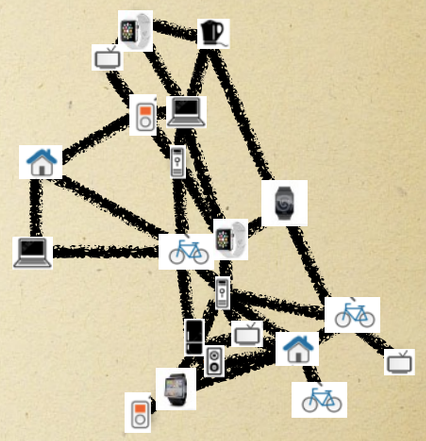
Amnesiac Flooding (the curious unique algorithm) and Self-Healing

AMITABH TREHAN
DURHAM UNIVERSITY



With

Walter Hussak (Loughborough University)
and Henry Austin, Max Gadouleau, George Mertzios (Durham University)

and Saia, Hayes, Rustagi, Pandurangan, Robinson, Gilbert, Castañeda, Dolev, Lefevre, Kutten, Peleg, …

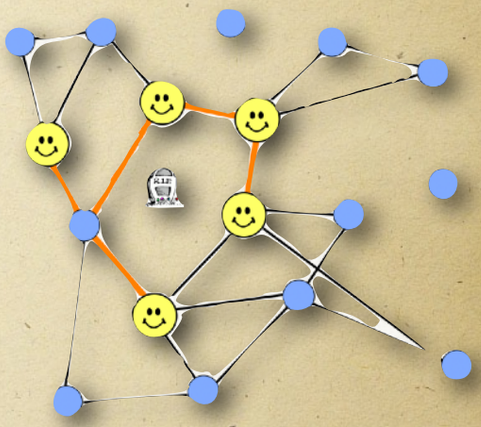
At The Warwick Venice Workshop!

May 2026

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

Games we play!

- Game 1:The amnesiac ambitious WhatsApper! (with Hussak, Austin et al …)

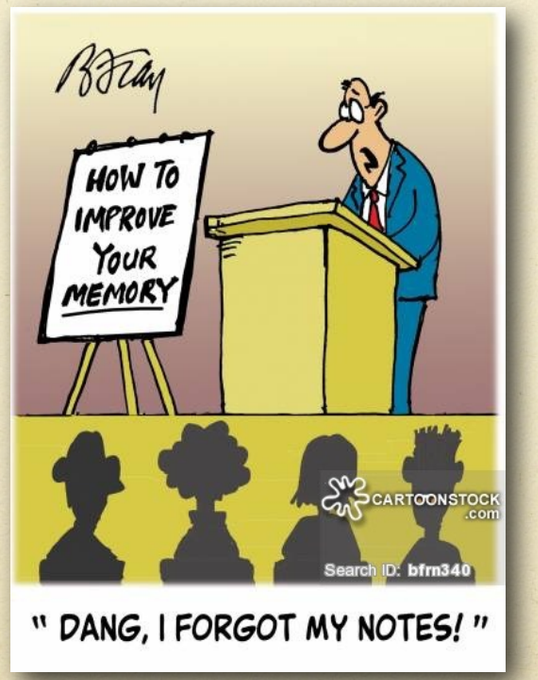


- Game 2: Graph reconstruction (self-healing) game! (with Saia, Hayes, Dolev, Pandurangan, Robinson, Gilbert et al …)

- Game 3: Too many friends, too little memory! (*Compact Local Streaming (CLS)* with Castaneda et al)

Short Term memory loss!

- Game 1: Amnesia! (From dictionary)- Partial or total loss of memory, usually resulting from shock, psychological disturbance, brain injury, or illness.

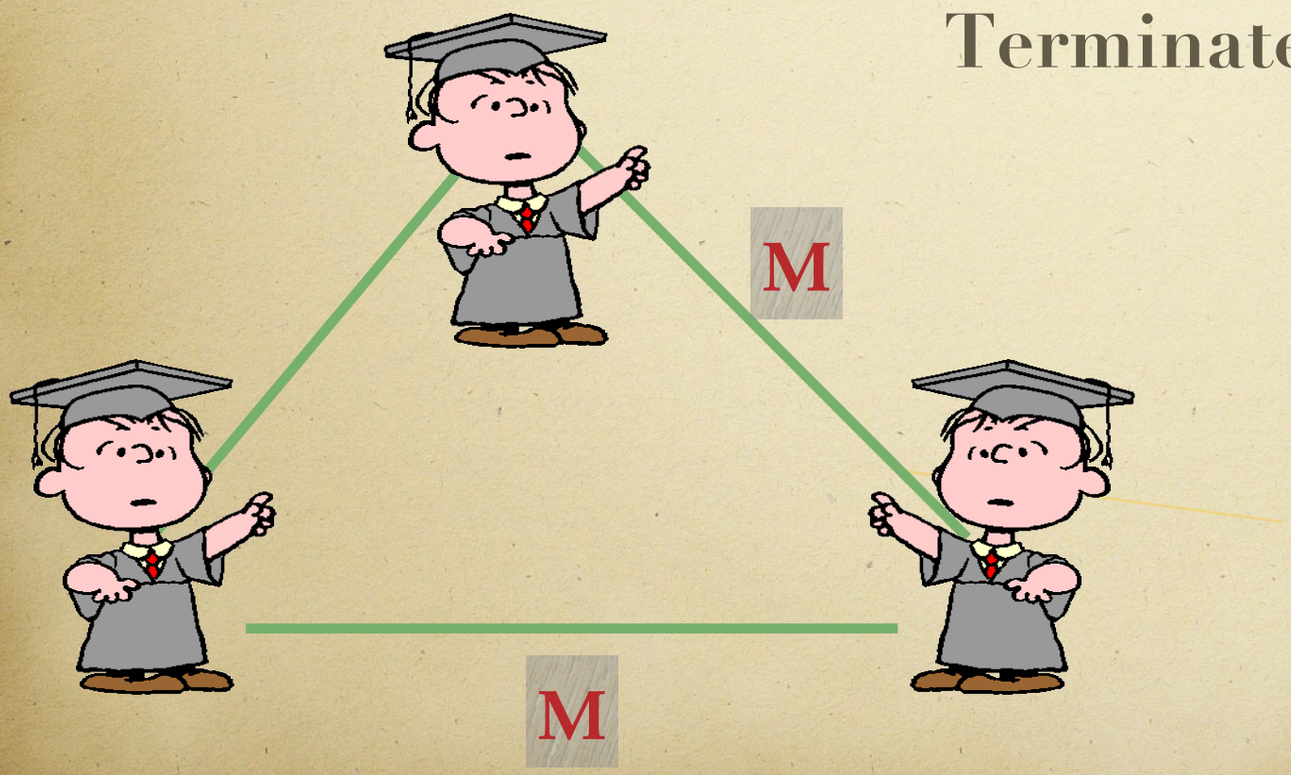


Memory not bounded but short term = no state =stateless!

(Classic) `Dumb' Flooding

Flooding is about the simplest of all distributed algorithms. It is dumb and expensive but easy to implement …[James Aspnes]

Objectives: (Source s has message M). **Broadcast** M to all nodes. **Terminate.**



(Classic) `Dumb' Flooding

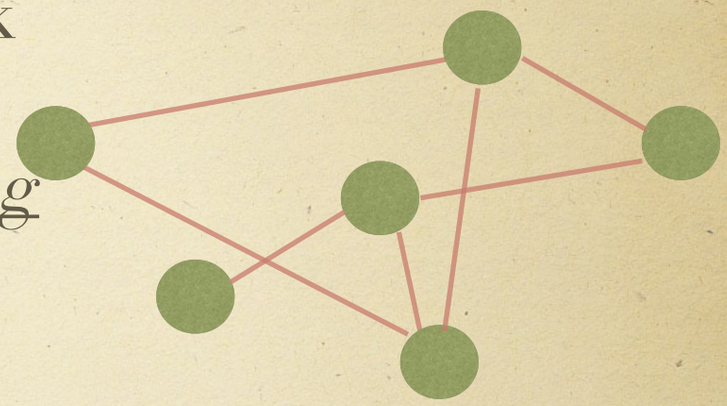
Flooding is about the simplest of all distributed algorithms. It is dumb and expensive but easy to implement …[James Aspnes]

Objectives: (Source s has message M). **Broadcast** M to all nodes.
Terminate.

Broadcast: Given source(s), every node receives M
Terminate: The transmission ceases i.e. no node forwards M

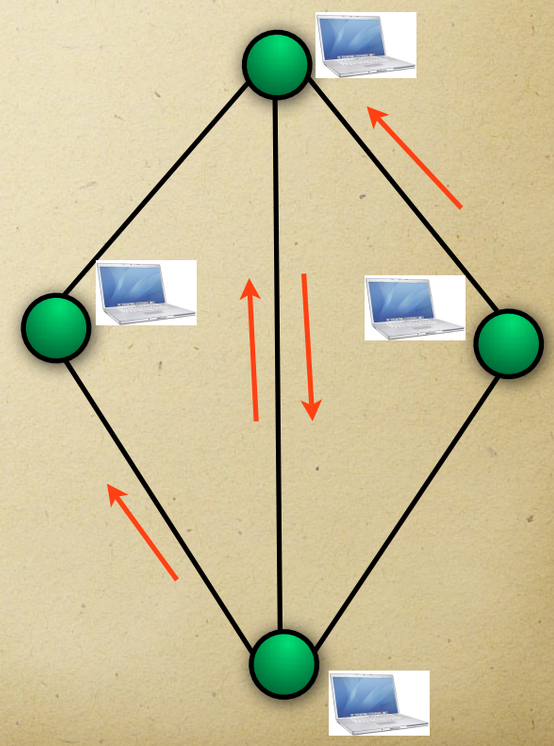
Message Passing: Formal Model

- A graph **G(V,E)** is a formal model for a network
- **Communication:** Purely by *sending and receiving messages.*



A. Synchronous and Reliable: communication in synchronous rounds, messages delivered by end of the round sent in.

B. Adaptive Round asynchronous: 'global' rounds but adaptive adversary decides the delay on each edge (i.e. delays are integers)



(Stateful)/Traditional Flooding

Ver 1:

- Start: A distinguished node *l*
- *Round 1: l* sends message *M* to all neighbours
- *Round i (> 1):*
  - *A.* If node *v* receives *M* for the first time, *v* floods to *n(v)* (i.e. all of v's neighbours)
  - *B.* On subsequent receipts, *v* ignores M

(Stateful)/Traditional Flooding

Ver 2:

- Start: A distinguished node *l*
- *Round 1: l* sends message *M* to all neighbours
- *Round i (> 1):* If node *v* receives *M* from neighbours *R* in round *i-1:*

  A. If node *v* receives *M* for the <u>first time</u>, *v* floods to *n(v)/R* (i.e. neighbours besides *R*)

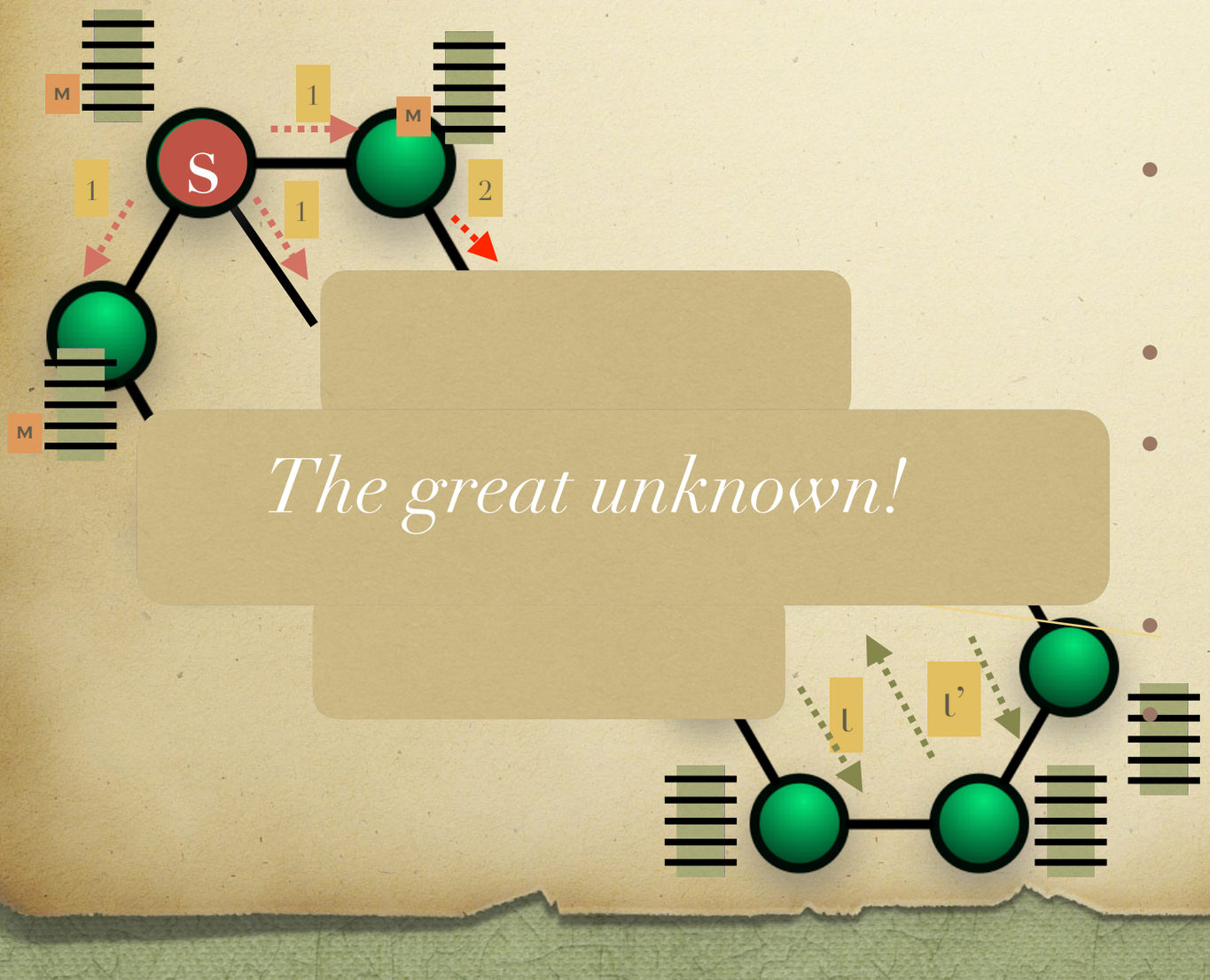
  B. On subsequent receipts, *v* ignores *M*

(Ver 1 and Ver 2): *How is Step B executed?*

(Classic) `Dumb' Flooding

How to terminate?

(i.e. no more messages)



- Keep a copy of M in the *local stack!*
- Check if incoming message is M.
- If yes, discard!
- Overhead and history for every flood.
- *Is this necessary?*

Amnesiac Flooding: `Dumber' yet `smarter?'

What if I forget to use the local stack?
Will flooding still terminate? ???

- Introducing **Amnesiac Flooding** (**AF**)
  = **Send and Forget!**

Amnesiac Flooding: `Dumber' yet `smarter?''

What if I forget to use the local stack?
Will flooding still terminate? ???

- Introducing **Amnesiac Flooding** (**AF**)
  = **Send and Forget!**
- Just one rule: *In every round, send M to all neighbours who did not send in the previous round!*
- *AF is memoryless (`amnesiac') and stateless!*

Hussak and Trehan
[PODC'19, STACS'20, Dist Computing'24]

Amnesiac Flooding (AF)

- More formally: ***Amnesiac Flooding (AF)***
- Start: A distinguished node *l*
- *Round 1: l* sends message *M* to all neighbours
- *Round i (> 1):* If node *v* receives *M* from neighbours *R* in round *i-1*, *v* floods to *n(v)/R* (i.e. neighbours besides *R*)

Q1: Does AF terminate?

Q2: If AF terminates, how long does it take?

AMNESIAC FLOODING: TERMINATION

Amnesiac Flooding

- Start: A distinguished node *l*
- *Round 1: l* sends message *M* to all neighbours
- *Round i (> 1):* If node *v* receives *M* from neighbours *R* in round *i-1*, *v* floods to *n(v)/R* (i.e. neighbours besides *R*)

Q1: Does this process terminate?

- Let's try some examples

1. Line Graph:

Round 1 Round 2 Round 3

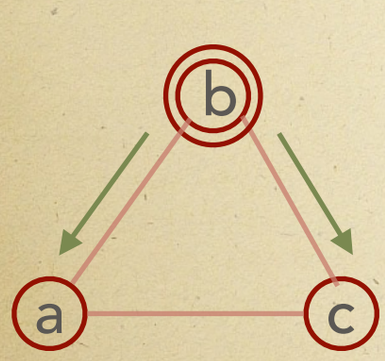
AMNESIAC FLOODING: TERMINATION

Amnesiac Flooding

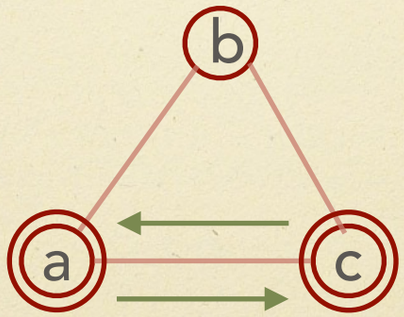
- Start: A distinguished node *l*
- *Round 1: l* sends message *M* to all neighbours
- *Round i (> 1):* If node *v* receives *M* from neighbours *R* in round *i-1*, *v* floods to *n(v)/R* (i.e. neighbours besides *R*)

Q1: Does AF terminate?

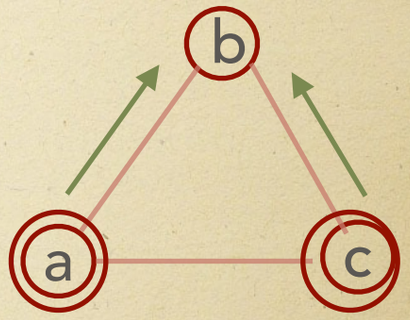
2. Triangle/Clique/Odd Cycle:



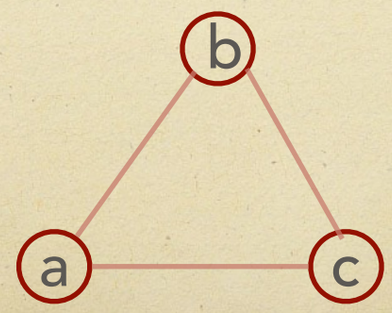
Round 1



Round 2



Round 3



Round 4

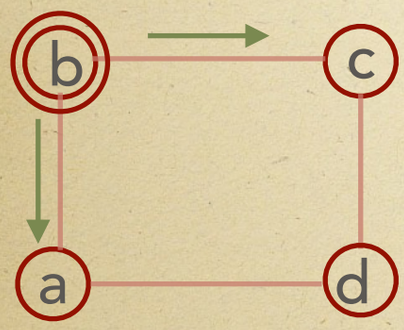
AMNESIAC FLOODING: TERMINATION

Amnesiac Flooding

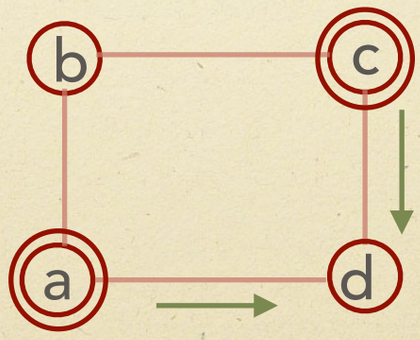
- Start: A distinguished node *l*
- *Round 1: l* sends message *M* to all neighbours
- *Round i (> 1):* If node *v* receives *M* from neighbours *R* in round *i-1*, *v* floods to *n(v)/R* (i.e. neighbours besides *R*)

Q1: Does AF terminate?

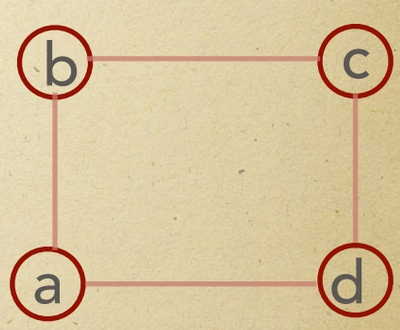
3. Even Cycle:



Round 1



Round 2



Round 3

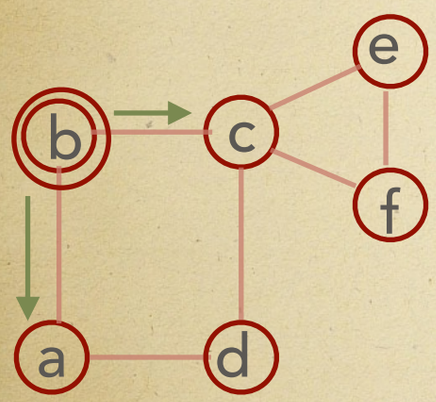
AMNESIAC FLOODING: TERMINATION

Amnesiac Flooding

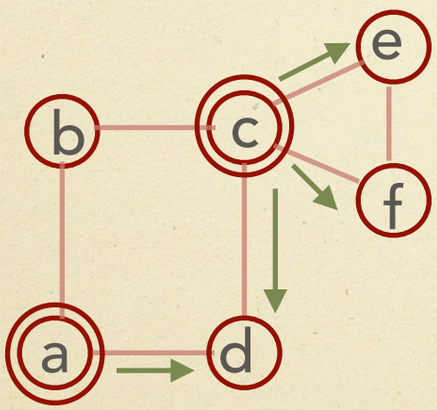
- Start: A distinguished node *l*
- *Round 1: l* sends message *M* to all neighbours
- *Round i (> 1):* If node *v* receives *M* from neighbours *R* in round *i-1*, *v* floods to *n(v)/R* (i.e. neighbours besides *R*)

4. More complex topologies:

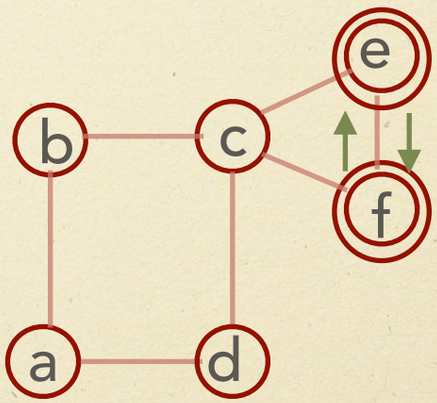
Q1: Does AF terminate?



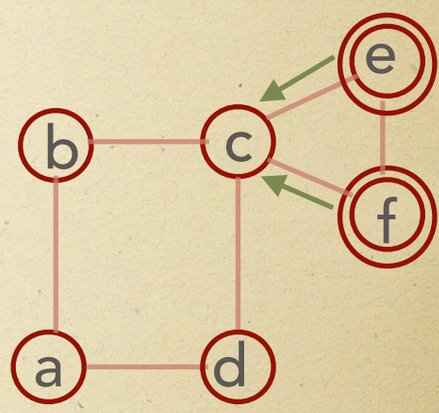
Round 1



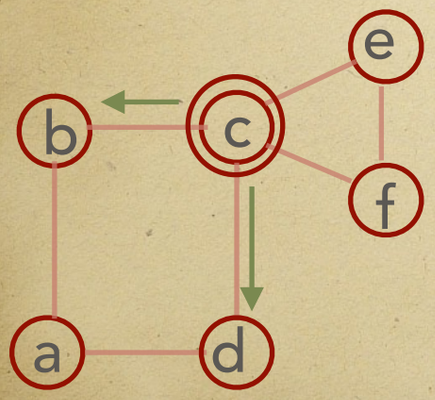
Round 2



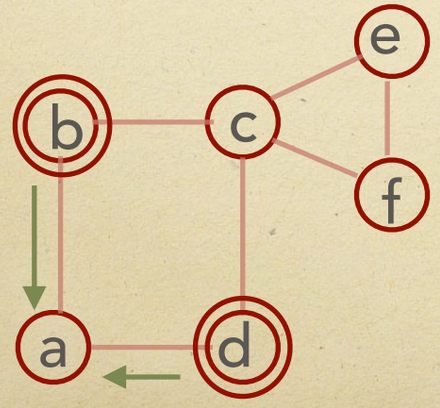
Round 3



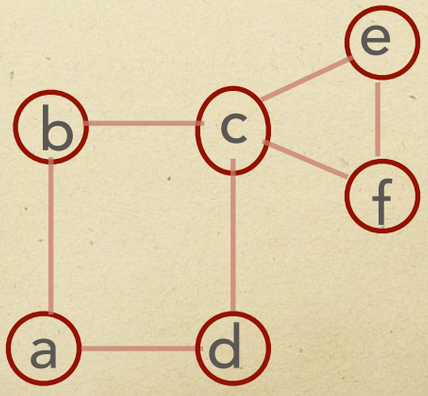
Round 4



Round 5



Round 6

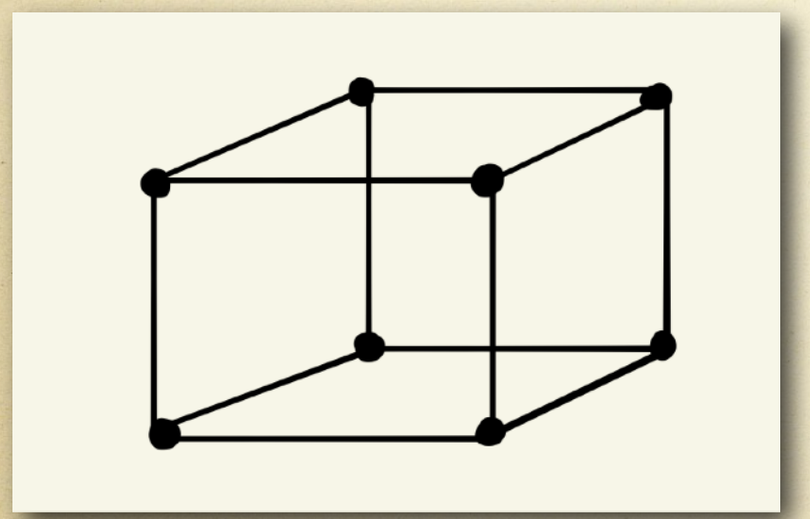


Round 7

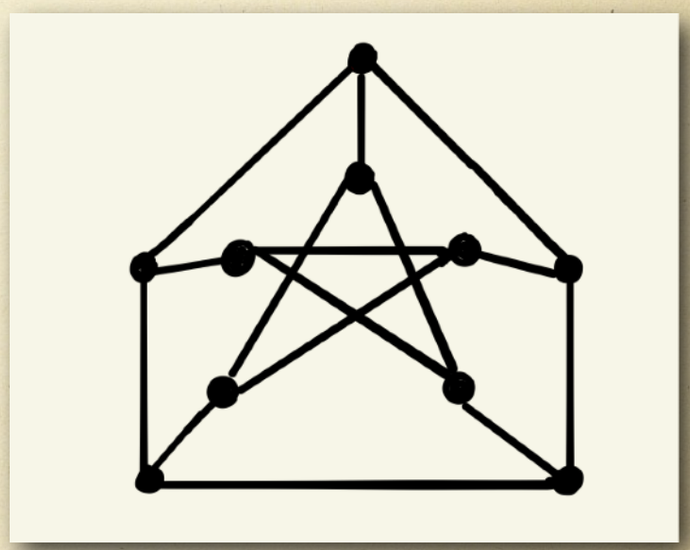
AMNESIAC FLOODING: TERMINATION

Q1: Does AF terminate?

5. .. And Some Well known Topologies:



Hypercube

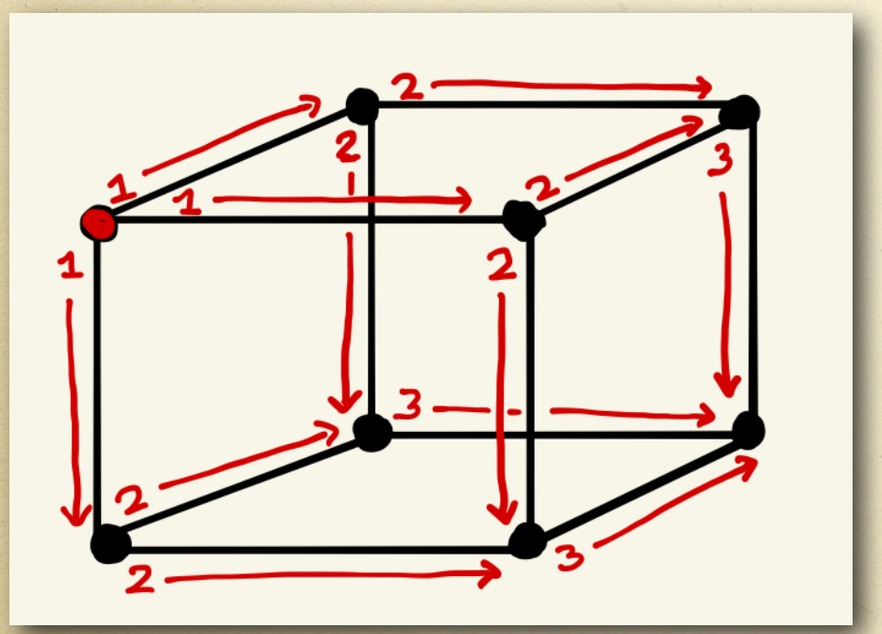


Petersen Graph

AMNESIAC FLOODING: TERMINATION

Q1: Does AF terminate?

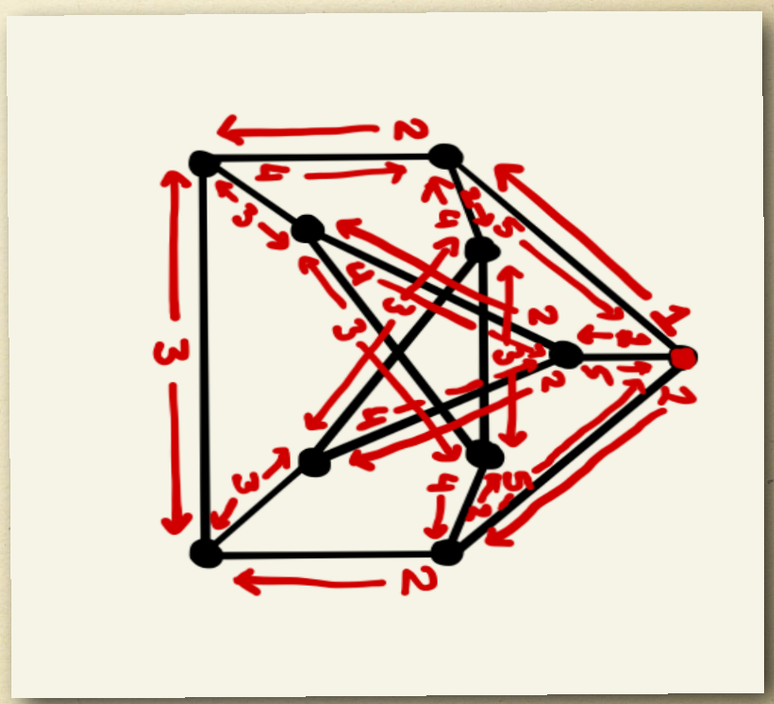
5. .. And Some Well known Topologies:



Hypercube

T = 3 rounds

Diameter = 3



Petersen Graph

T = 5 rounds

Diameter = 2

Why such different termination times?

AF TERMINATION

DOES AF TERMINATE?

ON EVERY GRAPH?

IS IT QUICK?

AF TERMINATION

DOES AF TERMINATE?

YES

ON EVERY GRAPH?

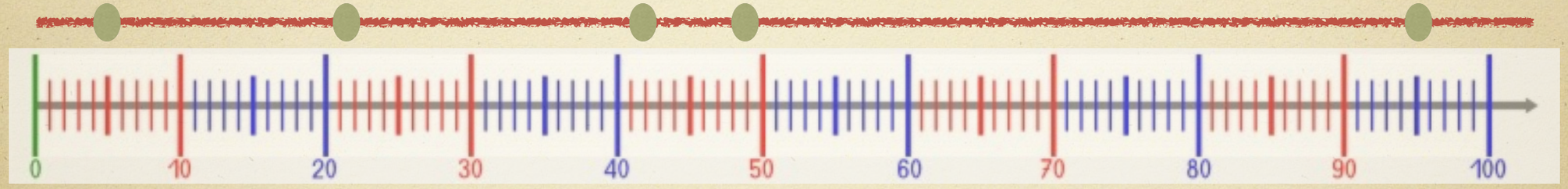
YES

IS IT QUICK?

ACTUALLY, YES (WE'LL SEE)

Thought experiment!
Non-terminating processes

- Consider a non-terminating process on a discrete time line! i.e the same event repeating infinitely.



- What properties do such sequences have?
- Can negation of some such property prove process is terminating?

AF Termination

Theorem 1. *Given a finite graph G, Amnesiac Flooding (AF) from a single source will terminate in a finite number of rounds*

Proof.

Proof is by contradiction

High level idea:

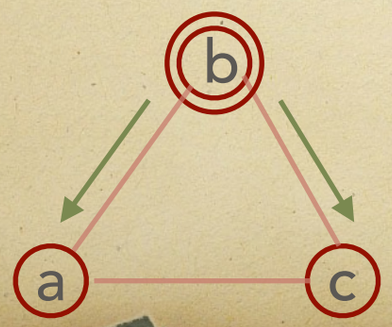
- Define ***round-sets*** as set of nodes receiving M in a particular round
- Consider sequences $\mathbf{R^e}$ of round-sets E of even duration:

Condition of non-termination:

There must be at least one E having the same node repeated!

- We show this is not possible!

E.g.



$R_0 = \{b\}$

R1 = {a,c} R2 = {a,c}

R3 = {b} R4 = {}

AF TERMINATION

Theorem 1. *Given a finite graph G, Amnesiac Flooding (AF) from a single source will terminate in a finite number of rounds*

Claim 1. If AF does not terminate, R^e will be non-empty

- **Detailed Proof.**

Assume R^e is non-empty
Take the *first smallest* sequence in the set R^e!

Define
md: shortest duration of any sequence
ms: earliest starting point of such sequences

Consider node ***x*** common to R_{ms} and R_{ms+md} (exists by assumption)
and node **y** which sent M to node x in round R_{ms+md} :

Case 1. y also sent M to x in round ms
OR
Case 2. x sent M to y in round ms+1

AF TERMINATION

Theorem 1. *Given a finite graph G, Amnesiac Flooding (AF) from a single source will terminate in a finite number of rounds*

Case 1. y also sent M to x in round ms

Proof by picture

Thus, round ms-1 is either 0 and y is origin node *l*

Or y received M in round ms-1 (> 0):

Either way, there is a sequence
of even min-duration md but earlier start point ms-1
with
Contradiction.

Claim 1. If AF does not terminate, $\mathbf{R}^e$ will be non-empty

The *first smallest* sequence in the set $\mathbf{R}^e$!

node *x* in R_{ms} and R_{ms+md}
node y: sent M to x in R_{ms+md}

AF TERMINATION

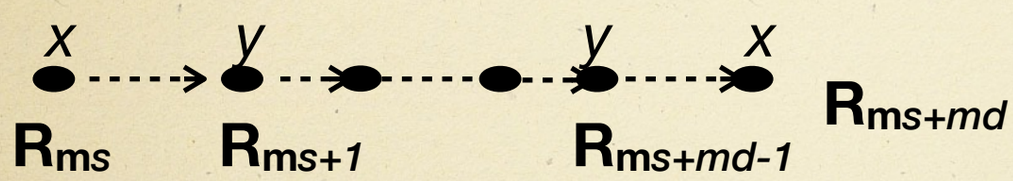
Theorem 1. *Given a finite graph G, Amnesiac Flooding (AF) from a single source will terminate in a finite number of rounds*

Claim 1. If AF does not terminate, R^e will be non-empty

The *first smallest* sequence in the set R^e!

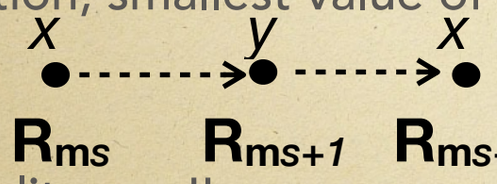
node *x* in R_{ms} and R_{ms+md}
node y: sent M to x in R_{ms+md}

Case 2. x sent M to y in round ms+1



Proof by picture

By definition, smallest value of md is 2 i.e.

Maybe  but this is not possible

due to politeness!!

Thus, there is a sequence $\underline{R^{*''}} = R_{ms+1}, \ldots, R_{ms+md-1}$

Of even duration md - 2 with y repeating i.e. $R_{ms+1} \cap R_{ms+md-1} \neq \emptyset$

Contradiction.

Hence proved.

TERMINATION TIMES

Revisiting the proof:

Theorem 1. *Given a finite graph G, Amnesiac Flooding (AF) from a single source will terminate in a finite number of rounds*

Claim 1. If AF does not terminate, R^e will be non-empty

Don't exist!

Proof. G is finite, therefore, if AF does not terminate, a node x must occur in infinitely many round-sets.

Consider the first 3 such round-sets (e.g. 2,5, and 7);

Surely at least two of these are evenly spaced.

L1: In AF from a single source, a node can be visited at most twice!

T2: AF from a single source terminates in at most 2n rounds where n is the number of nodes

However, there are better bounds ....

AF TERMINATION

HOW LONG DOES IT TAKE?

BIPARTITE GRAPHS: <= D STEPS

NON-BIPARTITE GRAPHS: <= 2D + 1 STEPS

(where D is the diameter of the graph)

Note: A hypergraph is bipartite (termination takes D)
but Petersen graph is not (termination takes 2D+1)

AF TERMINATION (MORE PRECISELY)

HOW LONG DOES IT TAKE?

BIPARTITE GRAPHS:

A graph is Bipartite if and only if termination time t of AF from origin a is $t = e(a) \leq D$ *rounds*

(where e(a) is the eccentricity of the origin node a and D the diameter of the graph)

(SIMPLER CASE) TERMINATION IN BIPARTITE GRAPHS

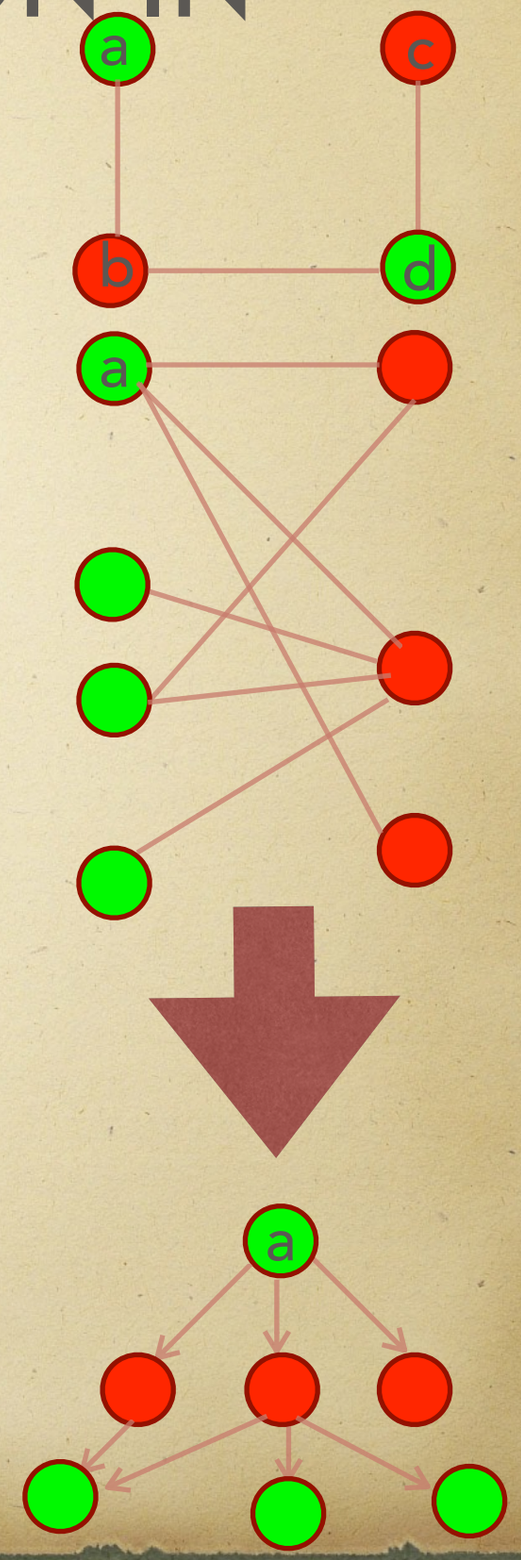
Bipartite Graph: A set of graph vertices decomposed into two disjoint sets (say, red and green) such that no two graph vertices within the same set are adjacent.

Theorem. In a connected bipartite graph, AF terminates in ***#rounds = e(a)***, where e(a) is the eccentricity of the vertex a, where a is the origin node.

Proof Sketch.

Consider the BFS traversal from the source!

There are no cross edges, therefore, nodes are explored at the earliest and by AF, there are no cycles.



AF TERMINATION (MORE PRECISELY)

HOW LONG DOES IT TAKE?

NON-BIPARTITE GRAPHS:

In a non-bipartite graph, AF from an origin node a terminates in t rounds where

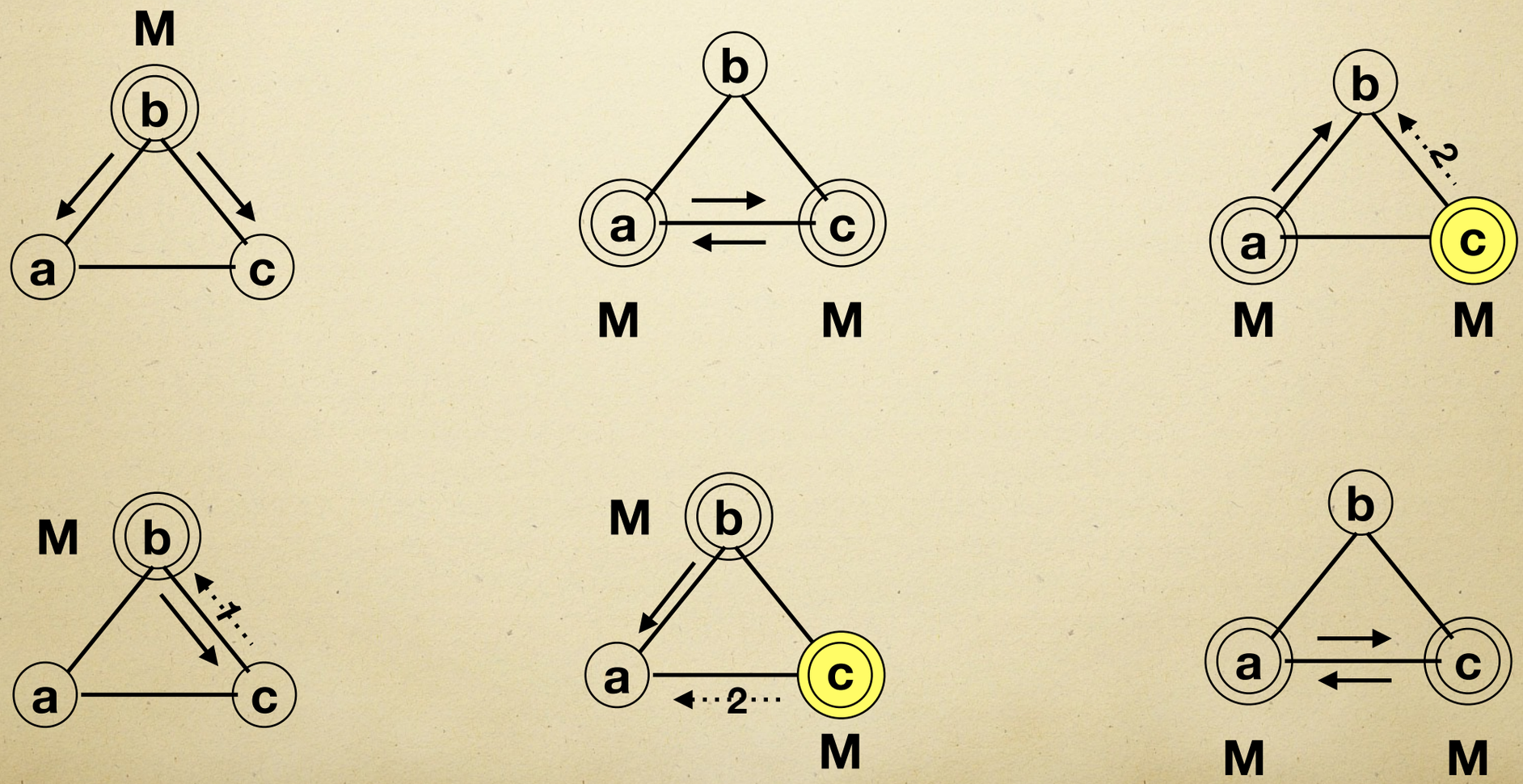
$$e(a) < t \leq e(a) + D + 1 \leq 2D + 1$$

(where e(a) is the eccentricity of the origin node a and D the diameter of the graph)

ASYNCHRONOUS FLOODING

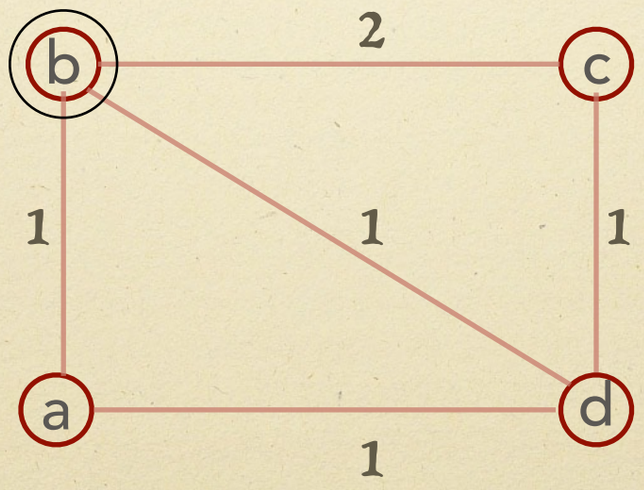
WHAT IF MESSAGES COULD BE DELAYED?

- Non-termination could be forced if an adversary could control delays!



Non-Termination!

- Even fixed delays can cause non-termination!



- However, Asynchronous AF will always terminate on Acyclic graphs (in maximum eccentricity of any origin node steps!)

Questions?

- Does synchronous AF with multiple initiators terminate?
  - What if the initiations are at different times?
- What if the graph is changing during AF?

Questions?

- Does synchronous AF with multiple initiators terminate?

YES

  - What if the initiations are at different times?

YES

- What if the graph is changing during AF?

YES - FOR DELETION OF EDGES AND NODES. MAYBE NOT ON INSERTIONS.

Uniqueness of AF! [PODC'25, DISC'25]

Special properties

- Vertices have no internal state and cannot retain information other than their port labellings.
- Deterministic
- Vertex behaviour does not depend on message contents

Amnesiac Flooding is unique (A., Gadouleau, Mertzios, Trehan 2025)

Amnesiac Flooding is the **only** *correct* and *terminating* algorithm obeying those three conditions.

Amnesiac Flooding is unique at KT(1) (A., Gadouleau, Mertzios, Trehan 2025)

Amnesiac Flooding is the **only** *correct* and *terminating* algorithm obeying those three conditions, even if agents have unique IDs and knowledge of their neighbour's IDs.

AF is a Fixed Point? [PODC'25, DISC'25]

Amnesiac Flooding is (mostly) unique

Special properties

- Vertices have no internal state and cannot retain information other than their port labellings.
- Deterministic
- Vertex behaviour does not depend on message contents

Amnesiac Flooding is unique at KT(1) (A., Gadouleau, Mertzios, Trehan 2025)

Amnesiac Flooding is the **only** *correct* and *terminating* algorithm obeying those three conditions, even if agents have unique IDs and knowledge of their neighbour's IDs.

Sharpness (A., Gadouleau, Mertzios, Trehan 2025)

There exist correct and terminating algorithms **obeying any two of the three conditions**.

References

[Wiki] https://en.wikipedia.org/wiki/Amnesiac_flooding

[**PODC'19**] **On Termination of a Flooding Process** (**Brief Announcement**). PODC 2019 <https://dl.acm.org/doi/10.1145/3293611.3331586>

[STACS'20] Walter Hussak<https://dblp.org/pid/60/6563.html>, Amitabh Trehan:On the Termination of Flooding. STACS 2020<https://dblp.org/db/conf/stacs/stacs2020.html#HussakT20>: 17:1-17:13

[AF-ARXIV'20] Walter Hussak<https://dblp.org/pid/60/6563.html>, Amitabh Trehan: Terminating cases of flooding. CoRR abs/2009.05776<https://dblp.org/db/journals/corr/corr2009.html#abs-2009-05776> (2020)

[Distributed Computing'23] Walter Hussak, Amitabh Trehan: Termination of amnesiac flooding (2023)

[PODC'25, DISC'25] Henry Austin, Maximilien Gadouleau, George B Mertzios, Amitabh Trehan: B Amnesiac Flooding: Easy to Break, Hard to Escape (2025) DISC 2025: 10:1-10:23

FOLLOW UP WORK AND OPEN DIRECTIONS

- Alternate analysis by an auxiliary graph reduction from non-bipartite to bipartite graphs (Turau*)
- K-starters Amnesiac flooding problem: Given k starters, what is the placement of these starters that gives the smallest termination time (Turau*)
- Why? Fundamental reasons behind termination? (Partially addressed in AGMT, PODC'25 (Brief Announcement), DISC'25
- How can these results be used? Distributively estimating diameter/topology?
- Relations to temporal graph theory/other areas?
- Self-healing flooding: Termination against a graph modifying adversary?

** Turau, Volker, Analysis of Amnesiac Flooding, Arxiv (https://arxiv.org/abs/2002.10752)*

The(A) *Distributed* Matrix….

Architecture	Message Passing (Point-to-Point, Broadcast…)	Sensor, Wireless, Reconfigurable, Overlay, Cloud…	Shared Memory
Timing	Synchronous	Partial ..	Asynchronous
Identity	Anonymous	ID	
Knowledge	Local (Own + neighbour IDs), + size, dia estimates?	Restricted	Global
Local Memory	Unlimited	SuperLinear, Linear, Logarithmic	Constant
Bandwidth	*LOCAL* (Unlimited)	*CONGEST* (polylog)	?
Failure	Crash (node)	Edge, Memory, Transient …	Byzantine
Behaviour	Obedient!	Selfish (Game Theory)	Byzantine

Game 2: Self-healing (Model)

- Start: a (reconfigurable) distributed system/network G

Run forever or till possible

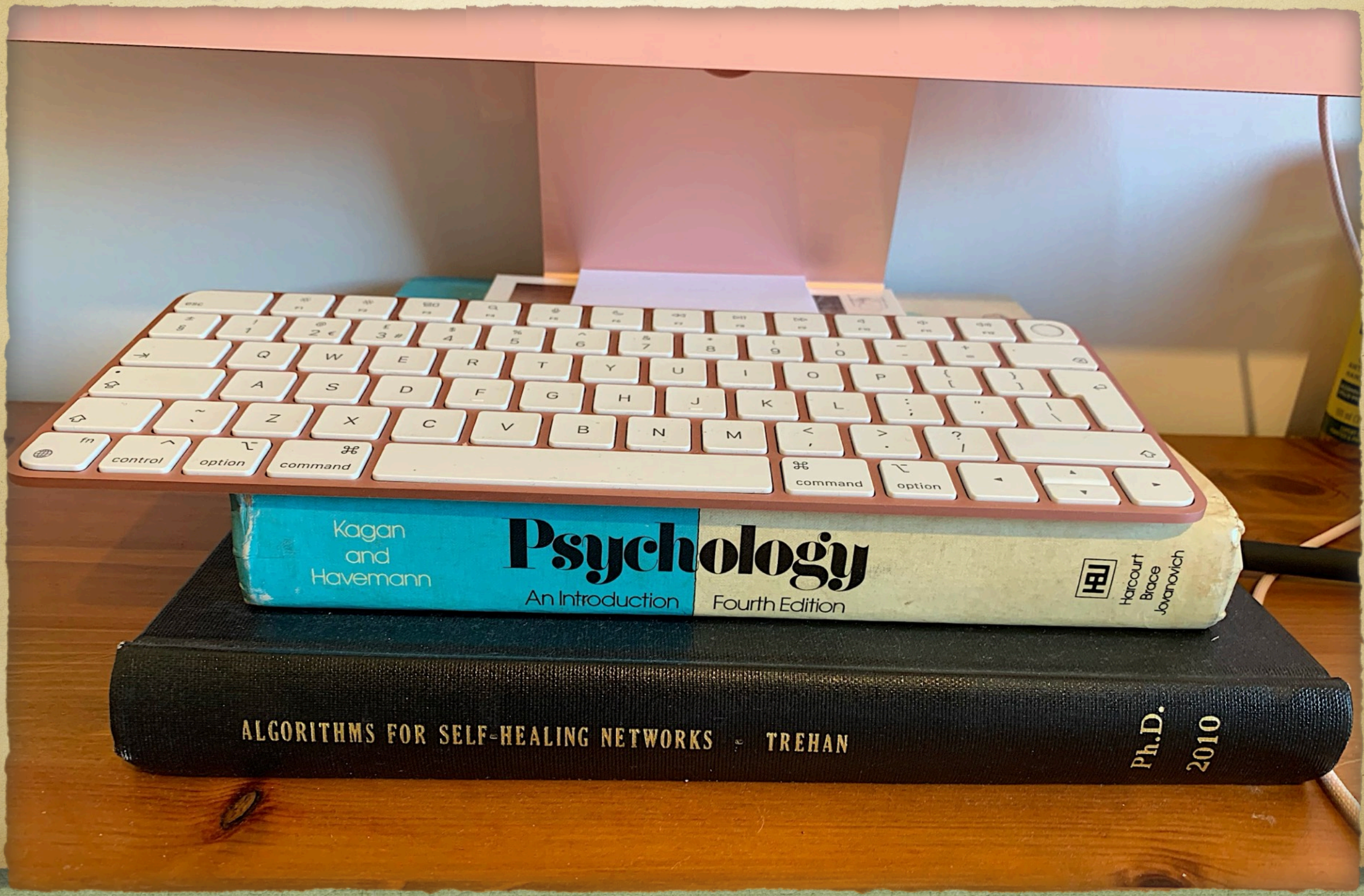
{

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

}

- Node dynamic as opposed to edge dynamic!

Supportive Self-Healing!



Two new non-citations!

- 10 Baruch A., Trehan A. (2011). “On the impossibility of local self-healing in expander graphs,” in Proceedings of the 30th Annual ACM Symposium Principles of Distributed Computing (PODC) (ACM), 273–282.

- 55 Pandurangan G., Trehan A. Upfal E. (2012). Distributed construction of self-healing expander networks. ACM Trans. Algorithms 8, 1–29.

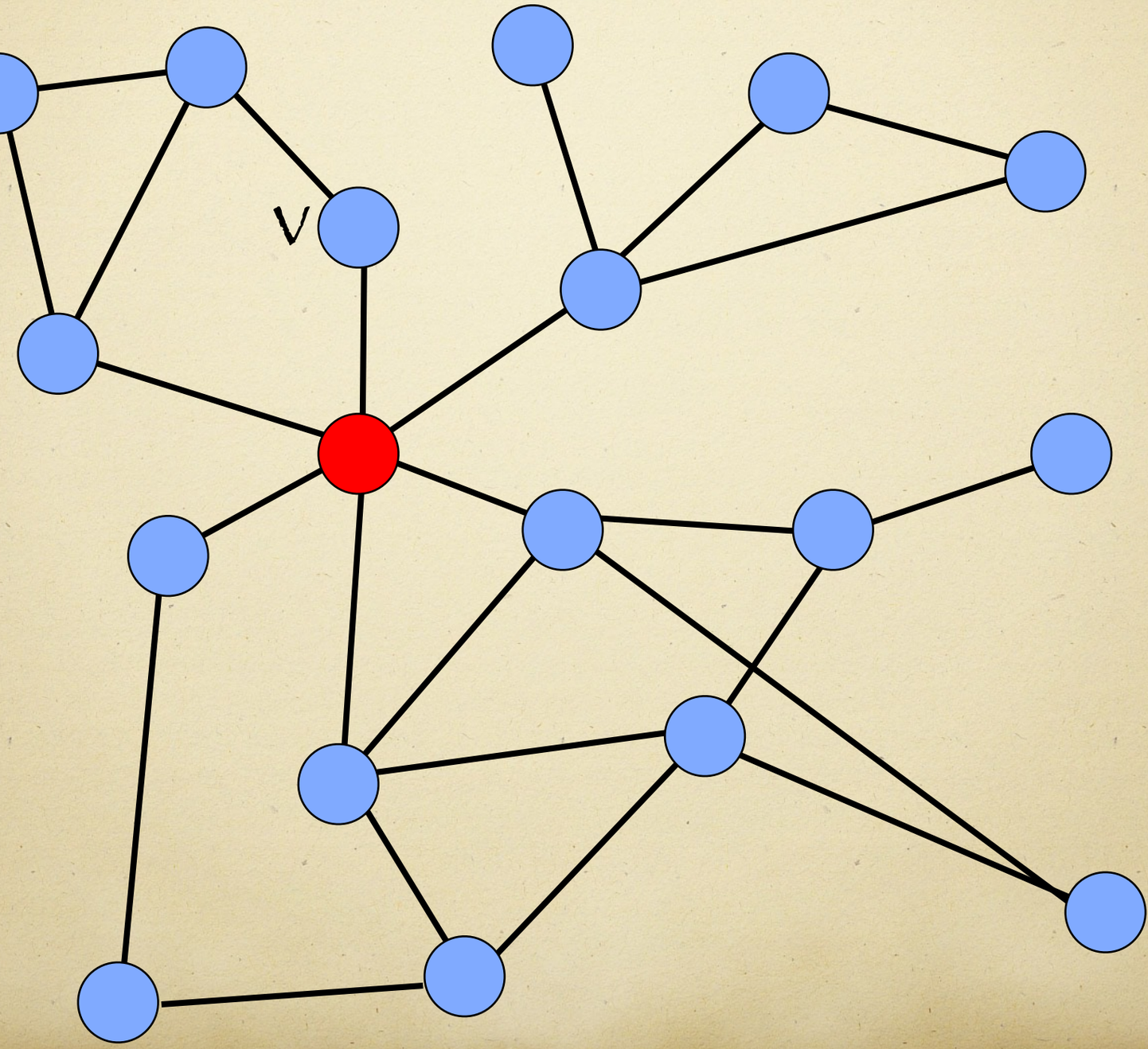
PS: The **above papers don’t exist** but appear in a recent ‘Frontiers in Computer Science’ journal article! This one does:

- 54 Pandurangan G., Robinson P., Trehan A. (2016). Dex: self-healing expanders. Distrib. Comput. 29, 163–185. doi: 10.1007/s00446-015-0258-3

Read the story at https://www.linkedin.com/feed/update/urn:li:activity:7459376568935747584/

Illustration

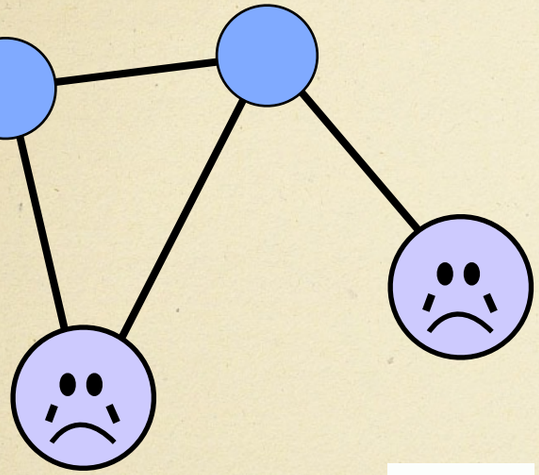
Attacker!

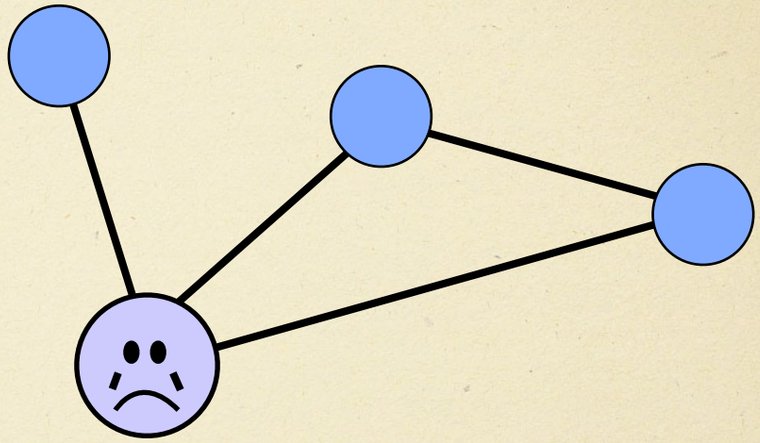


Healer!

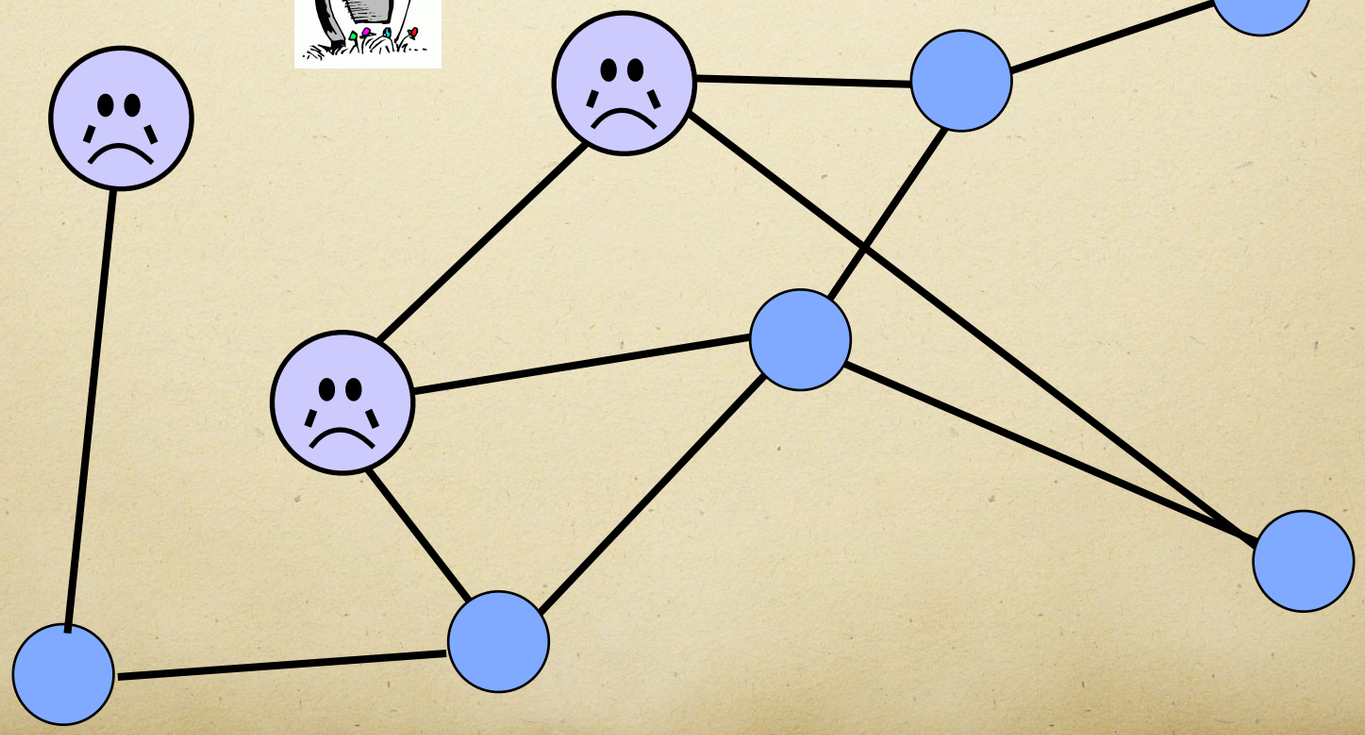
Illustration

Attacker!



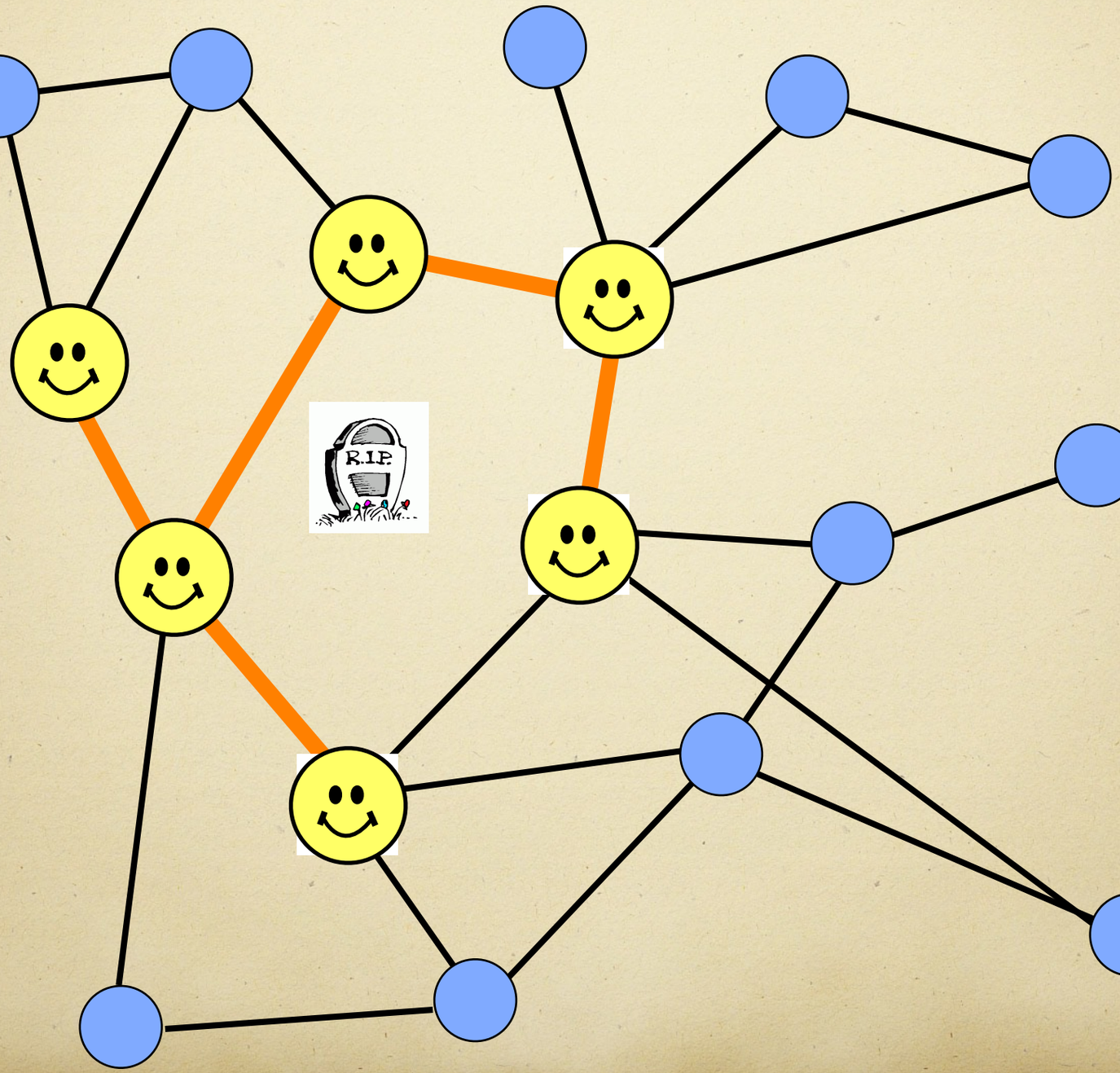


Healer!



Illustration

Attacker!



Healer!

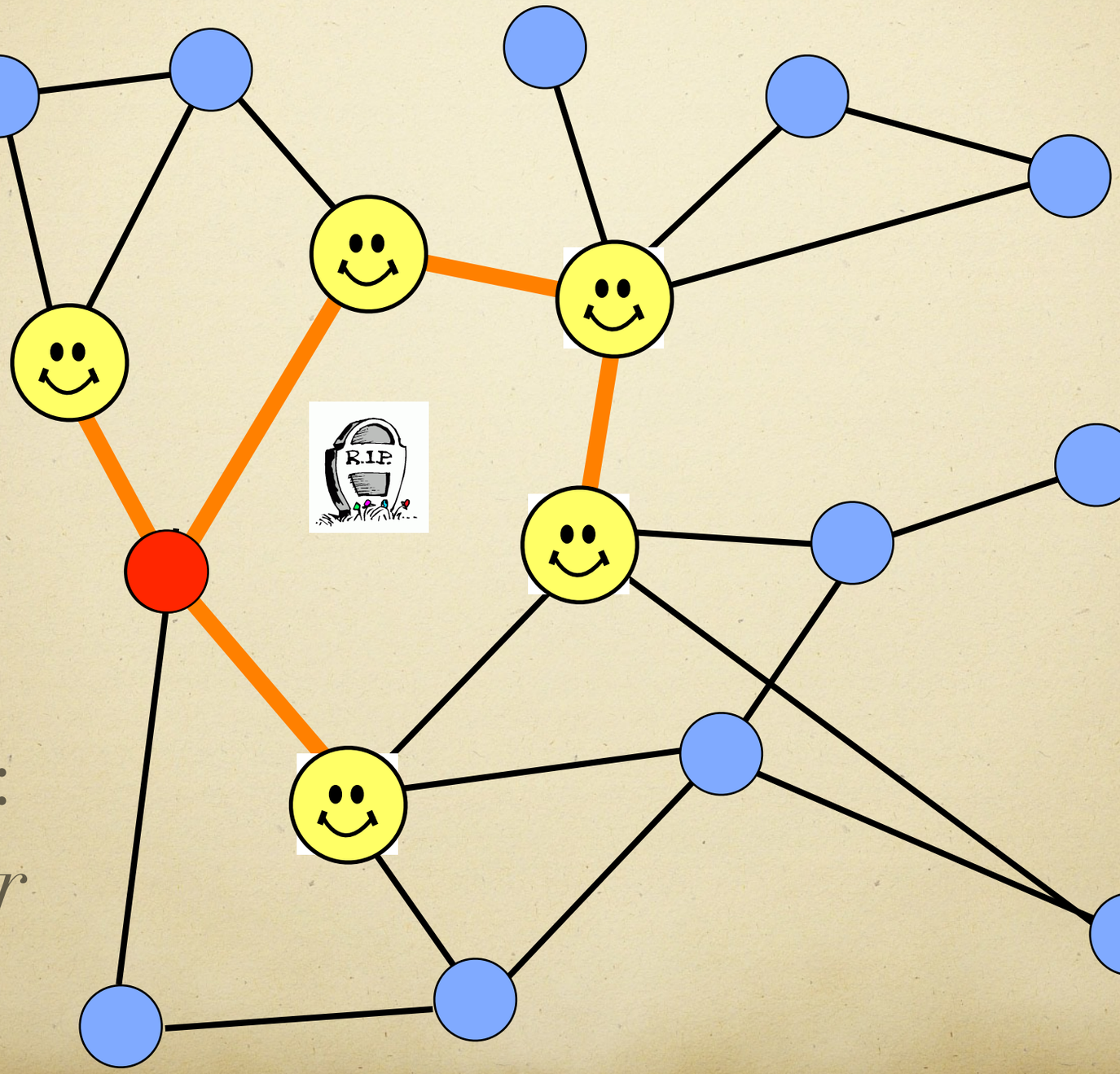
- Question: What is a good adversary attack strategy?

- Question: What is a good adversary attack strategy?
- One possible answer: Attack the neighbour of the target!

Illustration

Attacker!

Strategy:
Neighbour attack!

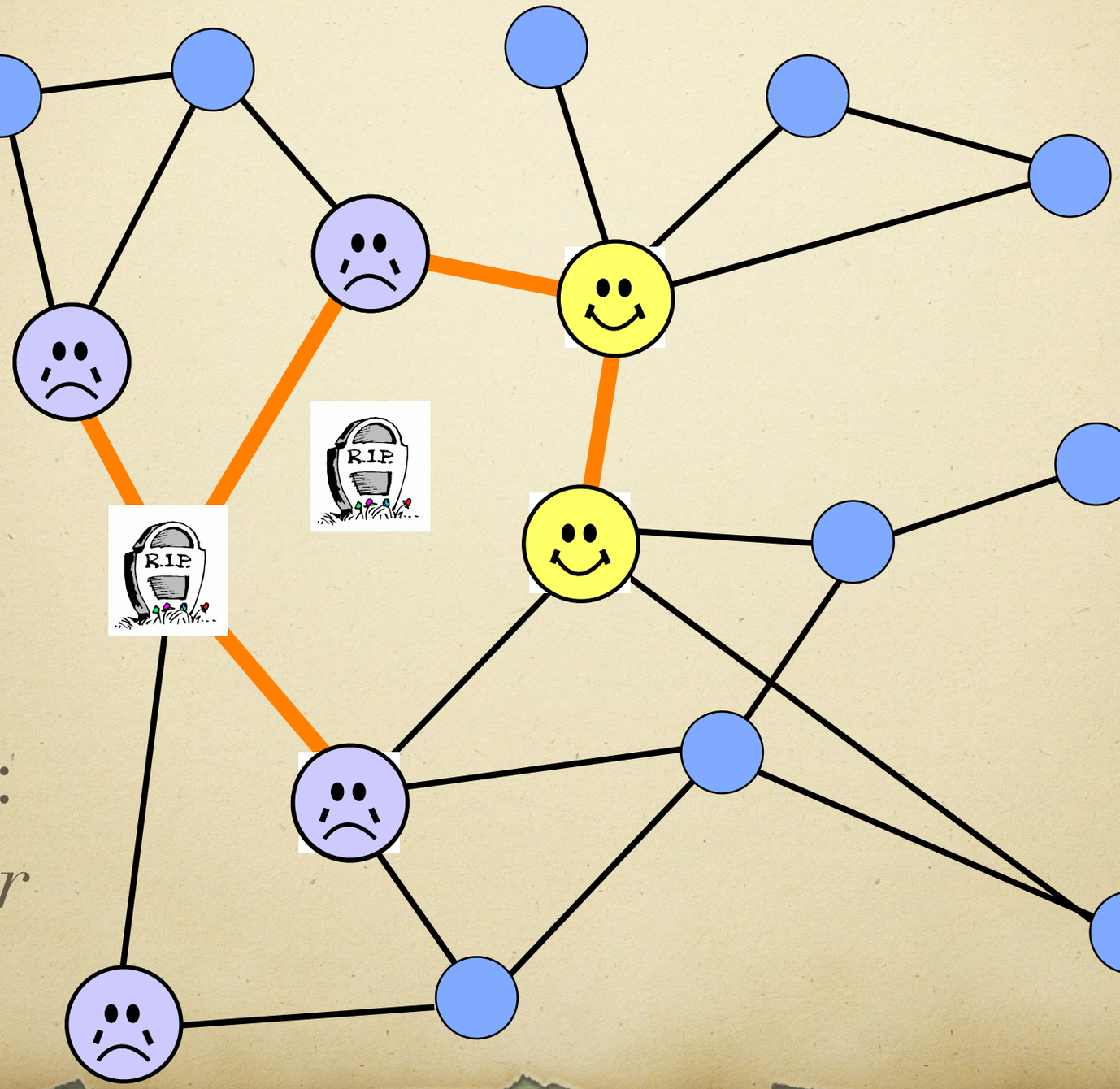


Healer!

Illustration

Attacker!

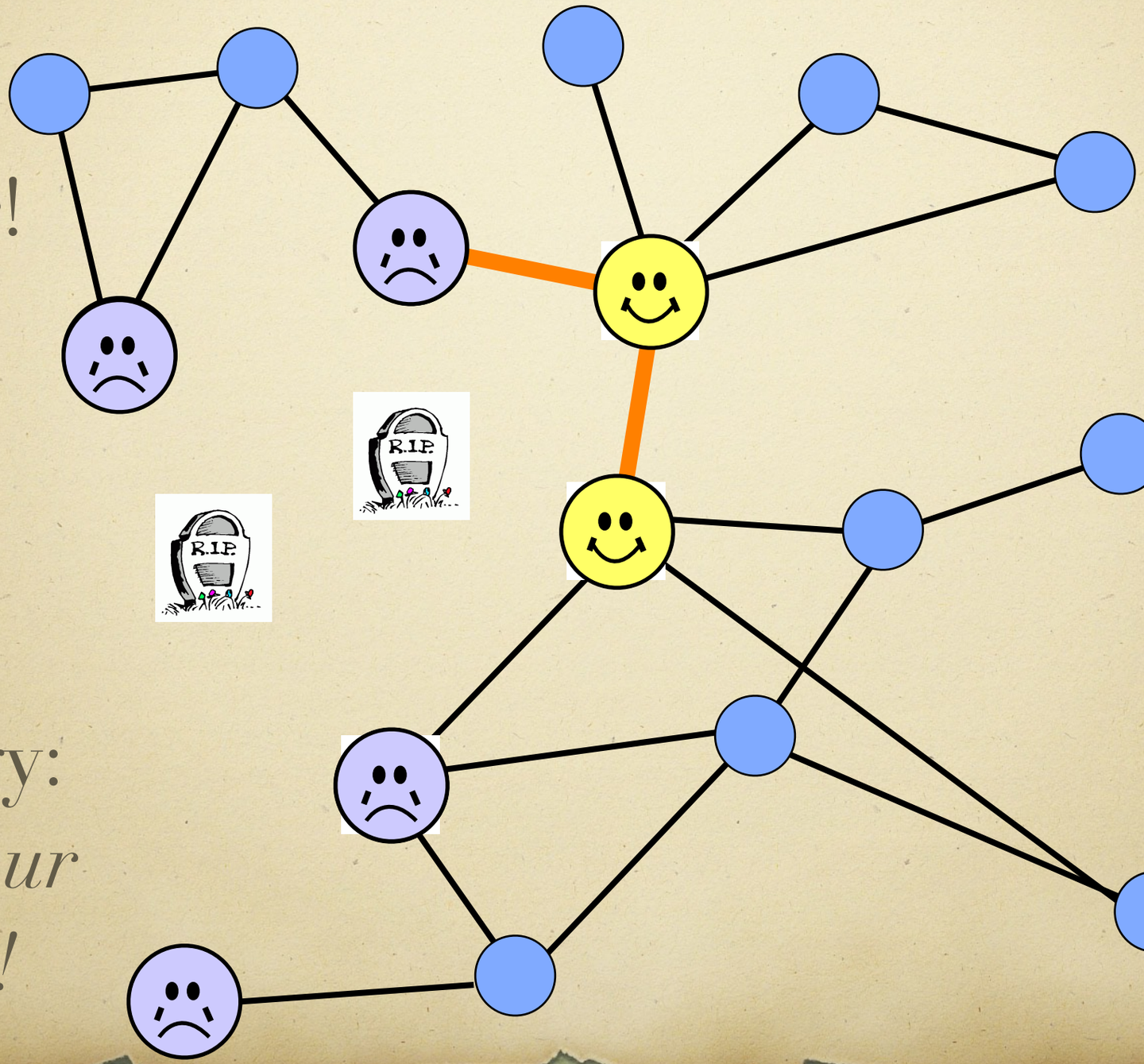
Strategy:
Neighbour attack!



Healer!

Illustration

Attacker!



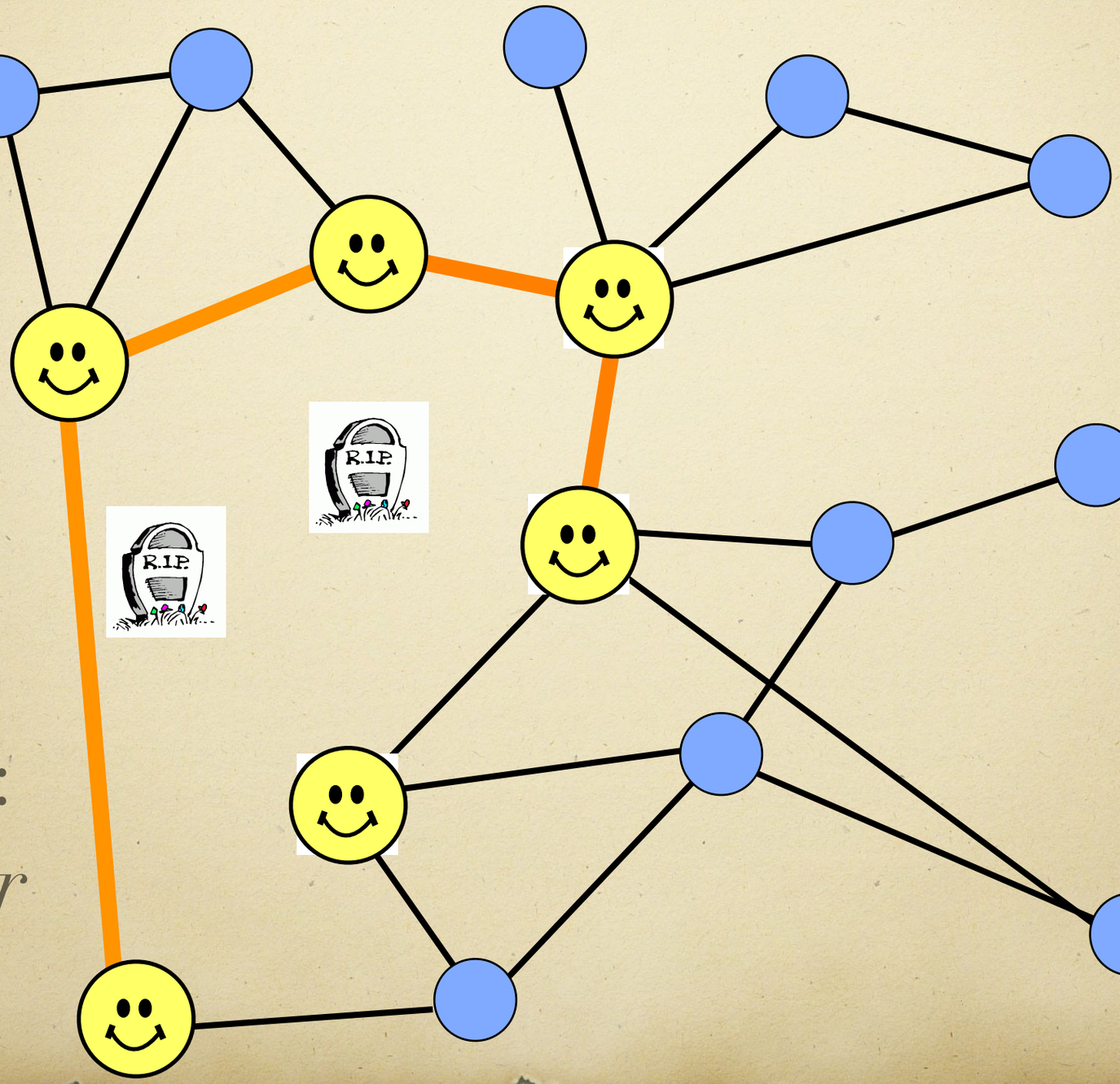
Healer!

Strategy:
Neighbour attack!

Illustration

Attacker!

Strategy:
Neighbour attack!

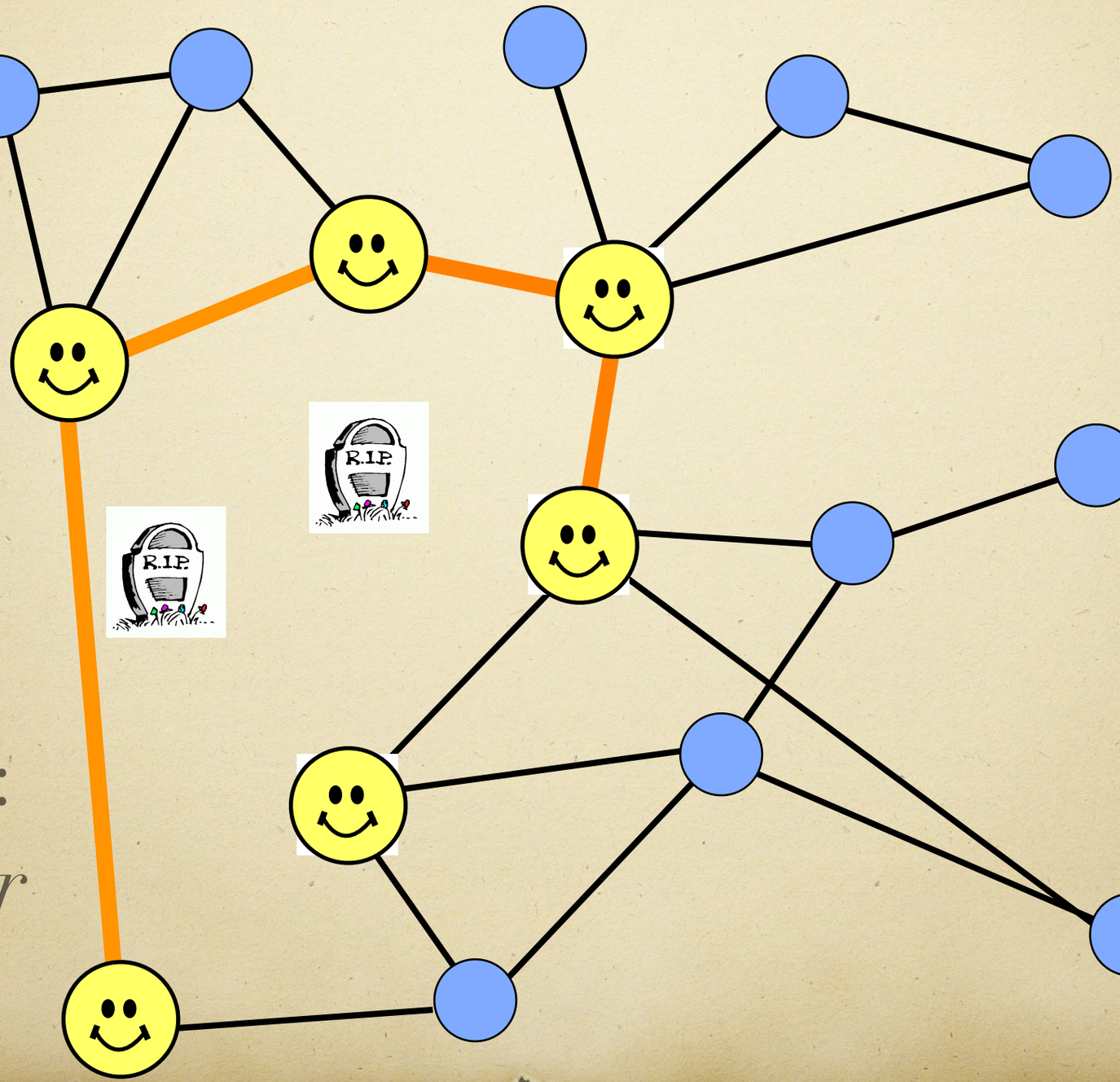


Healer!

Naive?

Attacker!

Strategy:
Neighbour attack!

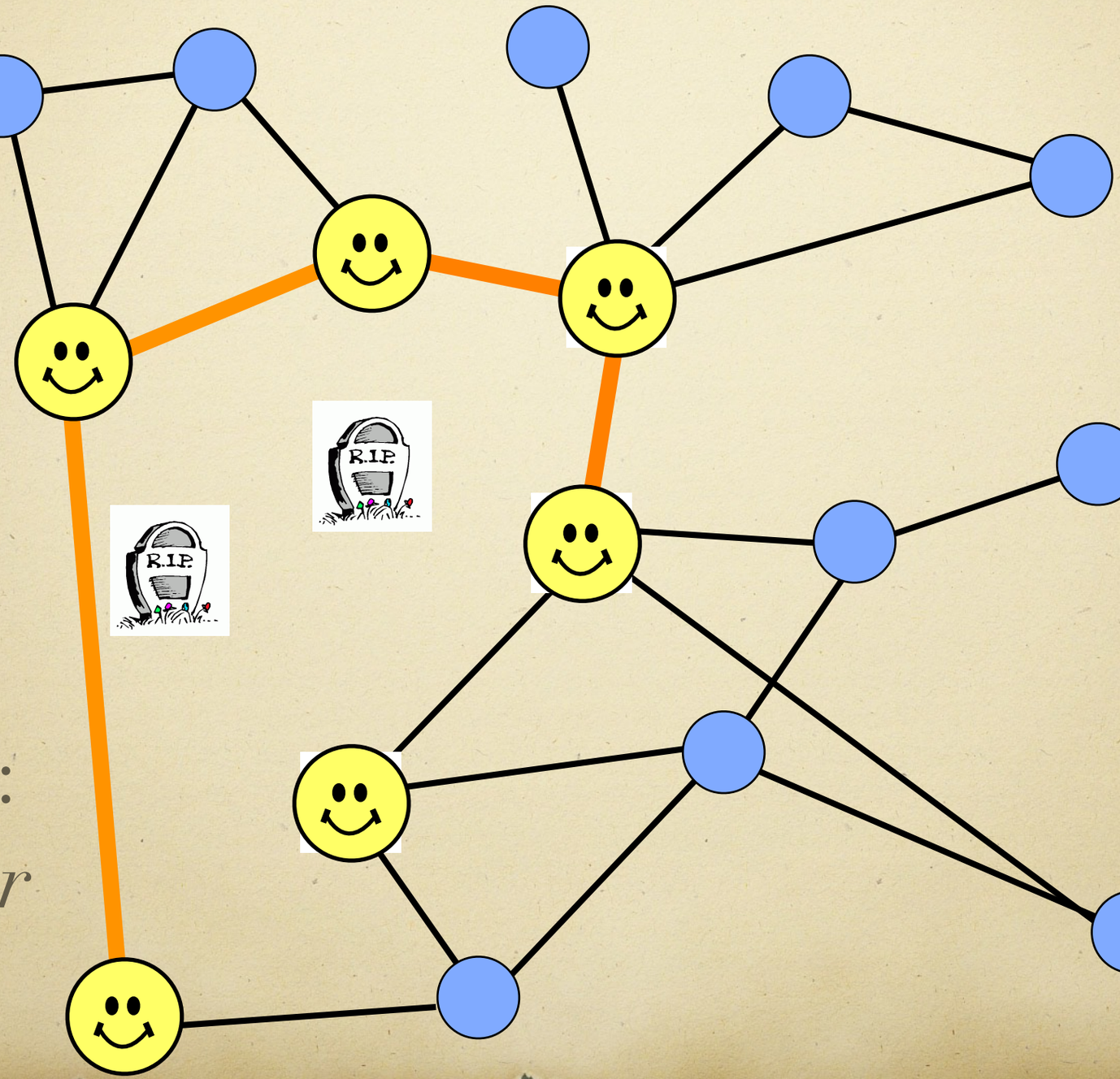


Healer!

Illustration

Attacker!

Healer!

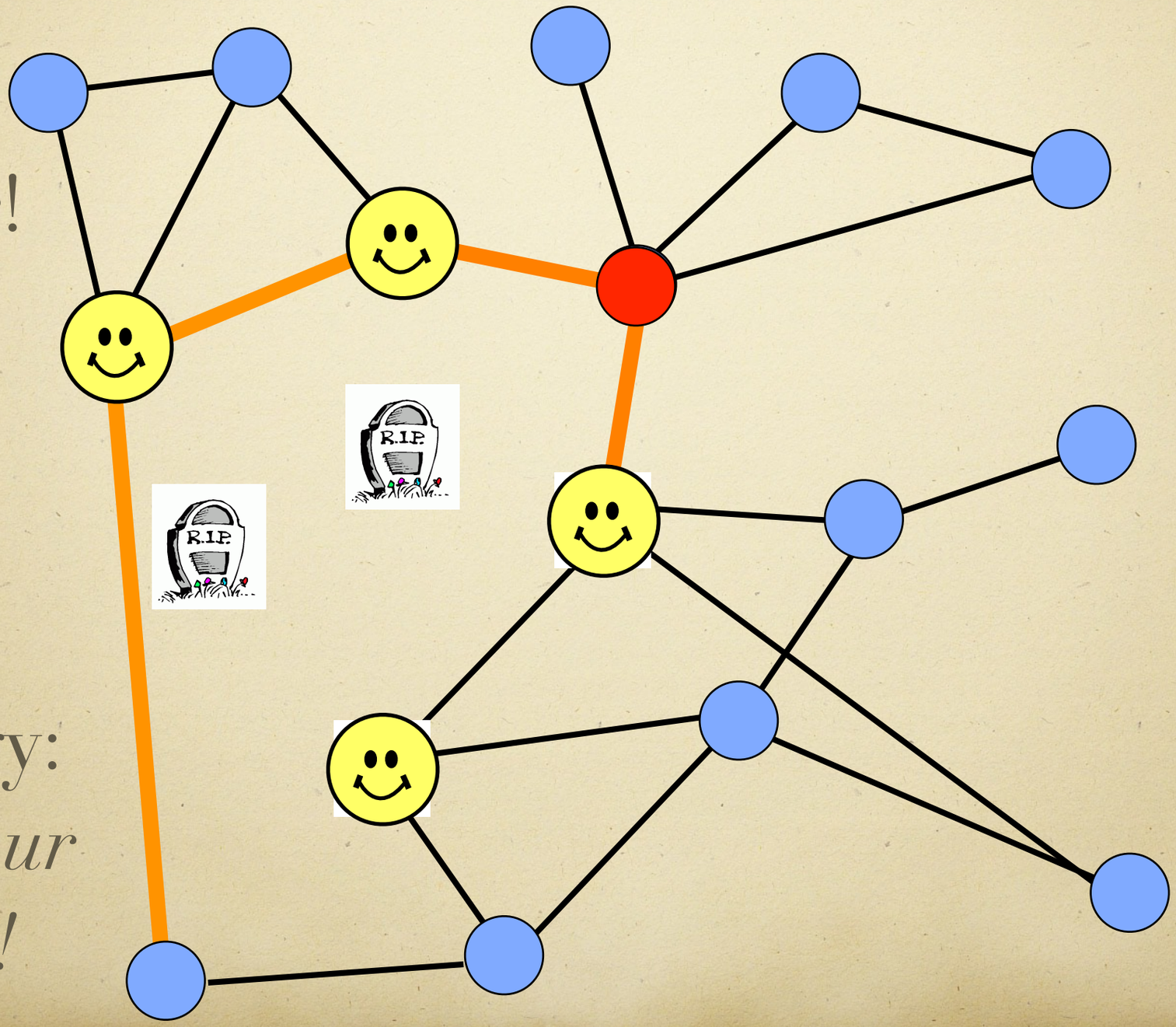


Strategy:
Neighbour attack!

Illustration

Attacker!

Healer!

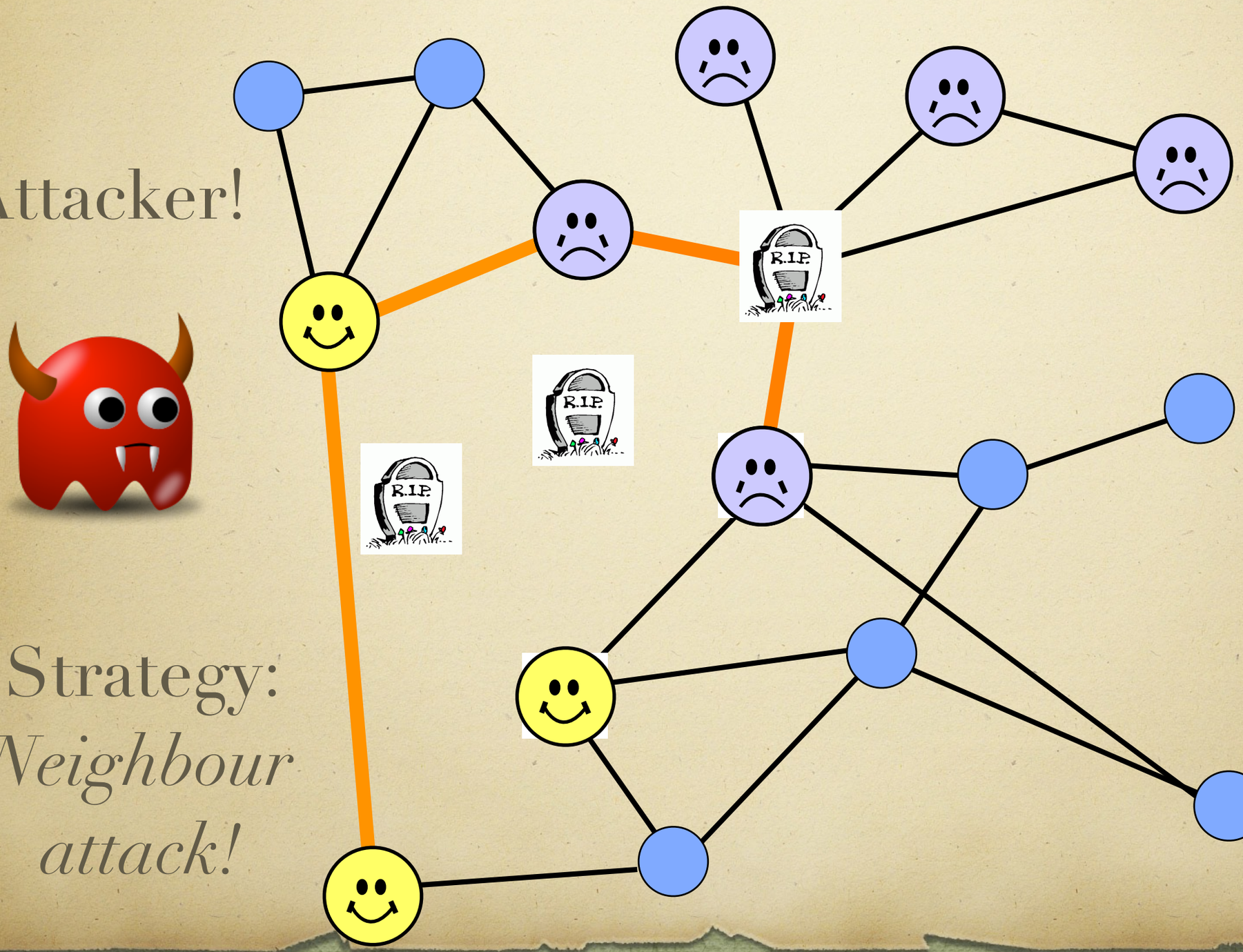


Strategy:
Neighbour attack!

Illustration

Attacker!

Healer!



Strategy:
Neighbour attack!

Illustration

Attacker!

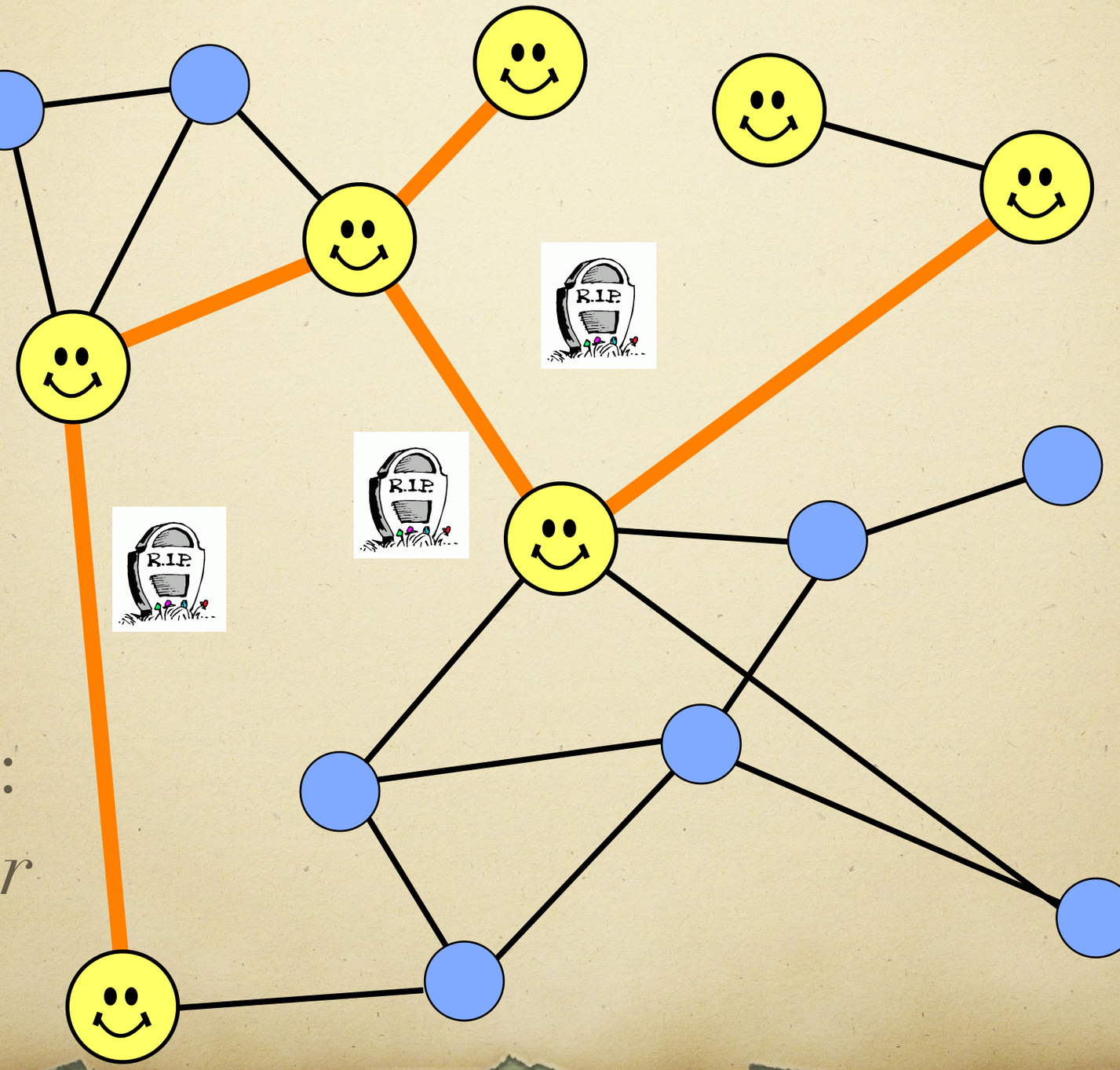
Healer!

Strategy:
Neighbour attack!

Illustration

Attacker!

Strategy:
Neighbour attack!

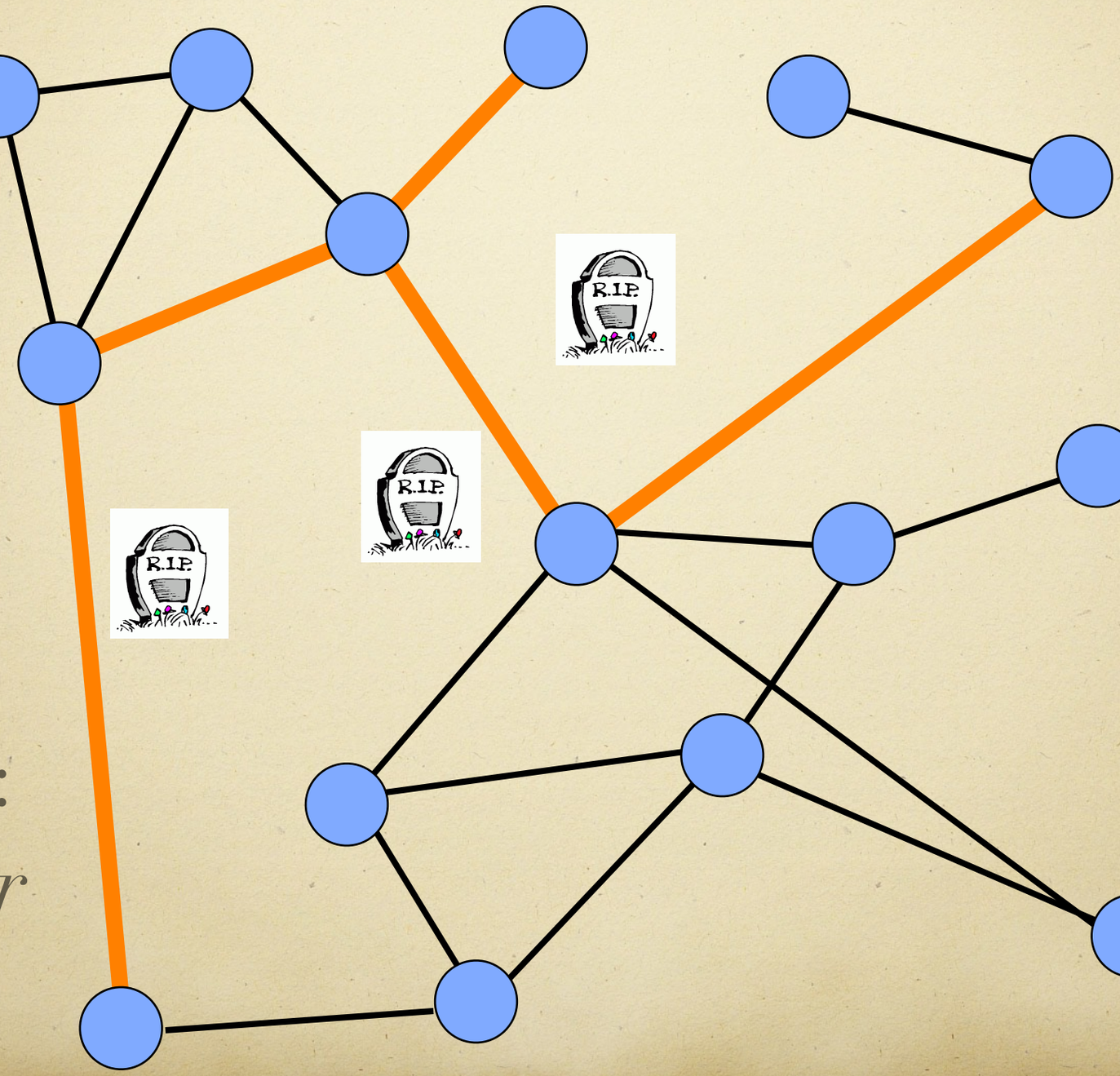


Healer!

Illustration

Attacker!

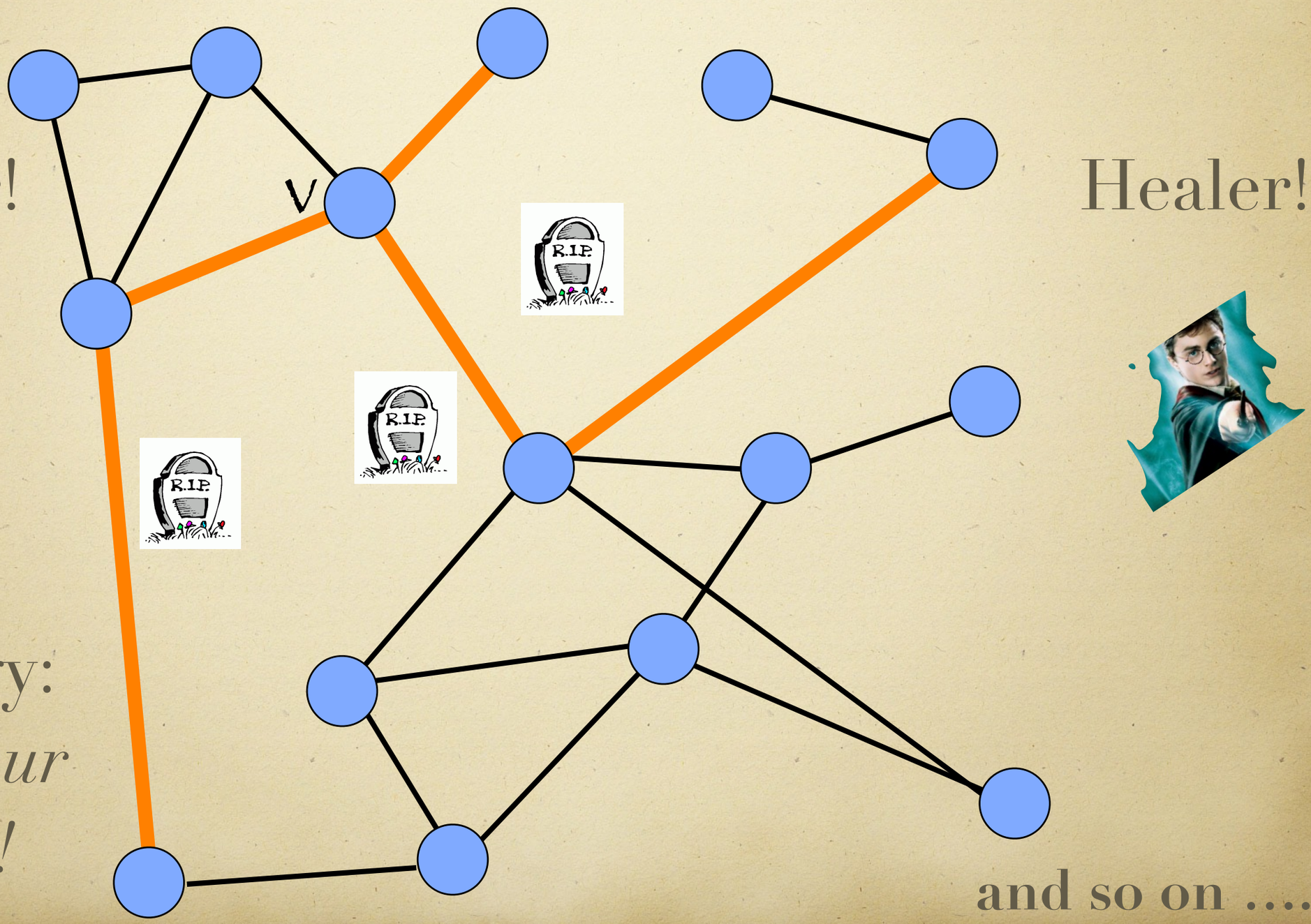
Strategy:
Neighbour attack!



Healer!

Illustration

Attacker!



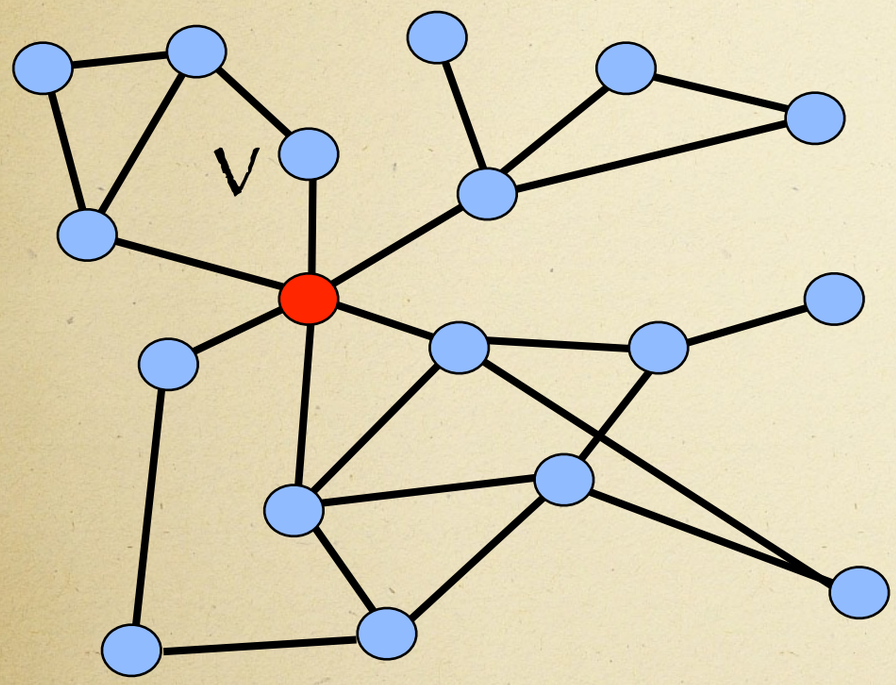
Healer!

Strategy:
Neighbour attack!

and so on ....

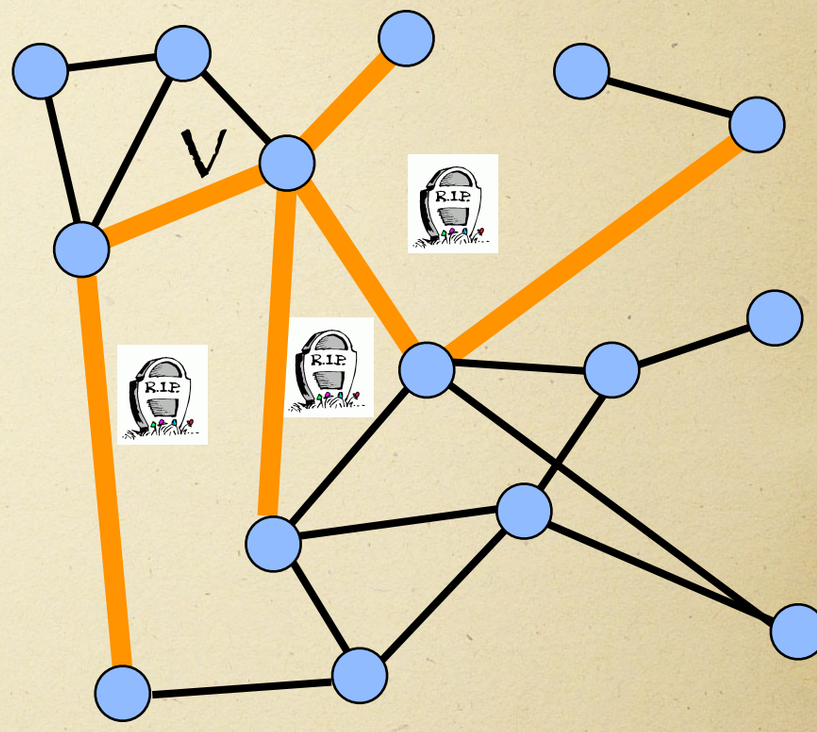
- Suggest different healing strategies and reflect on them wrt node distances (~network latency) and individual node degree increase (~processing load)!

Problem



G_0

Degree(v,G_0) = 2

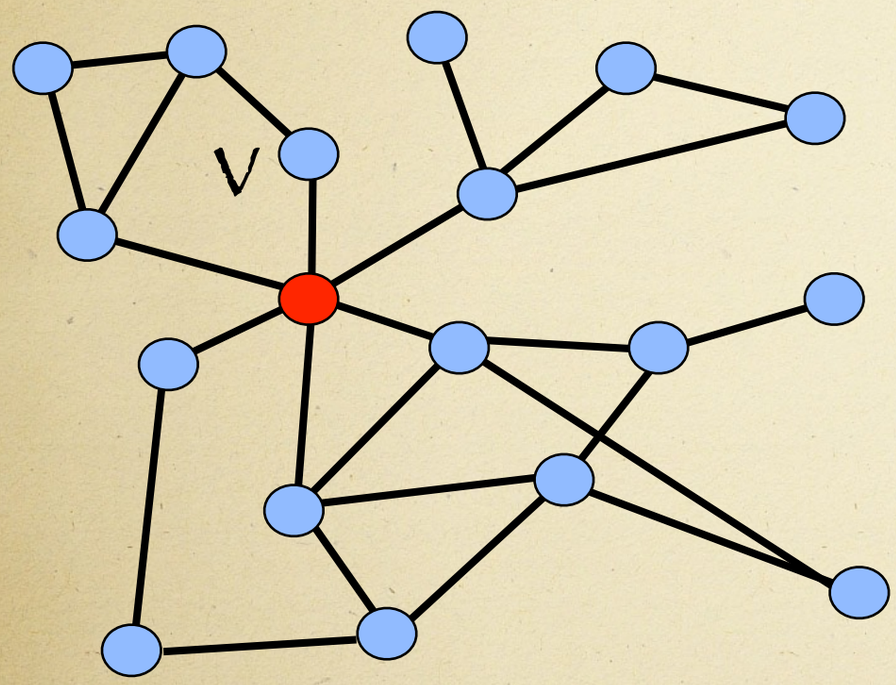


G_3

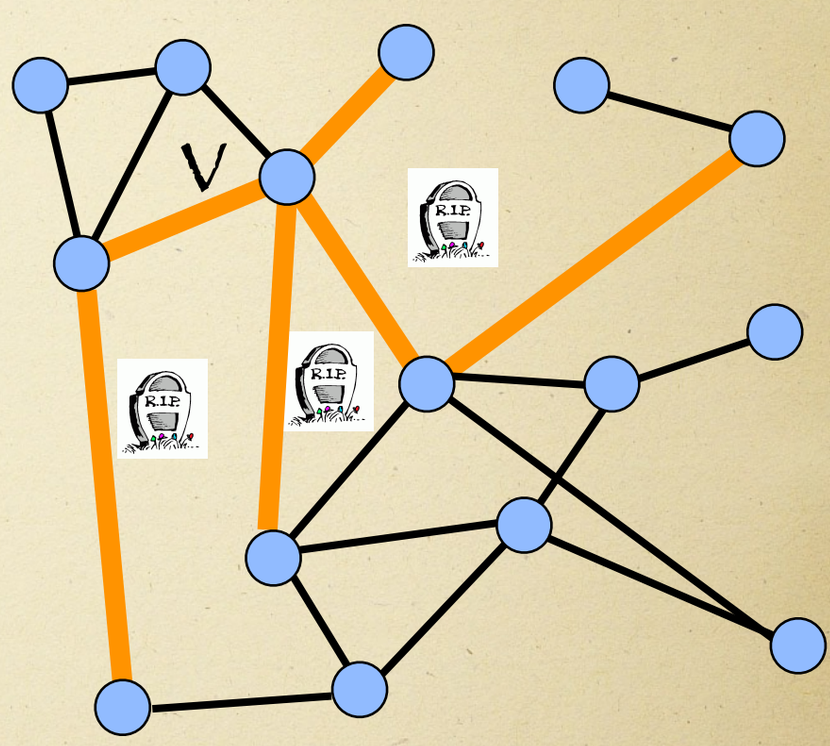
Degree(v,G_3) = 5

Possible healing topologies:

Line Graph



G_0

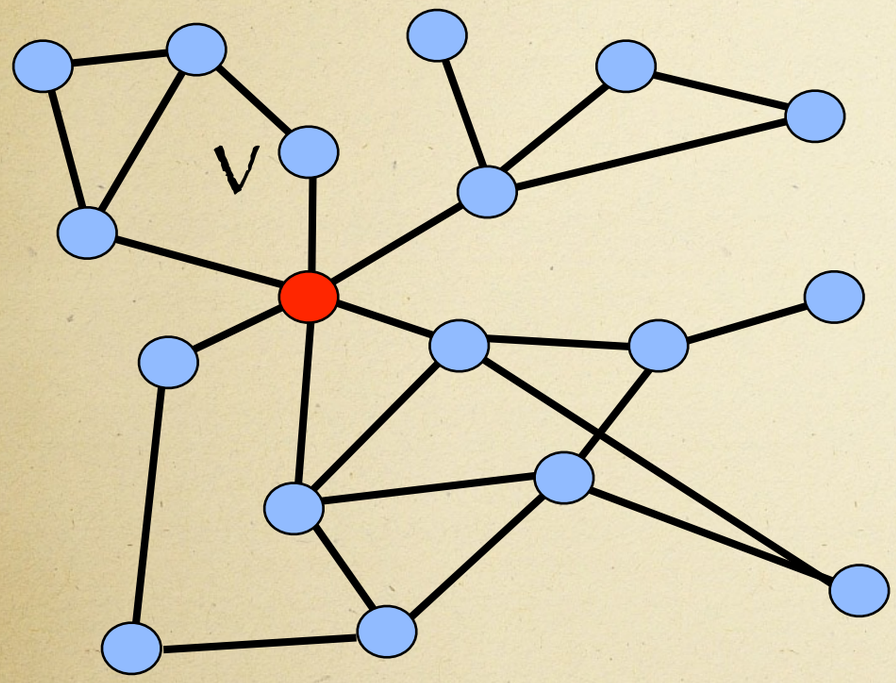


G_3

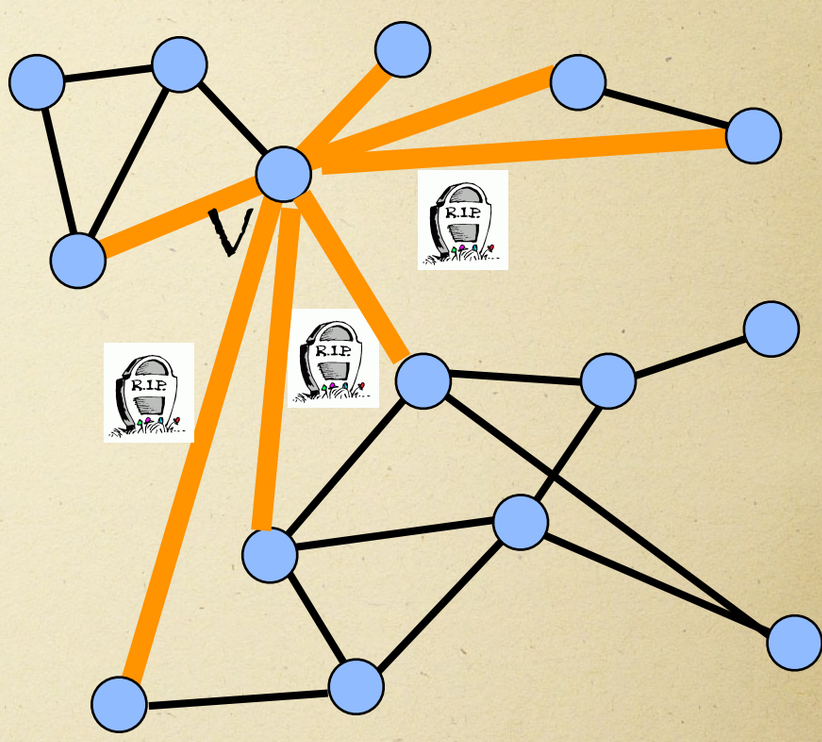
Low degree increase but diameter/ distances blow up

Possible healing topologies:

Star Graph



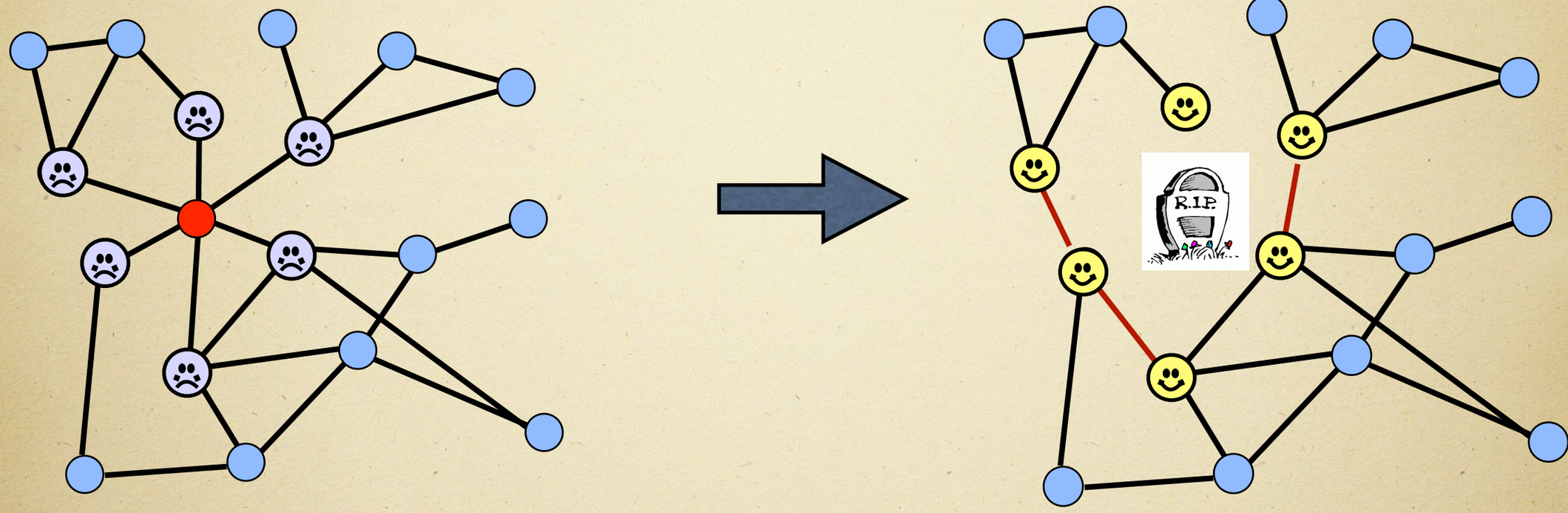
G_0



G_3

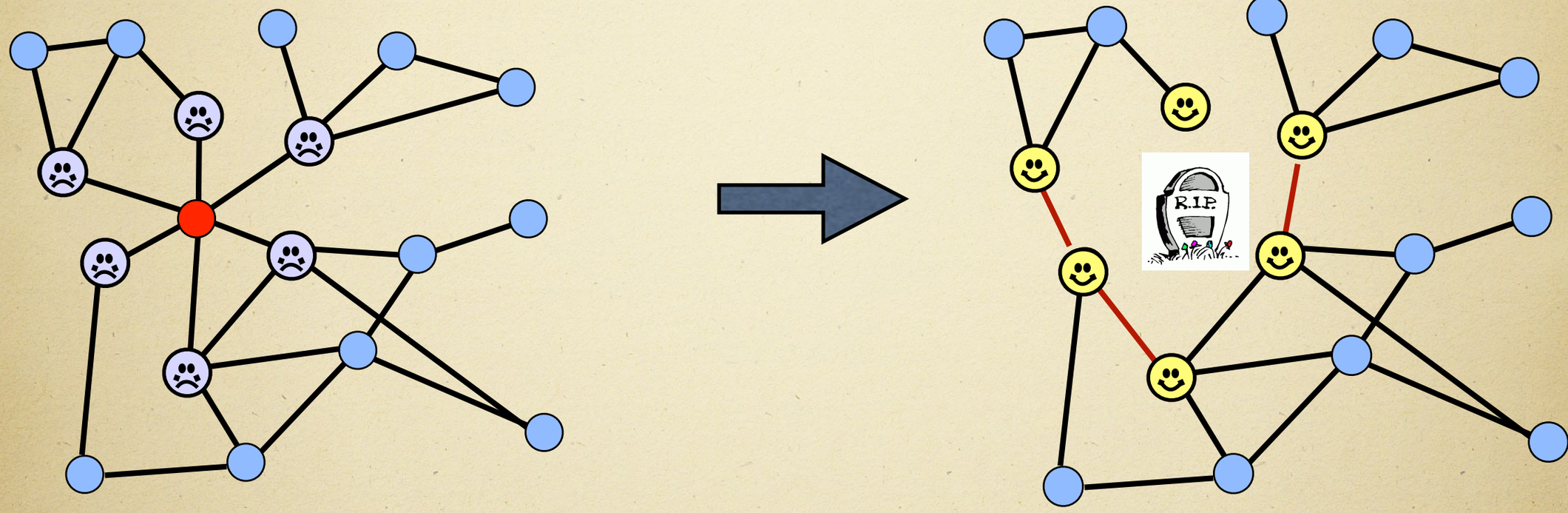
Low distances but degree blows up

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 constraint*

Self-healing (topological) Goals

- Healing should be fast.
- Certain (topological etc…) properties should be maintained within bounds:
  - Connectivity
  - Degree (quantifies the work done by algorithm)
  - Diameter/ Stretch
  - Expansion/ Spectral properties
  - *<Add your own> ….*
- Cost Metrics: Time, #messages per healing and change in properties (dilation) over lifetime.

A Self-healing System

- System: a responsive reconfigurable system
- Attack action: An adversary <attacks>
- Healing action: After each adversary action, we <*repair*> locally (and quickly)!

Why *(P2P) Self-healing*?

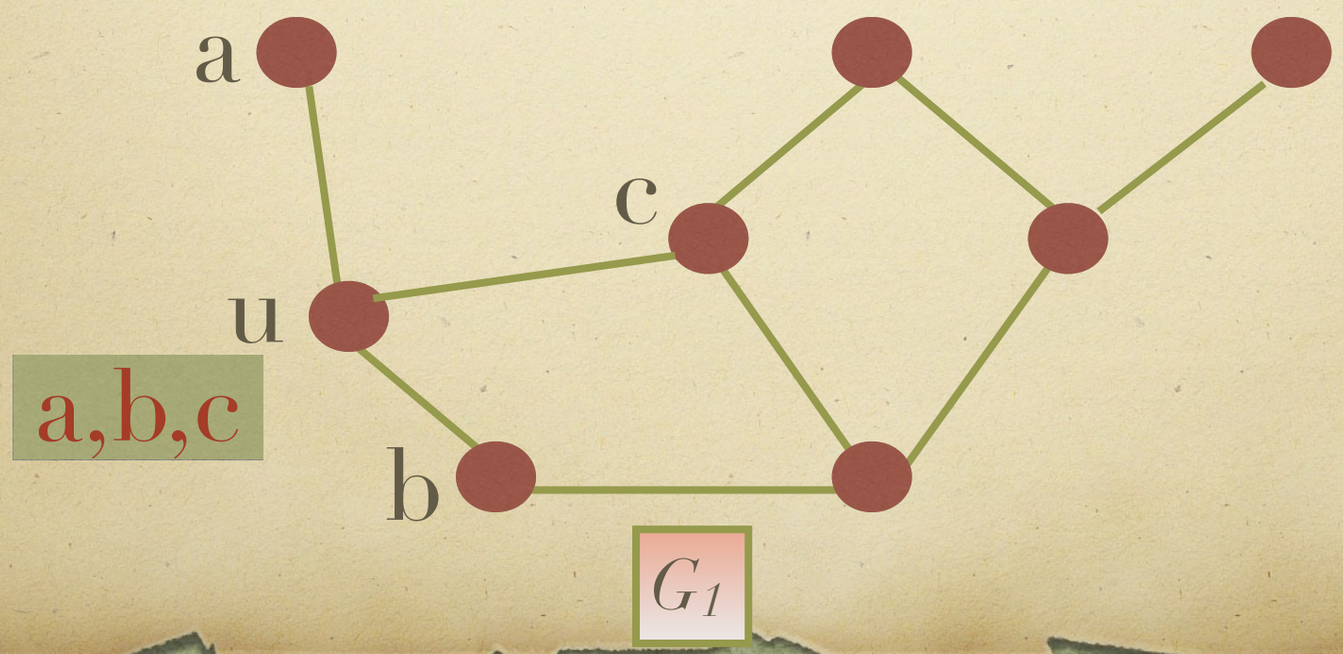
Start from an arbitrary P2P network, converge to a `good' topology (e.g. an expander network), and then maintain it using self-healing.

Formal Distributed model: P2P-CONGEST

[*DConstructor*…, Gilbert, Pandurangan, Robinson, T, ACM PODC'20]

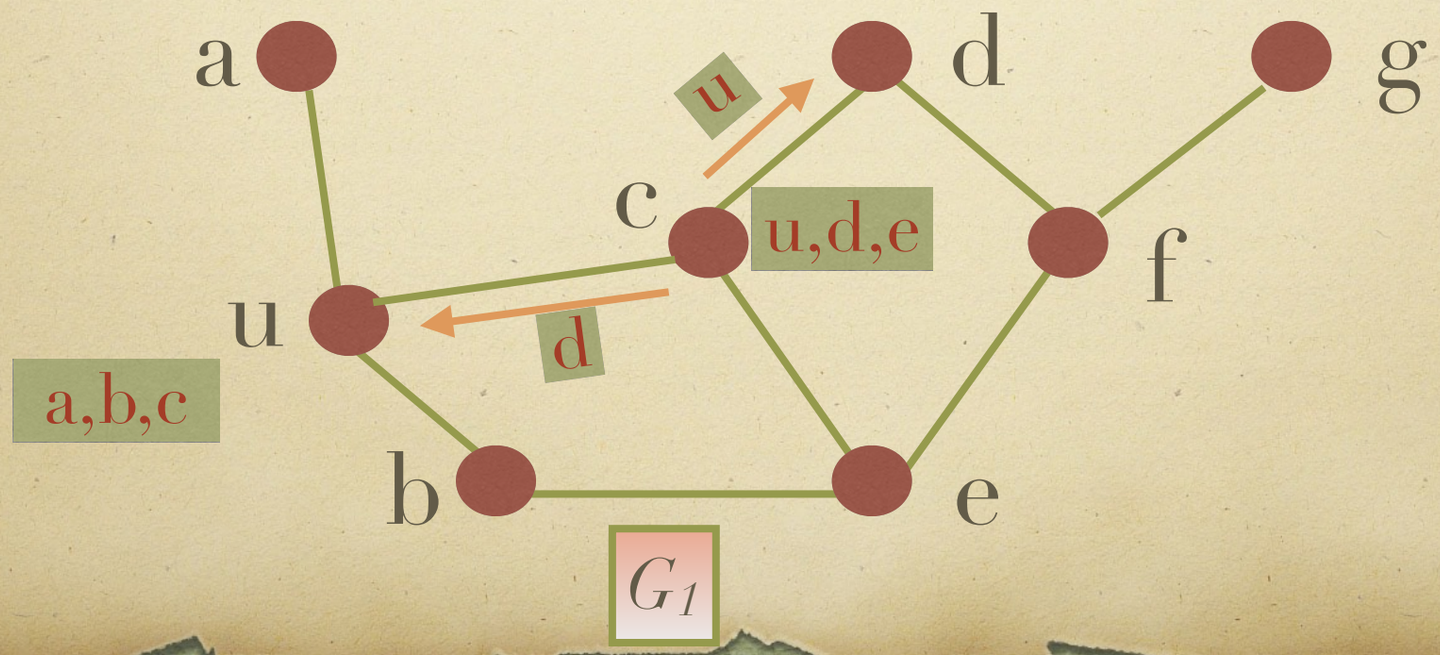
- **P2P-CONGEST: CONGEST** model extended to P2P(Overlay) Reconfigurable networks:

- Graphs $G_1, G_2, \ldots$, where G_1 is the initial network. Initially, nodes know only neighbours in G_1.



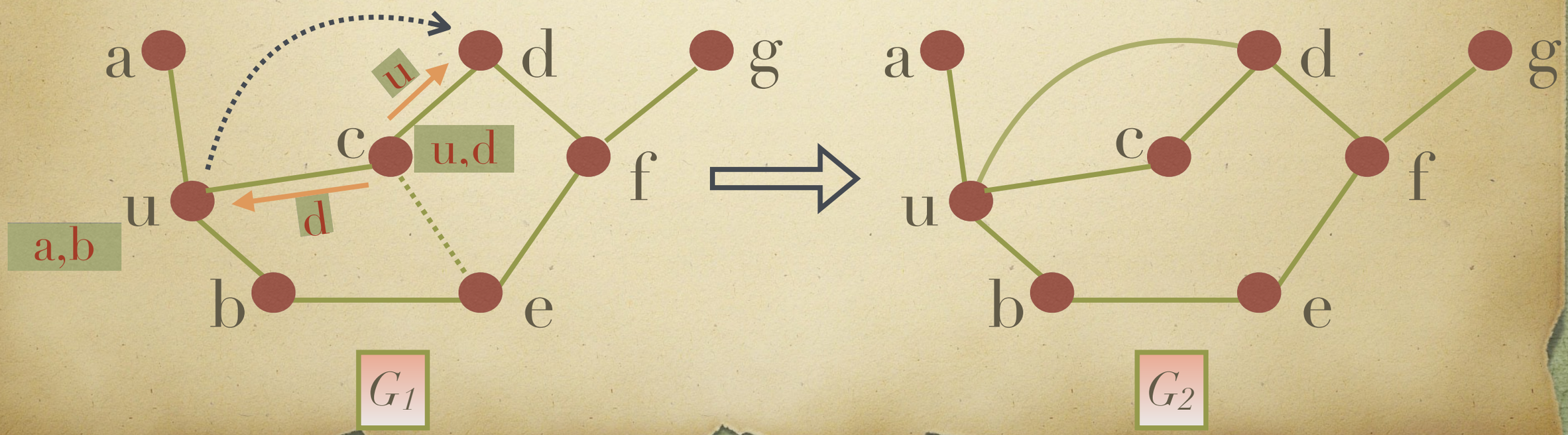
P2P-CONGEST

- **P2P-CONGEST: CONGEST** model extended to P2P(Overlay) Reconfigurable networks:
  - Graphs $G_1, G_2, \ldots$, where G_1 is the initial network. Nodes have unique IDs and know only neighbours in G_1
  - *(Synchronous, CONGEST):* Each round:
    1. nodes receive, process, and send a single message of size O(log n) per neighbour

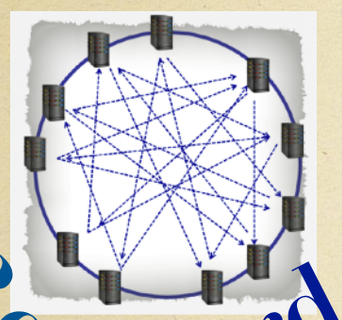


P2P-CONGEST

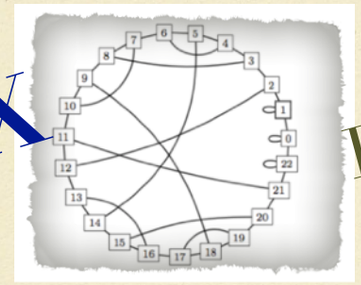
- **P2P-CONGEST: CONGEST** model extended to P2P(Overlay) Reconfigurable networks:
  - Graphs $G_1, G_2, \ldots$, where G_1 is the initial network. Nodes have unique IDs and know only neighbours in G_1
  - *(Synchronous, CONGEST):* Each round:
    1. nodes receive, process, and send a single message of size O(log n) per neighbour
    2. *(Reconfigure)* Nodes can add/drop edges i.e If node u knows ID of node v, it can add or drop edge (u,v). Nodes add/drop edges at end of each round.



P2P Construction and Maintenance



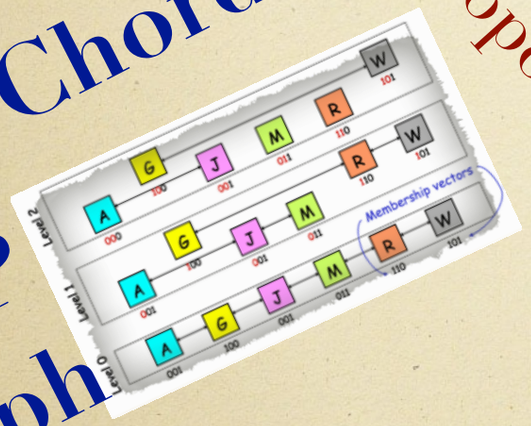
Re-Chord



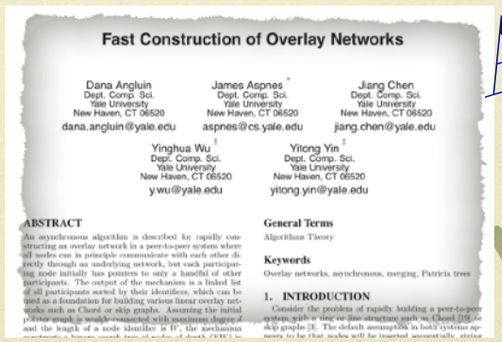
DEX

Fault-tolerant (Self-Healing) ✓

✗ Fixed topologies

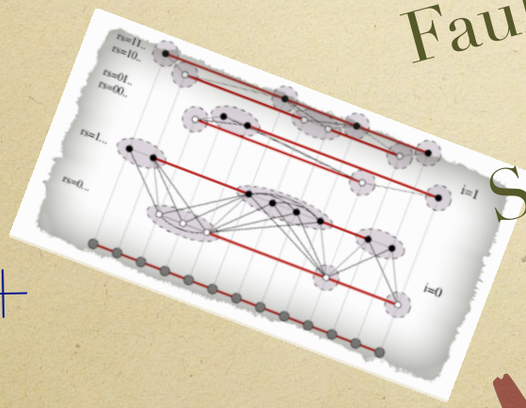


SKIP Graph



Angluin et al.

✓ Asynchronous



SKIP+

Fault-tolerant (Self-Stabilising) ✓

Large Messages (LOCAL) ✗

Not Fault/Churn tolerant ✗

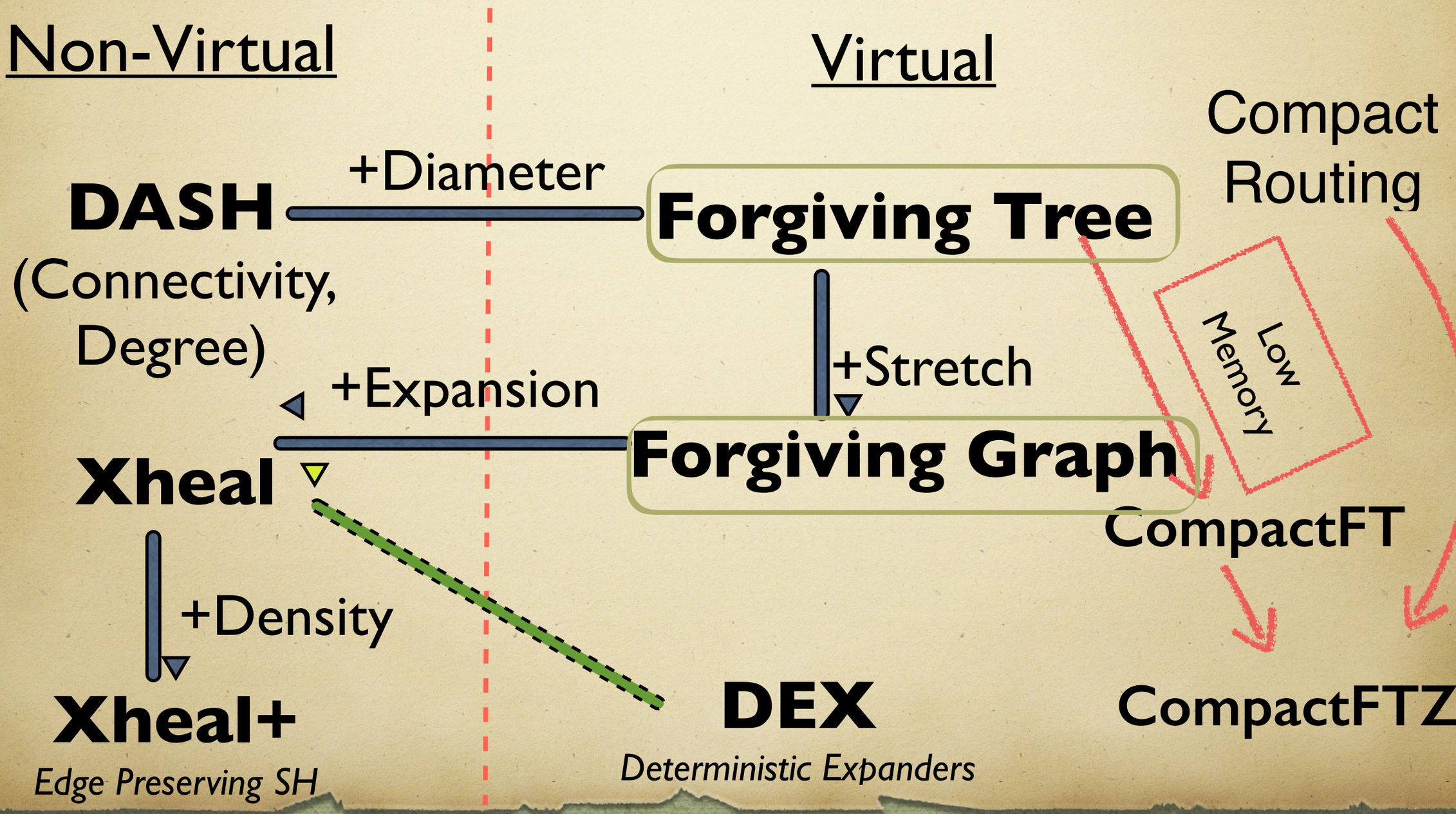
DConstructor (PODC'20)

✓ Asynchronous Self-Repairing ✓

Healing!

Courtesy: https://www.dreamstime.com (royalty free)

Our Self-healing Algorithms



The forgiving tree: a self-healing distributed data structure

Tom Hayes, Navin Rustagi, Jared Saia, Amitabh Trehan, ACM PODC' 2008

The Forgiving Tree: A Self-Healing Distributed Data Structure

Tom Hayes* Navin Rustagi † Jared Saia † Amitabh Trehan †

ABSTRACT

We consider the problem of self-healing in peer-to-peer networks that are under repeated attack by an omniscient adversary. We assume that the following process continues for up to n rounds where n is the total number of nodes initially in the network: the adversary deletes an arbitrary node from the network, then the network responds by quickly adding a small number of new edges.

We present a distributed data structure that ensures two key properties. First, the diameter of the network is never more than $O(\log \Delta)$ times its original diameter, where Δ is the maximum degree of the network initially. We note that for many peer-to-peer systems, Δ is polylogarithmic, so the diameter increase would be a $O(\log \log n)$ multiplicative factor. Second, the degree of any node never increases by more than 3 over its original degree. Our data structure is fully distributed, has $O(1)$ latency per round and requires each node to send and receive $O(1)$ messages per round. The data structure requires an initial setup phase that has latency equal to the diameter of the original network, and requires, with high probability, each node v to send $O(\log n)$ messages along every edge incident to v. Our approach is orthogonal and complementary to traditional topology-based approaches to defending against attack.

zation]: Performance of Systems- *Reliability, availability, and serviceability*; E.1 [**Data**]: Data Structures - *Distributed data structures, Graphs and networks, Trees*; H.3.4 [**Information Systems**]: Information Storage and Retrieval: Systems and Software - *Distributed systems, Information networks*

General Terms

Algorithms, Design, Reliability, Security, Theory

Keywords

Self-healing, reconfigurable, responsive, distributed, networks, peer-to-peer, data structure

1. INTRODUCTION

Many modern networks are *reconfigurable*, in the sense that the topology of the network can be changed by the nodes in the network. For example, peer-to-peer, wireless and mobile networks are reconfigurable. More generally, many social networks, such as a company's organizational chart; infrastructure networks, such as an airline's transportation network; and biological networks, such as the human brain, are also reconfigurable. Unfortunately, our mathematical and algorithmic tools have not developed to the

The Forgiving Tree: Model

- Start: a network G.
- Nodes fail in unknown order $v_1, v_2, \ldots, v_n$
- (P2P-Congest) After each node deletion, we can add and/or drop some edges between pairs of nearby nodes, to "heal" the network

The Forgiving Tree: Model

- Start: a network G.
- Nodes fail in unknown order v_1, v_2, …, v_n
- (P2P-Congest) After each node deletion, we can add and/or drop some edges between pairs of nearby nodes, to "heal" the network

Each node of G_0 is a processor.
Each processor starts with a list of its neighbors in G_0.
Pre-processing: Processors may send messages to and from their neighbors.
for $t := 1$ to n **do**
Adversary deletes a node v_t from G_{t-1}, forming H_t.
All neighbors of v_t are informed of the deletion.
Recovery phase:
Nodes of H_t may communicate (in parallel) with their immediate neighbors. These messages are never lost or corrupted, and may contain the names of other vertices.

During this phase, each node may insert edges joining it to any other nodes as desired. Nodes may also drop edges from previous rounds if no longer required.
At the end of this phase, we call the graph G_t.
end for
Success metrics: Minimize the following "complexity" measures:

1. **Degree increase.** $\max_{t<n} \max_v \text{degree}(v, G_t) - \text{degree}(v, G_0)$
2. **Diameter stretch.** $\max_{t<n} \text{diam}(G_t)/\text{diam}(G_0)$
3. **Communication per node.** The maximum number of bits sent by a single node in a single recovery round.
4. **Recovery time.** The maximum total time for a recover round, assuming it takes 1 bit no more than 1 time unit to traverse any edge and unlimited local computational power at each node.

Model 2.1: The Delete and Repair Model – Distributed View.

The Forgiving Tree: Main Result

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

The Forgiving Tree: Main Result

- A distributed algorithm, Forgiving Tree such that, for any network G with max degree Deg, for an arbitrary sequence of deletions,
- Graph stays connected
- Diameter increases by ≤ log(Deg) factor
- Degrees increase by ≤ 3 (additive)

} Matching lower bound

- Each repair takes constant time and involves O(Deg) nodes.

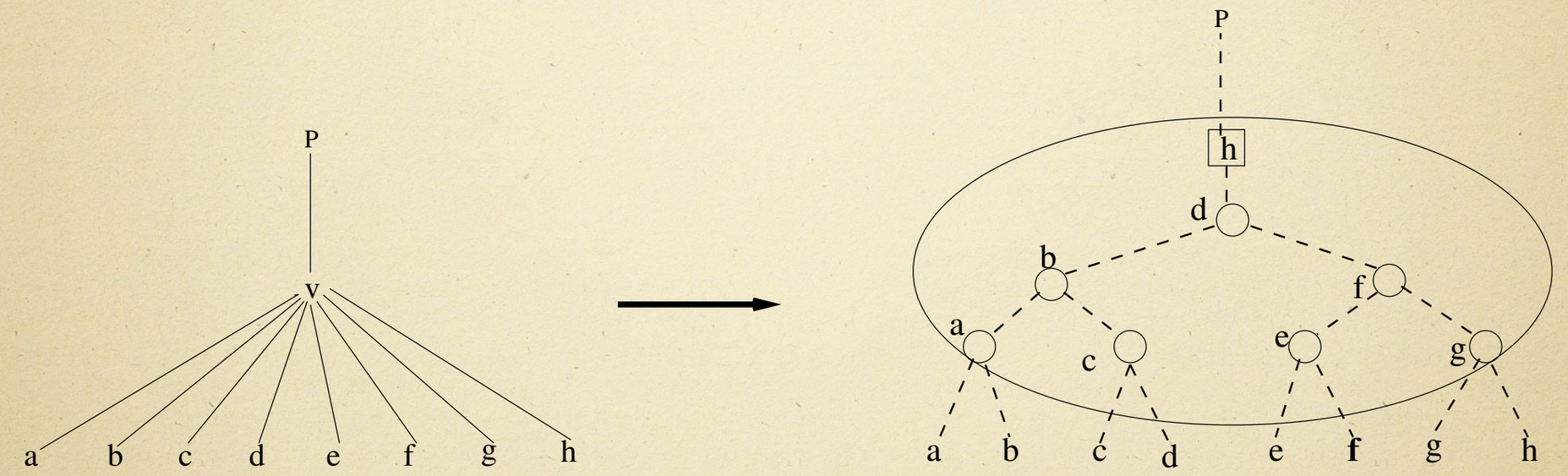
FT: Motivations

- 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

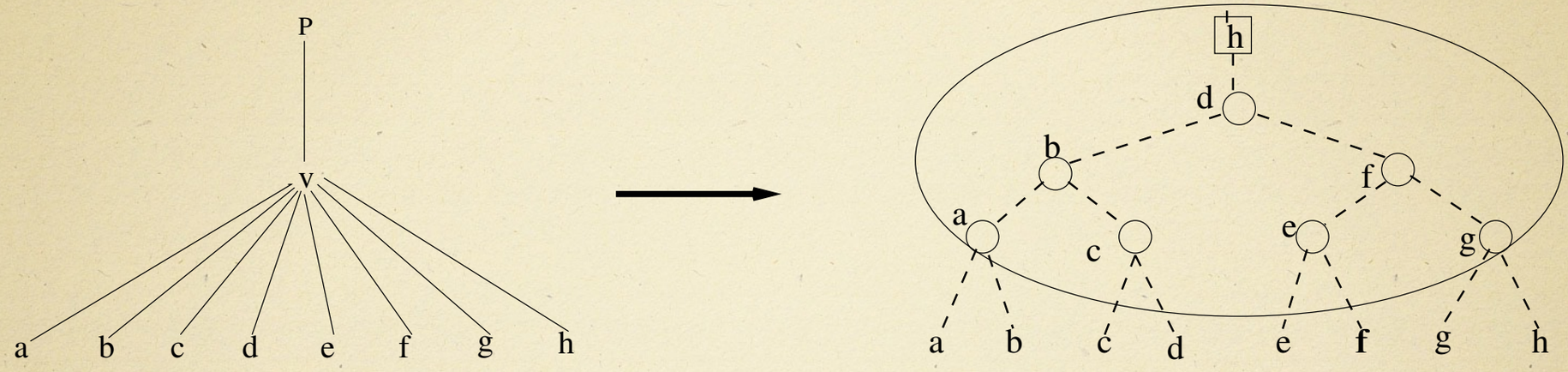
- Find a Dia approximating spanning tree of G.
- Choose some vertex to be the root, and orient all edges toward the root.
- When a node is deleted, replace it by a balanced binary tree of “virtual nodes”
- Short-circuit any redundant virtual nodes
- Somehow the surviving real nodes simulate the virtual nodes

FT and SH: Basic Strategy



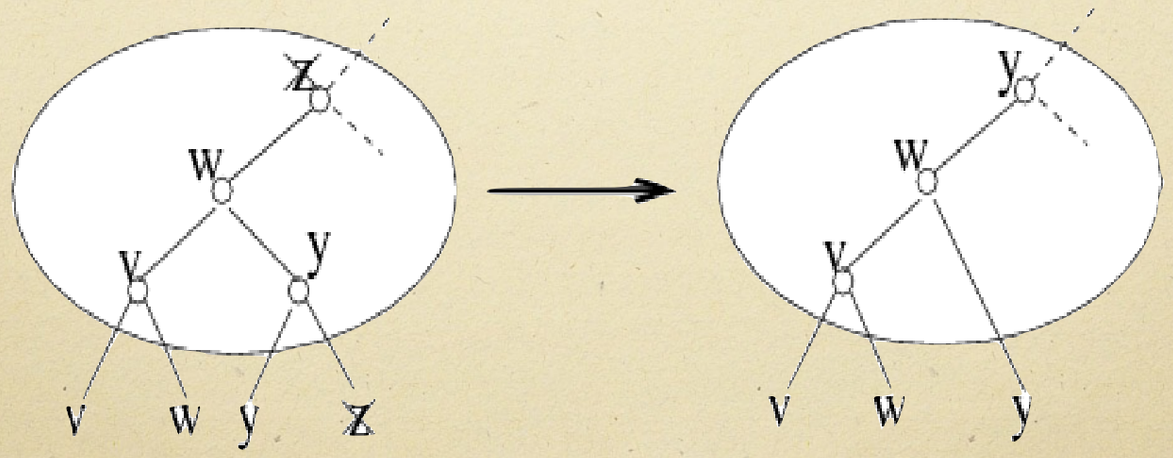
Replacing v by a balanced binary tree of virtual nodes

FT and SH: Basic Strategy



If v is non-leaf:

Replacing v by a balanced binary tree of virtual nodes

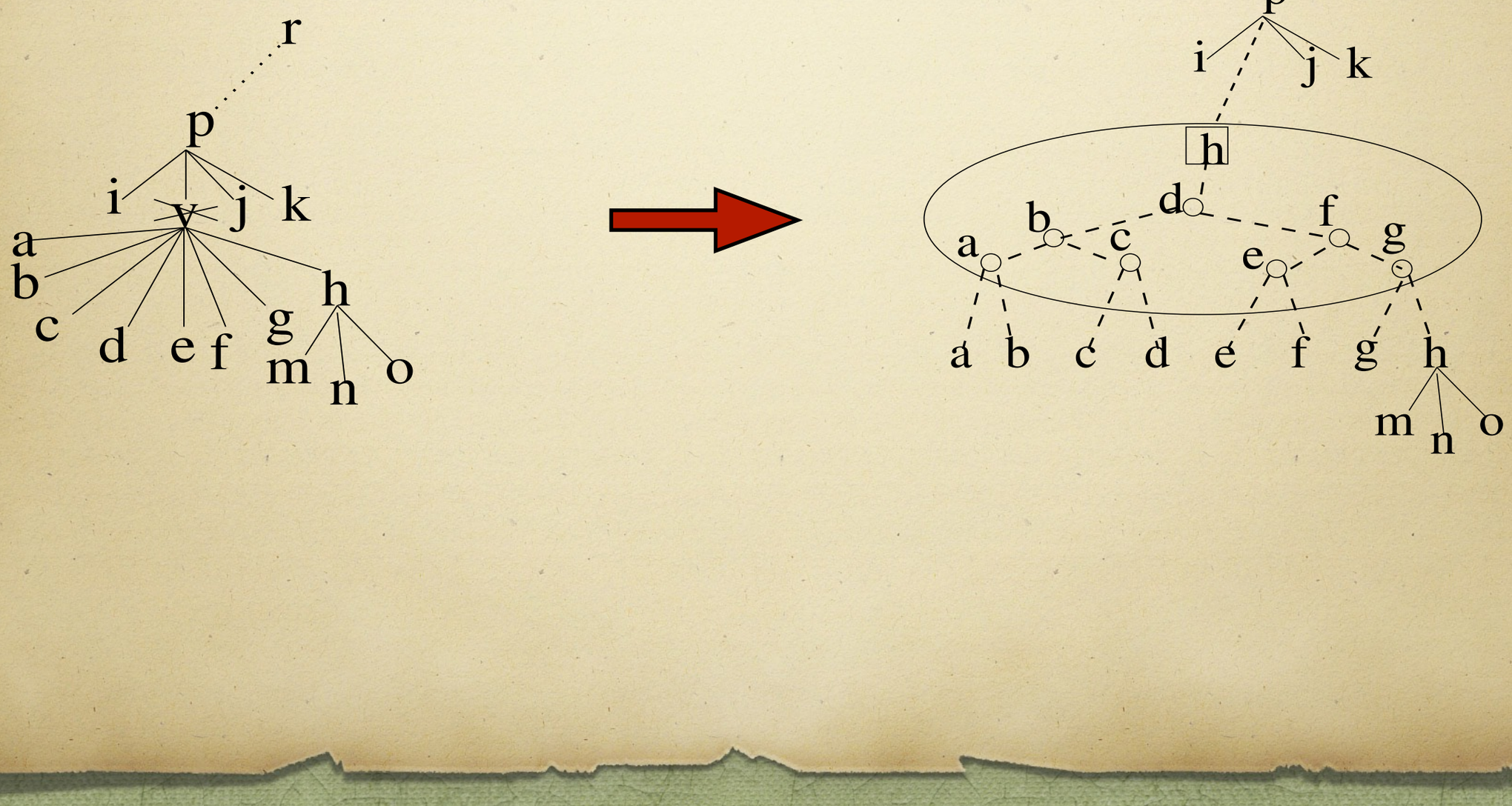


If v is a leaf:

Short-circuiting a redundant virtual node

FT in action

- Node v deleted:

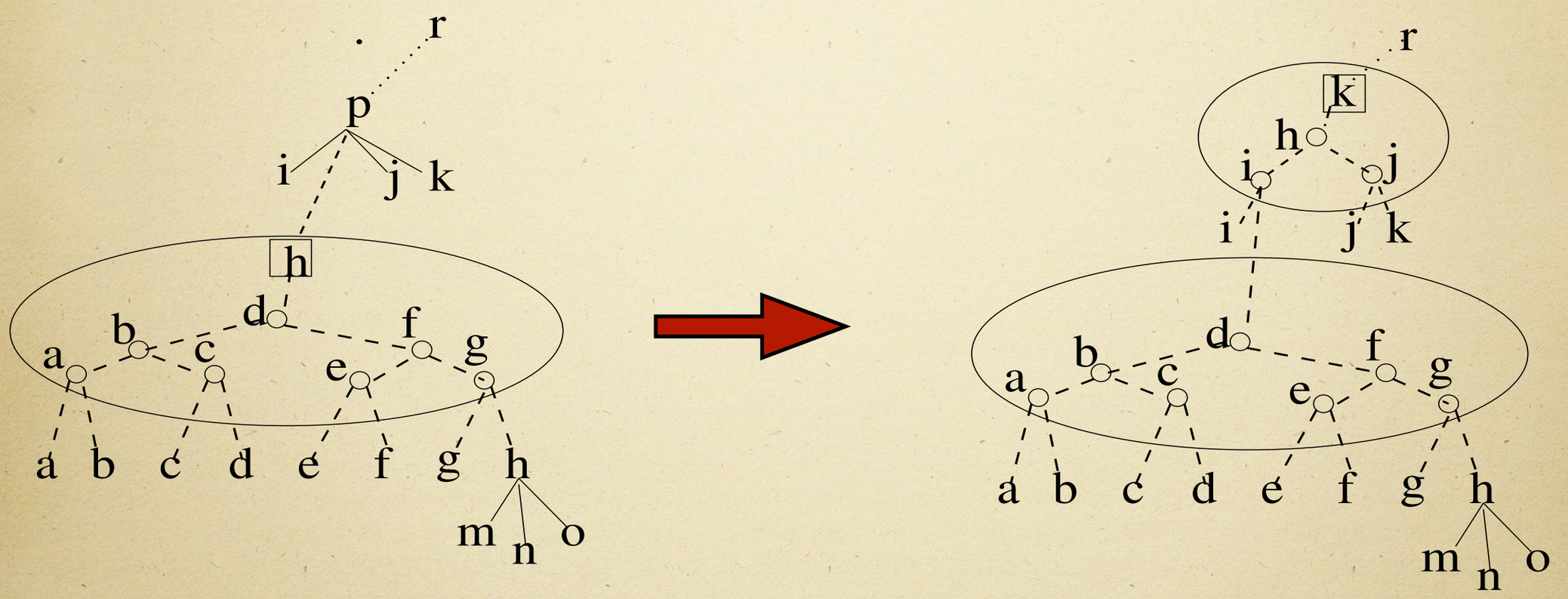


Heirs

- What happens when a node w simulating a virtual node is deleted?
  - (a) If w has children, one child is w's "heir" and takes over the virtual node.
  - (b) if the virtual node becomes redundant, short-circuit!

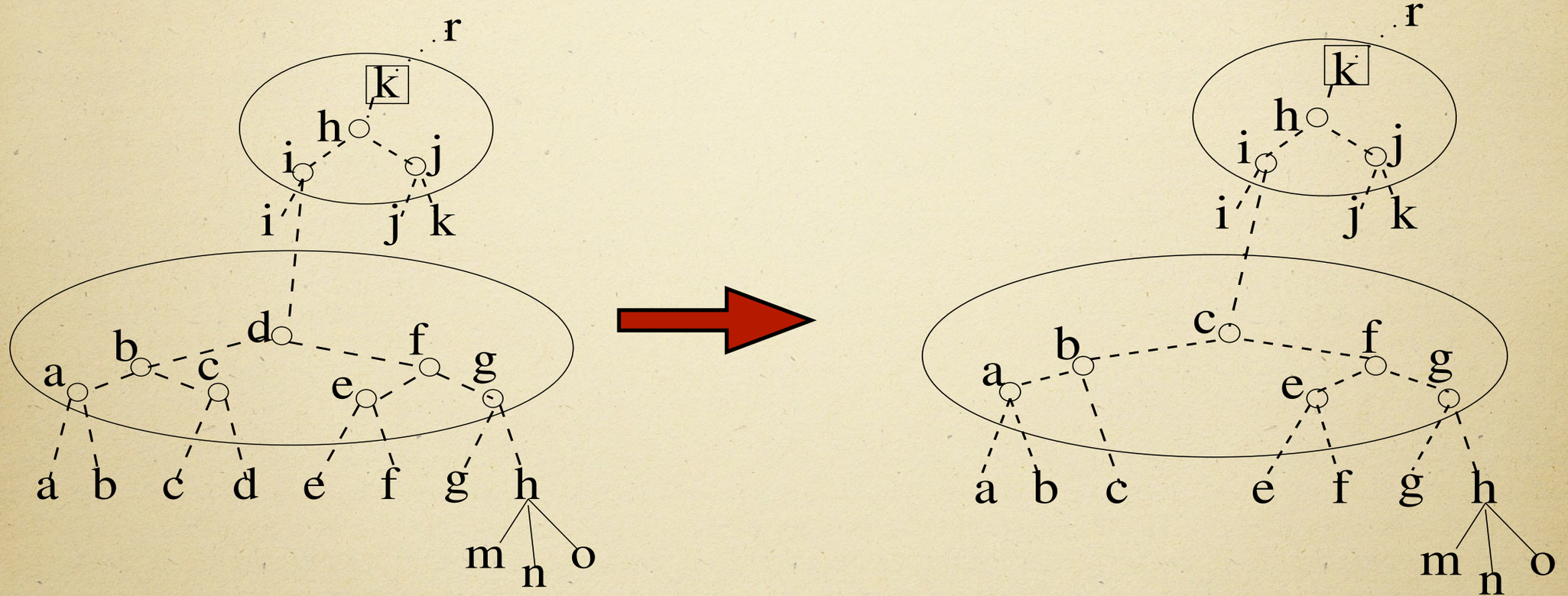
FT in action

- Node p deleted:



FT in action

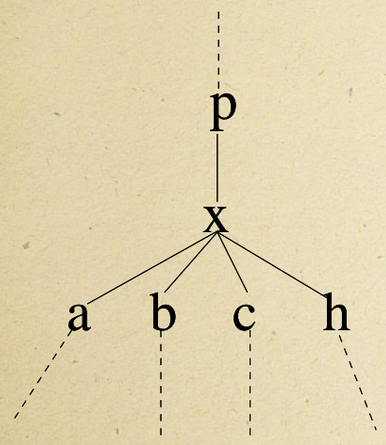
- Node d deleted:

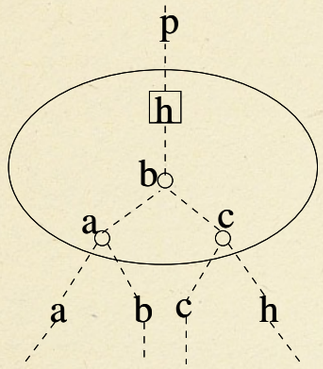


Distributed Implementation: Assigning virtual duties

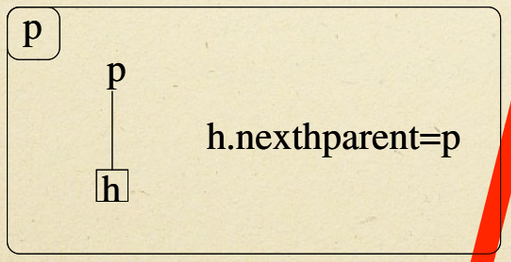
- Each non-leaf node v writes a **will** with duties for its children.
- Give relevant will portions to each child
- Keep the will updated!
- When **v is deleted, v's will is implemented**

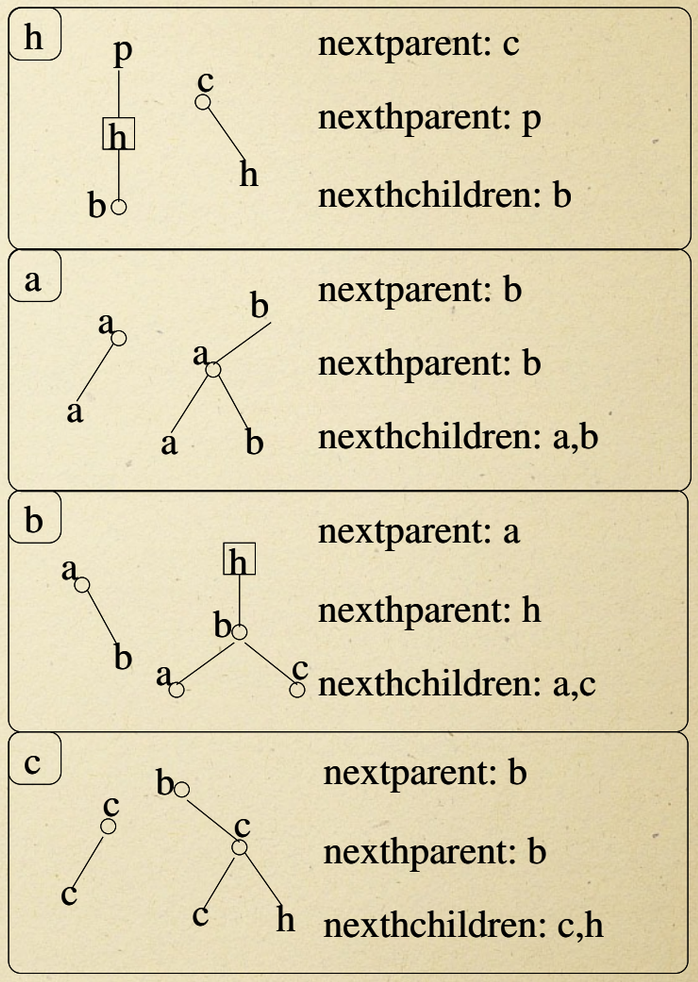
Where there's a will…





RT(x)

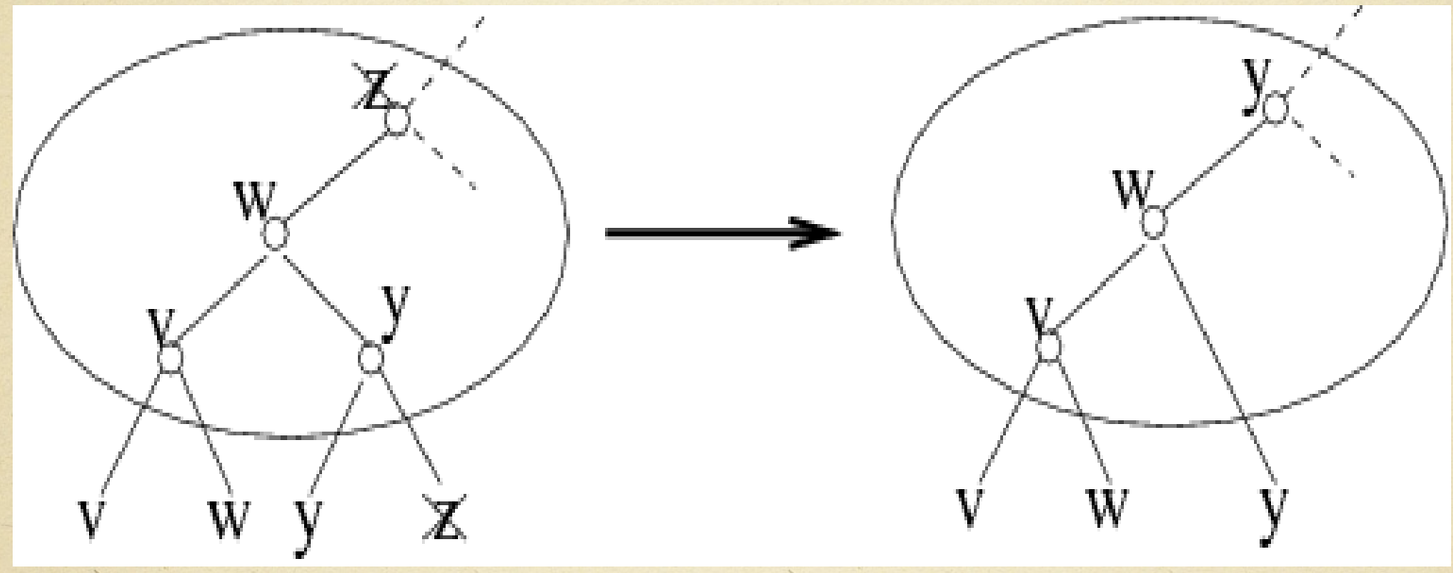




x assigns virtual nodes in his will h is the "heir"

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

Leaf deletions



z is deleted, making y's virtual node redundant.

Now y takes over z's virtual node (z had no heir).

Proofs

Lemma 1. *In the Forgiving Tree, a real node can simulate at most one helper node at a time.*

Lemma 2. *If an original node x is an ancestor of another original node v in FT_i for some time step i, then node x must also have been an ancestor of node v in FT_0.*

Lemma 3. *Let $\mathrm{danc}_i(v)$ be the number of ancestors of v in FT_0 that have been deleted by time step i.*

For all nodes v,

$$\mathrm{depth}_i(v) \leq \mathrm{depth}_0(v) + \log \Delta \times \mathrm{danc}_i(v)$$

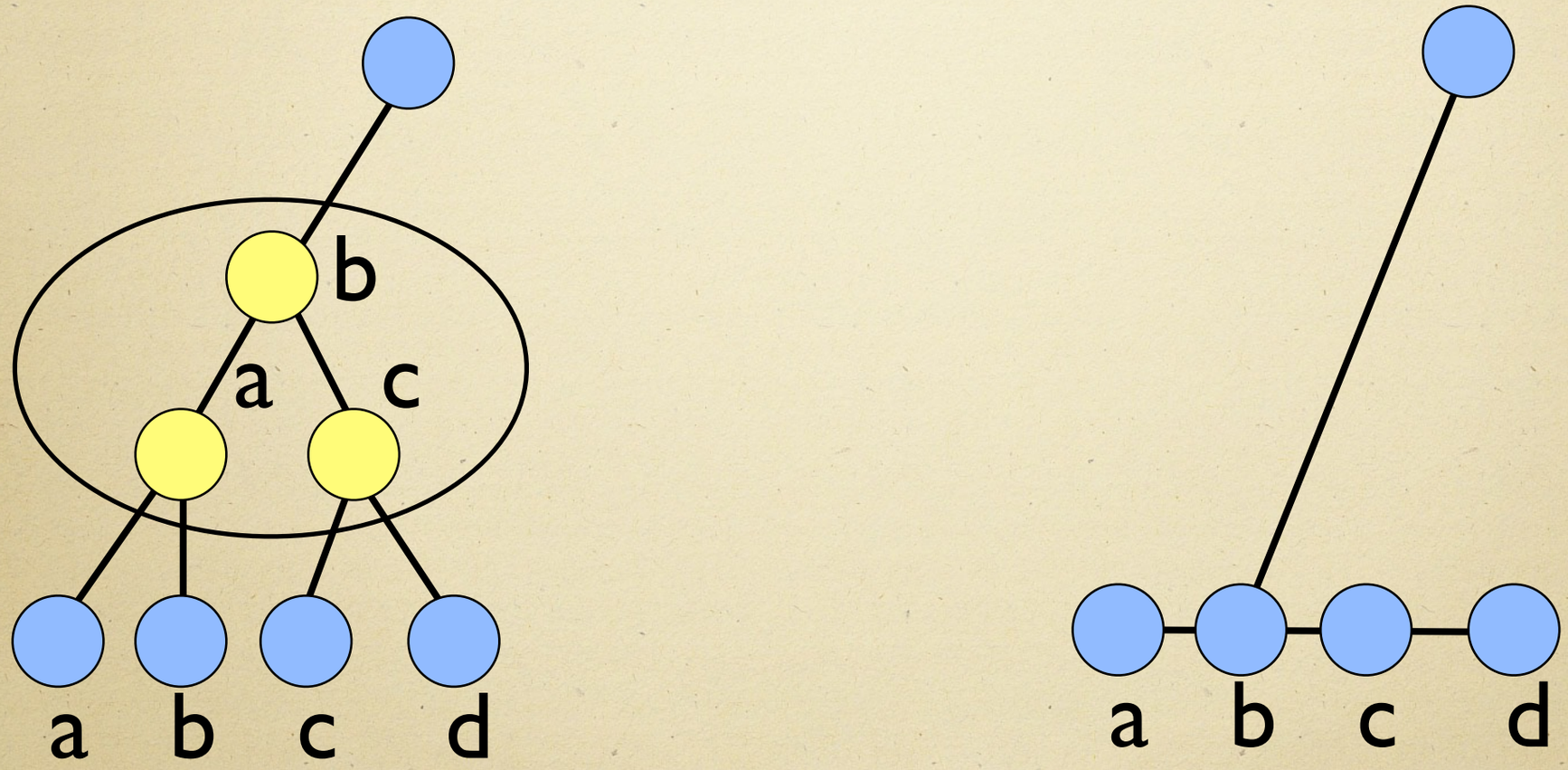
Theorem 1. *The Forgiving Tree has the following properties:*

1. *The Forgiving Tree increases the degree of any vertex by at most 3.*
2. *The Forgiving Tree always has diameter $O(D \log \Delta)$.*
3. *The latency per deletion and number of messages sent per node per deletion is $O(1)$; each message contains $O(1)$ node IDs and thus $O(\log n)$ bits.*

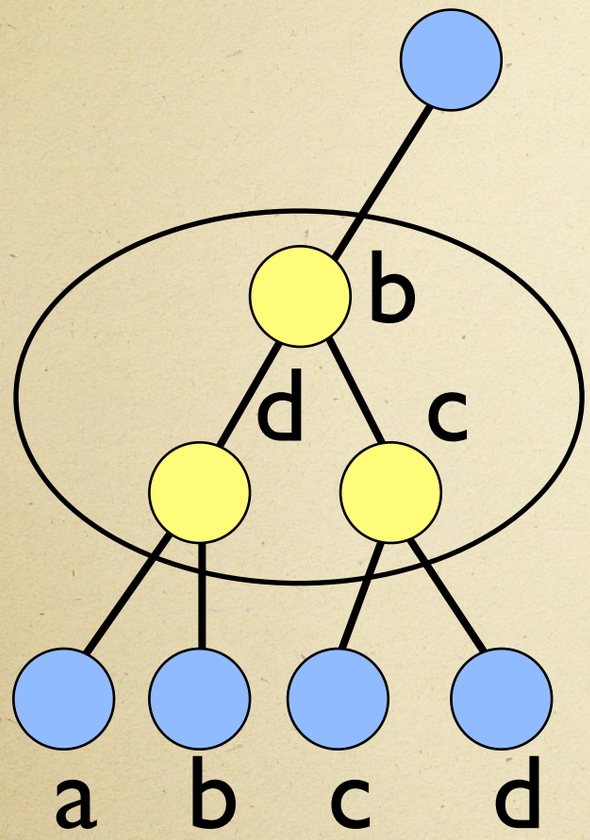
The `real' Network!

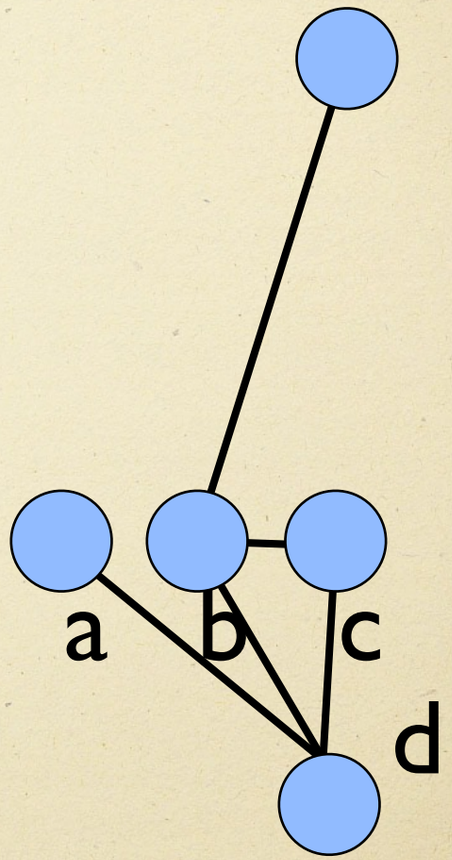
Homomorphism: Given $G_1 = (V_1, E_1), G_2 = V_2, E_2$

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



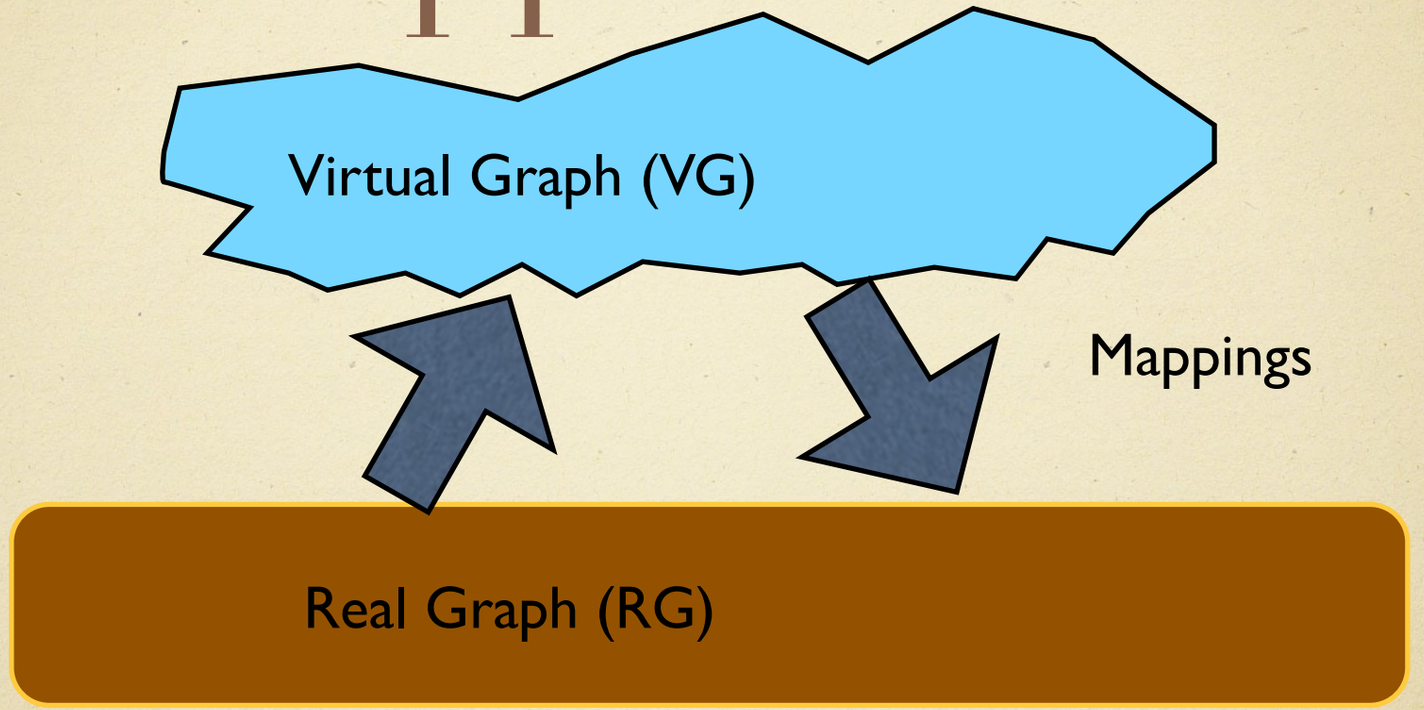
A virtual tree (left) and its homomorphic image (right)





- A different labeling of the virtual nodes, and the image. Note that in this case the image is not a tree.

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).

Advantages of the approach

- Useful for certain problem solutions
- Good ‘accounting structure’ (for graph properties)
- Easier to analyse
- Easier to visualize

Game 3

GAME 3: TOO MANY FRIENDS, TOO LITTLE MEMORY!

Game 3 + Game 2

- How to Route a message from a sender u to a receiver v despite node failures in a reconfigurable/overlay/P2P network of nodes with low memory?

  - *Compact Routing Messages in Self-Healing Trees. Castañeda, Dolev, Trehan.* *ICDCN 2016; Theoretical Computer Science, 2017.*

  - *Fully Compact Routing Messages in Self-Healing Trees.* Castañeda, Lefèvre, Trehan. *ICDCN 2020.*

- Adversary is Deletion only

- Each node uses less than linear (i.e. o(n)) memory **(=Compact)***, !

Sometimes the messenger may have an accident!

High level

1. Preprocessing (regular memory[ICDCN'16], Compact Memory[ICDCN'20]): BFS spanning tree of the network + DFS labelling and TZ setup + CompactFT data structure

Do forever (compact memory)

{

2. COMPACTFT (Compact version of ForgivingTree*): On node deletion, neighbours replace deleted node by its RT and maintain SH tree

3. **S + R:** TZ (Modified Thorup-Zwick^ Routing) on real nodes + BST routing on RTs (virtual nodes)

}

^M. Thorup and U. Zwick. Compact routing schemes. In SPAA, pages 1–10, 2001.

*Tom Hayes, Navin Rustagi, Jared Saia and Amitabh Trehan, "The forgiving tree: a self-healing distributed data structure", in Principles of Distributed Computing (PODC), 2008.

1. A compact Routing Scheme **S** (no failures)

2. If node failure, **compactly** self-heal by replacing node with reconstruction structure (RS) following Routing Scheme R.

3. **S + R:** Transparent integrated compact routing with small stretch….

TZ: Modified Thorup-Zwick^
Routing over trees

COMPACTFT: Compact version of ForgivingTree*

Binary Search Tree Traversal

COMPACTFTZ: SH Compact routing

^M. Thorup and U. Zwick. Compact routing schemes. In SPAA, pages 1–10, 2001.

*Tom Hayes, Navin Rustagi, Jared Saia and Amitabh Trehan, “The forgiving tree: a self-healing distributed data structure”, in Principles of Distributed Computing (PODC), 2008.

Why `compact' memory?

- Large networks of low memory nodes
- For example, small sensors forming dense networks - Internet of Things?
- Limit node memory to O(polylog n) bits even in arbitrary networks (i.e. O(n) neighbours)
- Implies a node cannot even store identities of neighbours
- Introduces a local streaming style computation (next slide)

A formal model

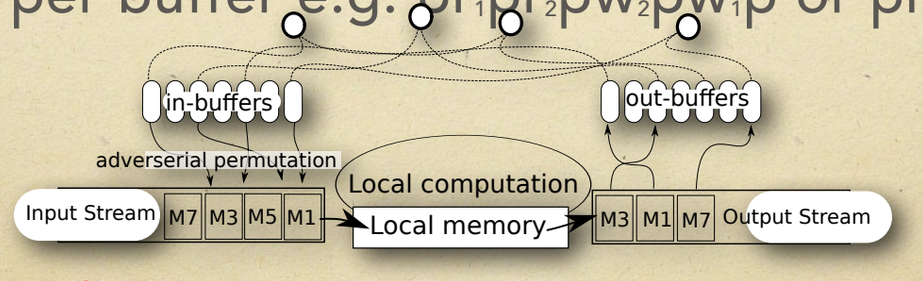
- **COMPACT LOCAL STREAMING (CLS) / COMPACT MESSAGE PASSING (CMP)** or COMPACT model

- Consider each message passing round at a finer granularity

- Computation (read-process-write) by rounds, every round executing a *sweep*:

1. Mutable reads: Once in-buffer read, value cleared

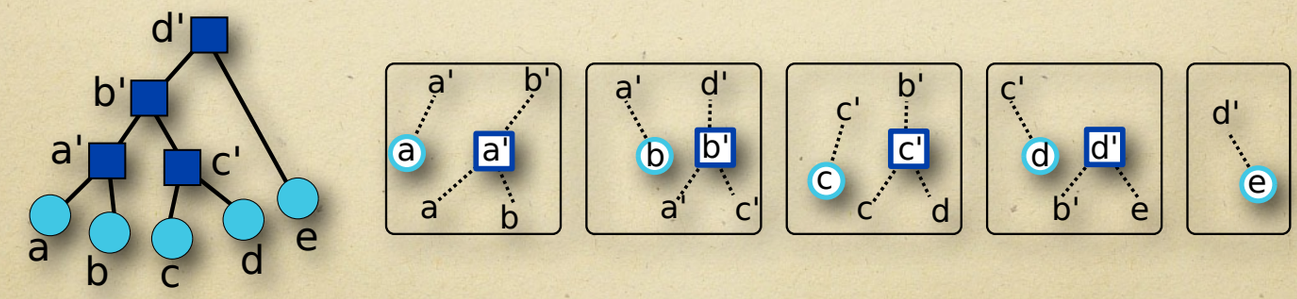
2. Fair Interleaving condition: read (r), write (w), process (p) in any order with only 1 read, write per buffer e.g. $pr_1pr_2pw_2pw_1p$ or $pr_1pw_2pr_3pw_1$



! Self-healing Routing and Other Problems in Compact Memory
A Castañeda, J Lefèvre, A Trehan arXiv preprint arXiv:1803.03042

For fully compact CompactFT

- Induced components of a labelled graph!
- In particular, binary search labelled half-full trees
- Half-full trees are binary trees used as Wills! *
- Haft(v,w) needs to return the components of Haft(v) that w, a neighbour of v appears in! (Note: v does not have enough memory to compute the whole tree in its memory!)



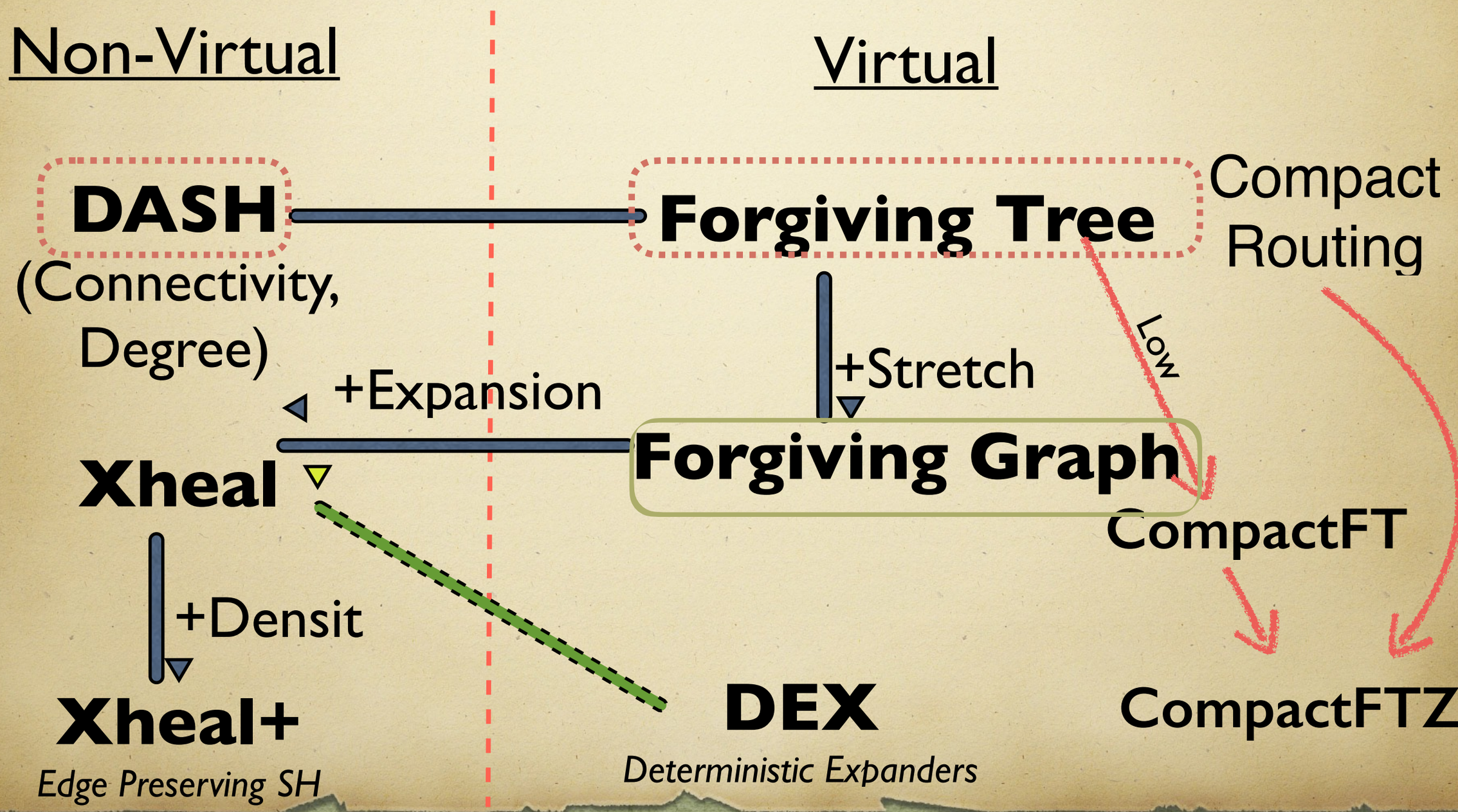
The forgiving graph: a distributed data structure for low stretch under adversarial attack

TP Hayes, J Saia, A Trehan Distributed Computing 25 (4), 261-278

Some Results in CLS

Algorithm	Internal memory	Time (# rounds)	# messages
* Leader Election by flooding	$O(\log n)$	$O(D)$	$O(m)$
* BFS Spanning Tree Construction	$O(\log n)$	$O(D)$	$O(m)$
* (regular) Convergecast	$O(\log n)$	$O(D)$	$O(n)$
Weight labelling with convergecast	$O(\log n)$	$O(D)$	$O(nD)$
Compact DFS relabelling of a tree	$O(\log n)$	$O(m)$	$O(m)$
* further Compact DFS walks	$O(\log n)$	$O(n)$	$O(n)$
'Light' Path by BFS construction	$O(\log^2 n)$	$O(D)$	$O(m)$
Wills (w/ half-full tree labelling) setup			
* ...with deterministic reads	$O(\log^2 n)$	$O(1)$	$O(n)$
...with adversarial reads	$O(\log n)$	$O(\Delta)$	$O(n\Delta)$
TZ preprocessing	$O(\log^2 n)$	$O(m)$	$O(m)$
CompactFT preproc. with adversarial reads	$O(\log n)$	$O(D+\Delta)$	$O(m+n\Delta)$
with deterministic reads	$O(\log n)$	$O(D)$	$O(m)$
CompactFTZ preproc. with adversarial reads	$O(\log^2 n)$	$O(m+\Delta)$	$O(m+n\Delta)$
with deterministic reads	$O(\log^2 n)$	$O(m)$	$O(m)$

Our Self-healing Algorithms



The Forgiving Graph: a distributed data structure for low stretch under adversarial attack

Thomas P. Hayes · Jared Saia · Amitabh Trehan

Received: 20 September 2010 / Accepted: 3 February 2012

Abstract We consider the problem of self-healing in peer-to-peer networks that are under repeated attack by an omniscient adversary. We assume that, over a sequence of rounds, an adversary either inserts a node with arbitrary connections or deletes an arbitrary node from the network. The network responds to each such change by quick "repairs," which consist of adding or deleting a small number of edges. These repairs essentially preserve closeness of nodes after adversarial deletions, without increasing node degrees by too much, in the following sense. At any point in the algorithm, nodes v and w whose distance would have been ℓ in the graph formed by considering only the adversarial insertions (not the adversarial deletions), will be at distance at most $\ell \log n$ in the actual graph, where n is the total number of vertices seen so far. Similarly, at any point, a node v whose degree would have been d in the graph with adversarial inser-

Graph, has low latency and bandwidth requirements. The Forgiving Graph improves on the Forgiving Tree distributed data structure from Hayes et al. (2008) in the following ways: 1) it ensures low stretch over all pairs of nodes, while the Forgiving Tree only ensures low diameter increase; 2) it handles both node insertions and deletions, while the Forgiving Tree only handles deletions; 3) it requires only a very simple and minimal initialization phase, while the Forgiving Tree initially requires construction of a spanning tree of the network.

Keywords Self-healing · Networks · Peer-to-peer · Stretch · Trees · Dynamic · Responsive

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

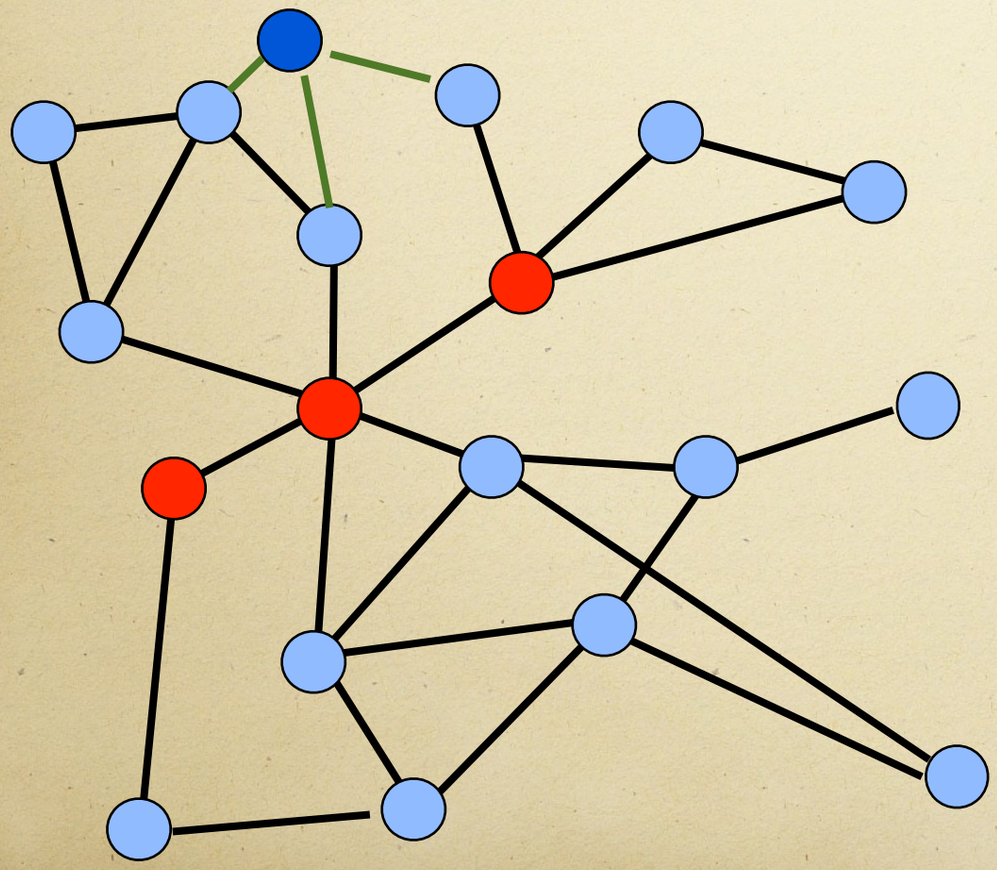
Big question:

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

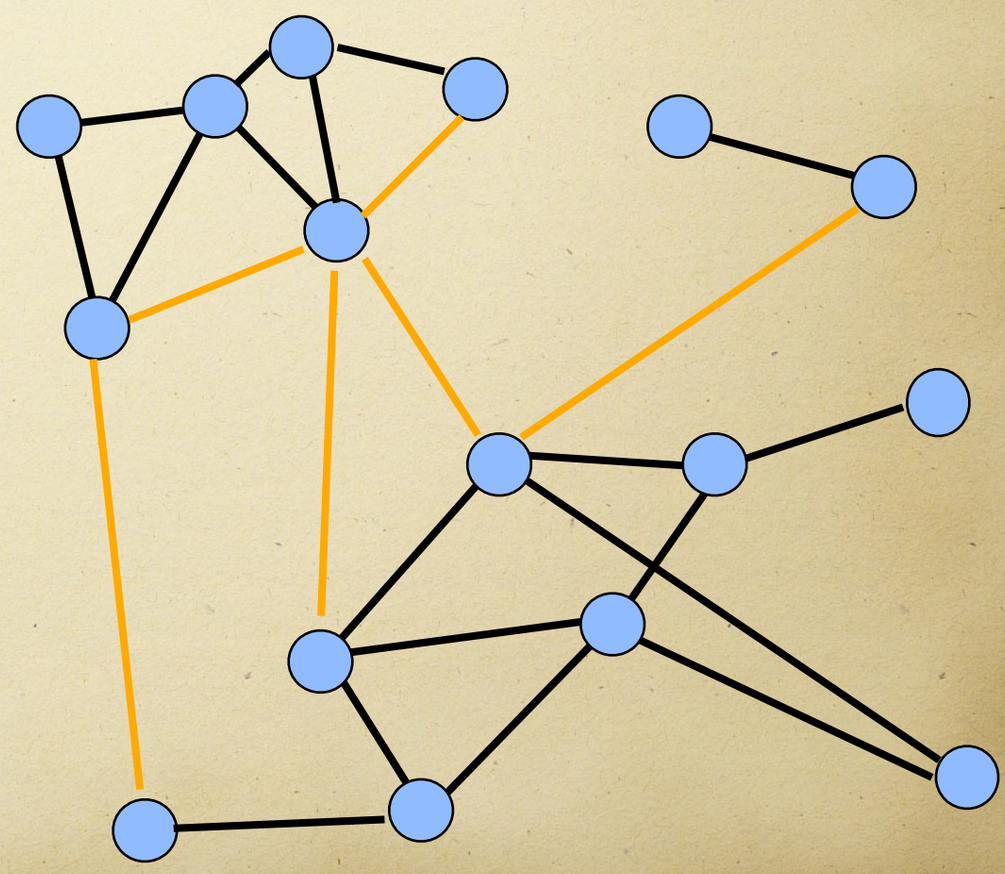
Or: *What can G_0 look like?*

Comparing results

G′: graph of only insertions and original nodes

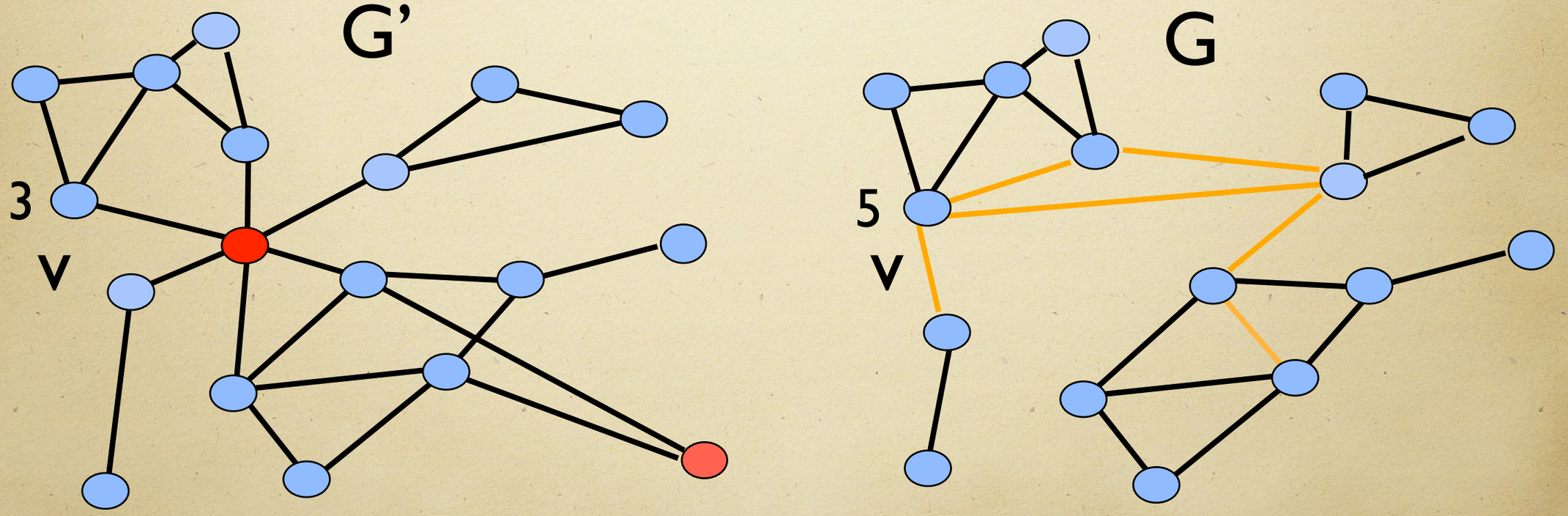


G: healed network



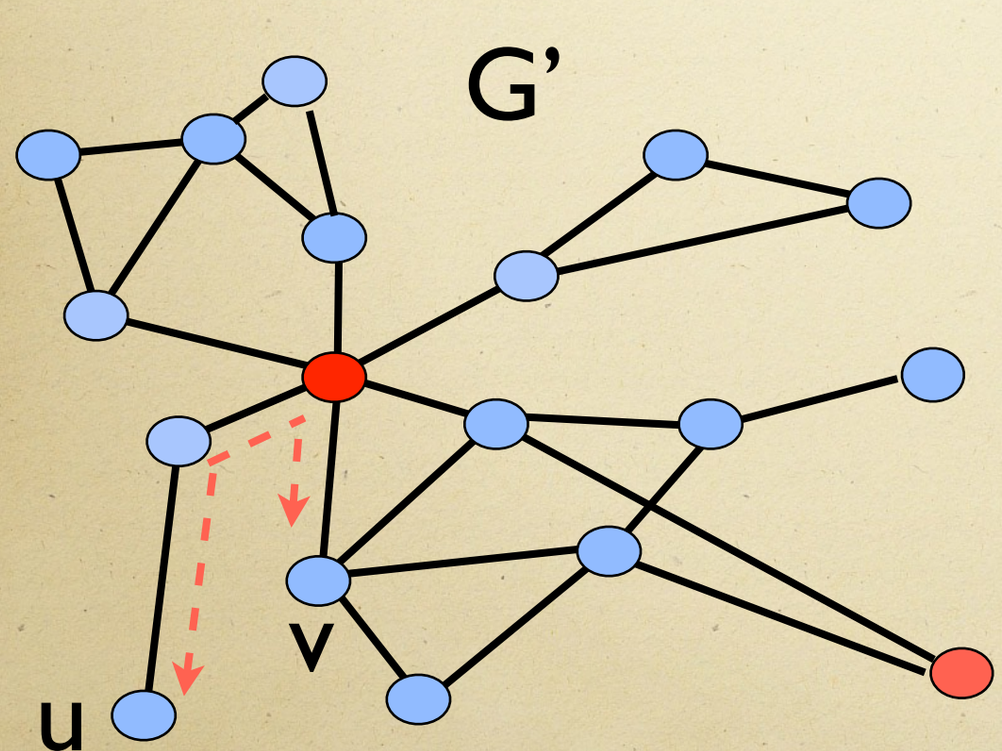
FG: Main Result

- A distributed algorithm, Forgiving Graph such that:
  - *Stretch*: Distance between any two nodes in G $\leq$ log n times their distance in G'

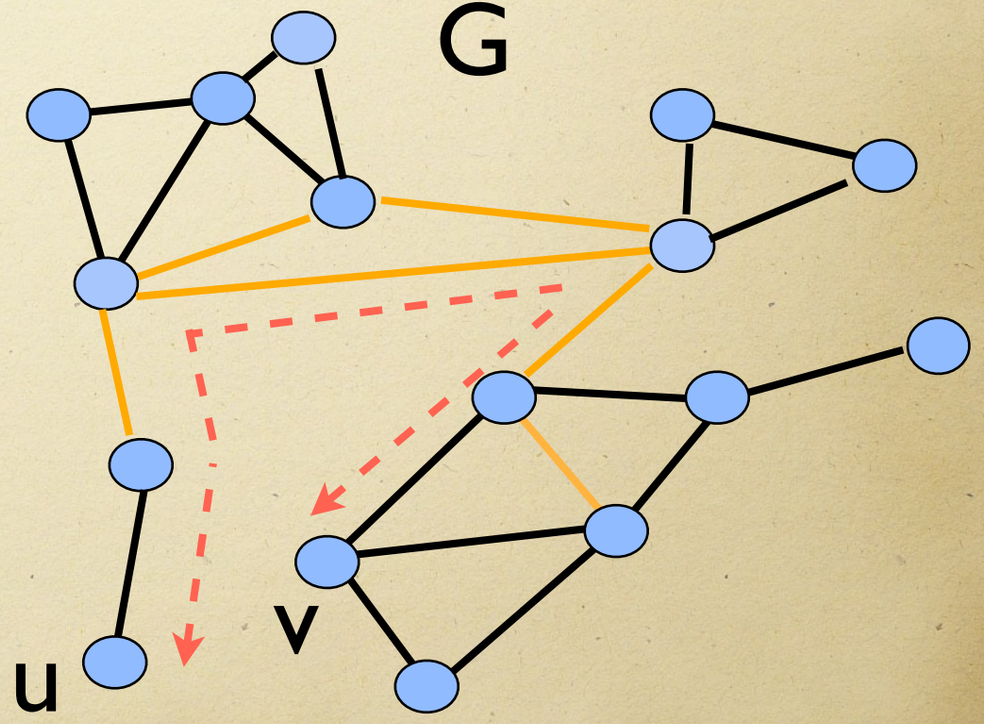


FG: Main Result

- A distributed algorithm, Forgiving Graph such that:
  - *Degree increase*: Degree of node in G ≤ 3 times degree in G'



d(u,v) = 3



d(u,v) = 5

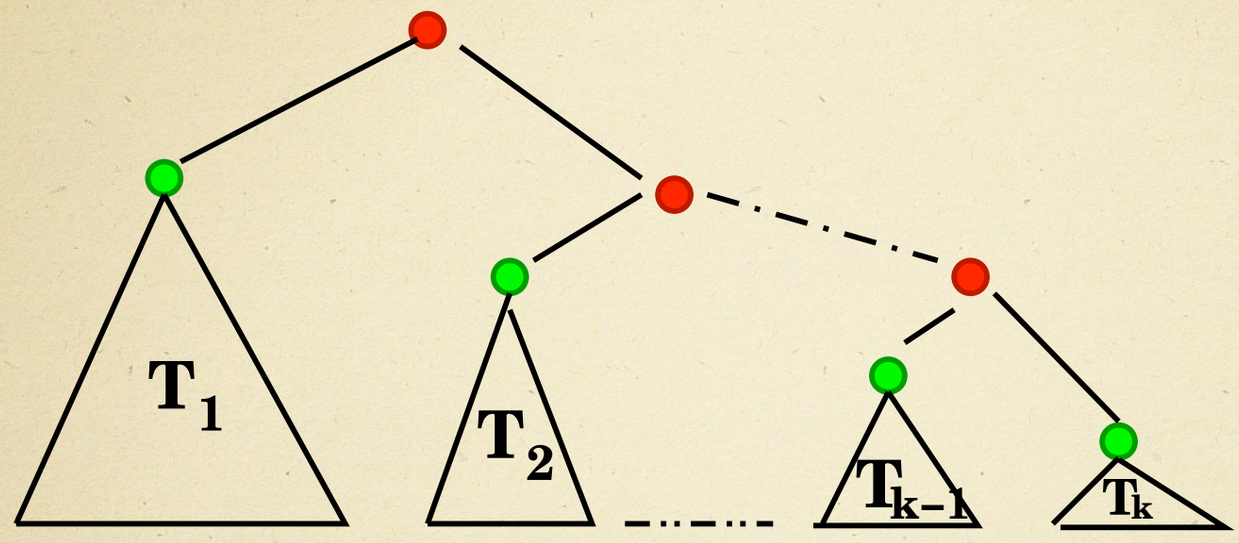
FG: Results and Optimaltiy

- 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′

Matching lower bound

- *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)

Half-Full Trees (hafts)



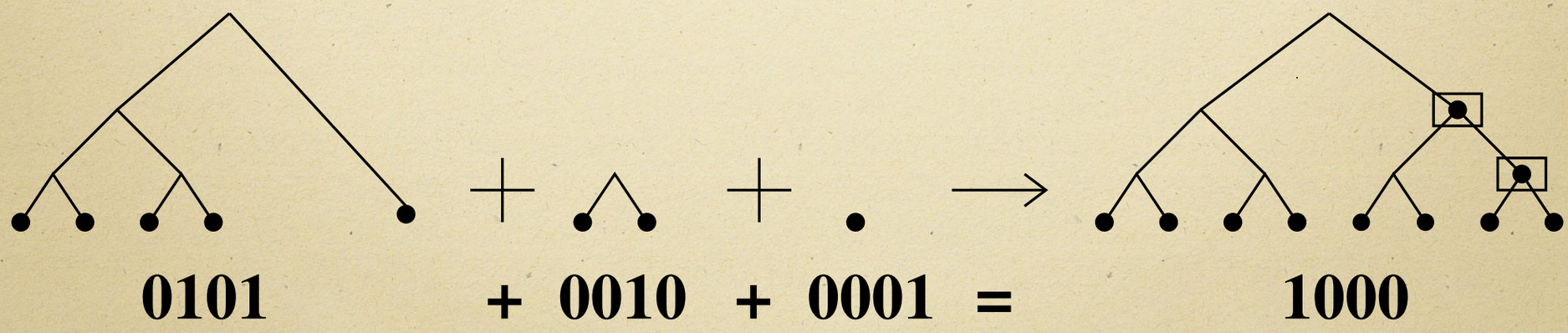
- A rooted binary tree in which every non-leaf node *v* has the following properties:
  - *v* has exactly two children.
  - The left child of *v* is the root of a complete binary subtree containing at least half of *v*'s children.

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.



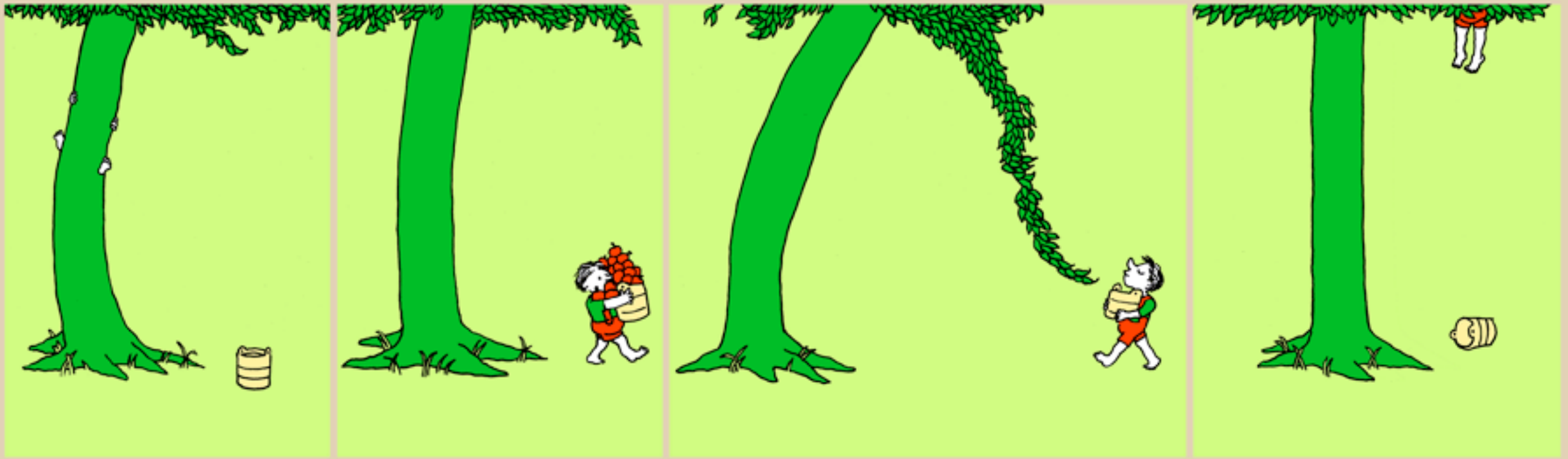
Questions and future work

- `Behavioural' Self-healing: Self-healing Amnesiac Flooding, General graph compact routing etc?
- Standard Graph problems in Self-healing/P2P-CONGEST?
- Understanding CLS (What's the right problem? Is it interesting?)
- Minimal history/working memory to solve particular problems?
- Theory of Unique/Countable solutions to problems?

Visit!

- Wiki Page: https://en.wikipedia.org/wiki/Amnesiac_flooding
- Me: www.durham.ac.uk/staff/amitabh-trehan/, amitabhtrehan.net
- My group: https://nestid.webspace.durham.ac.uk/ (Pls join our Seminars/ Seminar announcement list)
- Please join us for SIROCCO 2026 (June 9-11, 2026) at Durham!

pbfcomics.com (apologies, Silverstein)



THANK YOU

Features of (Stateful) Flooding

- Probably the most basic distributed algorithm
- Backbone of various algorithms (e.g. Ver 1 used for Peleg's time optimal Leader Election)
- Used to initiate a network and set up a communication spanning tree
- Terminating (due to Step B)
- Time optimal broadcast (At most diameter+1 steps)

AF TERMINATION

Theorem 1. *Given a finite graph G, Amnesiac Flooding (AF) from a single source will terminate in a finite number of rounds*

Detailed Proof.
Proof is by contradiction

Definition. **Round-sets:**

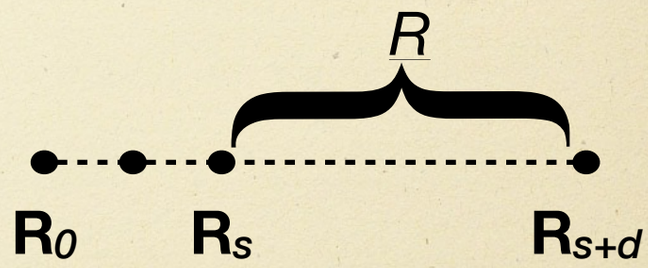
$R_0, R_1, \ldots$

R_0 : Singleton with node l

R_i : Set of nodes receiving a message at round i (>0)

Define **R** to be the set of finite sequences of the form

$\underline{R} = R_s, \ldots, R_{s+d}$ where $s \geq 0$, $d \geq 0$ and $R_s \cap R_{s+d} \neq \emptyset$



Consider R^e to be subset of R where d is even.

Claim 1. If AF does not terminate, R^e will be non-empty

AF TERMINATION

Theorem 1. *Given a finite graph G, Amnesiac Flooding (AF) from a single source will terminate in a finite number of rounds*

Claim 1. If AF does not terminate, R^e will be non-empty

Proof. G is finite:
=> if AF does not terminate, a node *x* must occur in infinitely many round-sets.
Take the first 3 such round-sets (e.g. 2,5, and 7);
Surely at least two of these are evenly spaced.

Thus, **R^e is non-empty.**

From above, assume R^e is non-empty and derive a contradiction.

TERMINATION IN BIPARTITE GRAPHS

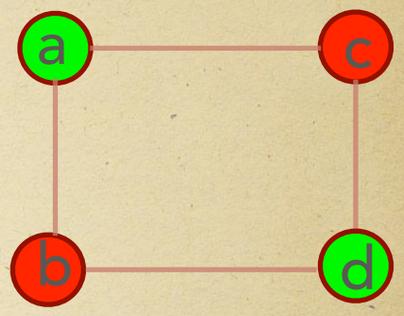
Theorem. In a connected bipartite graph, AF terminates in rounds = e(a), where e(a) is the eccentricity of the vertex a in graph B, where a is the origin node.

Since Diameter D is the largest possible e(a)

Corollary. In a connected bipartite graph, AF terminates in D rounds.

Bipartite Graphs

Revisiting the definition:



Bipartite Graph: A set of graph vertices decomposed into two disjoint sets (say, red and green) such that no two graph vertices within the same set are adjacent.

Or

Bipartite Graph: a bipartite graph is a graph that does not contain any odd-length cycles.

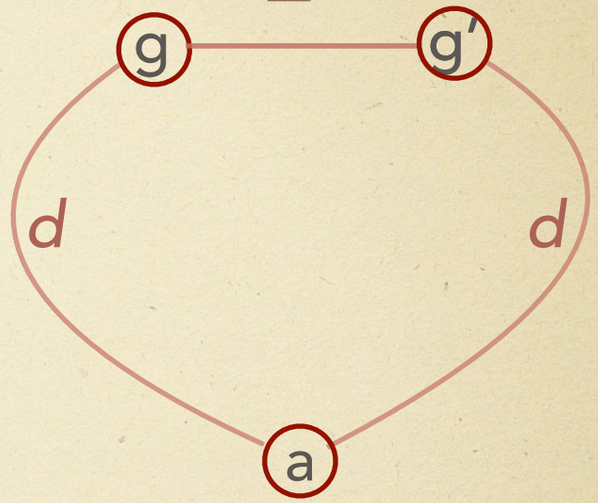
Or

Another equivalent definition wrt initiators on the next slide …

Bipartite vs. Non-Bipartite

- ***EC(Equidistantly-connected)* nodes** (from origin/initiator a)

G is bipartite iff it has no ec node



Proof. Equidistant nodes belong to the same *partite set*.
A graph is bipartite iff no edge connects two such nodes!

We will skip the termination time proof here …

Great!

Any hidden assumptions?

AF TERMINATION

DOES (SYNCHRONOUS) AF TERMINATE?

YES

ON EVERY GRAPH?

YES

IS IT QUICK?

YES

Questions?

- Does synchronous AF with multiple initiators terminate?

  YES. WHY?

  - What if the initiations are at different times?

    YES

- What if the graph is changing during AF?

  YES - FOR DELETION OF EDGES AND NODES. MAYBE NOT ON INSERTIONS.

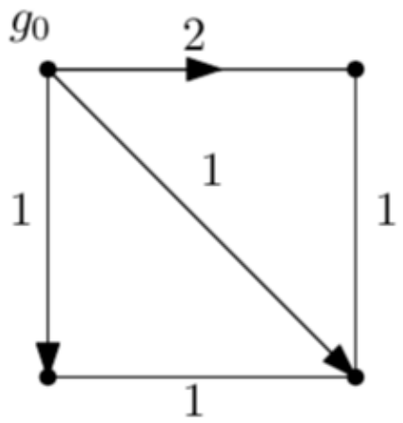
Clue: Earlier proofs had no dependence on initiator(s)!

- Redefine Initiator to be a set of initiators (I), EC and bipartiteness etc, and modify with technicalities.

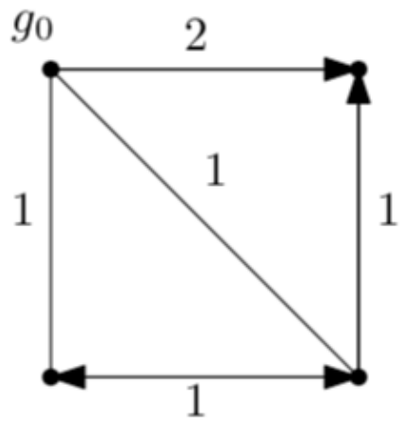
* has many more flooding variations.

* AF-ARXIV'20] Walter Hussak<https://dblp.org/pid/60/6563.html>, Amitabh Trehan: Terminating cases of flooding. CoRR abs/2009.05776

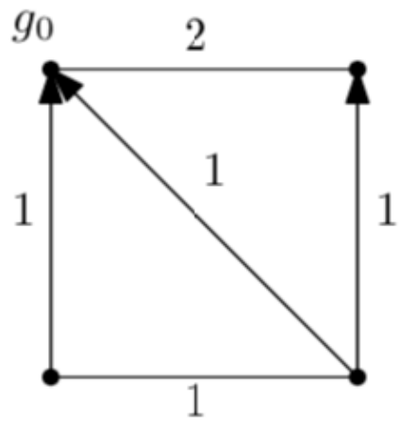
round 1



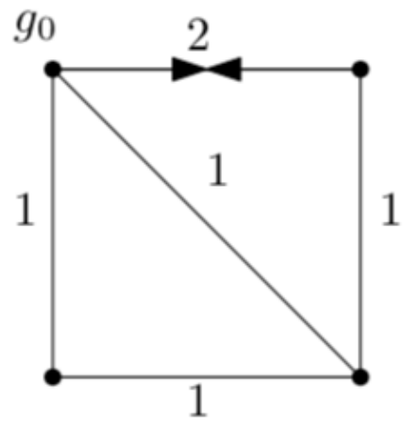
round 2



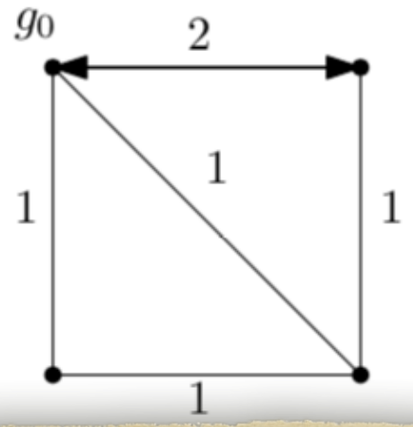
round 3



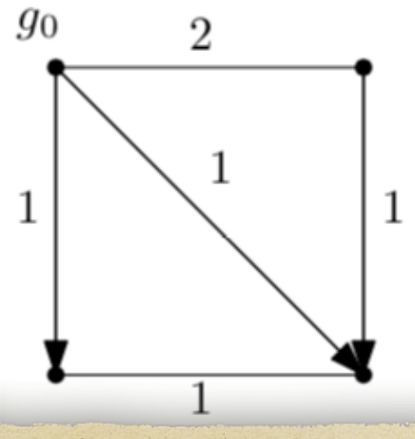
round 4



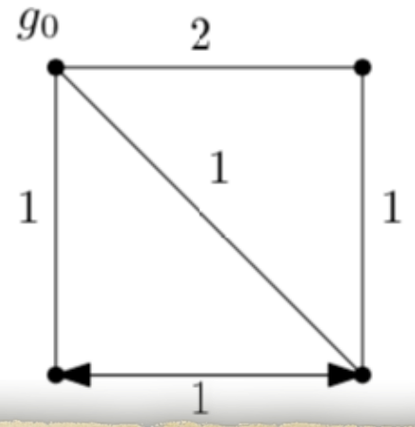
round 5



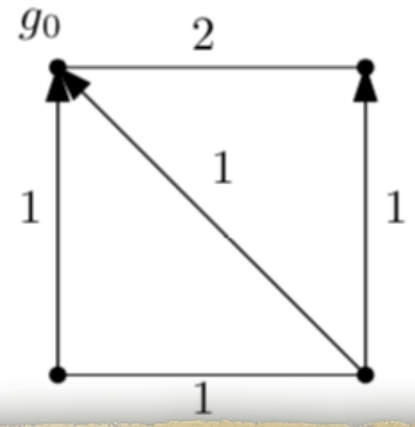
round 6



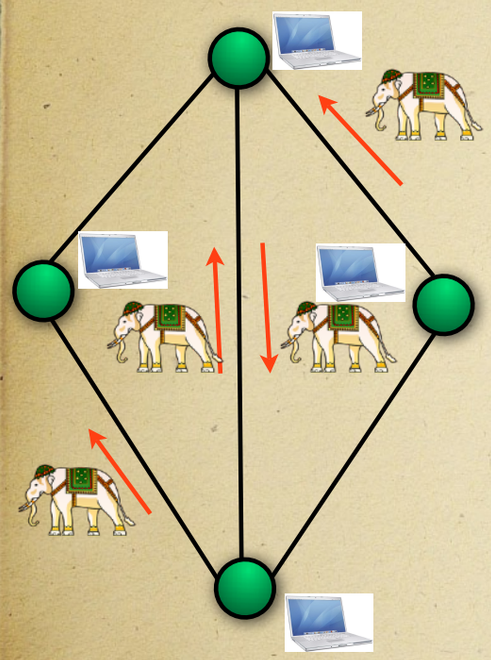
round 7



round 8



The Synchronous Message Passing Model



- Message synchronisation property:
  - Message sent from processor v to neighbour u at pulse p of v must arrive at u before pulse $p+1$ of u.

Computation Round Steps:

1. Send messages to (some of) the neighbours.
2. Receive messages from (some of) the neighbours.
3. Perform some local computation

Synchronous protocol: (Round based) Describes local computation and messages for every round.

Time Complexity = #Synchronous Rounds

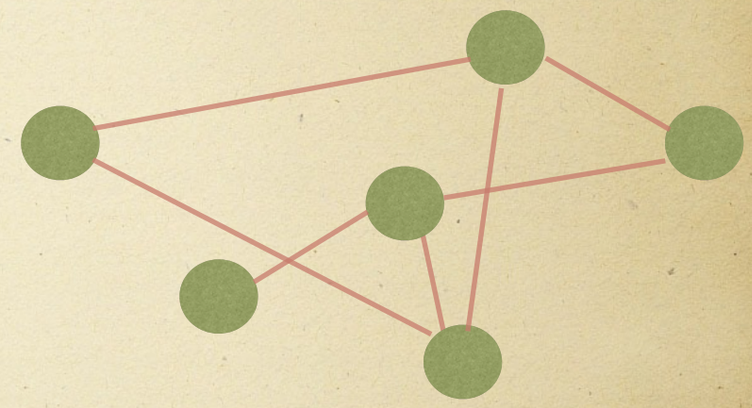
Message Complexity = #Messages exchanged

Motivation: Bitcoin

- Bitcoin runs a peer-to-peer overlay network
  - Overlay is used for all communication
  - Overlay is assumed to be reliable
- Basic Overlay maintenance:
  - Connect to few arbitrary neighbours
  - Maintain the number and accept incoming requests
- Hard: Run despite high churn and greedy users (byzantine)

Formal model: Message Passing Distributed Algorithms

- The graph **G(V,E)** is a formal model for a network.



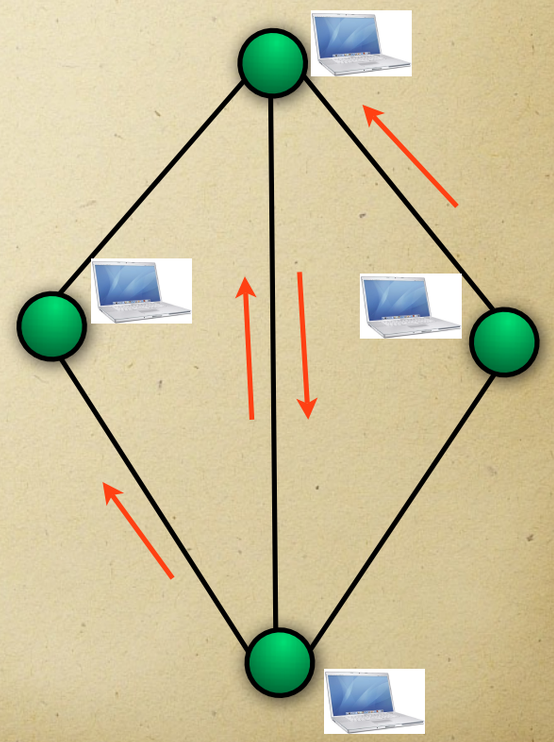
- Message passing: nodes only communicate by sending messages.

A. Synchronous and Reliable:
communication in synchronous rounds:

1. Process: Process and prepare new messages (possibly in response)

2. Send: Send messages to (all/any) neighbour(s) (outgoing)

3. Receive: Messages from all neighbours (incoming)



Objectives!

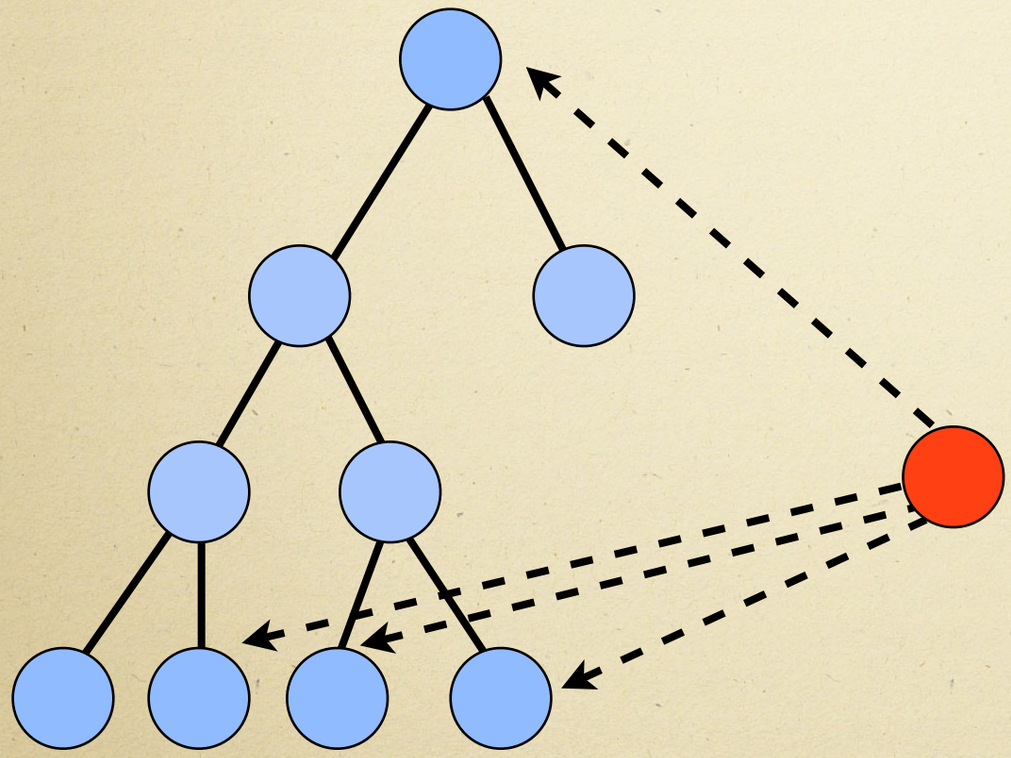
- In the self-healing model, where adversary deletes one node at a time, after any number of attacks:
  - (Graph thinking) Maintain diameter of graph within acceptable bounds (at most logarithmic)

Hint: Think Trees (use your past knowledge) - what trees approximate diameter of a graph well? Maybe we can maintain such a tree ad-infinitum?

Leaf deletions

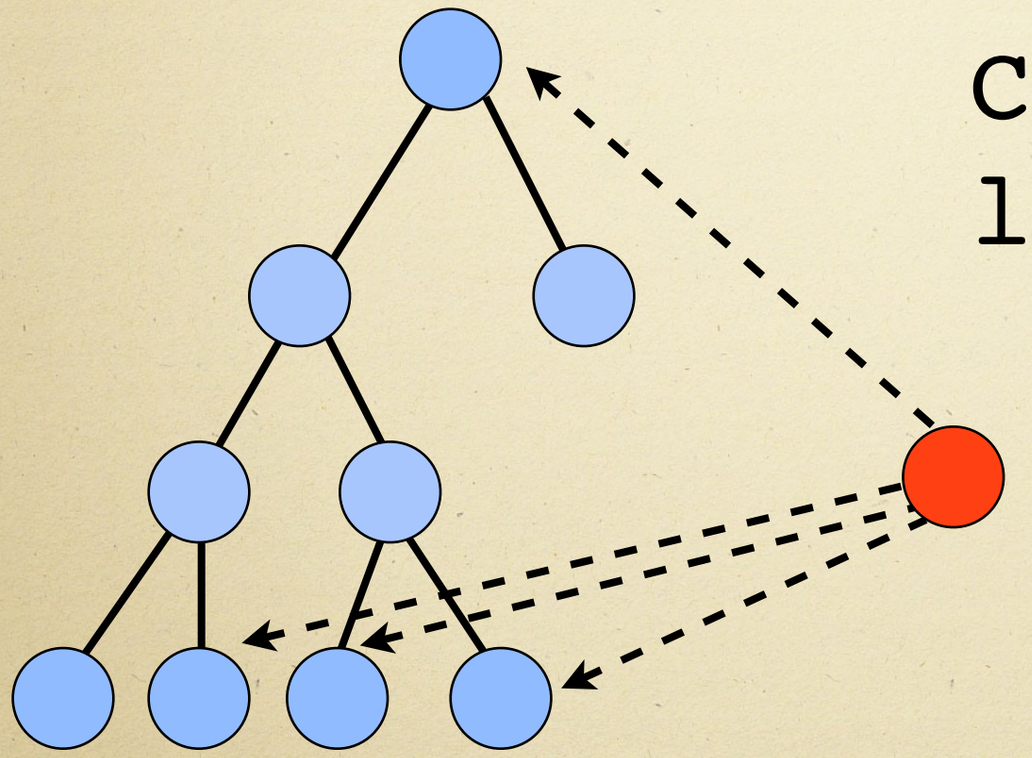
- When a leaf is deleted, there are two cases:
  - (a) if the parent is still alive, parent updates its will by reassigning v's virtual node.
  - (b) if the parent is already gone, v should have left a will with the simulator of the virtual node which will become redundant.

An open question: Inserting nodes



- Since a tree, only one of the incoming edges of the new node can be accepted

An open question: Inserting nodes



Heuristic 1:

`Connect to the lowest depth node`

Problem:

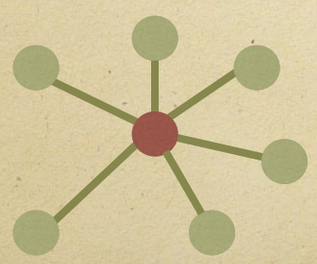
Tree diameter may blow up compared to graph diameter

Objectives!

- In the self-healing model, where adversary deletes one node at a time, after any number of attacks:
  - (Graph thinking) Maintain diameter of graph within acceptable bounds (at most logarithmic)
  - (Graph thinking) Contain degree increase per node (at most additive constant)
  - (Distributed thinking) Be efficient on each healing (both time and messages!)

Objectives!

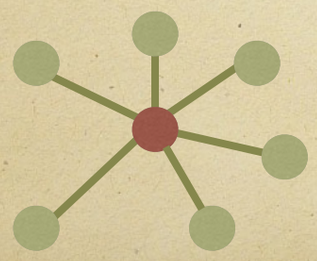
- In the self-healing model, where adversary deletes one node at a time, after any number of attacks:
  - (Graph thinking) Maintain diameter of graph within acceptable bounds (at most logarithmic)
  - (Graph thinking) Contain degree increase per node (at most additive constant)

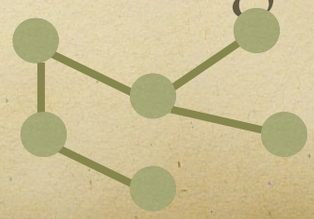
Idea: T=BFSTree(G). Replace each deleted node's star  by a low dia, low degree graph.

Which?

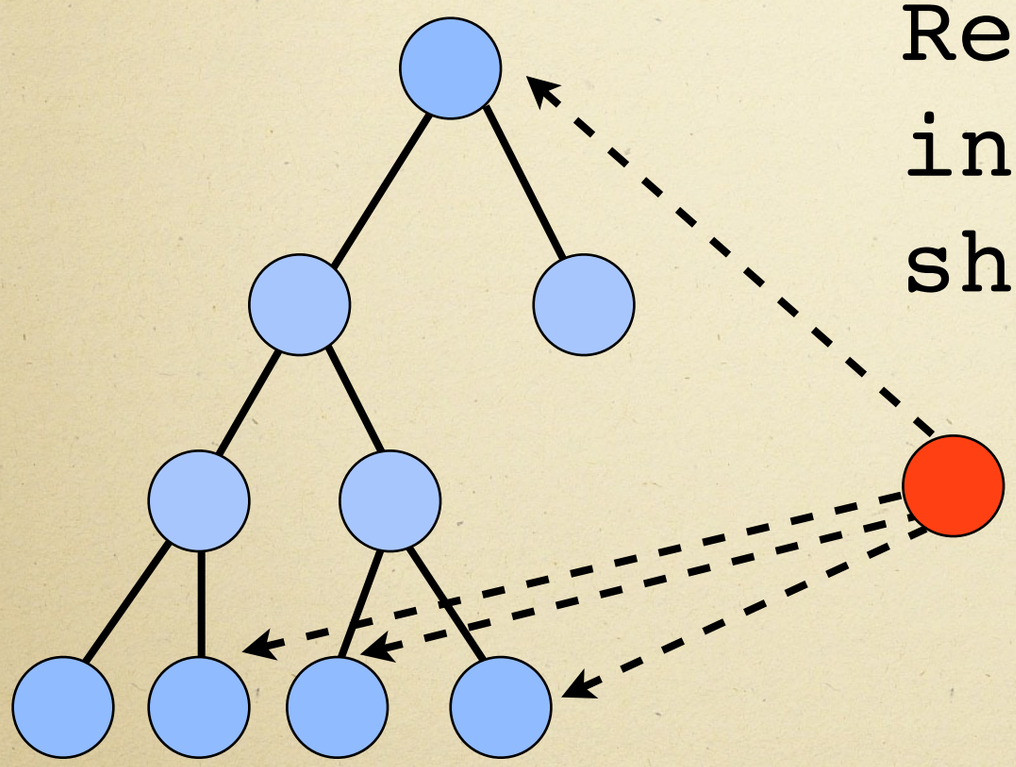
Objectives!

- In the self-healing model, where adversary deletes one node at a time, after any number of attacks:
  - (Graph thinking) Maintain diameter of graph within acceptable bounds (at most logarithmic)
  - (Graph thinking) Contain degree increase per node (at most additive constant)

Idea: T=BFSTree(G). Replace each deleted node's star  by a low dia, low degree graph.

Which? 

An open question: Inserting nodes



Heuristic 2:

`Retain all edges of incoming node that shorten distances`

Problem:

- Cascade effect (multiple nodes may be involved; reconstructing the tree)

- handling virtual RTs and parent-child relationships

Amnesiac Flooding (AF)

- More formally: ***Amnesiac Flooding (AF)***
- Start: A distinguished node *l*
- *Round 1: l* sends message *M* to all neighbours
- *Round i (> 1):* If node *v* receives *M* from neighbours *R* in round *i-1:*

  A. *v* floods to *n(v)/R* (i.e. neighbours besides *R*)

Amnesiac Flooding (AF)

- More formally: ***Amnesiac Flooding (AF)***
- Start: A distinguished node *l*
- *Round 1: l* sends message *M* to all neighbours
- *Round i (> 1):* If node *v* receives *M* from neighbours *R* in round *i-1*, *v* floods to *n(v)/R* (i.e. neighbours besides *R*)

Q1: Does AF terminate?

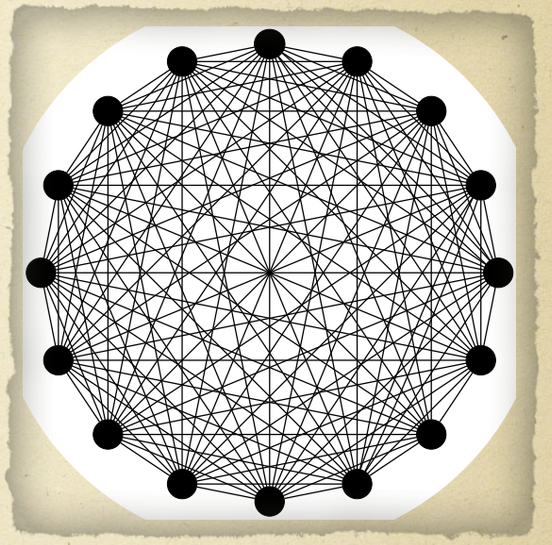
Q2: If AF terminates, how long does it take?

*Termination: M is not sent in a round (and subsequent rounds) by any node in the network

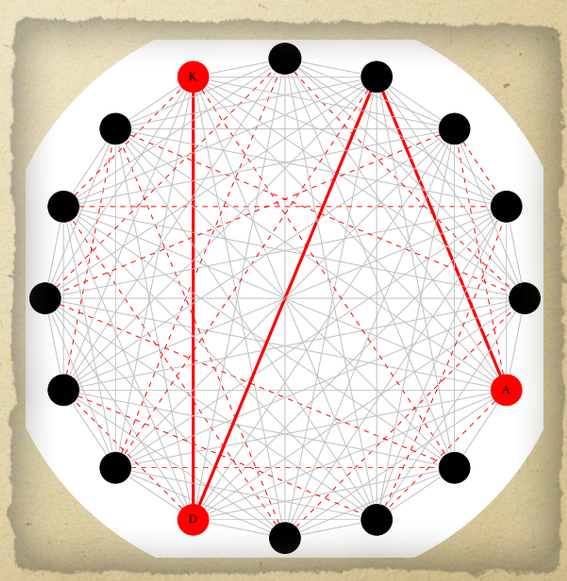
Termination time: The number of rounds AF takes i.e. last round with a transmission

The world of Overlay/P2P Networks

- Given: A collection of (changing) servers



- (Assuming fully connected (underlay): Choose a `good' subset of the edges s.t.
  - Subgraph has low degree
  - Subgraph has low diameter
  - Subgraph has good expansion?



Objectives!

- In the self-healing model, where adversary deletes one node at a time:
  - Maintain diameter of graph within acceptable bounds (at most logarithmic)
  - Contain degree increase per node (at most additive constant)
  - Be efficient on each healing (both time and messages!)

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 deletions,
- Graph stays connected
- Diameter increases by ≤ log(D) factor
- Degrees increase by ≤ 3 (additive)
- Each repair takes constant time and involves O(D) nodes.

Theorem 1. *The Forgiving Tree has the following properties:*

1. *The Forgiving Tree increases the degree of any vertex by at most* 3.
2. *The Forgiving Tree always has diameter* $O(D \log \Delta)$.
3. *The latency per deletion and number of messages sent per node per deletion is* $O(1)$*; each message contains* $O(1)$ *node IDs and thus* $O(\log n)$ *bits.*

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. (later).

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.