Distributed Computing over
Communication Networks:

Tree Algorithms

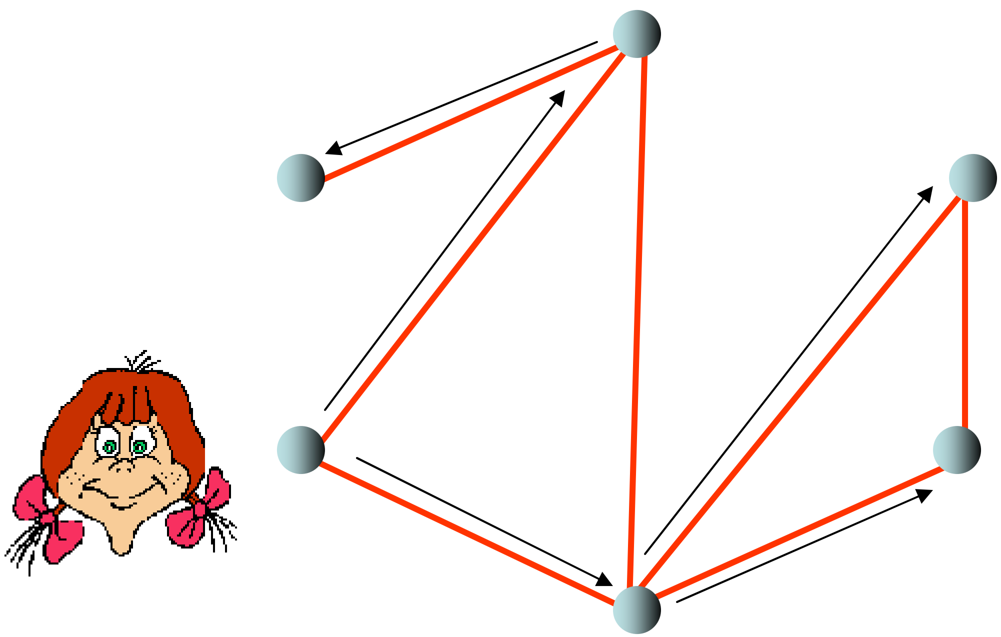
Why trees?

E.g., efficient broadcast, aggregation, routing, ...

Important trees?

E.g., breadth-first trees, minimal spanning trees, ...

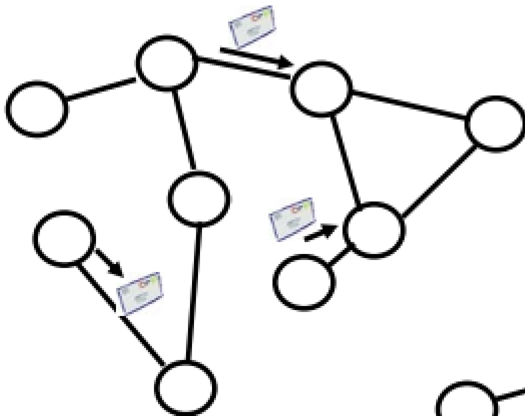
Broadcast



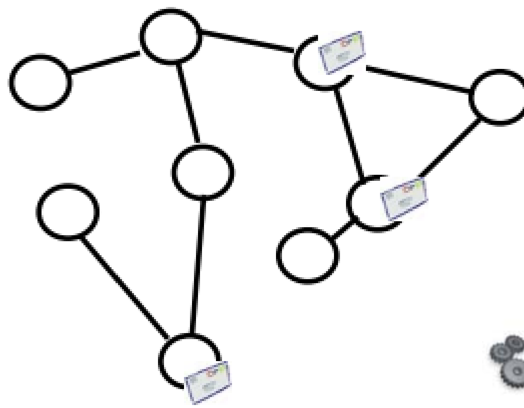
Lower bound for
time and messages?

Recall: Local Algorithm

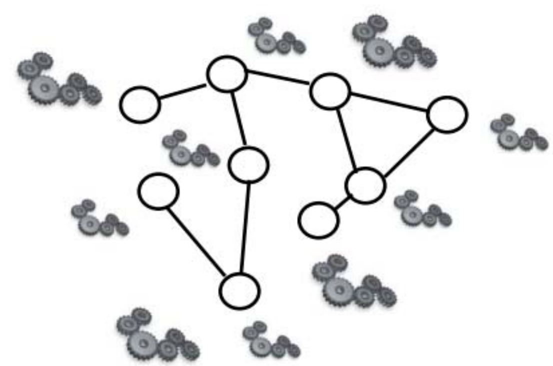
Send...



... receive...



... compute.



Broadcast

Broadcast

Message from one source to all other nodes.

Distance, Radius, Diameter

Distance between two nodes is # hops.
Radius of a node is max distance to any other node.
Radius of graph is minimum radius of any node.
Diameter of graph is max distance between any two nodes.

Relationship between R and D?

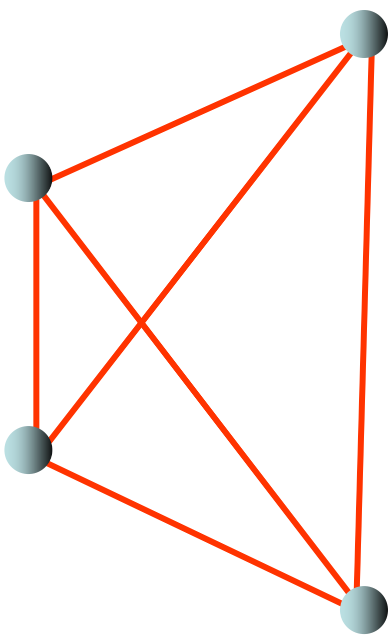
Examples....

Lemma (R, D)

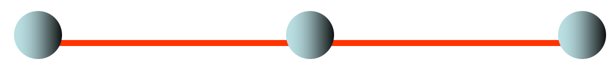
$R \leq D \leq 2R$

Where R=D?

Complete graph:

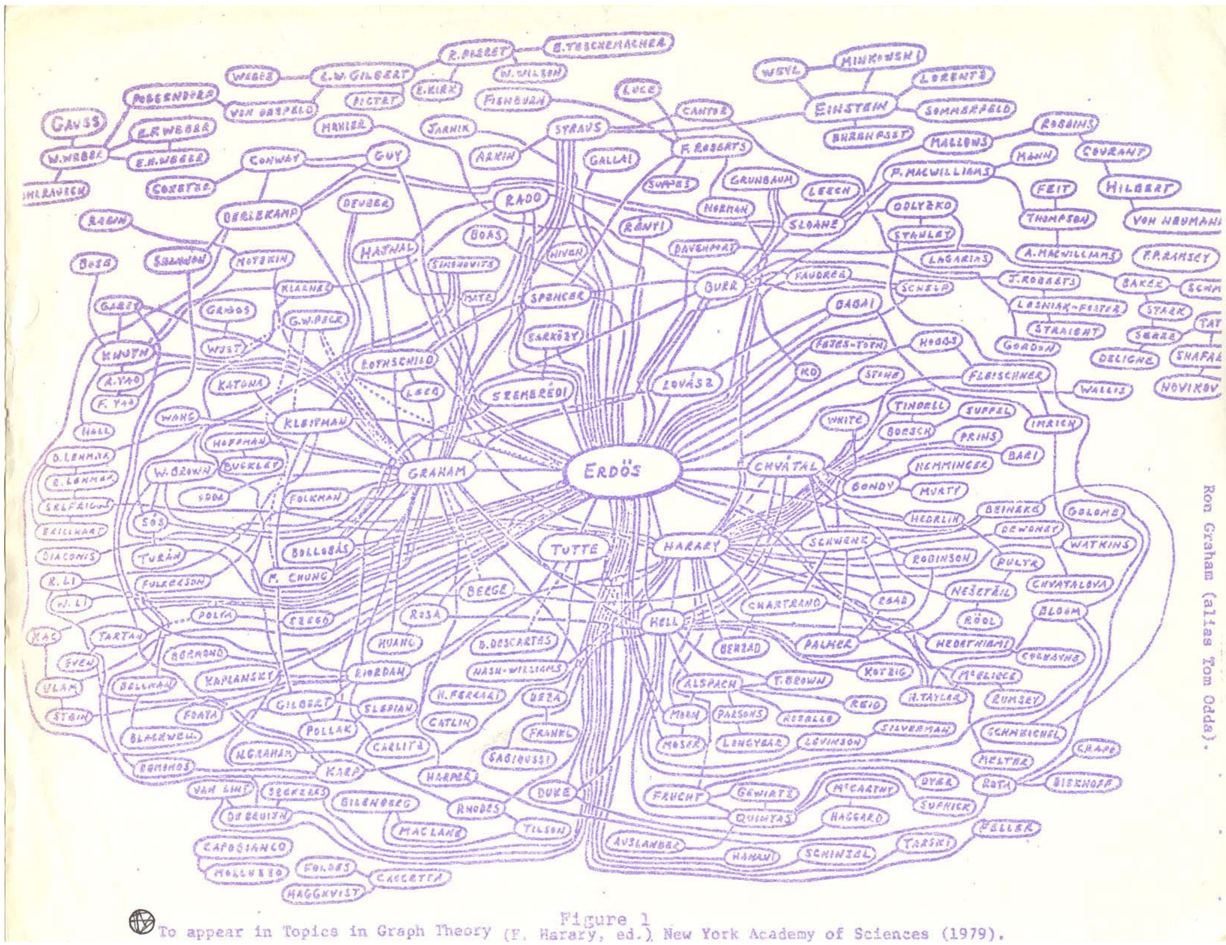


Where 2R=D?



Kevin Bacon, Paul Erdös, ...

People like to find nodes of small radius in a graph! E.g., movie collaboration (link = act in same movie) or science (link = have paper together)!

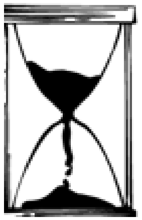


Lower Bound for Broadcast?

Message complexity?

Each node must receive message: so at least n-1.

Time complexity?



The radius of the source: each node needs to receive message.

How to achieve broadcast with n-1 messages and radius time?

Pre-computed breadth-first spanning tree...

Broadcast in Clean Networks?

Clean Graph

Nodes do not know topology.

Lower bound for clean networks?

Number of edges: if not every edge is tried, one might miss an entire subgraph!

How to do broadcast in clean network?

Flooding

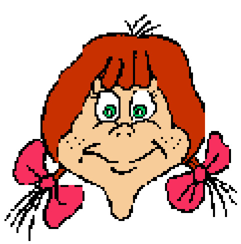
1. Source sends message to all neighbors.
2. Each other node u when receiving the message for the first time from node v (called u's parent), sends it to all (other) neighbors.
3. Later receptions are discarded.

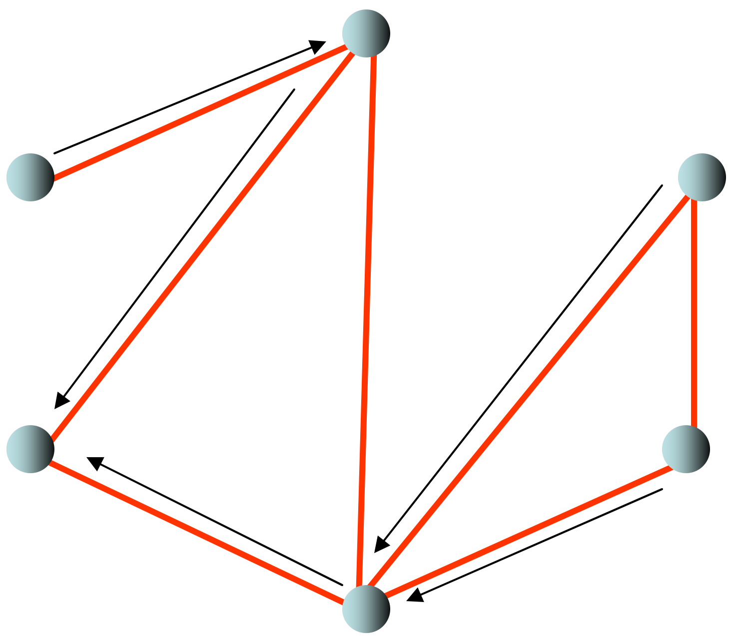
Note that parent relationship defines a tree!
In synchronous system, the tree is a breadth-first search spanning tree!

Convergecast

Convergecast

Opposite of broadcast: all nodes send message to a given node!





Purpose?

E.g., for aggregation!
E.g., find maxID!
E.g., compute average!
E.g., aggregate ACKs!

How?

Aggregation (last time, promised ☺)

Echo Algorithm

Echo Algorithm

0. Initiated by the leaves (e.g., of tree computed by flooding algo)
1. Leave sends message to its parent
2. If inner node has received a message *from each child*, it forwards message to parent

Application: convergecast to determine termination. How?

Sub-tree completed?

Complexities?

Echo on tree, but complexity of flooding to build tree...

BFS Tree Construction

How to compute a breadth-first tree?

Flooding gives parent-relationship, but...

... only if synchronous.

How to do it in asynchronous distributed system?

Dijkstra or Bellman-Ford style....

Do you remember the ideas??

Bellman-Ford: BGP in the Internet!

Dijkstra: grow on the „border“

Bellman-Ford: distances (distance vector)...

Asynchronous BFS Tree

Dijkstra: find next closest node („on border") to the root

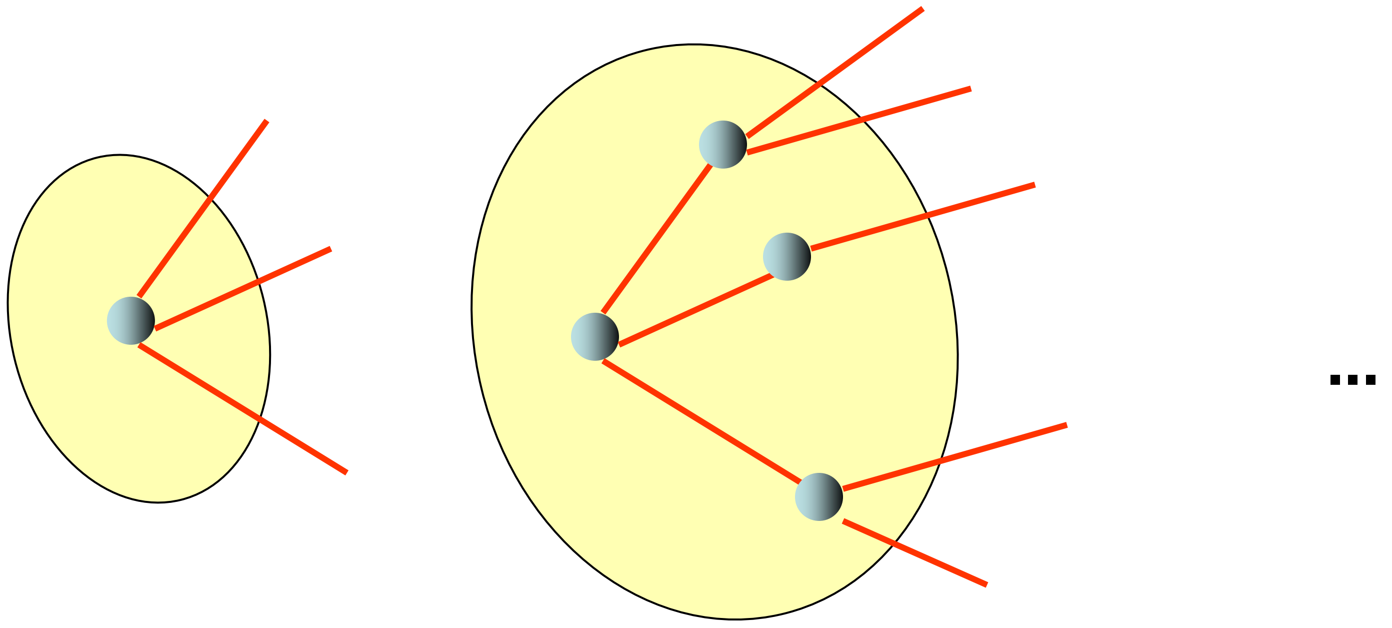
Dijkstra Style

Divide execution into *phases*. In phase p, nodes with distance p to the root are detected. Let T_p be the tree of phase p. T_1 is the root plus all direct neighbors.

Repeat (until no new nodes discovered):

1. Root starts phase p by broadcasting „**start p**" within T_p
2. A leave u of T_p (= node discovered only in last phase) sends „**join p+1**" to all quiet neighbors v (u has not talked to v yet)
3. Node v hearing „join" for first time sends back „**ACK**": it becomes leave of tree T_{p+1}; otherwise v replied „**NACK**" (needed since async!)
4. The leaves of T_p collect all answers and start Echo Algorithm to the root
5. Root initates next phase

Asynchronous BFS Tree: Idea



Phase 1

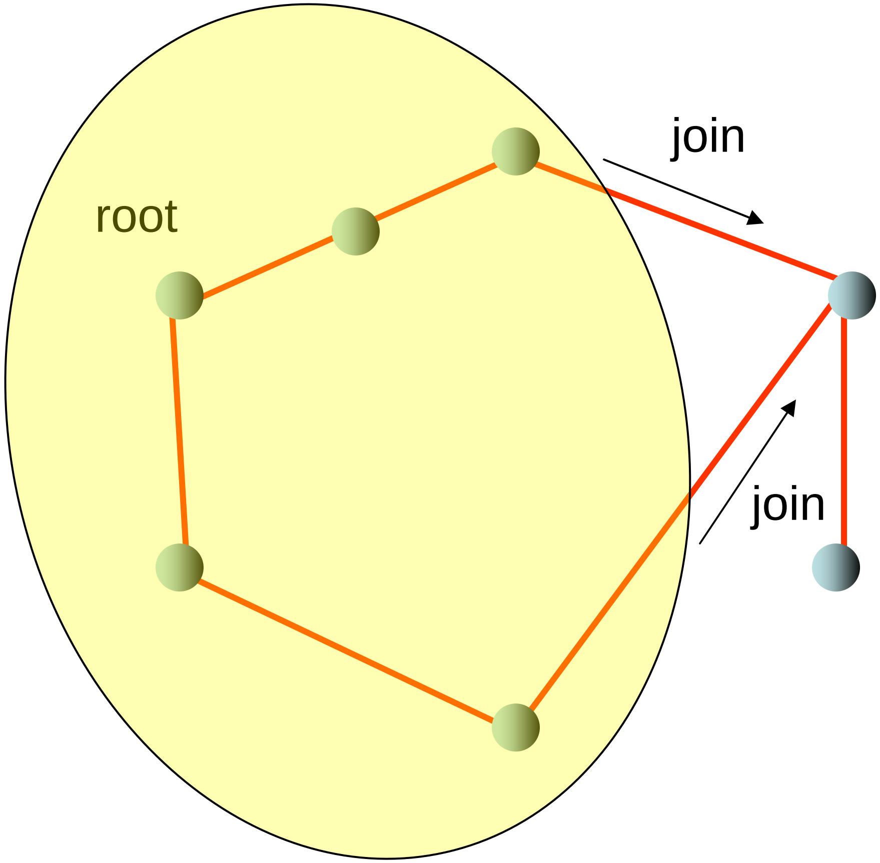
Phase 2

Wait until all
next hops explored...

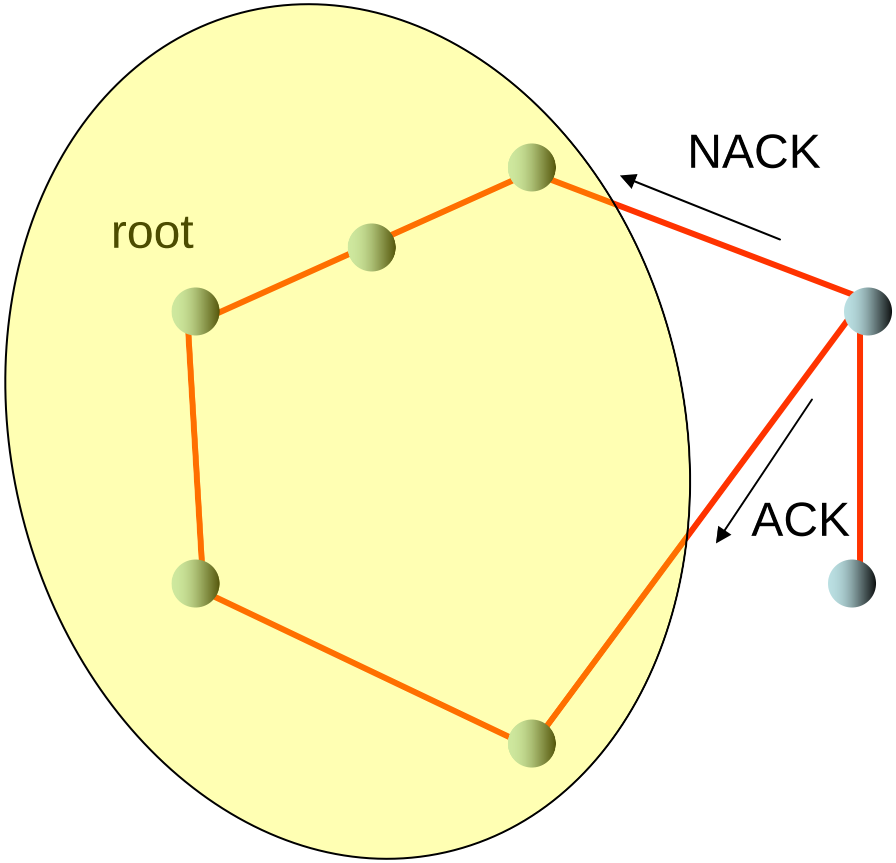
Wait until all
next hops explored...

Asynchronous BFS Tree

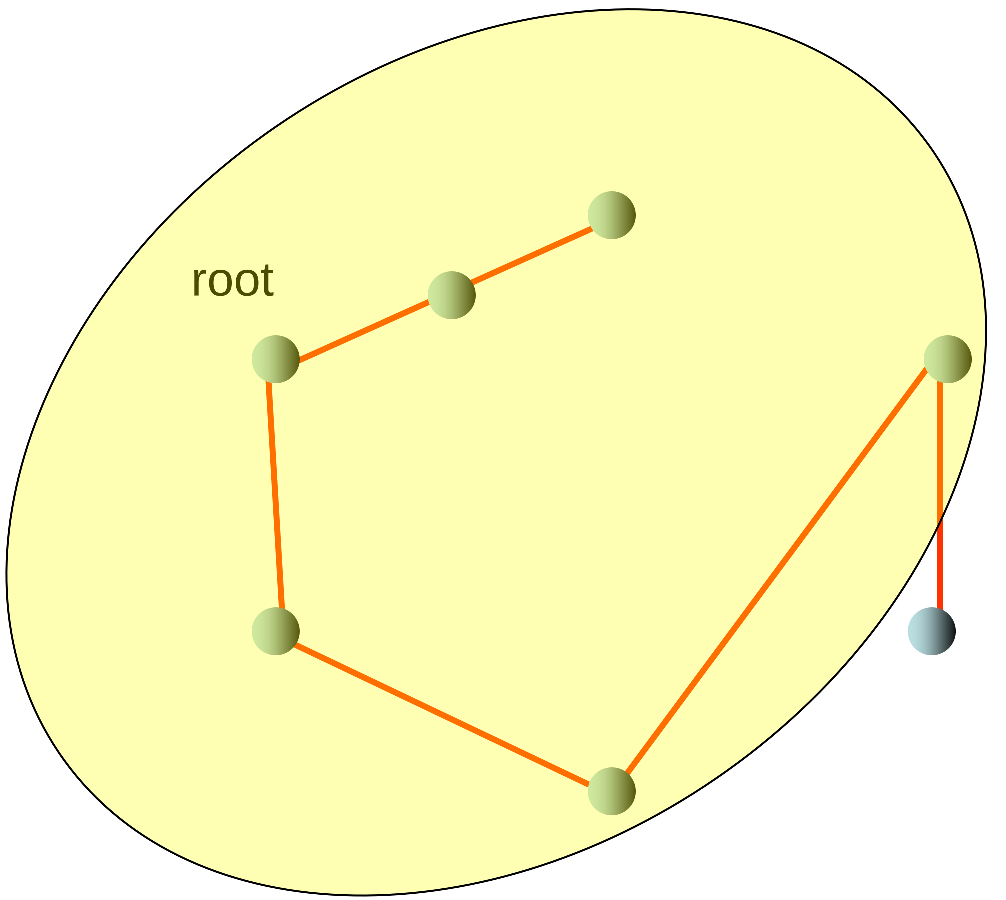
P



Asynchronous BFS Tree

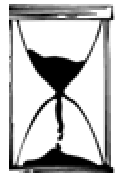


Asynchronous BFS Tree



Analysis

Time Complexity?



$O(D^2)$ where D is diameter of graph...

... as convergecast costs O(D), and we have D phases.

Message Complexity?

O(m+nD) where m is number of edges, n is number of nodes.

Because: Convergecast has cost O(n), one per link in tree, so over all phases O(nD). On each edge, there are at most two join messages (both directions), and there is at most an ACK/NACK answer, so +m...

Alternative algo?

Asynchronous BFS Tree

Bellman-Ford: compute shortest distances by flooding an all paths; best predecessor = parent in tree

Bellman-Ford Style

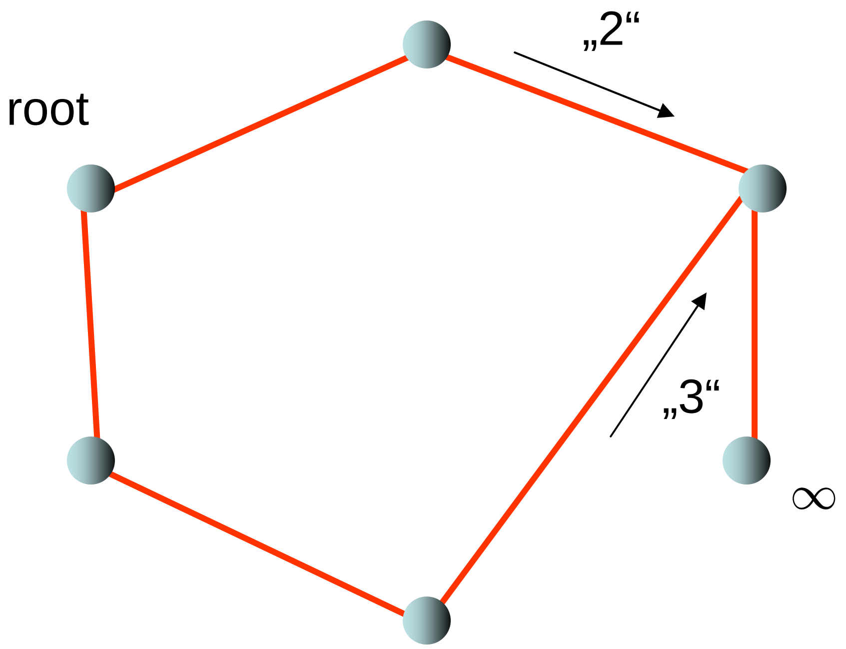
Each node u stores d_u, the distance from u to the root. Initially, $d_{root}=0$ and all other distances are ∞. Root starts algo by sending „1“ to all neighbors.

1. If a node u receives message „y“ with $y<d_u$

   $d_u := y$

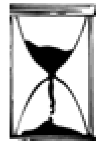
   send „y+1“ to all other neighbors

Asynchronous BFS Tree



Analysis

Time Complexity?



O(D) where D is diameter of graph.

By induction: By time d, node at distance d got „d“.
Clearly true for d=0 and d=1.
A node at distance d has neighbor at distance d-1 that got „d-1“ on time by induction hypothesis. It will send „d“ in next time slot...

Message Complexity?

O(mn) where m is number of edges, n is number of nodes.

Because: A node can reduce its distance at most n-1 times (recall: asynchronous!). Each of these times it sends a message to all its neighbors.

Discussion

Which algorithm is better?

Dijkstra has better message complexity, Bellman-Ford better time complexity.

Can we do better?

Yes, but not in this course... ☺

Remark: Asynchronous algorithms can be made sychronous... (e.g., by central controller or better: local synchronizers)

MST Construction

MST

Tree with edges of minimal total weight.

Another spanning tree? Why?

For weighted graphs: tree of minimal costs...
useful building block (approximation algorithms etc.)!

Assume all links have different weights. So...
MST is unique.

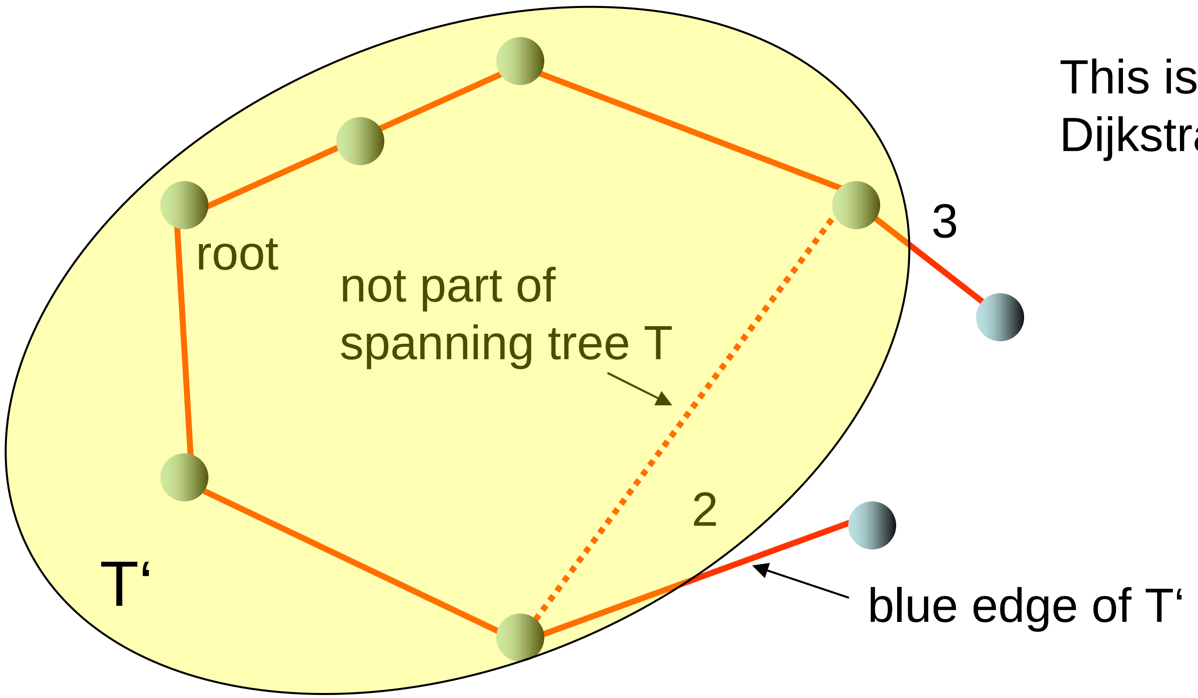
How to compute in a distributed manner (synchronously...)?!
(How to do it classically?)

Kruskal, Prim, ...

Idea

Blue Edge

Let T be a spanning tree and T‘ a subgraph of T.
Edge e=(u,v) is outgoing edge if u $\in$ T‘ but v is not.
The outgoing edge of minimal weight is called *blue edge*.



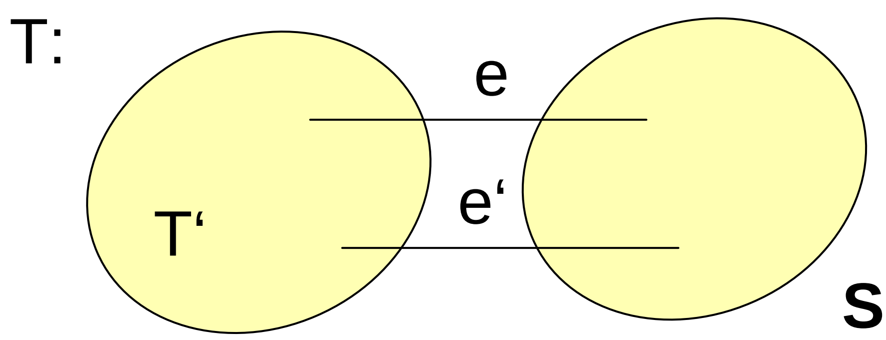
This is like Dijkstra....

Lemma

If T is the MST and T‘ a subgraph, then the blue edge of T‘ is also part of T.

Proof idea?

By contradiction! Suppose there is an other edge e‘ connecting T‘ to the rest of T. If we add the blue edge e and remove e‘ from the resulting cycle, we still have a spanning tree, but with lower cost...



So what?!

Distributed Kruskal

Note: every node must be incident to a blue edge!
We do not have to grow just one component, but can do many fragments in parallel!

This is „distributed Kruskal" so to speak. ☺

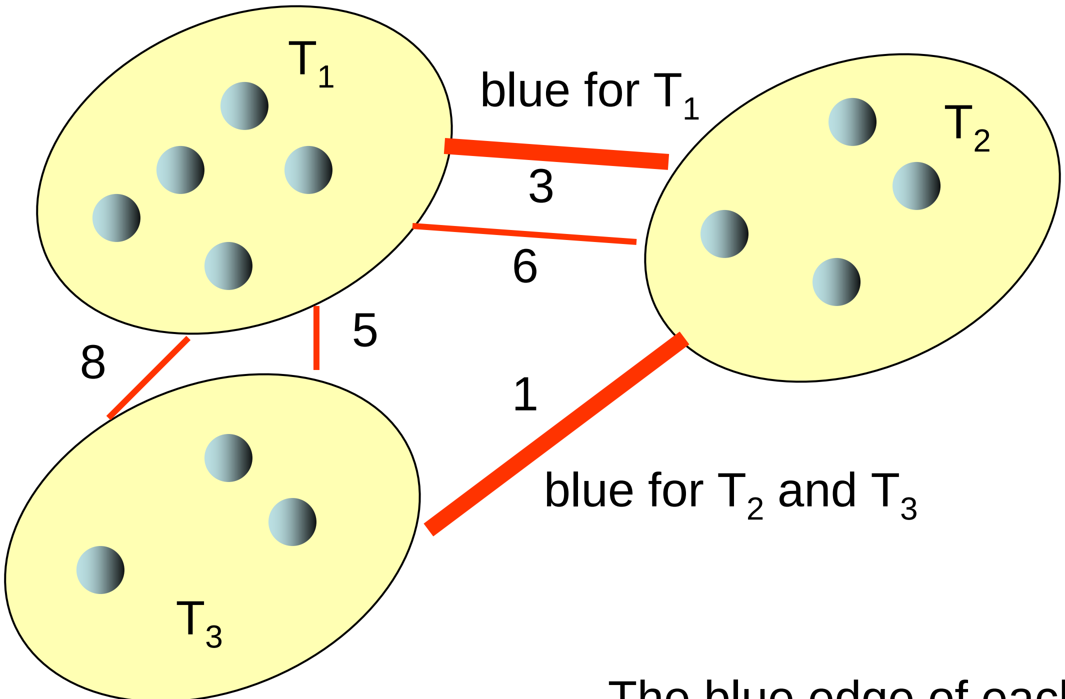
Gallager-Humblet-Spira

Initially, each node is root of ist own fragment.

Repeat (until all nodes in same fragment)

1. nodes learn ID of neighbors
2. root of fragment finds blue edge (u,v) by convergecast
3. root sends message to u
4. if v also sent a merge request over (u,v), u or v becomes new root depending on smaller ID (make trees directed)
5. new root informs fragment about new root (convergecast on „MST" of fragment)

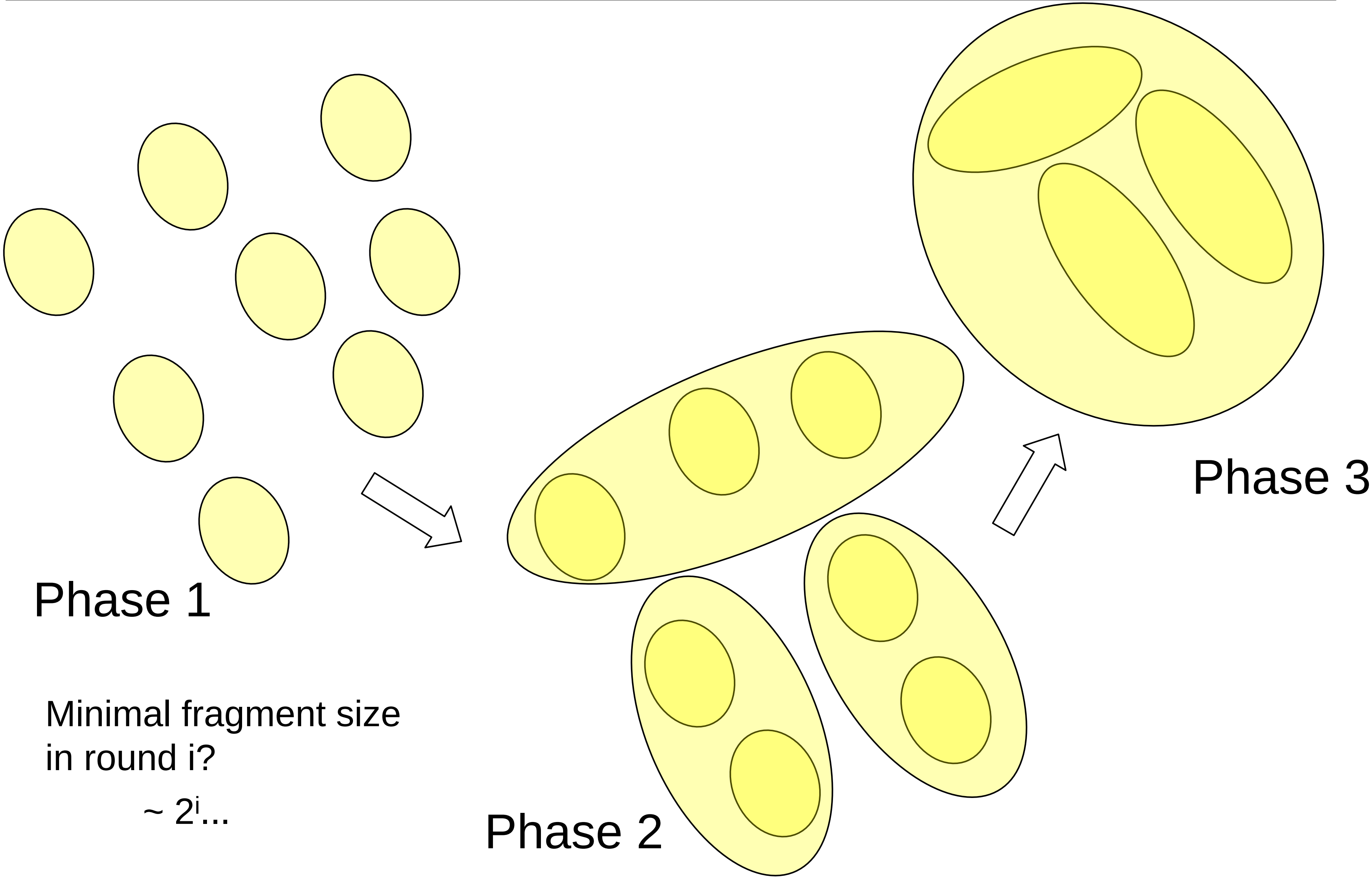
Distributed Kruskal: Idea



The blue edge of each fragment can be taken for sure: cycles not possible! (Blue edge lemma!)

So we can do it in parallel!

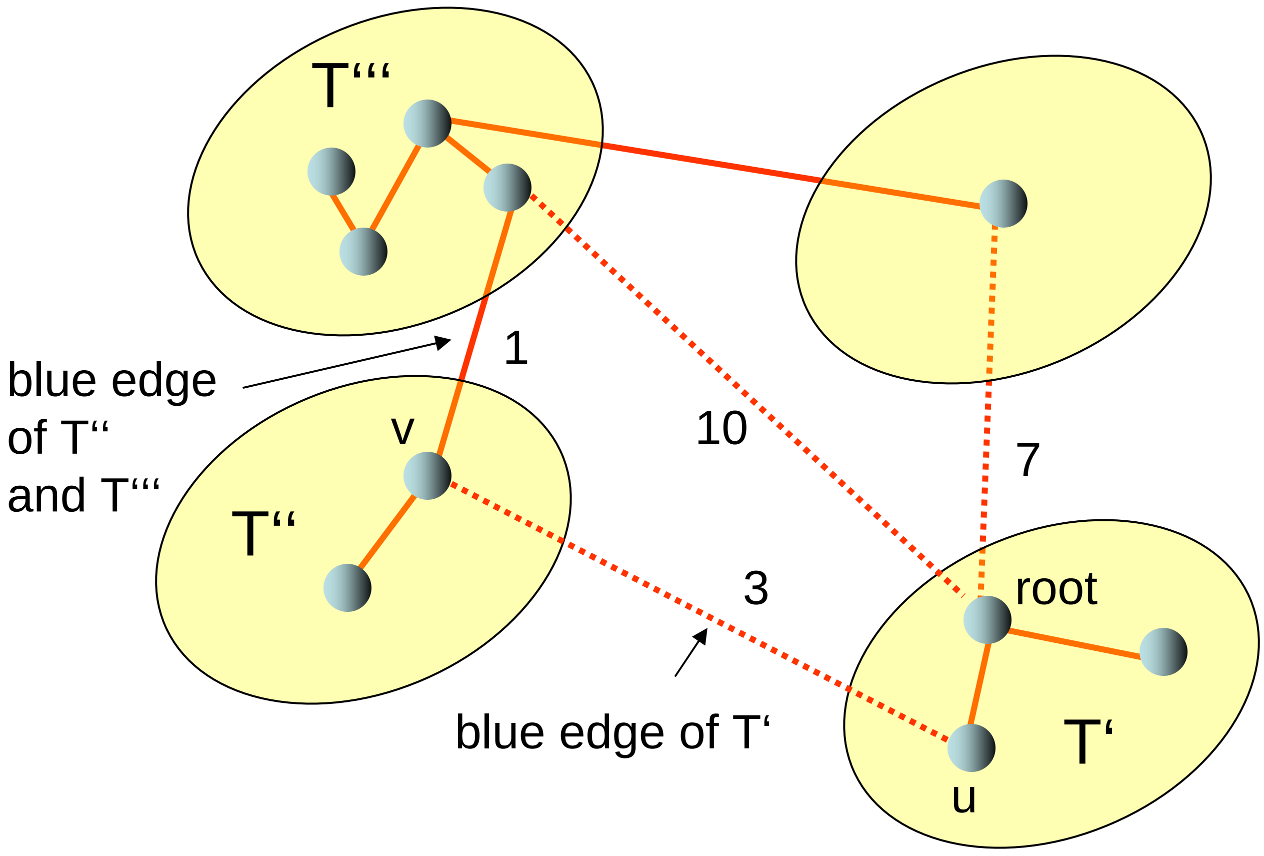
Distributed Kruskal: Idea



Minimal fragment size
in round i?

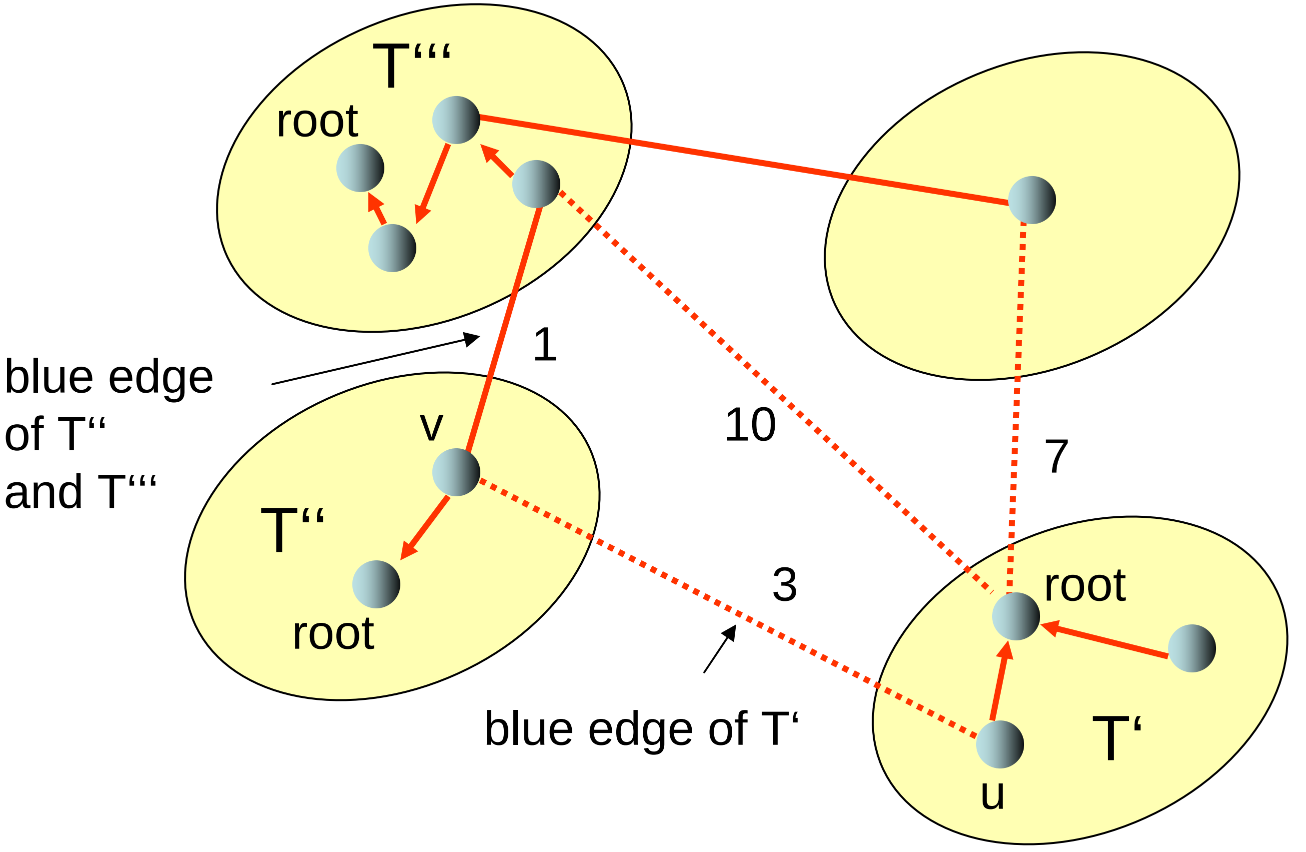
~ 2^i...

Distributed Kruskal



Who becomes overall leader of T and T‘?
Make trees directed...

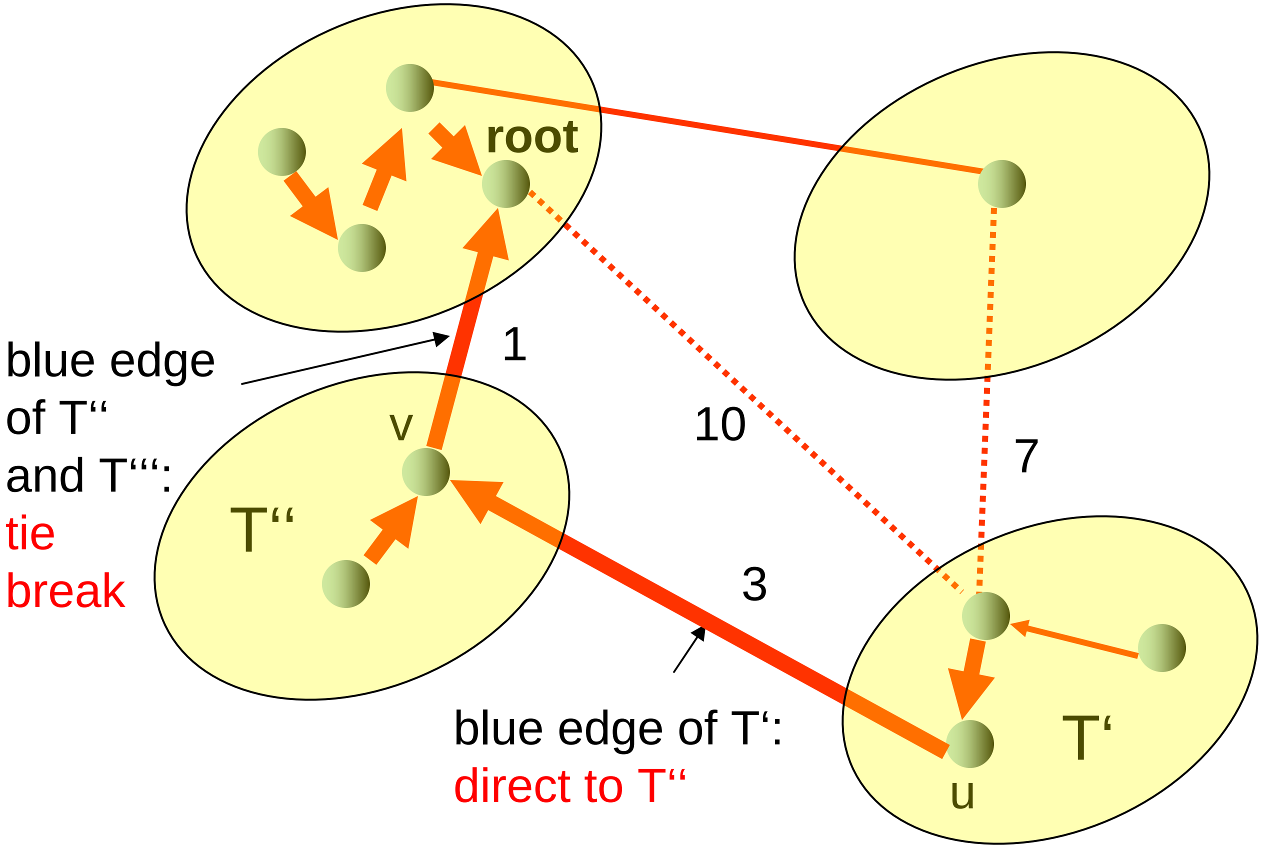
Distributed Kruskal



All trees rooted! How to merge on blue edge (u,v)?

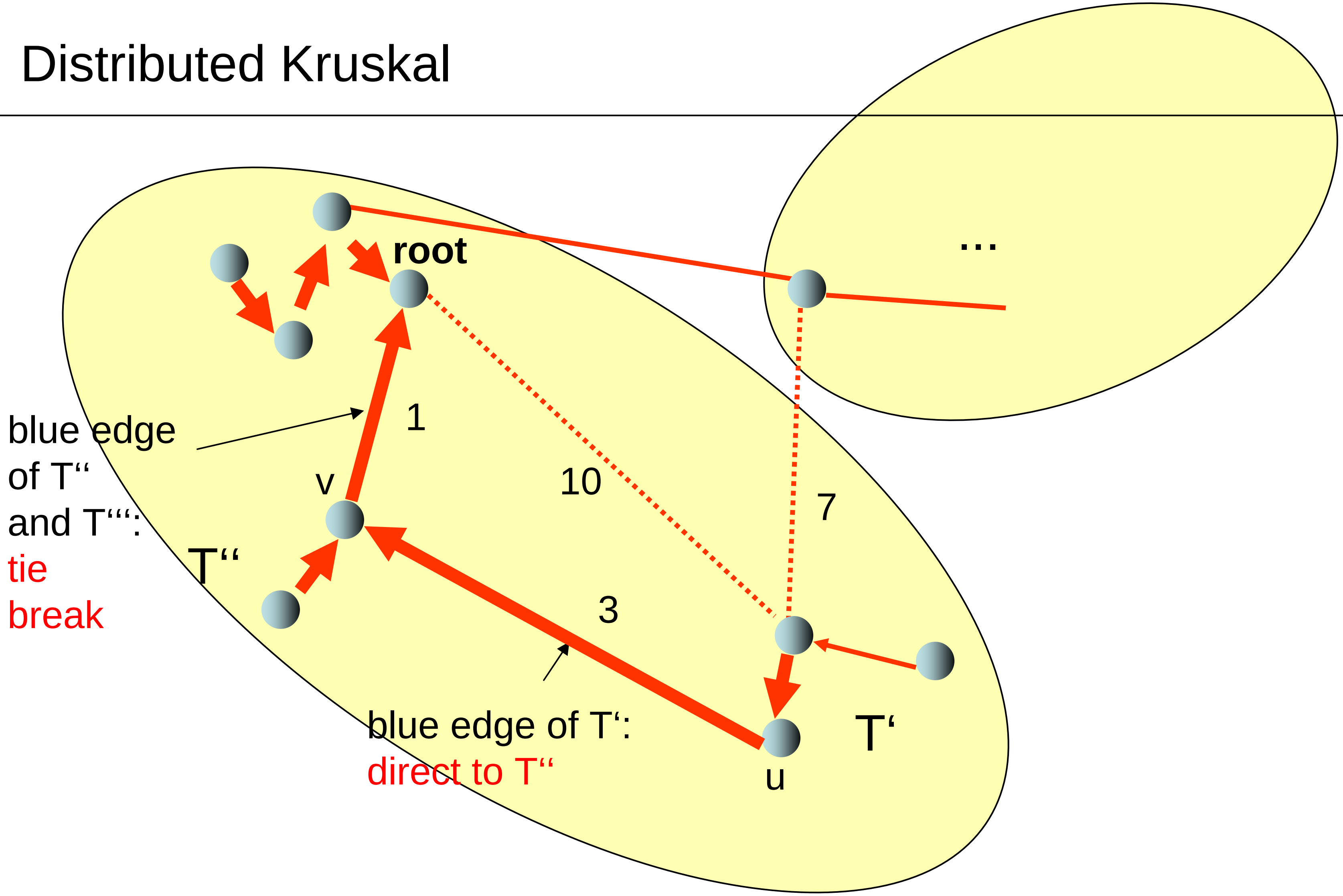
1. Invert path from root to u (u is temporary root)
2. If u and v sent message over blue edge: point blue edge to smaller ID; otherwise v is parent of u..

Distributed Kruskal



New directed tree with new root! ☺
T‘‘‘‘ connects somewhere else...

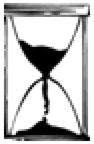
Distributed Kruskal



Merged fragments!

Analysis

Time Complexity?

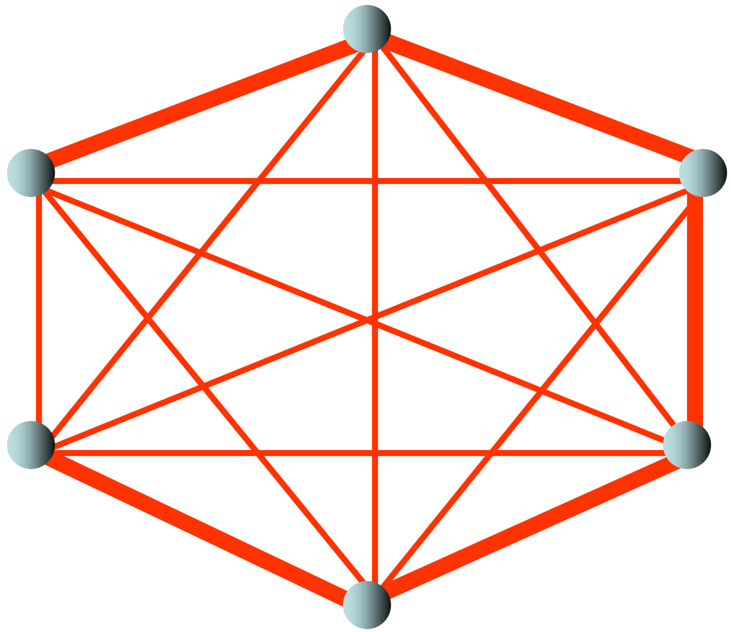


Message Complexity?

Each phase mainly consists of two convergecasts, so O(D) time and O(n) messages per phase?

Analysis

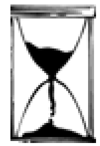
Careful: diameter of MST may be larger than diameter of graph!



O(n) time for convergecast, and not O(1)...

Analysis

Time Complexity?



O(n log n) where n is graph size.

Message Complexity?

O(m log n) where m is number of edges.

Each phase mainly consists of two convergecasts, so O(n) time and O(n) messages. In order to learn fragment IDs of neighbors, O(m) messages are needed (e.g., first phase!).
How many phases are there?

The size of the smallest fragment at least doubles in each phase, so it‘s logarithmic.

Yes, we can do better. ☺
(Is it a good idea to distribute Prim‘s algorithm?)

Literature for further reading:

- Peleg‘s book (as always ☺)

End of lecture