#ASLI ENGINEERING

Minimum Spanning Tree in Distributed Systems

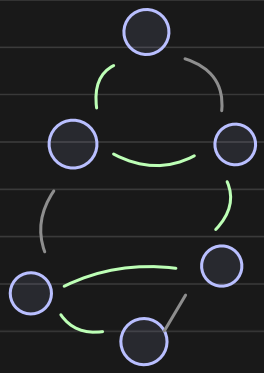
SWIPE

BY

ARPIT BHAYANI

Minimum Spanning Tree in distributed system

Say, a node wants to broadcast a message in a distributed network, but it has to be extremely efficient about it. This is where having a Minimum Spanning Tree makes a difference.



Spanning Tree

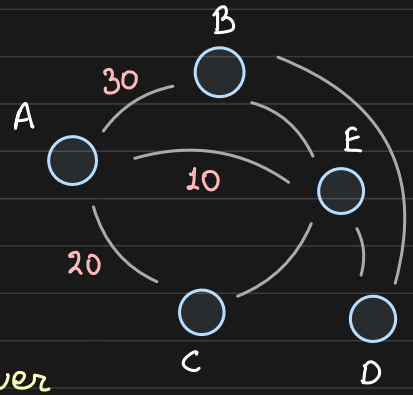
A tree that covers all the nodes through selective edges

Minimum spanning Tree is a spanning tree built such that the total cost of edges included is minimum.

↑ classic graph algorithm

But what does it mean for a distributed system?

Say, nodes are connected and the communication lines have different tariff



Nodes = machines

edges = communication lines

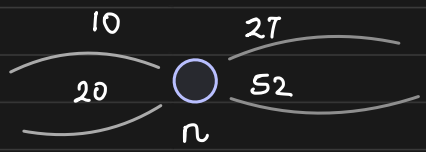
weight = cost to move packet across

For an efficient broadcast, we have to cover all nodes by covering edges such that cost is minimal

Distributed Setup

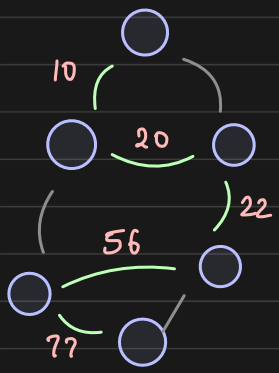
Given that we are working in a distributed setup, there would not be any node that has the entire topological information, and hence we assume

every node knows weight of all its incident edges every node knows the total nodes in the network



Problem Statement

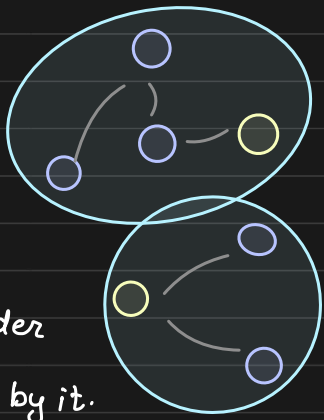
Find minimum-weight spanning tree for the entire network such that each node/machine decides which of its incident edges are and are not part of the spanning tree



Core Strategy

Start small with a forest of spanning trees and repetitively merge them to create a single spanning tree.

The GHS algorithm



The algorithm operates on 'level' and each level is a collection of spanning trees that we call Spanning Forest

Every component (tree) at each level has a leader and the spanning tree is identified / represented by it.

GHS algorithm is synchronous and hence every node proceeds synchronously in rounds.

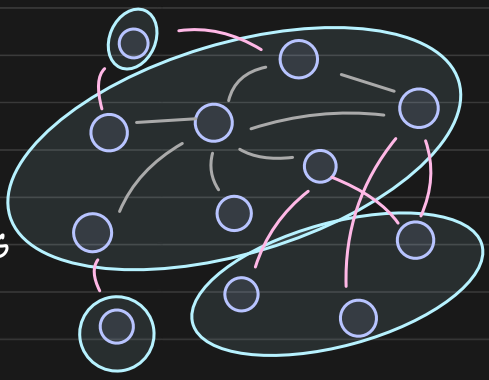
level 0 consist of all the nodes and no edges. Each node is a spanning tree and for n nodes we have a spanning forest of n spanning trees.

At this stage, every node knows

- total number of nodes 'n'
- incident edges to it
- UID of the leader of its component (spanning tree)

Each node within the component sends a search message within the component to get MWOE

Minimum Weight Outgoing Edge



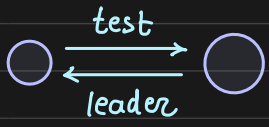
Search message is sent using Broadcast BFS

Each node receiving the search message,

- searches for an outgoing edge that is going out of the component (incident on that node)
- out of all such edges it selects the one with the minimum weight

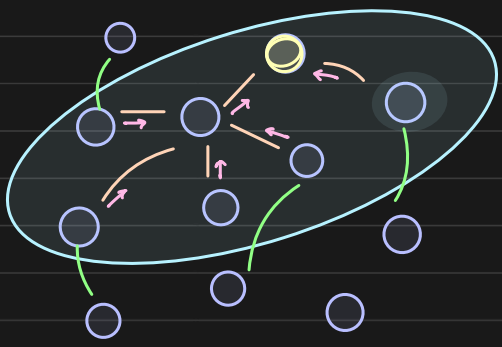
But how would the node know if the other node along its outgoing edge is within the same component or not ?

* Every node sends a test message to its neighbours neighbour can respond with the leader UID of its component and comparing it the node knows if the edge is connecting it to some other component or not.



* We can use any other way to determine this.

Now that every node knows of one * if applicable
outgoing edge that connects itself and
the component to another component
it sends this edge and node info
to the leader of the component



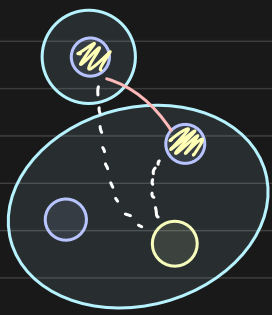
Out of all the minimum edges that
leader receives, it evaluates the global minima and that becomes
the edge that is MWOE of the entire component.

Merging

The components combine over this MWOE and form a larger component. But how?

leader of the component talks to the nodes
connected over MWOE, on either side

- mark the edge to be in the new tree



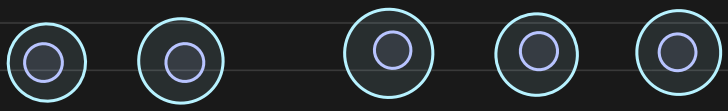
New leader of this merged component will be
the node with larger UID, of the two connected
over MWOE. This info is Broadcasted across component.

level by level

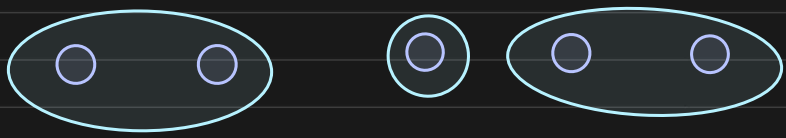
Because each node is independent, so there is a chance that nodes may form conflicting components and merging them forms a cycle, hence we need to bring order...

Hence the algorithm proceeds level by level

level 0 : each node is a component in itself



level 1 : two level 0 components merge to become level 1

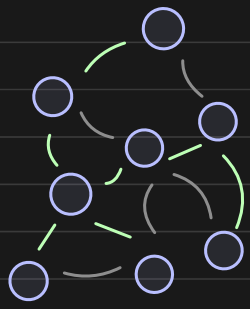


one level 1 and level 0 merge to become level 1

Process continues and level k nodes keep on merging and forming higher levels

Termination of algorithm

After some number of levels, the spanning forest will contain just one tree containing all the nodes. Each node will know the edge part of the spanning tree.



When the algorithm will try to find MWOE, it would not find any. The leader will know about this because it would get empty responses from peripheral nodes suggesting the completion

Complexity Analysis

At each level there are at least 2^k nodes

Because the components are merged by level

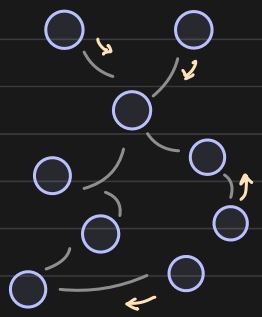
Total levels are at most $\log n$

Each level takes $O(n)$ time hence time complexity is $O(n \log n)$

Communication complexity = $O((n+|E|)\log n)$

at each level $O(n)$ msg sent & $O(E)$ message for deciding MWOE.

Leader Election with MST



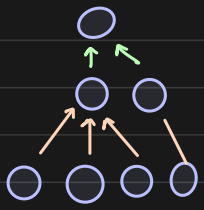
Once MST is constructed, leader election becomes pretty simple, so long as we have UIDs information.

A node with no outgoing edge is a leaf and sitting at peripheral is a leaf node

Leaves initiates a convergecast.

Internal node waits to hear from all but one neighbours

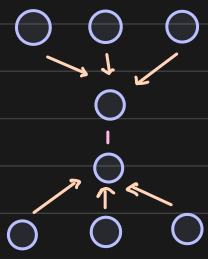
Then it sends message to the remaining neighbour



Termination:

if a node hears from all its neighbours without itself sending any message

it declares itself as the new leader



if two neighbouring nodes receive messages from each other in the same round,

One with the higher UID declares itself as the new leader