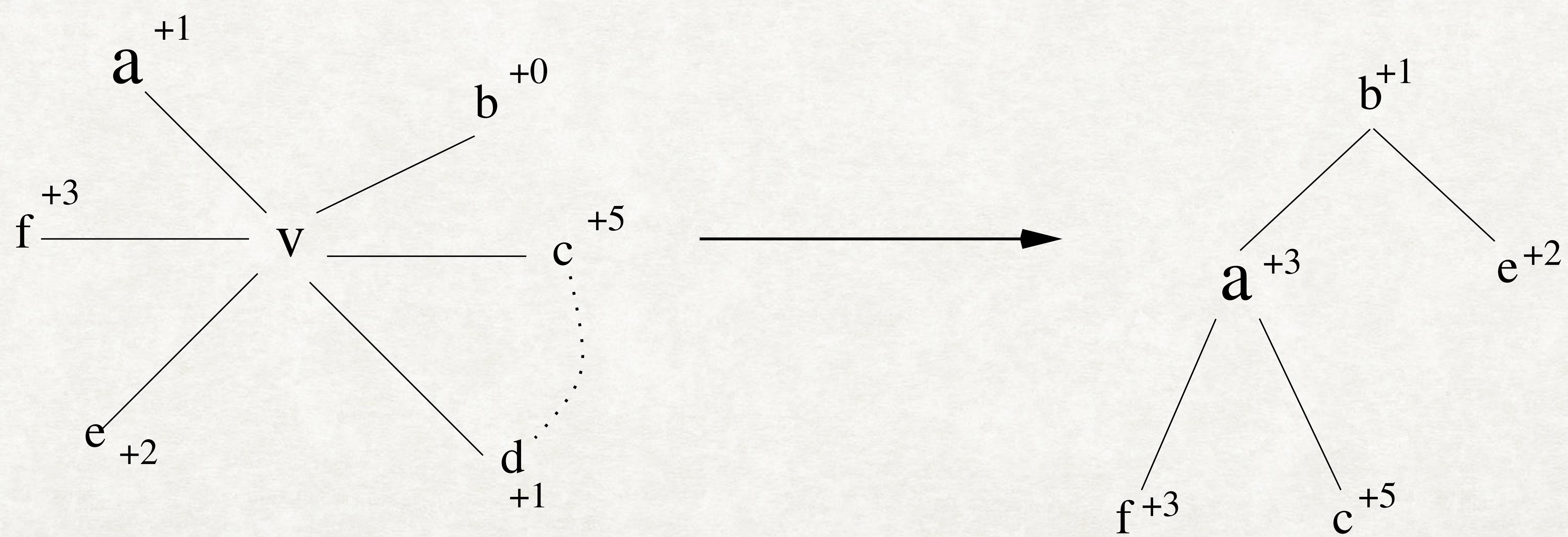
Algorithm Intuition:

- Keep track of load (degree increase) of nodes
- After each deletion, Insert a binary tree of neighbours of deleted node with more loaded nodes as leaves

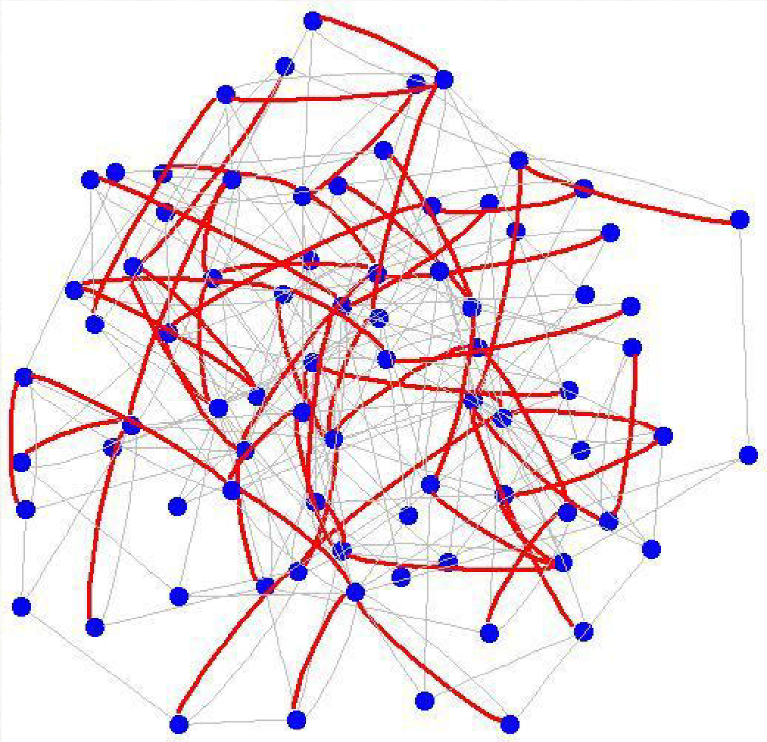
*Jared Saia, AT, Picking up the Pieces: Self-Healing in reconfigurable networks. IPDPS 2008

DASH: DEGREE ASSISTED SELF-HEALING

- Certain neighbours of the deleted node reconnect as a tree sorted on degree increase; degree of any vertex increases by at most 2 log n; no guarantees on diameter.



DASH: DEGREE ASSISTED SELF-HEALING



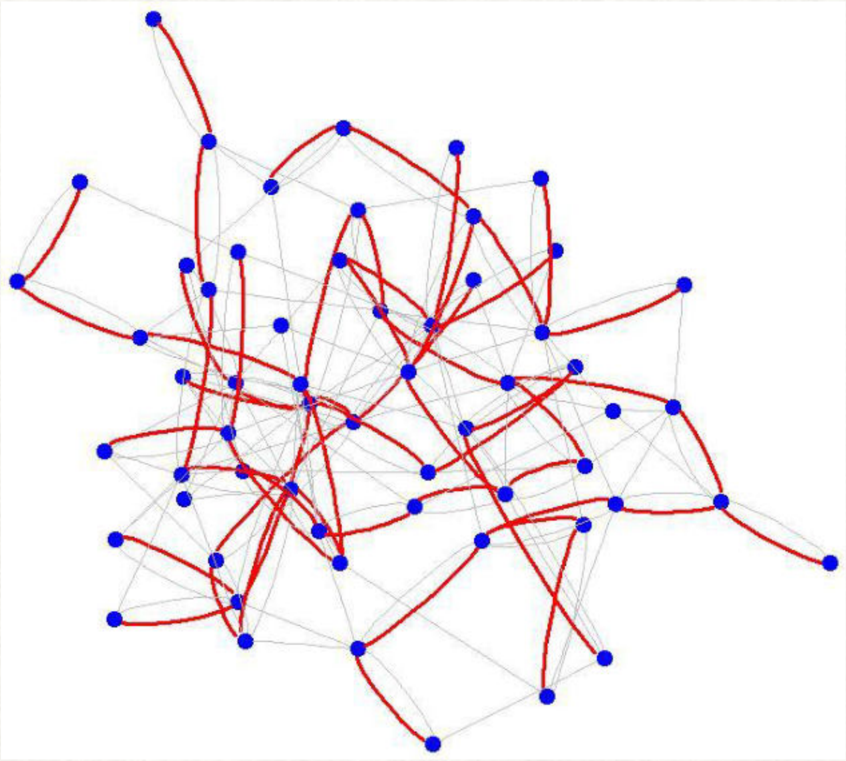
Healed Graph, n = 70; t =30

Algorithm:

- Keep track of load (degree increase) of nodes
- After each deletion, Insert a binary tree of neighbours of deleted node with more loaded nodes as leaves

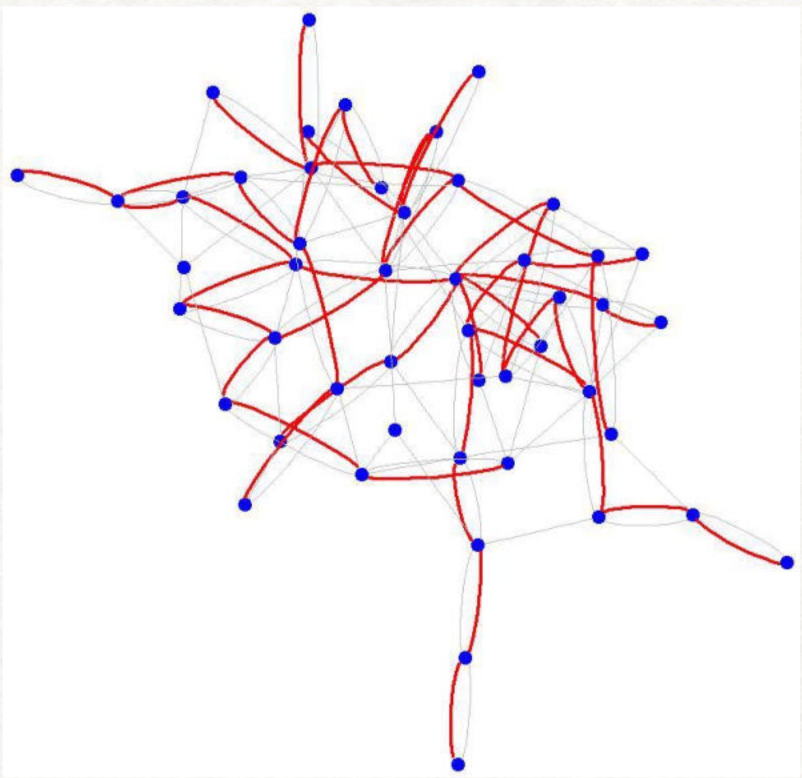
*Limits degree increase to O(log n) over series of deletions; empirical analysis of stretch over various attack strategies done.

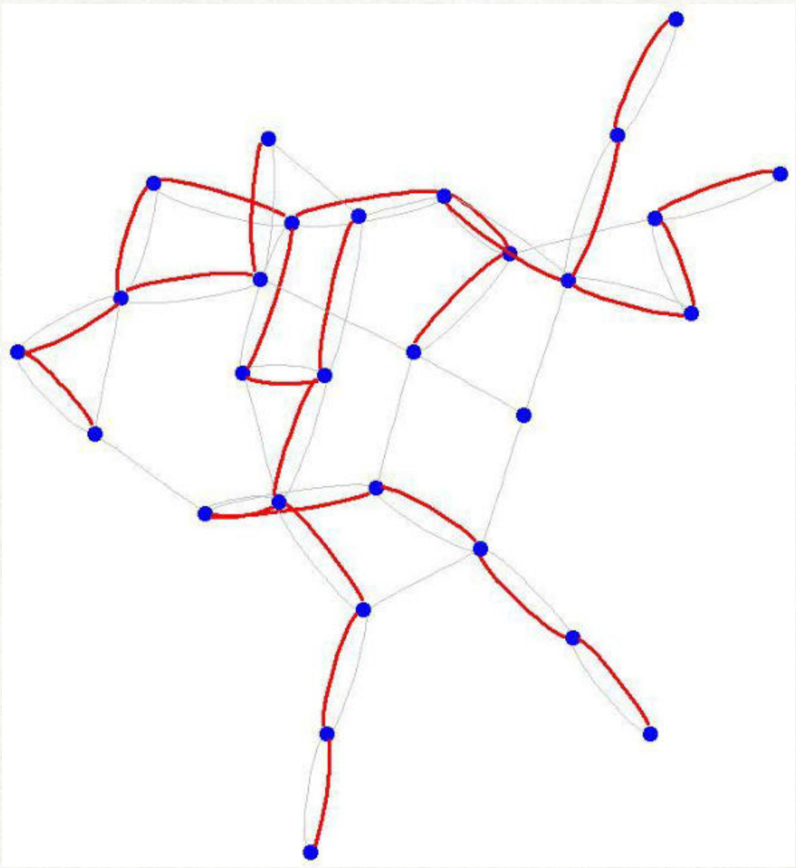
DASH: DEGREE ASSISTED SELF-HEALING

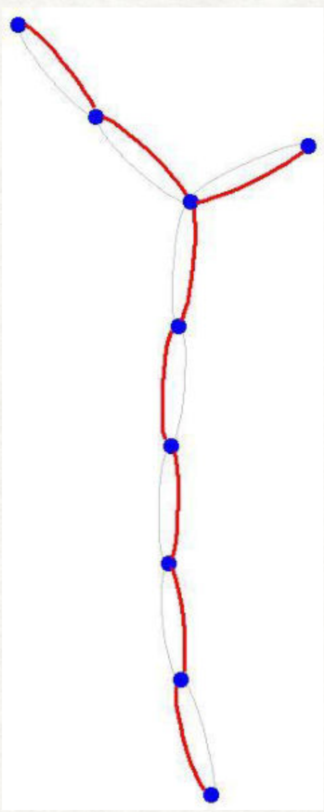


Graph, n = 50; t =50

DASH: DEGREE ASSISTED SELF-HEALING





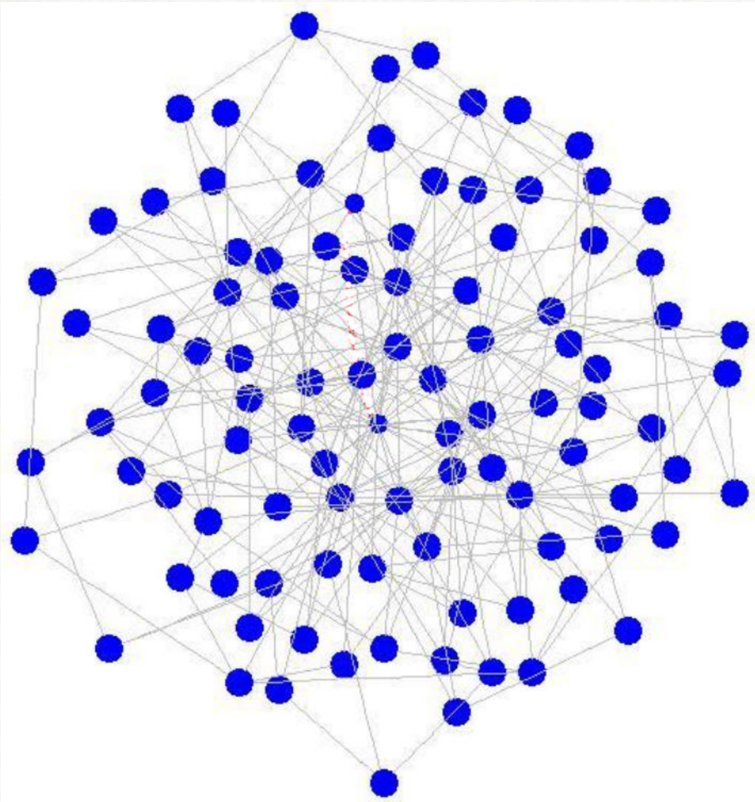


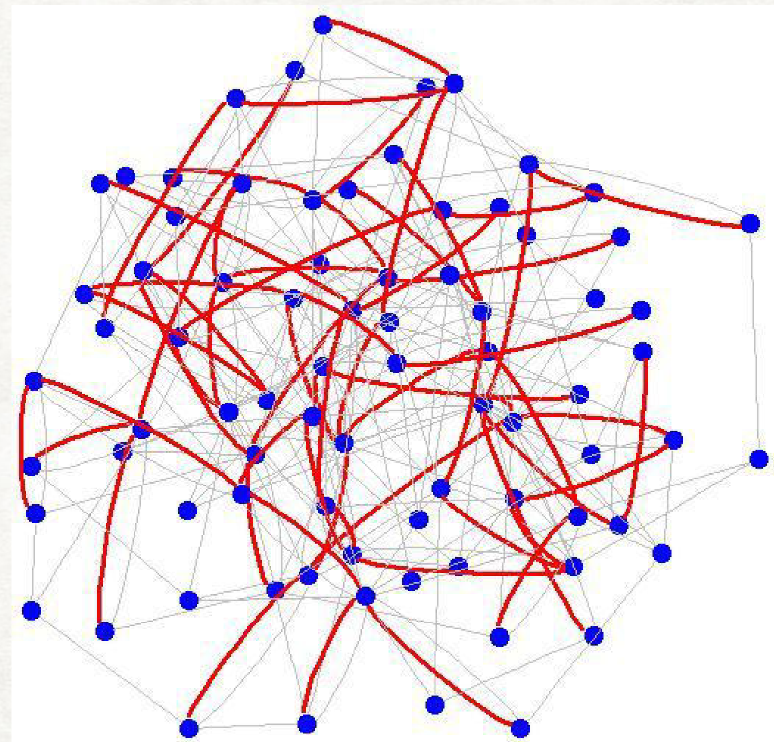
Graph, n = 30; t = 70

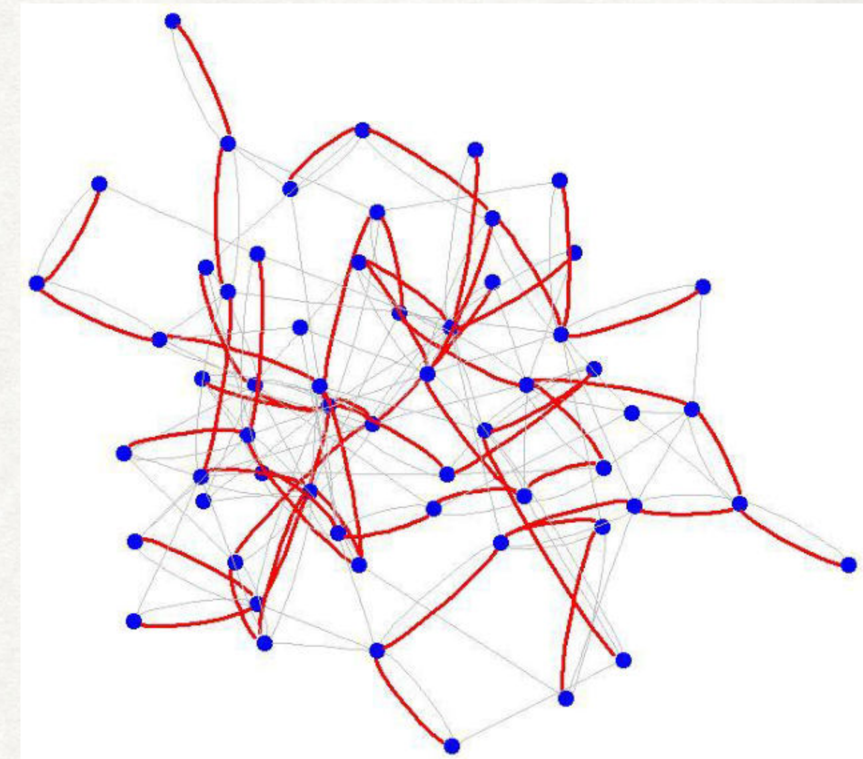
Graph, n = 20; t =80

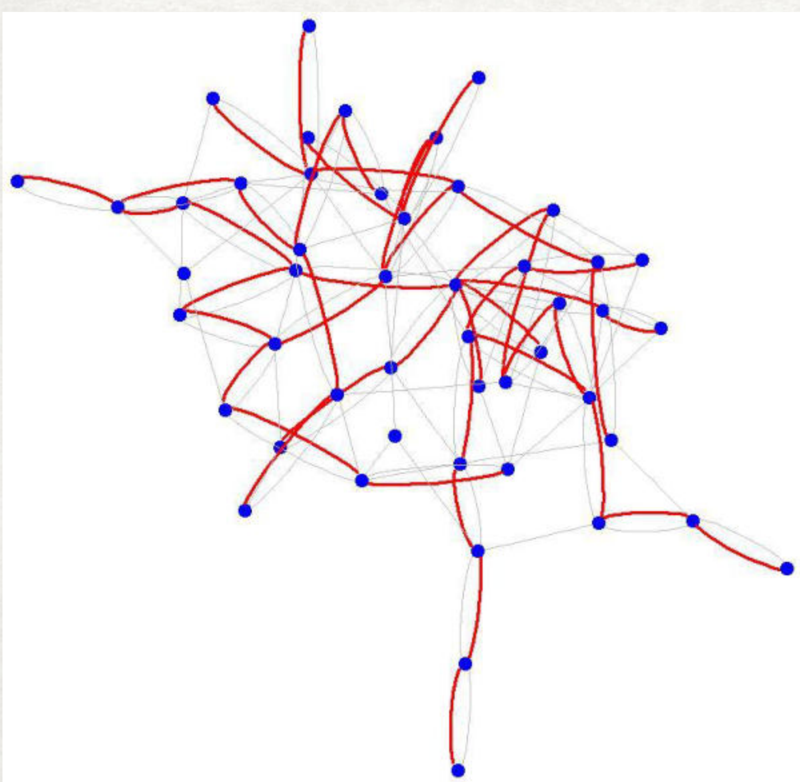
Graph, n = 10; t =90

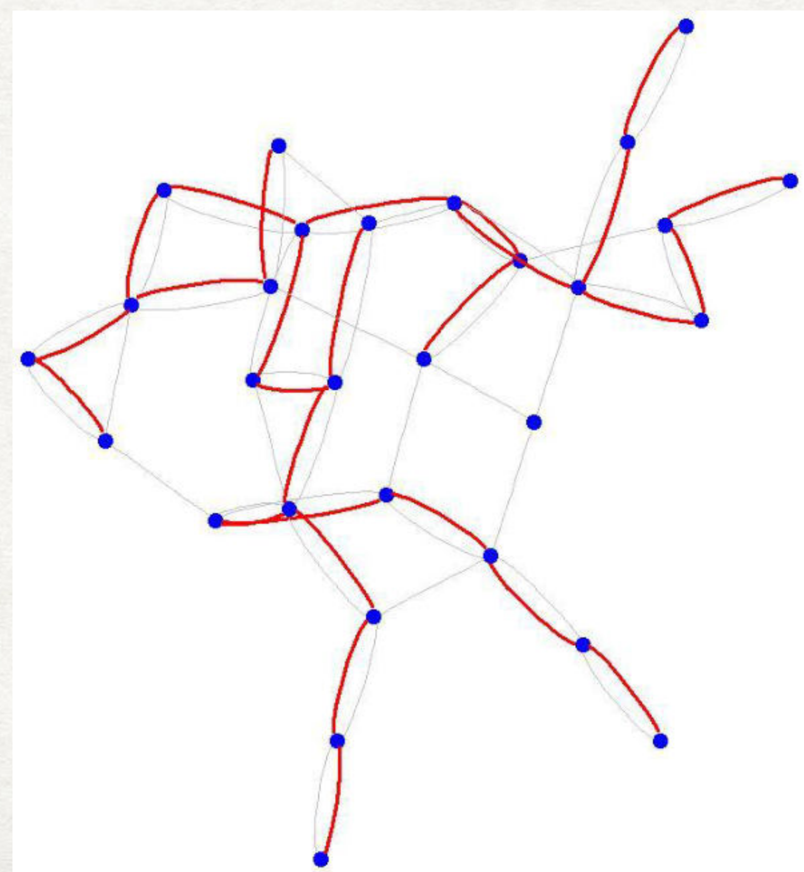
DASH: ALL TOGETHER

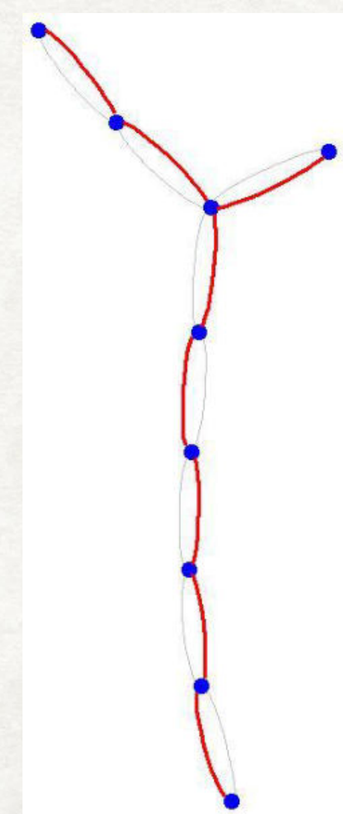




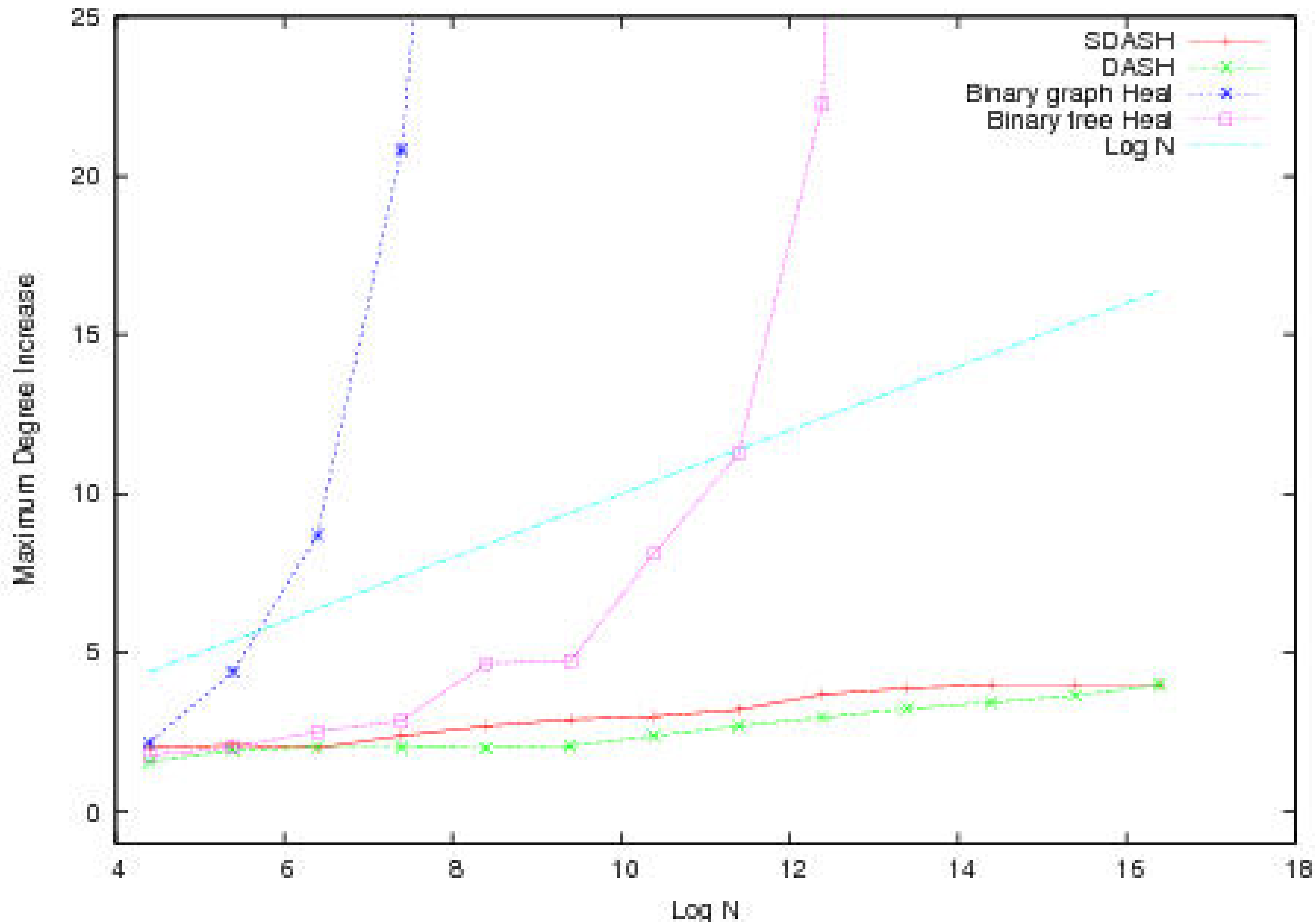






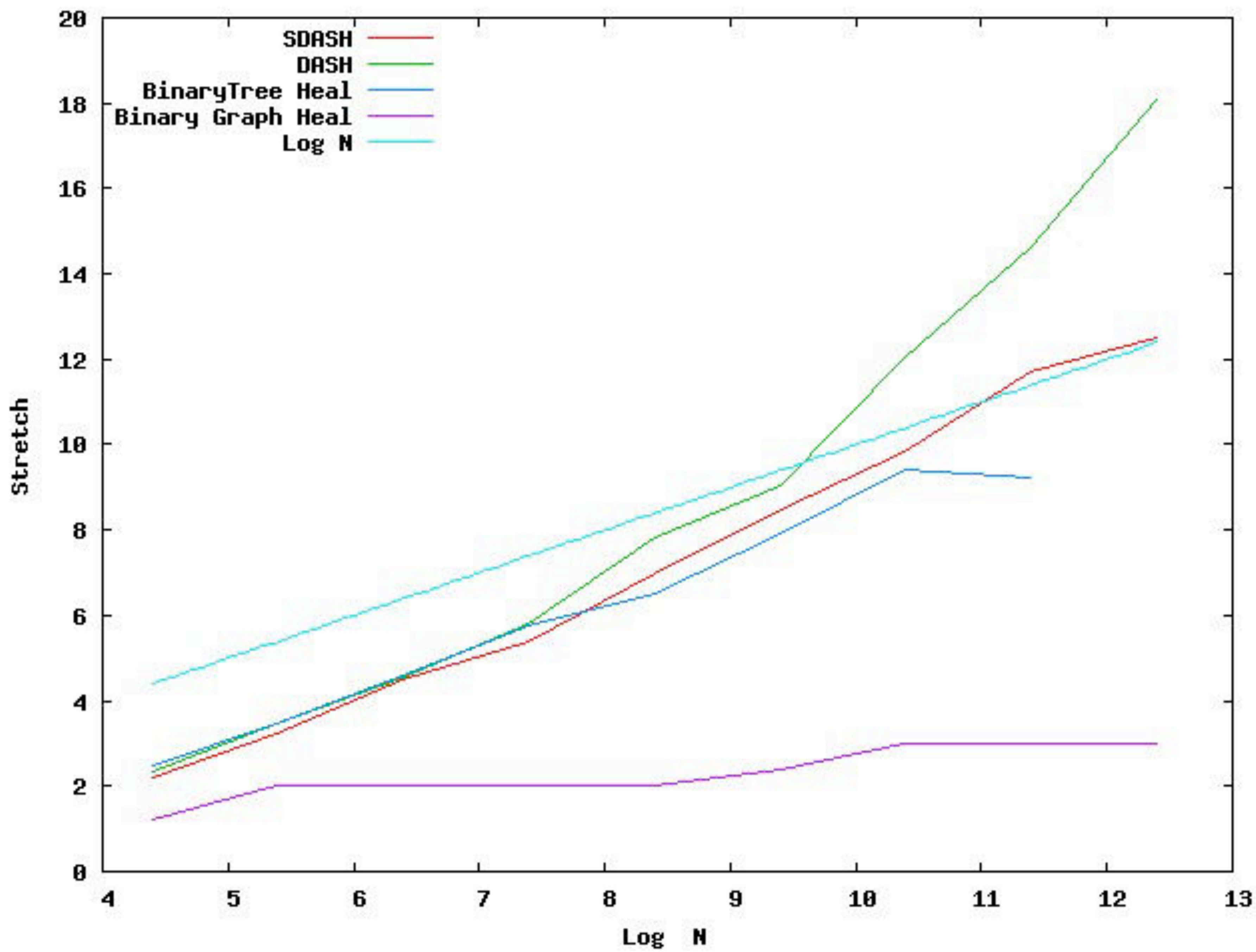


SIMULATIONS: DEGREE

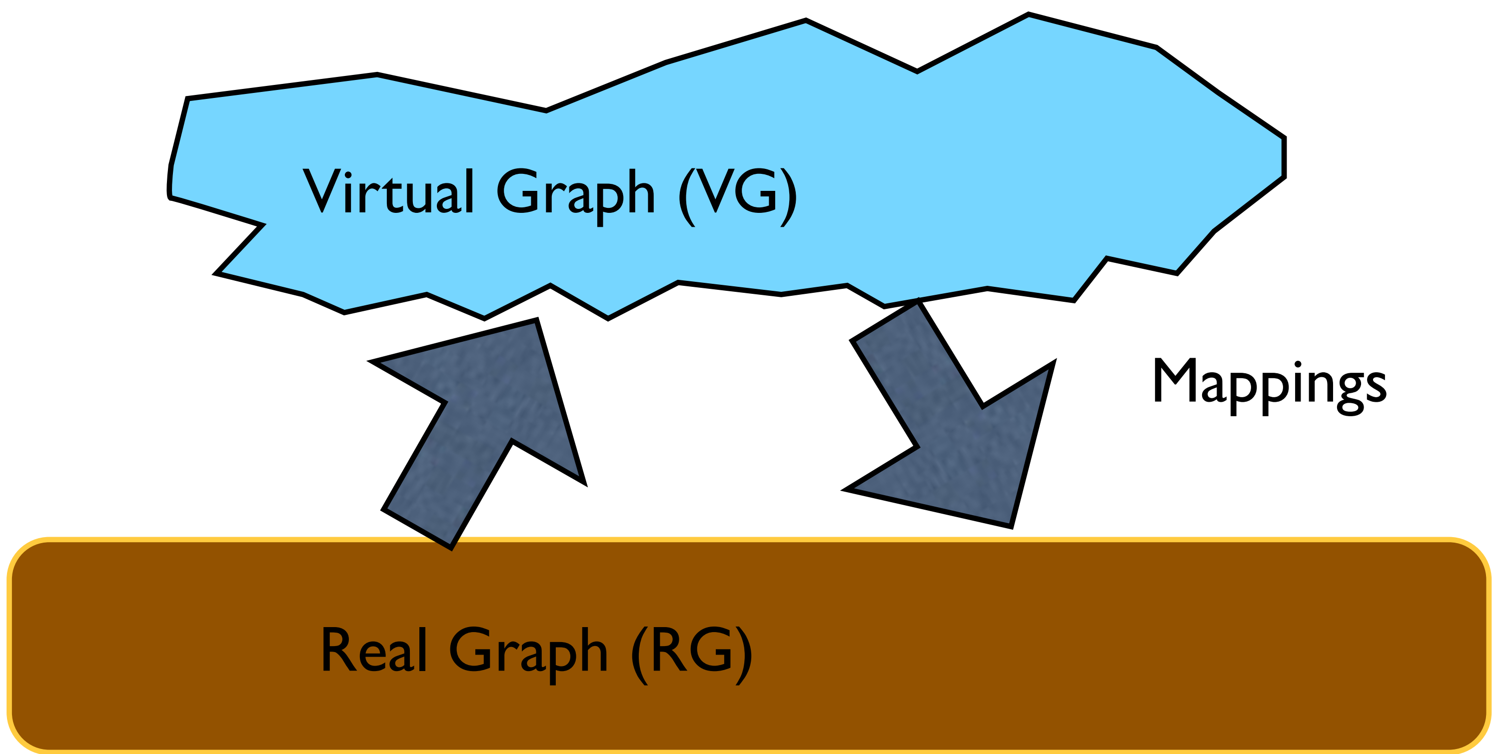


SIMULATIONS: STRETCH

Stretch: Maximum over all pairs of nodes u,v : Distance(G_t,u,v) / Distance (G_0, u,v)



VIRTUAL GRAPHS HEALING APPROACH

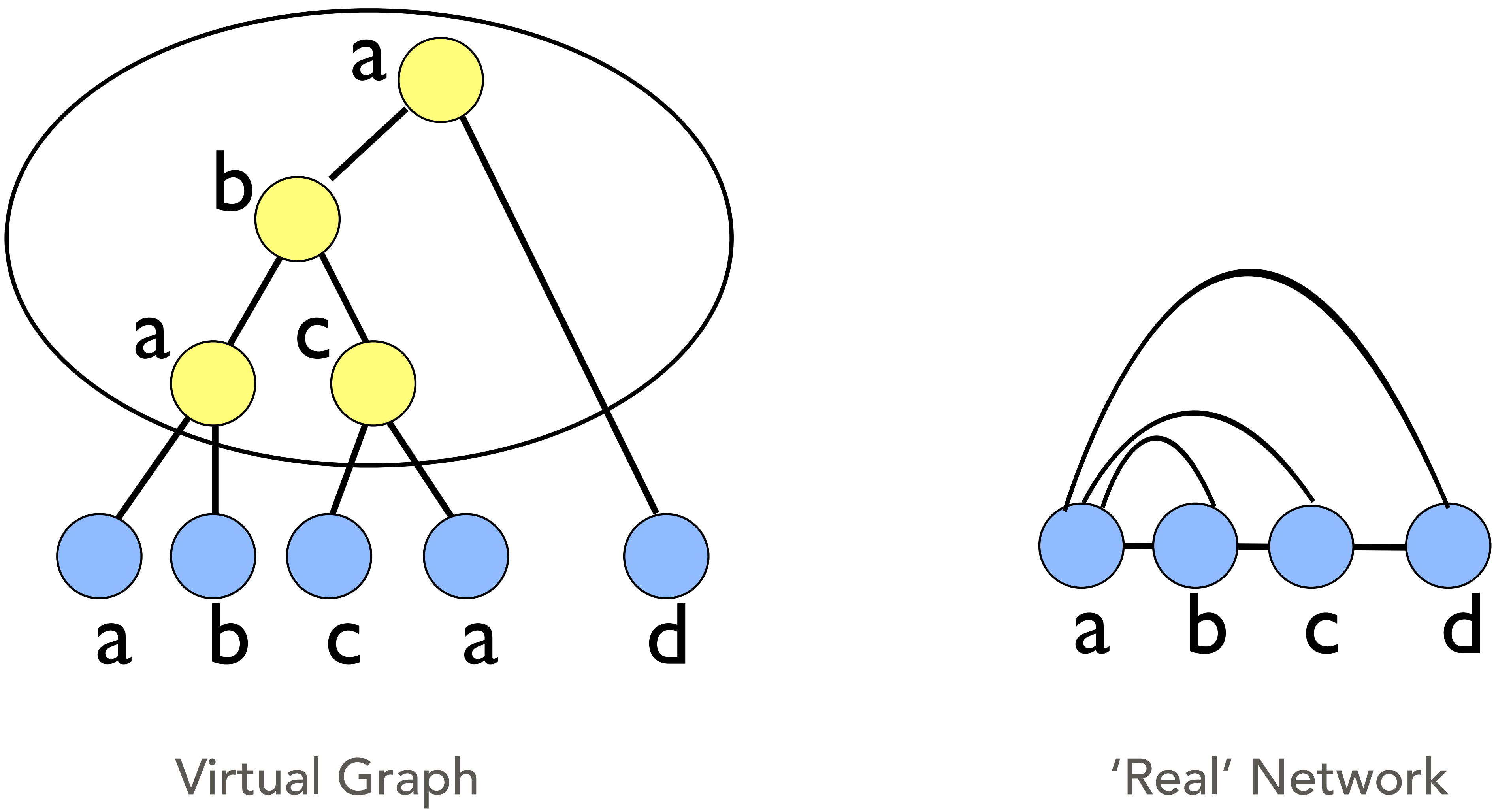


Method: Setup virtual graph (VG) on the real graph (RG). Maintain (self-heal) VG.

Required: If property A is maintained on VG, it is also maintained on RG (i.e. the correct mappings).

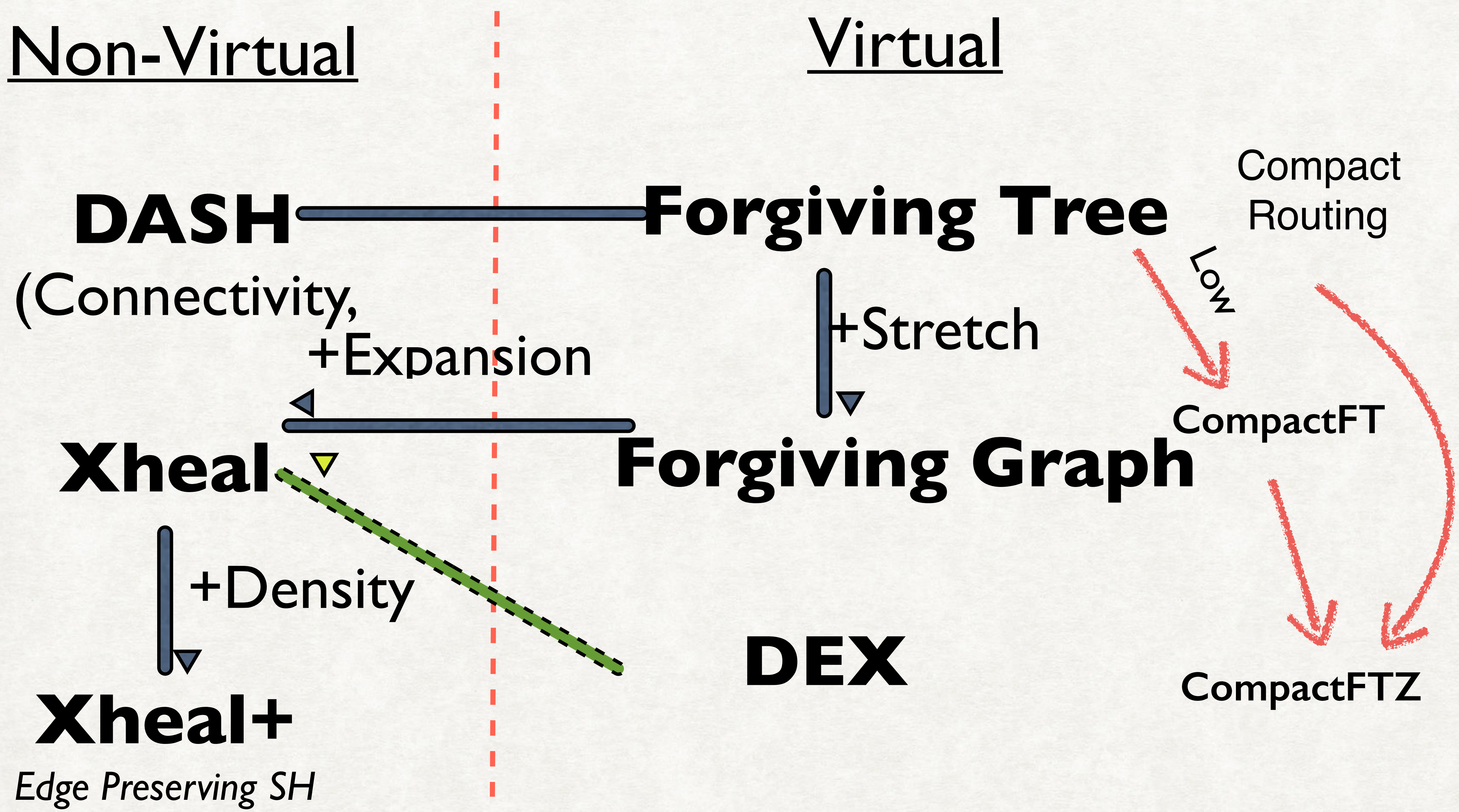
Homomorphism: Given $G_1 = (V_1, E_1), G_2 = V_2, E_2$

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



A virtual tree (left) and its homomorphic image (right)

OUR SELF-HEALING ALGORITHMS



FORGIVING TREE*

*Tom Hayes, Jared Saia, AT, The forgiving tree: a self-healing distributed data structure. PODC 2008

THE FORGIVING TREE: MODEL

- Start: a network G_0.
- 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

THE FORGIVING TREE: MAIN RESULT

- A distributed algorithm, Forgiving Tree such that, for any network G with max degree D, **for an arbitrary sequence of t deletions:**
- G_t stays connected
- Diameter(G_t) ≤ log(D). Diameter(G_0)
- For any node v in G_t, degree(G_t,v) ≤ degree(G_0,v) + 3
- Each repair takes constant time and involves O(D) nodes.

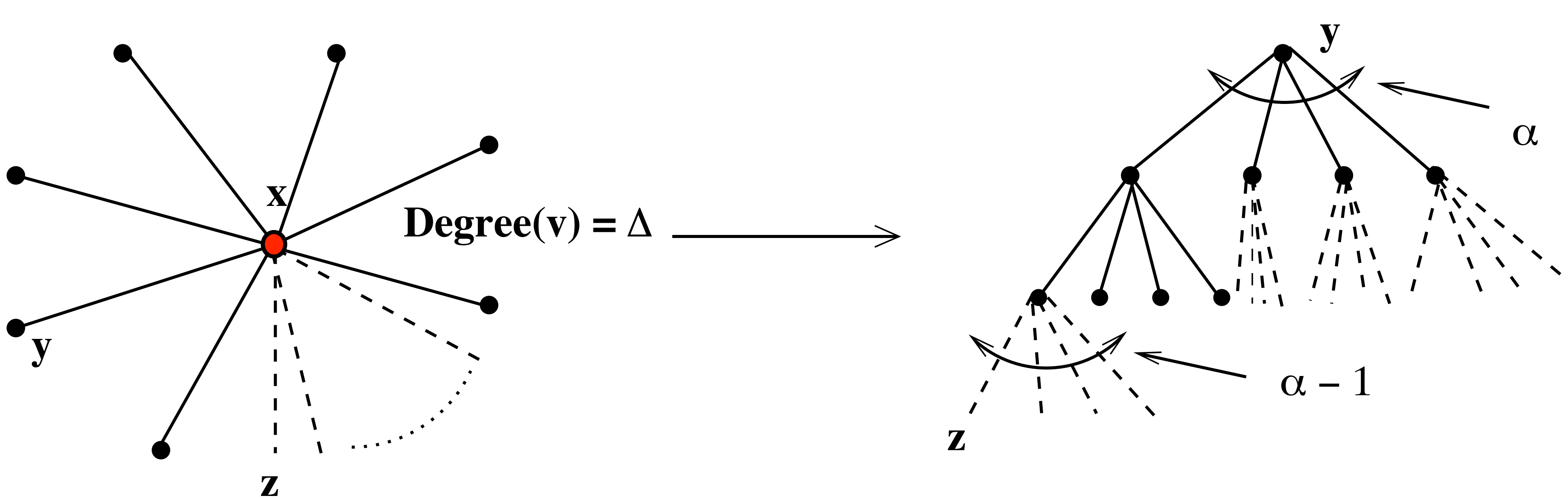
THE FORGIVING TREE: MAIN RESULT

- A distributed algorithm, Forgiving Tree such that, for any network G with max degree D, **for an arbitrary sequence of t deletions:**
- G_t stays connected
- Diameter(G_t) ≤ log(D). Diameter(G_0)
- For any node v in $G_{t,}$ degree($G_{t,}$v) ≤ degree($G_{0,}$v) + 3
- Each repair takes constant time and involves O(D) nodes.

} Matching lower bound

THE LOWER BOUND

- Adversary can force, for any self-healing algorithm:
  - Degree increase $\leq \alpha \Rightarrow$ stretch of $\Omega(\log_\alpha(n-1))$



G_T

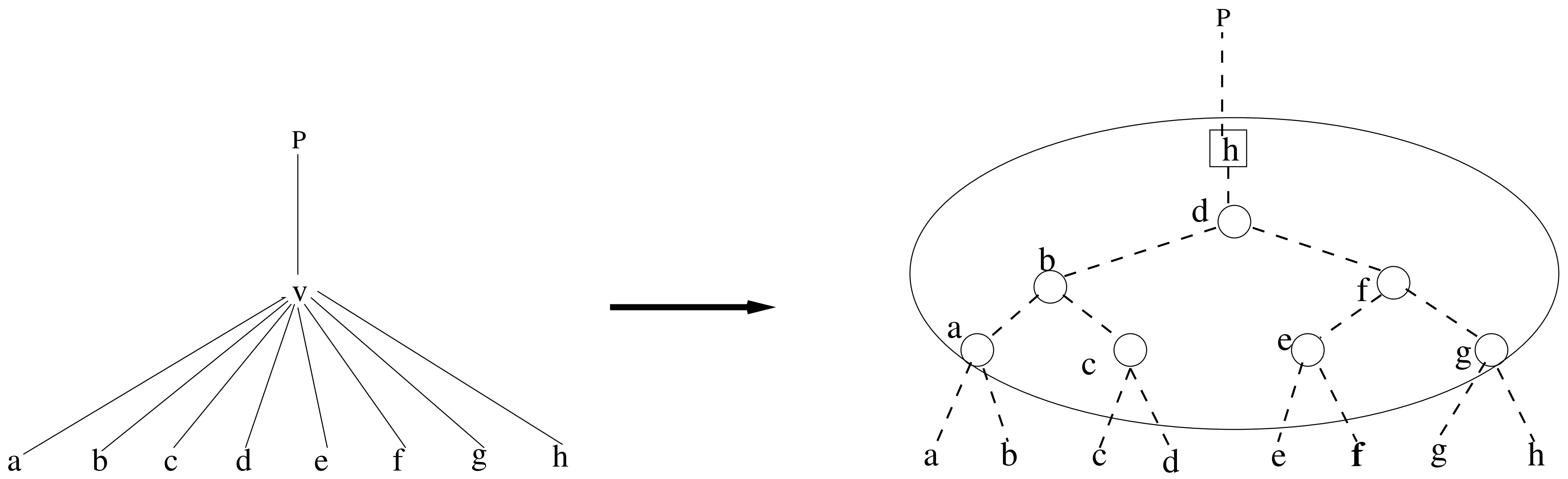
BFS Tree of y in G_{T+1}

THE FORGIVING TREE: MOTIVATIONS

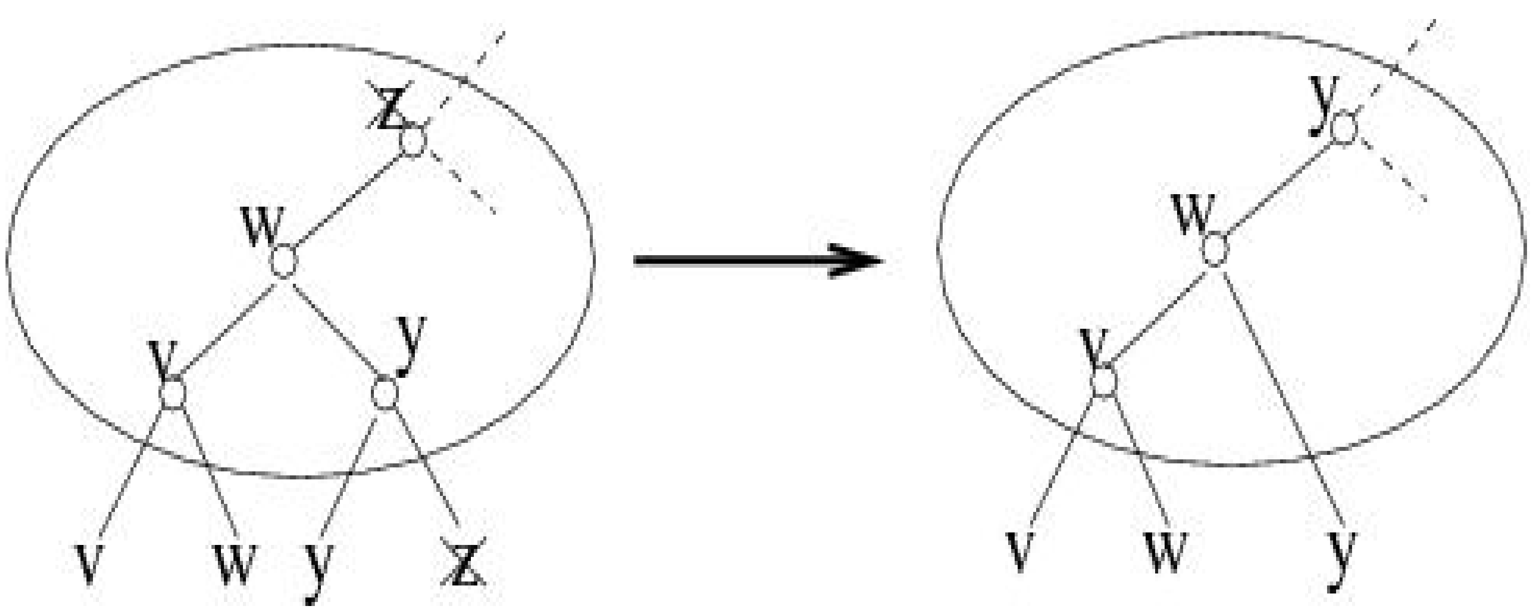
- Trees are the "worst case" for maintaining connectivity. Suppose we are given one.
- Our algorithm is based on maintaining a virtual tree. This helps us keep track of which vertices can afford to have their degrees increased, and also avoid blowing up distances.

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



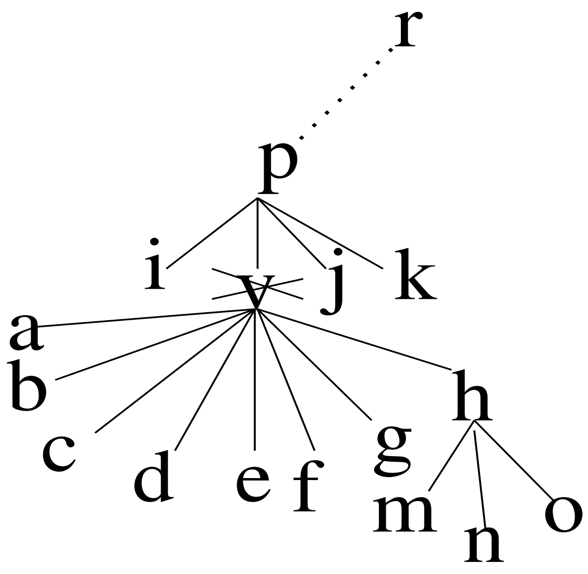
Replacing v by a balanced binary tree of virtual nodes

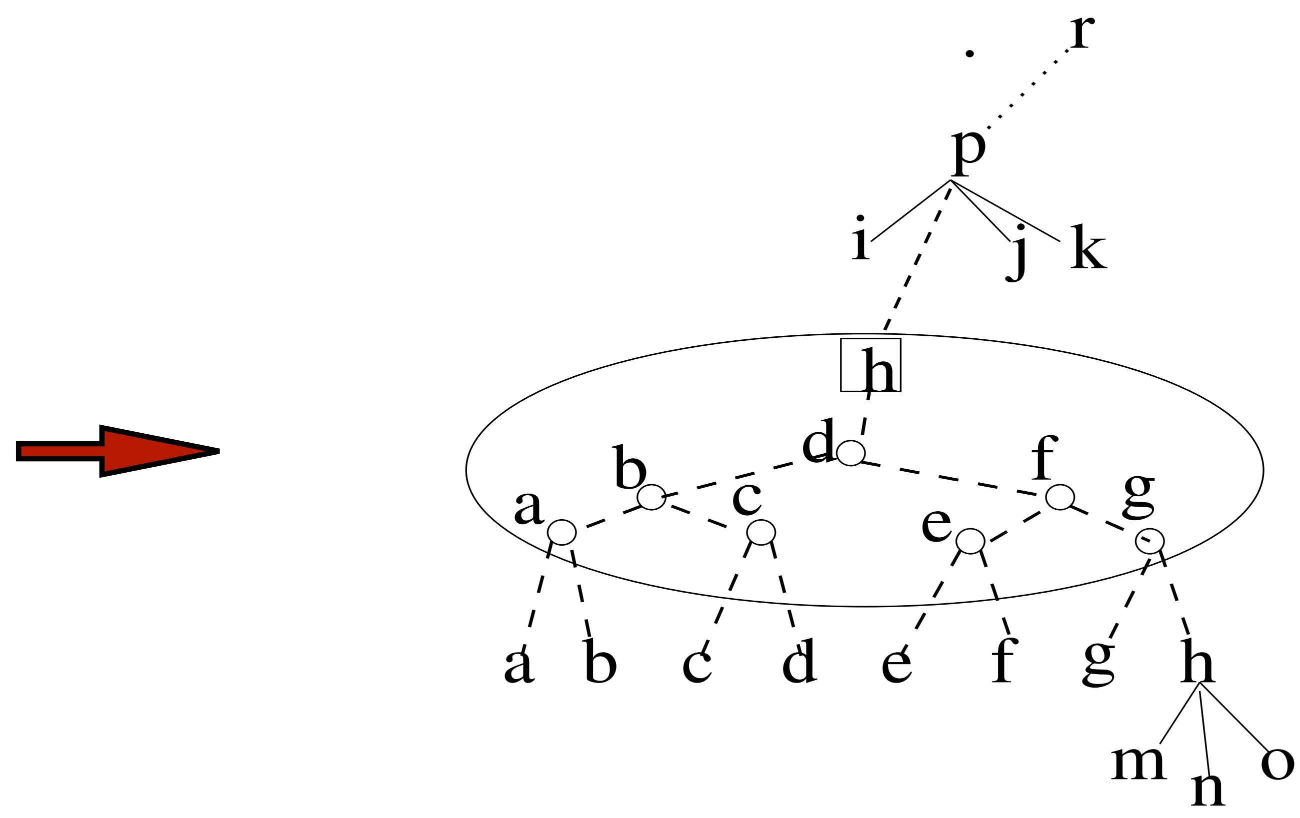


Short-circuiting a redundant virtual node

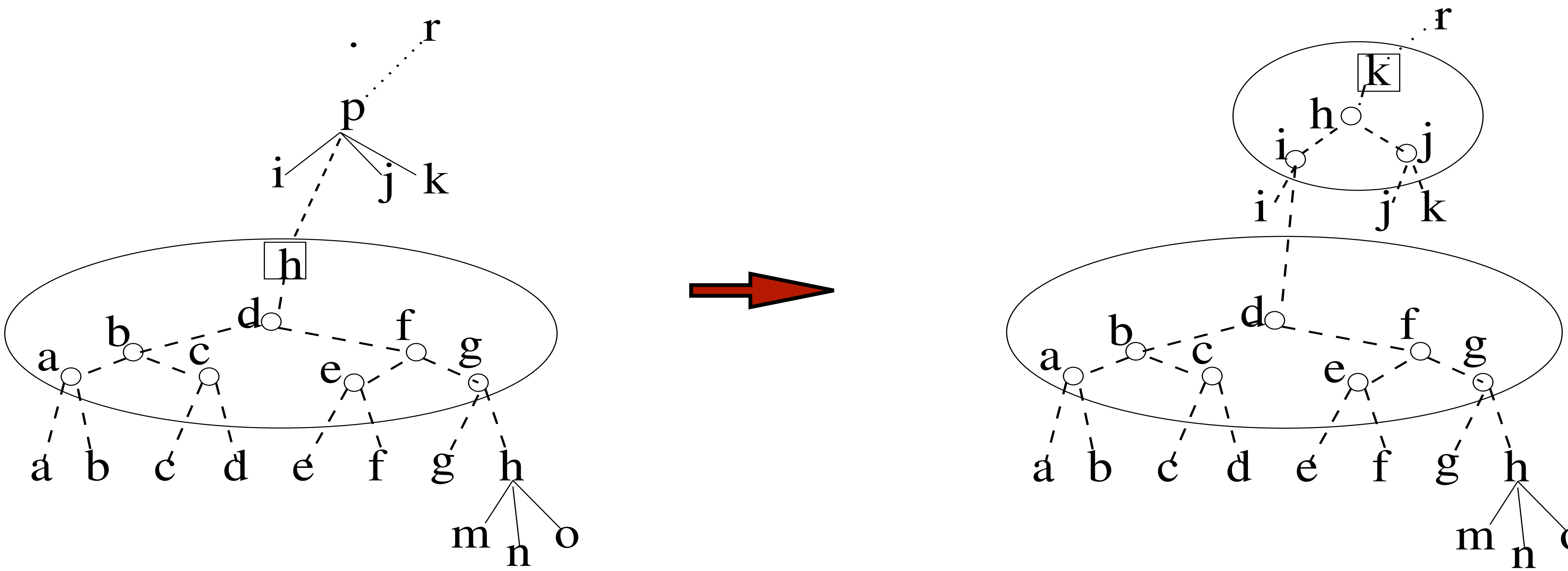
Algorithm in action

Node v deleted:

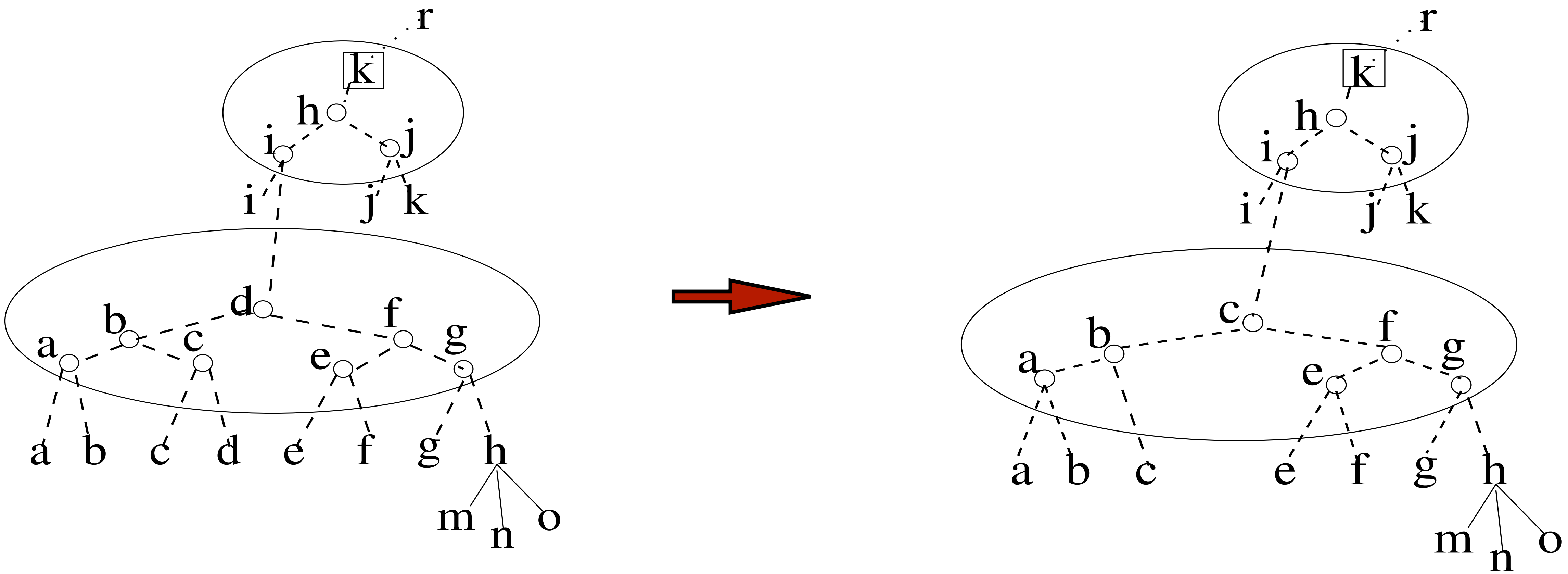




Node p deleted:



Node d deleted:



THE FT: MAIN RESULT INTUITION

- A distributed algorithm, Forgiving Tree such that, for any network G with max degree D, **for an arbitrary sequence of t deletions:**
- G_t stays connected: *Since the healing graph is connected*
- Diameter(G_t) ≤ log(D). Diameter(G_0): *The largest healing binary tree is on D nodes and never increases!*
- For any node v in G_t, degree(G_t, v) ≤ degree(G_0, v) + 3: *Every real node simulates at most one virtual node!*
- Each repair takes constant parallel time and involves O(D) nodes: *By the wills mechanism (not discussed)*

FORGIVING GRAPH*

*Tom Hayes, Jared Saia, AT, The Forgiving Graph: A distributed data structure for low stretch under adversarial attack. PODC 2009, Distributed Computing 2012

FORGIVING GRAPH (FG)

- FG extends Forgiving Tree:
  - Fully dynamic: has both insertions and deletions
  - Bounds the stronger property of stretch (as opposed to diameter only)
  - More complex and slower than Forgiving Tree

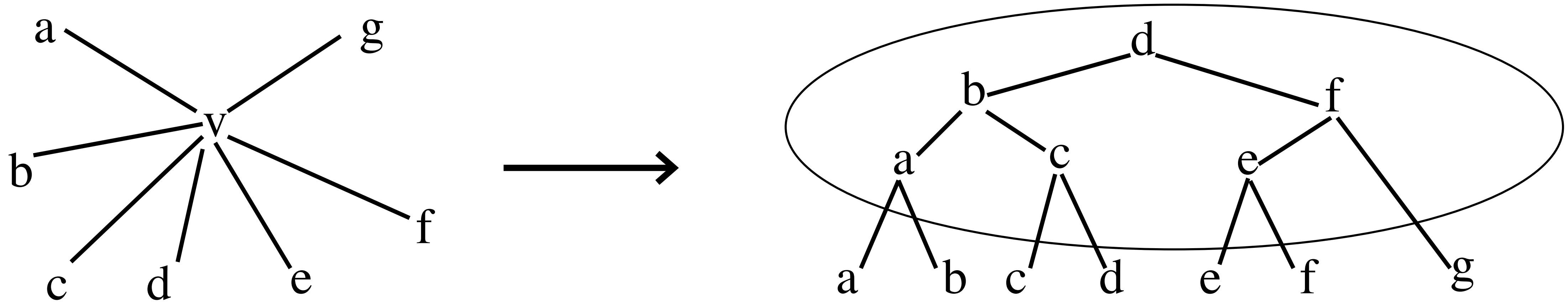
FORGIVING GRAPH: INSERTIONS PERMITTED

Big question:
How to analyse a self-healing algorithm which has insertions?

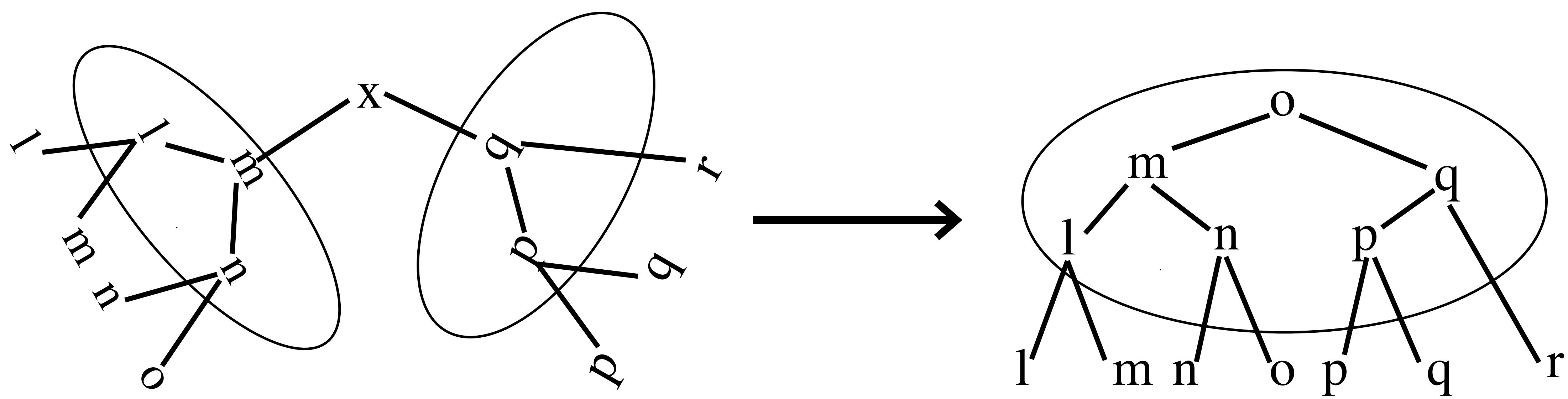
Hint: *What can G_0 look like?*

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.

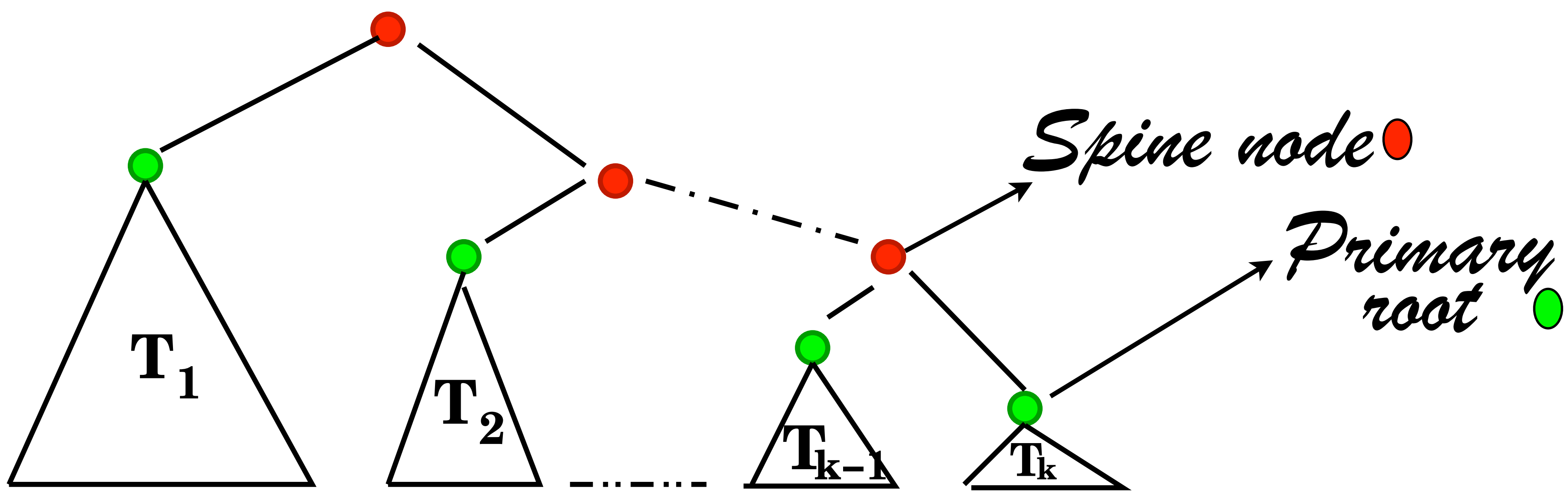


Merging two reconstruction trees on deletion of x

ANALYSIS FROM 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.

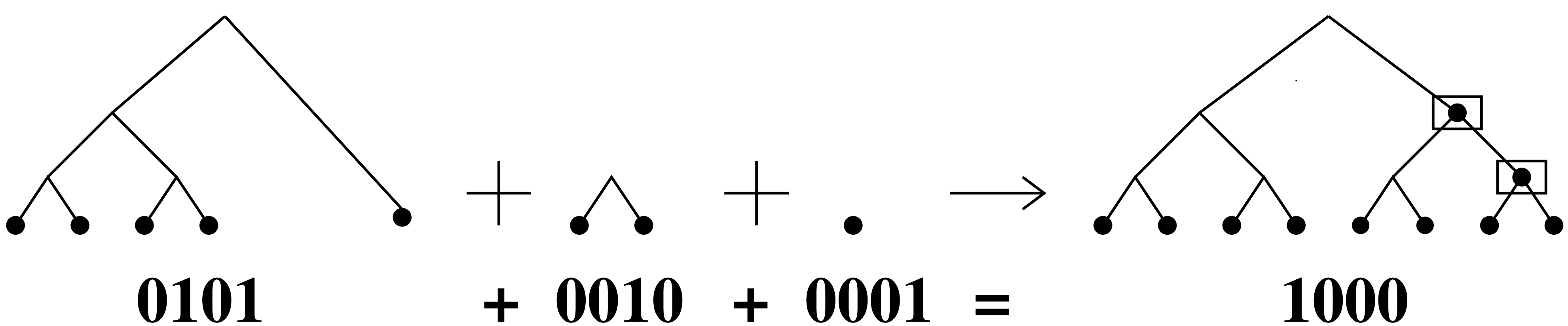
HALF-FULL TREES (HAFTS)



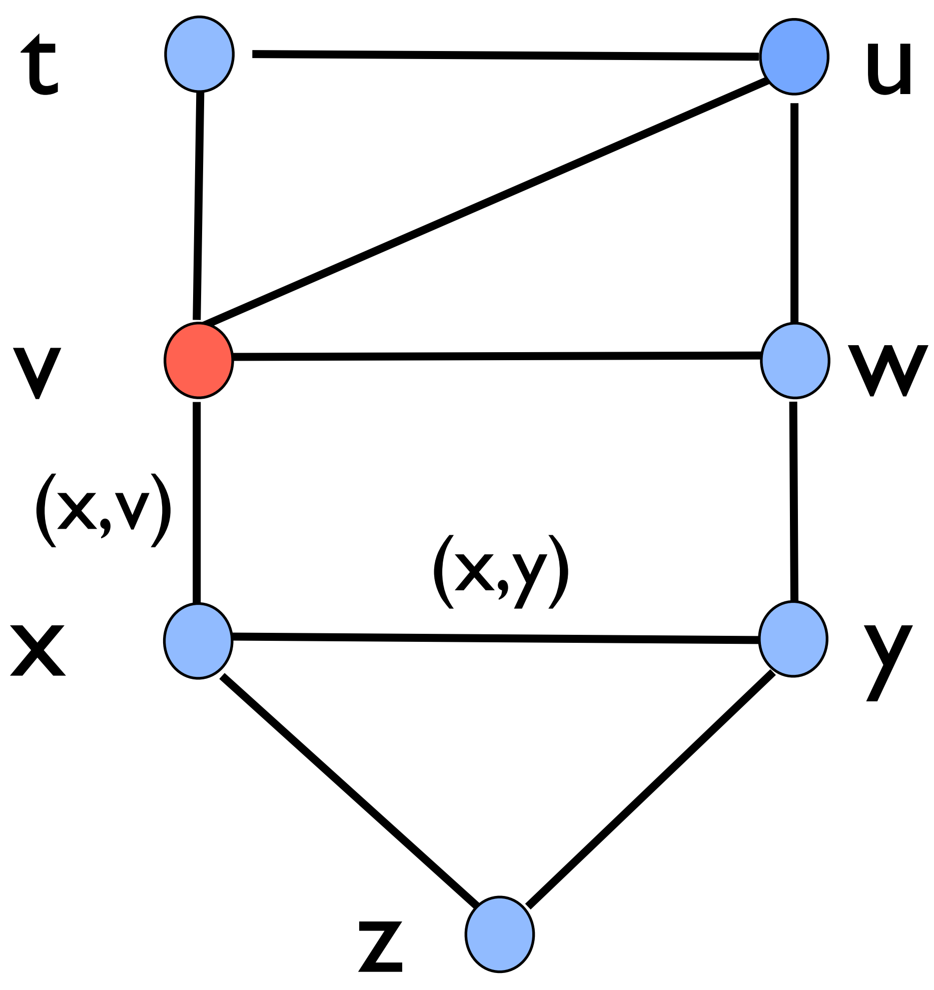
- A rooted binary tree in which every non-leaf node *v* has the following properties:
  - v has exactly two children.
  - The left child of v is the root of a complete binary subtree containing at least half of v's children.

OPERATIONS ON HAFTS

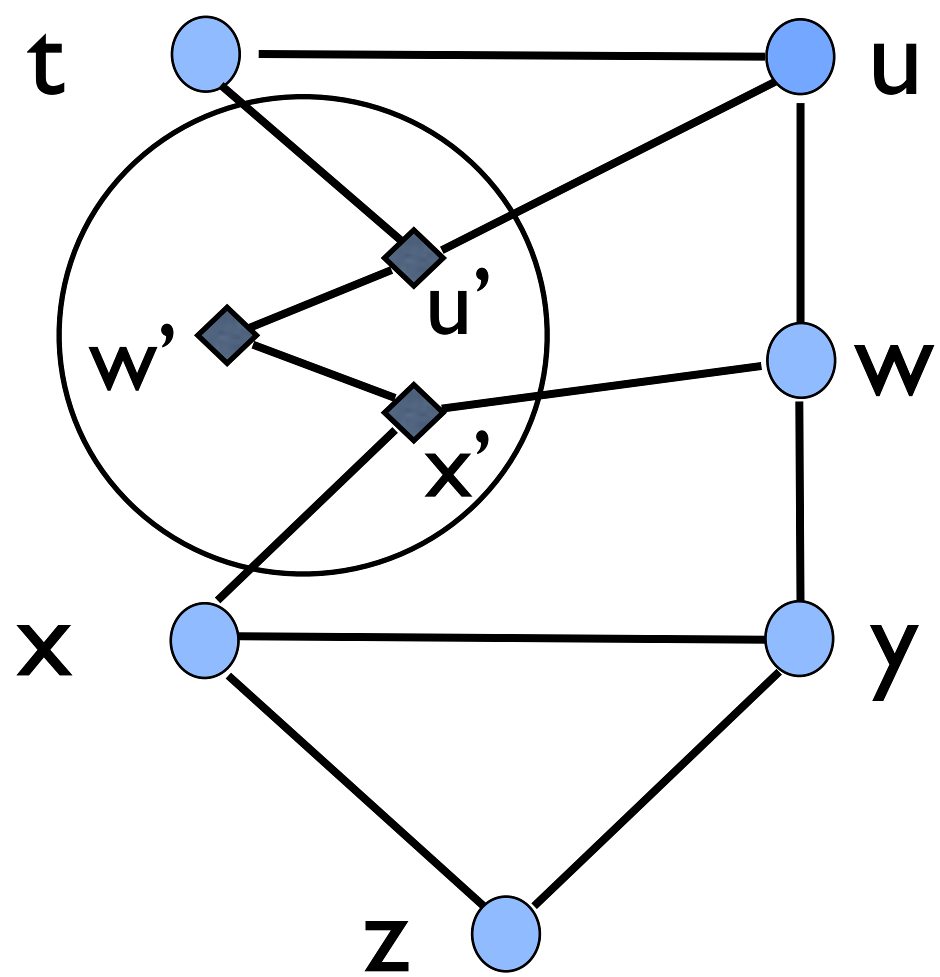
- Merge: Recombine hafts to make new haft. Analogous to binary addition.
  - Strip to get forest of complete trees.
  - Join adjacent trees with a new node as root, larger tree as left child.



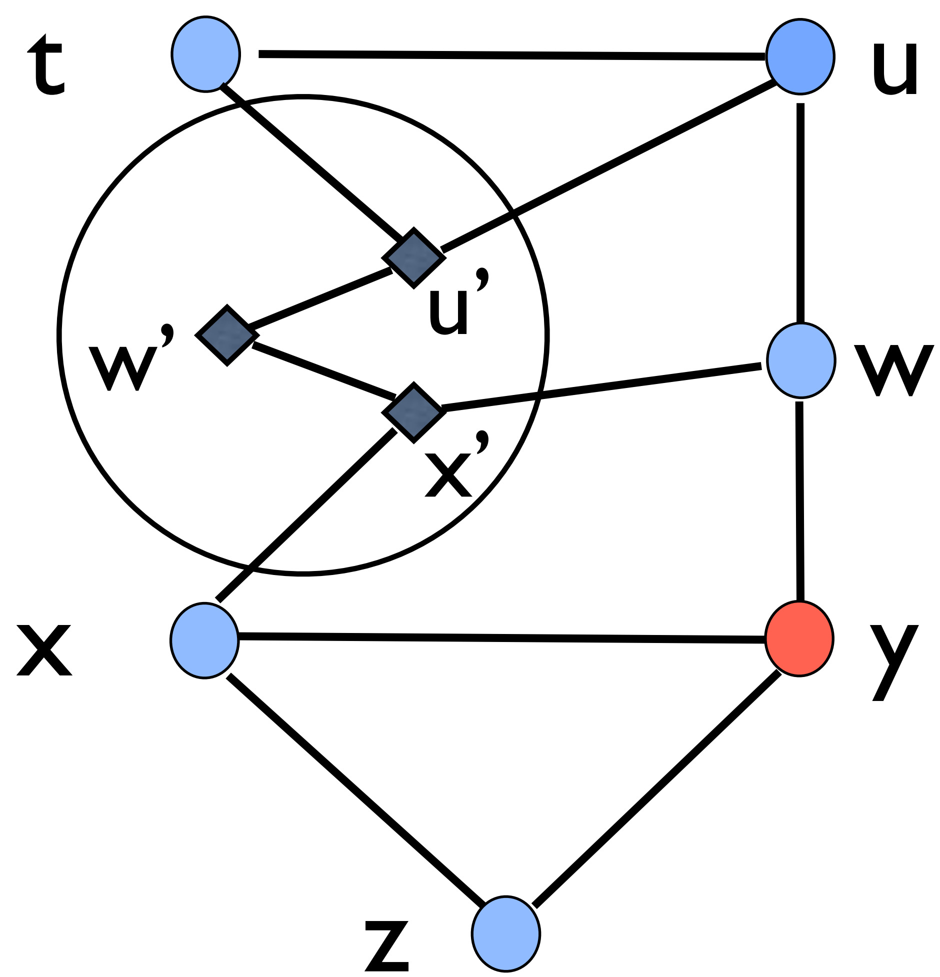
FG IN ACTION



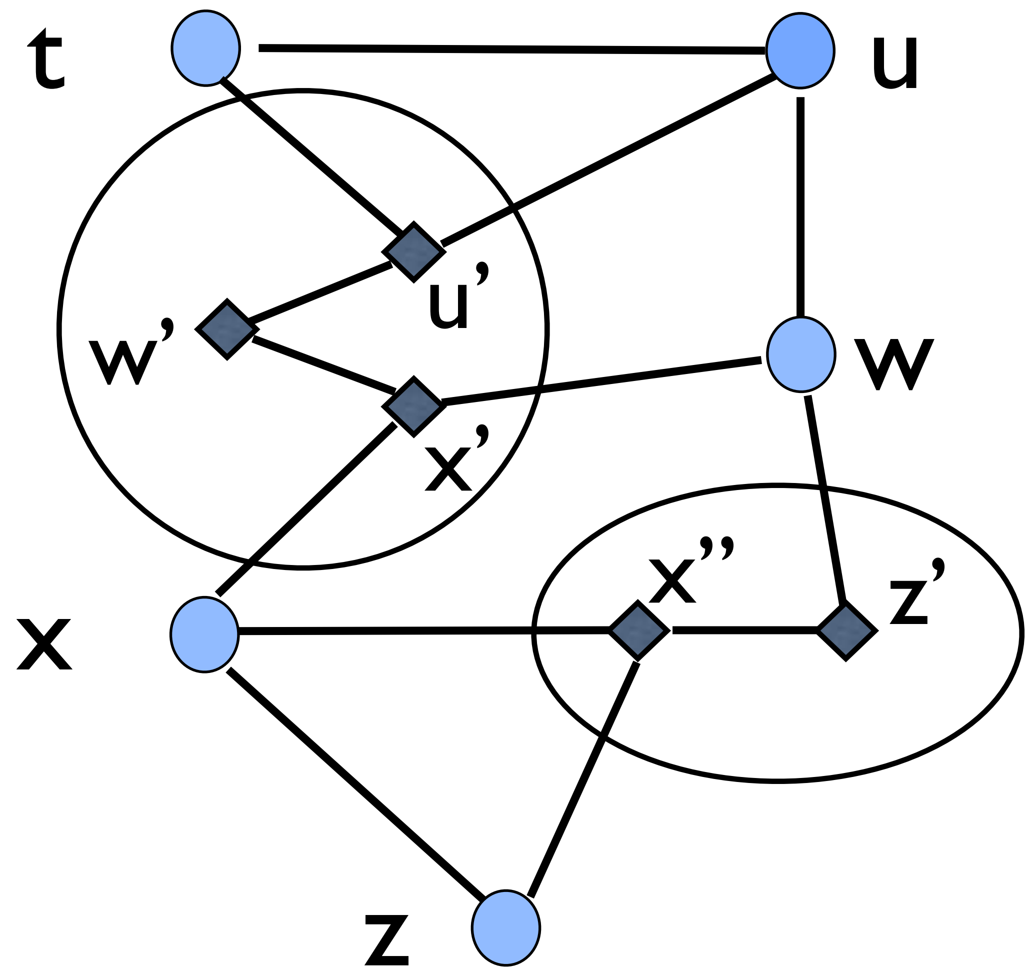
Node v deleted ...



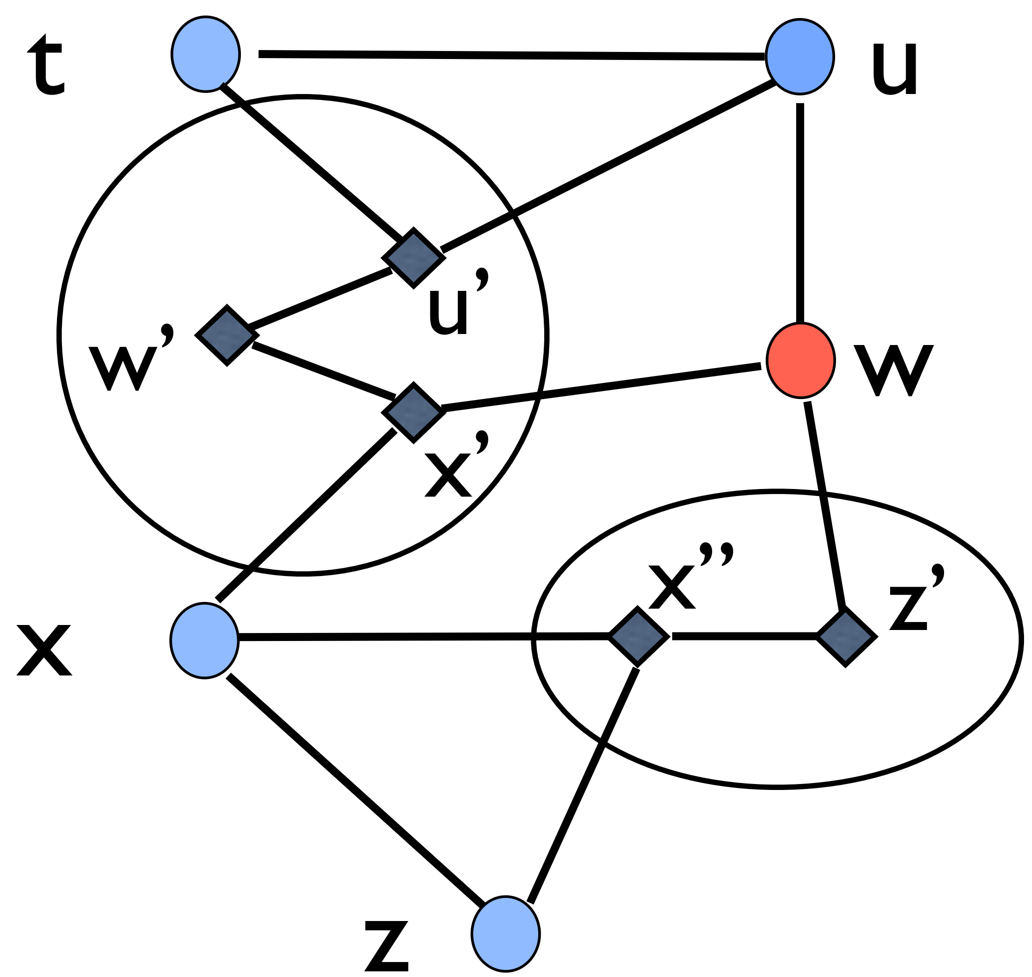
replaced by RT(v)



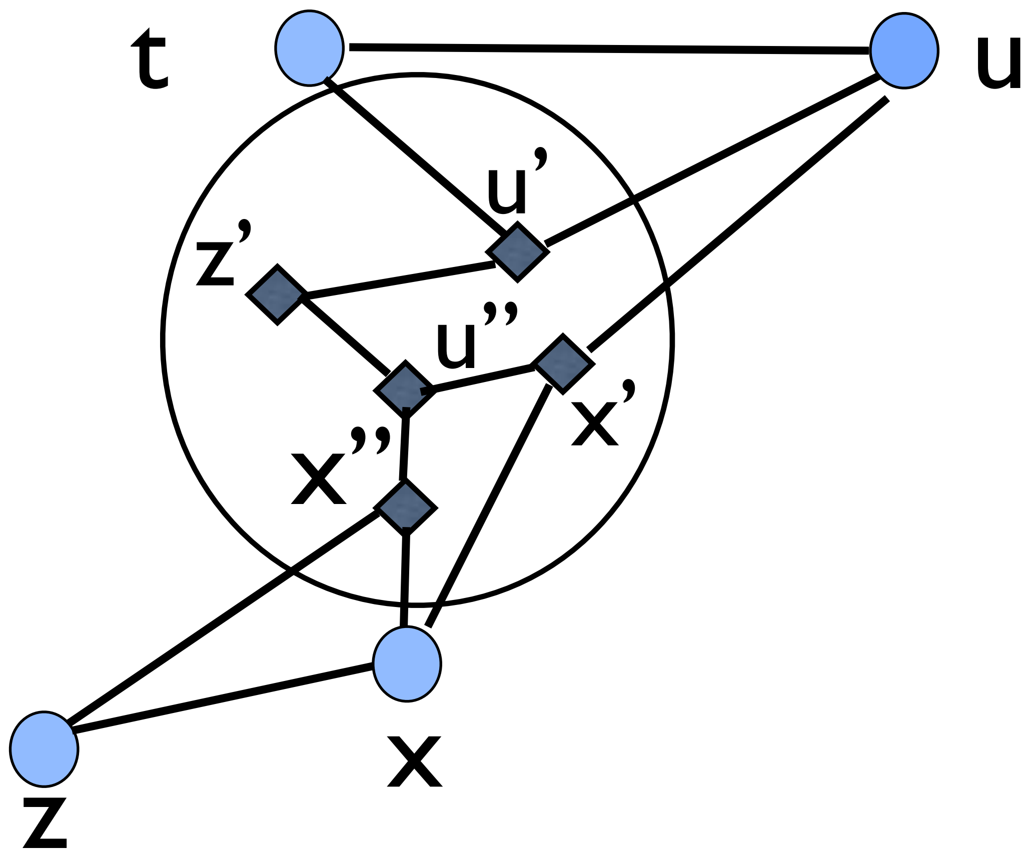
Node y deleted...



replaced by RT(y)

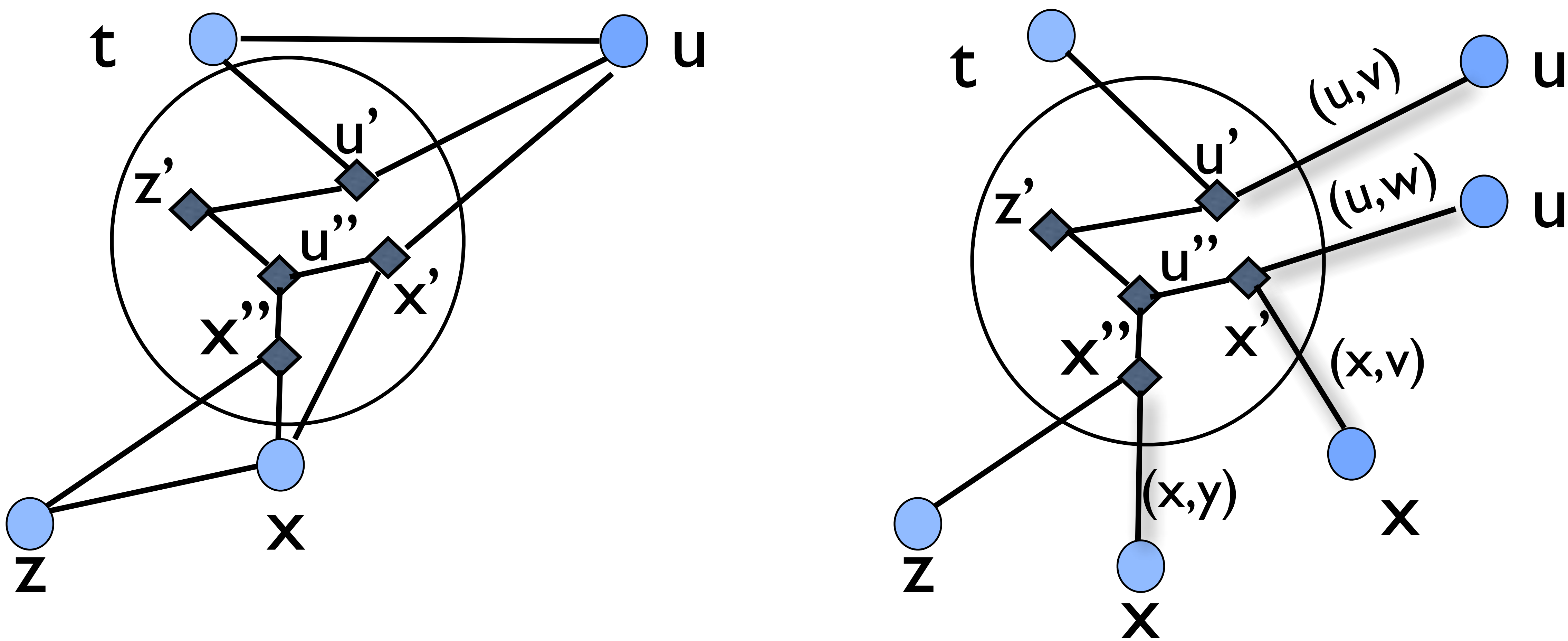


Node w deleted...



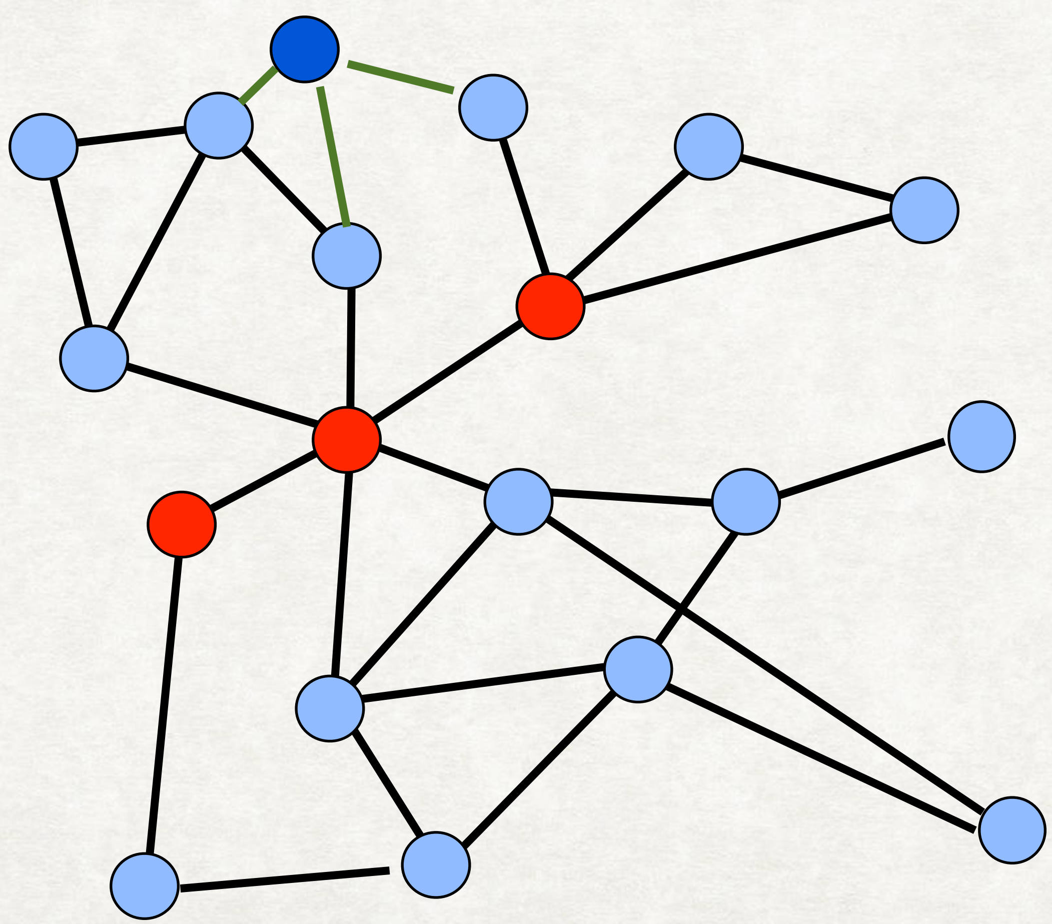
RT(v), RT(w) and u merge.

WHERE'S THE HAFT?

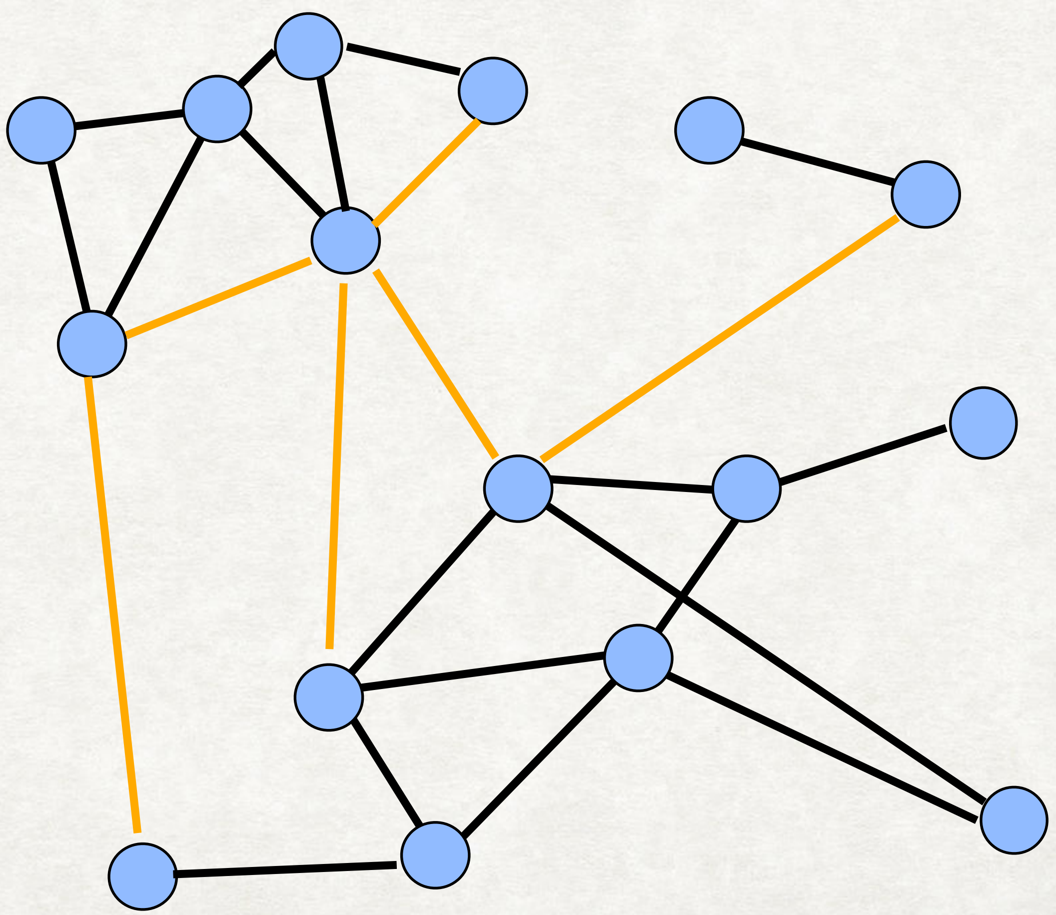


COMPARING RESULTS

G′: graph of only insertions and original nodes

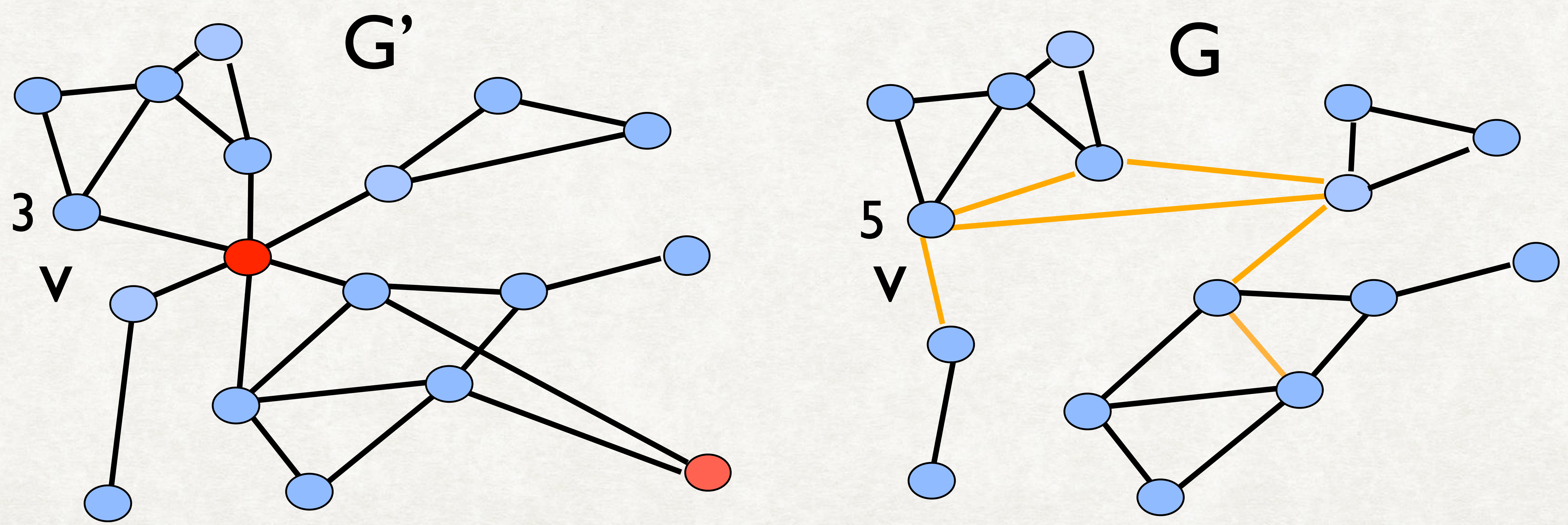


G: healed network



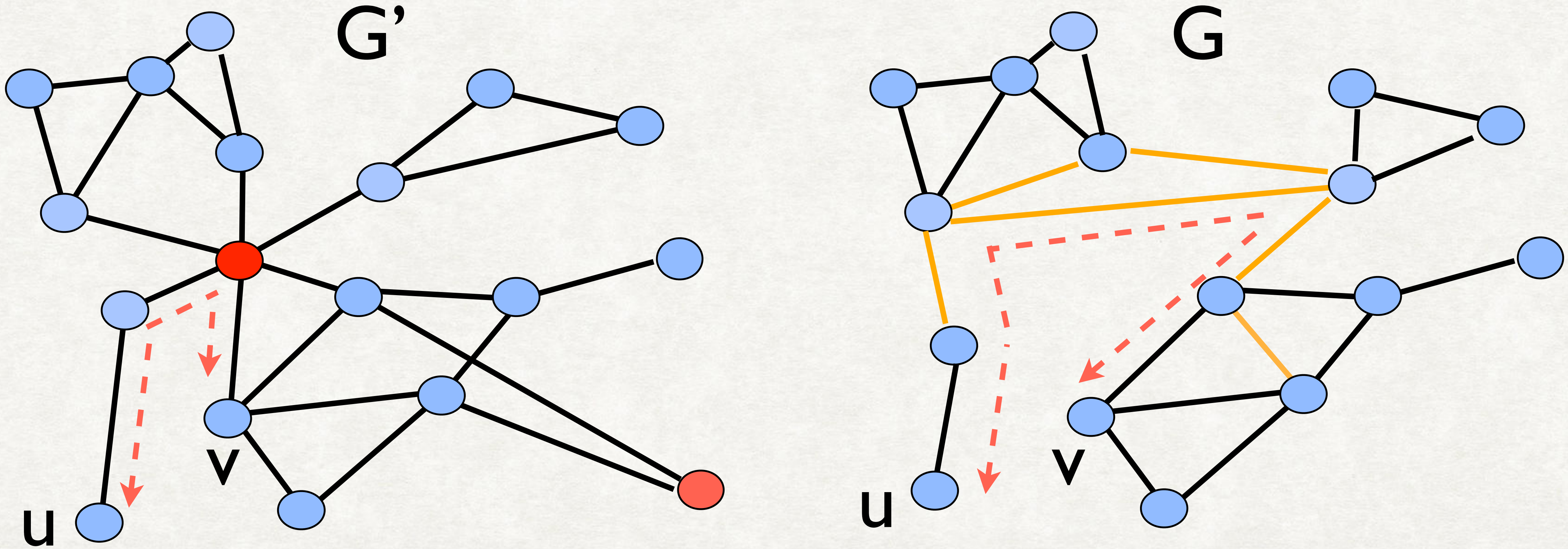
MAIN RESULT

- 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) = 3

d(u,v) = 5

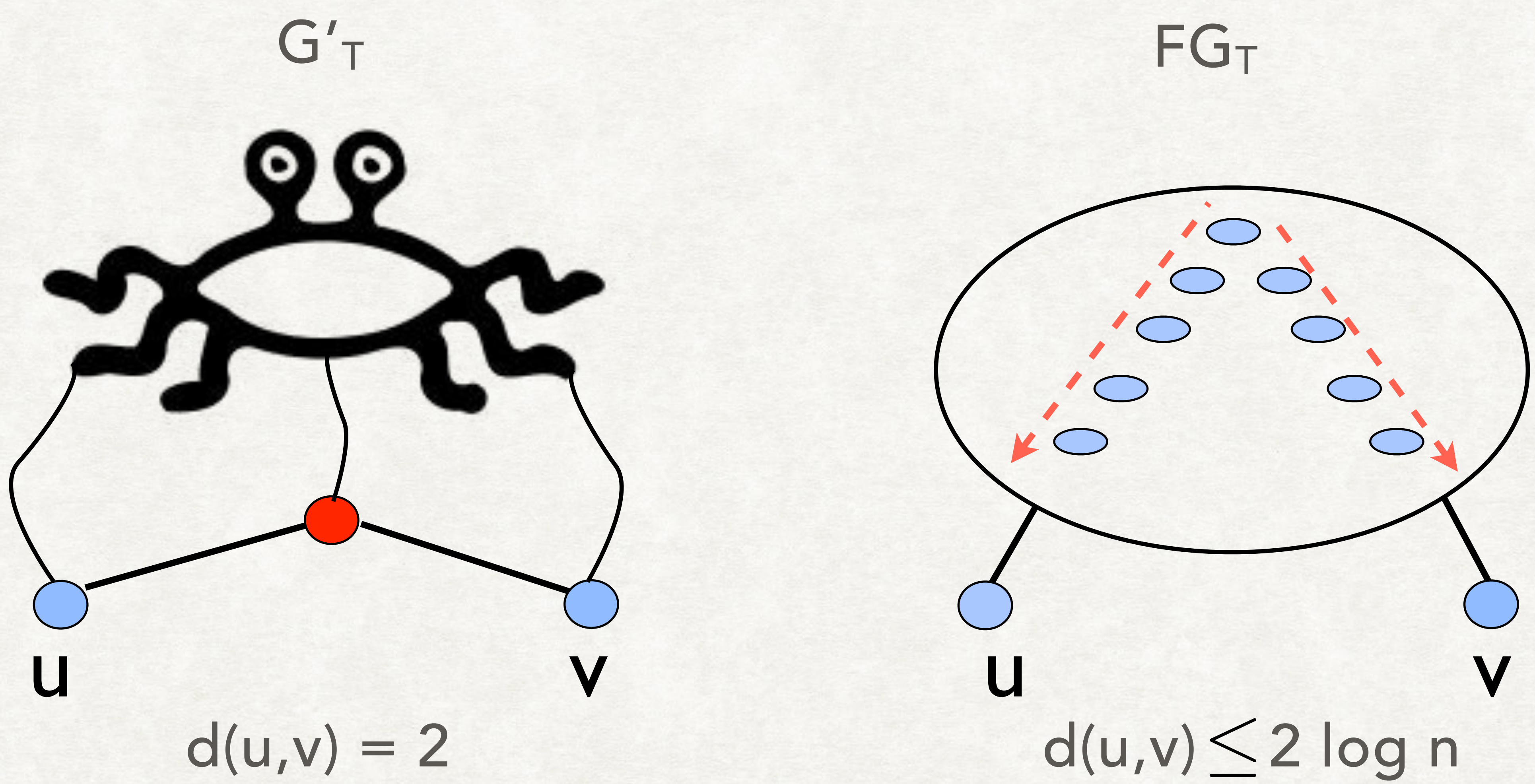
FG: RESULTS AND OPTIMALTIY

- A distributed algorithm, Forgiving Graph such that:
  - Degree of node in G ≤ 3 times degree in G′
  - Distance between any two nodes in G ≤ log n times their distance in G′
  - *Cost*: Repair of node of degree d requires at most O(d logn) messages of length O($\log^2$n) and time O(logd logn)

} Matching lower bound

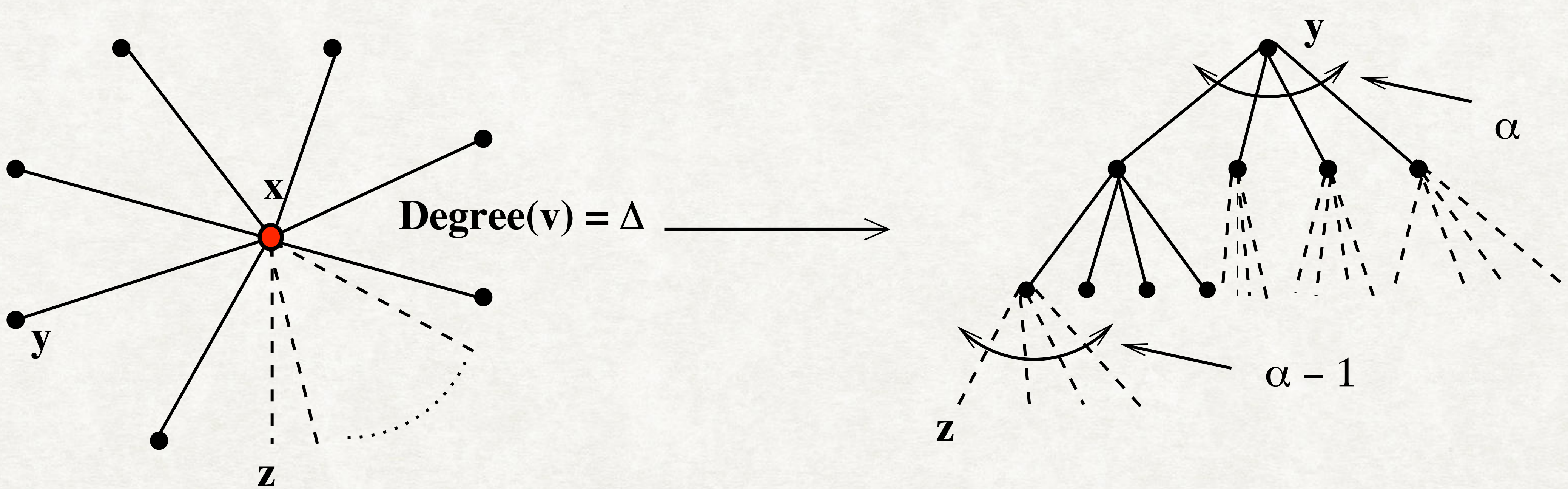
LOWER BOUND AGAIN

- *Stretch*: Distance between any two nodes in G_T ≤ log n times their distance in G'_T



LOWER BOUND AGAIN

- Adversary can force, for any self-healing algorithm:
  - Degree increase. $\leq \alpha \Rightarrow$ stretch of $\Omega(\log_{\alpha}(n-1))$



G'_T BFS Tree of y in FG_T

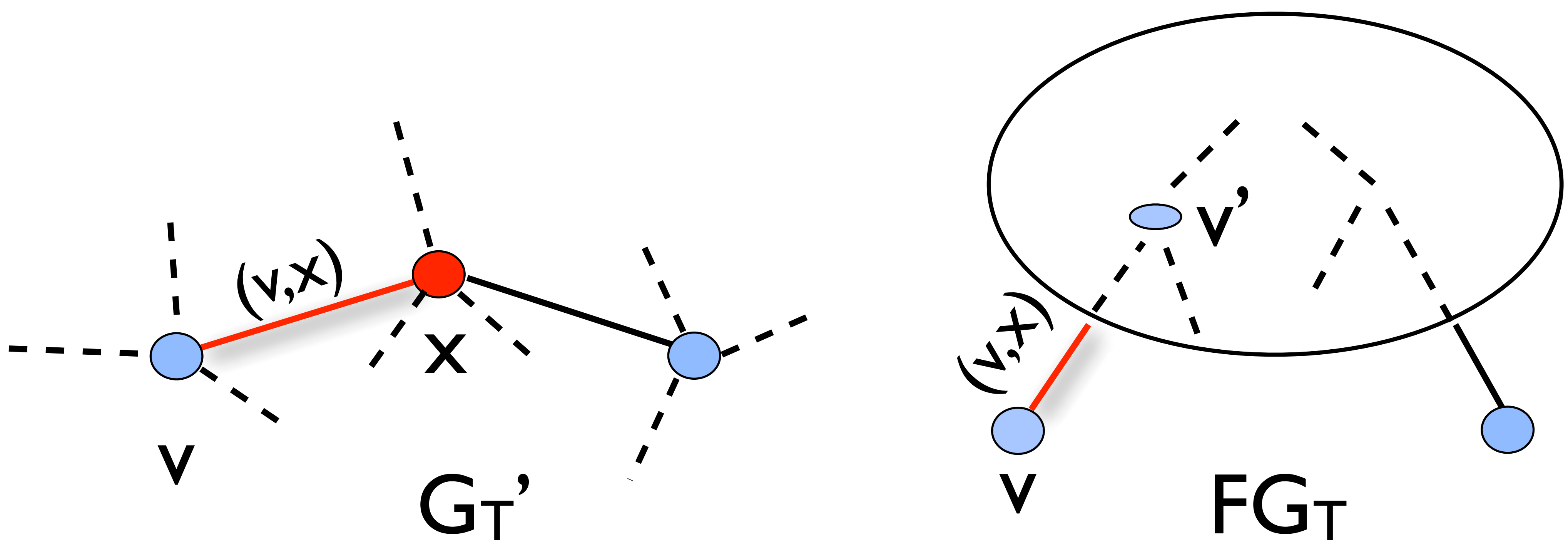
PROVE IT!

- A distributed algorithm, Forgiving Graph such that, at time T:
  - Degree of node in $G_T \leq 3$ times degree in G'_T
  - Distance between any two nodes in $G_T \leq \log n$ times their distance in G'_T
  - *Cost*: Repair of node of degree d requires at most $O(d \log n)$ messages of length $O(\log^2 n)$ and time $O(\log d \log n)$

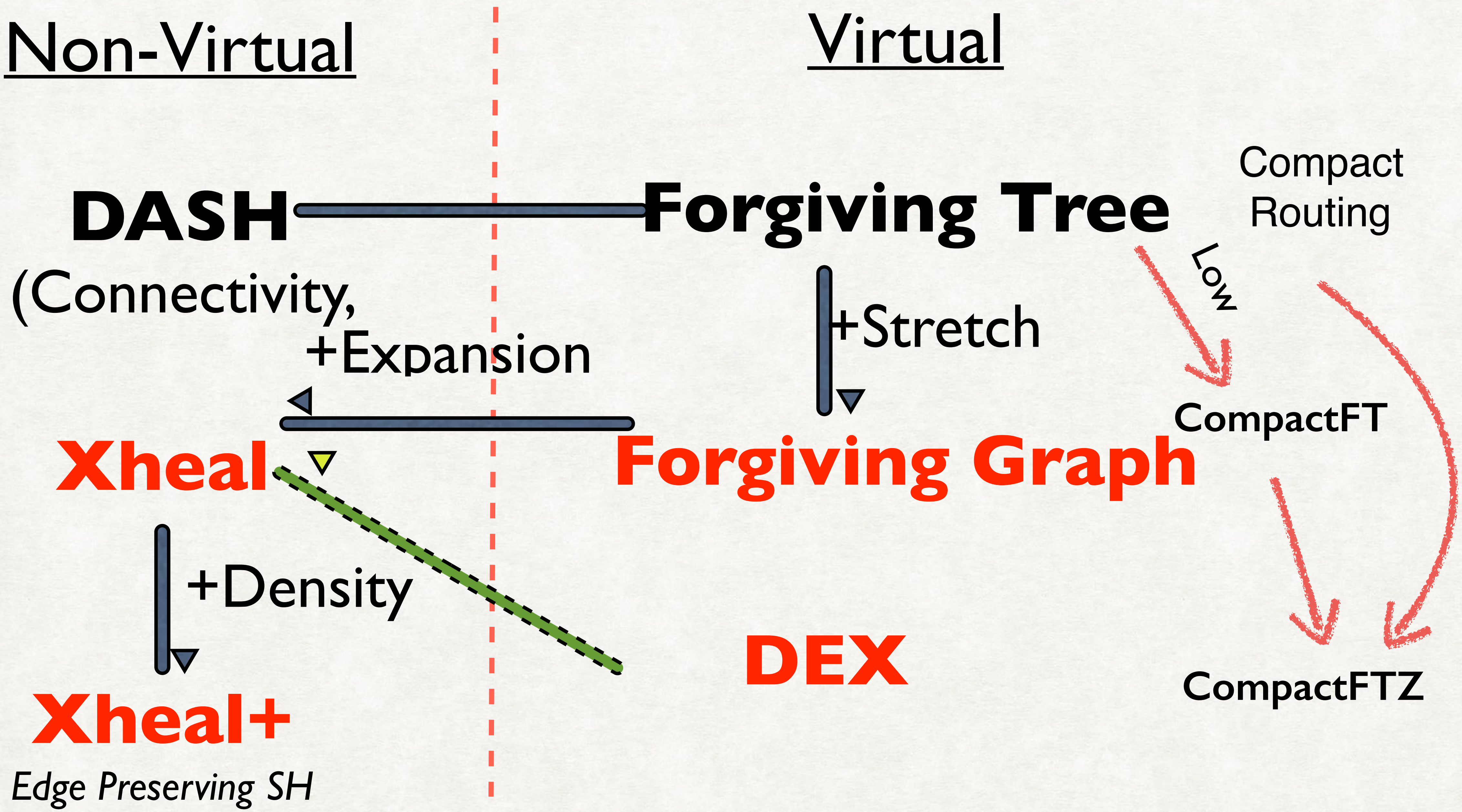
- *Degree increase*: Degree of node in $G_T \leq 3$ times degree in G'_T:

  i. An internal node of a binary tree has degree at most 3

  ii. Each edge in G'_T has at most one corresponding helper node in FG_T



OUR SELF-HEALING ALGORITHMS

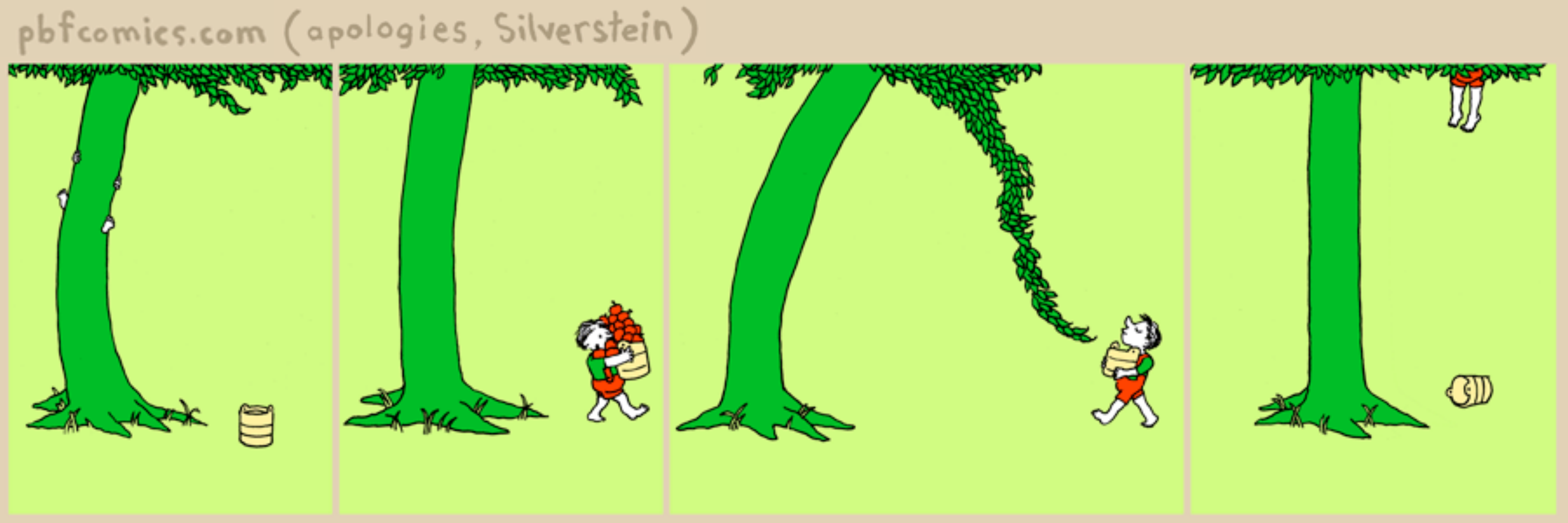


The red algorithms are fully dynamic!

TEMPORAL QUESTIONS AND FUTURE WORK

- What is the best way to analyse fully node dynamic algorithms (say, self-healing graphs)?
- Can edge dynamic temporal theory help? In some use cases, possibly node dynamic are contained in Edge dynamic!
- Other interactions between distributed algorithms and temporal theory
- Temporal self-healing and memory constrained Processes? - Routing* etc…
- A general theory for dynamicity - routing schemes as compositions/operators on self-healing networks

*Armando Castanader, Danny Dolev, AT. Compact routing messages in self-healing trees. ICDCN 2016, Theor. Comp. Sci. 2018



THANK YOU