#ASLI ENGINEERING

Shortest Path in a Distributed System

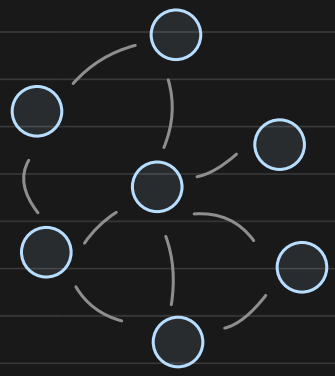
SWIPE

BY

ARPIT BHAYANI

Shortest Path in Distributed Systems

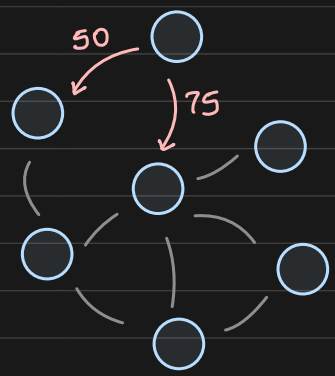
Determining shortest path in a distributed system is an important problem to address. It finds its usecase across many usecases and affects the performance of most functionalities like



- delivering message to a certain node
- efficient routing in a P2P network
- internet works on similar algorithm

Note: shortest is not just about the distance, instead it can also be about - congestion, communication overhead, expensive communication channels, third party cable infra, etc.

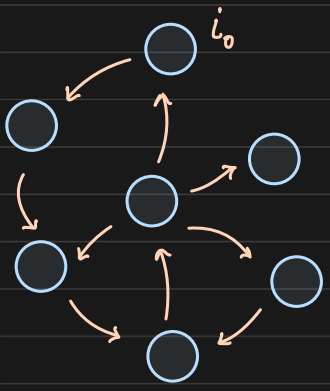
Hence, as a general problem statement, we say each edge connecting two node is directed and has a 'weight' assigned to it. This weight is the quantification of cost / congestion / distance across nodes



Problem Statement

In a distributed network, where nodes are connected via paths/edges having some weight assigned, find a shortest path from a specific source to all the nodes.

Shortest path : path with minimum weight



Bellman Ford algorithm in a distributed setup

Note: Because it is a distributed network no node knows the entire topology and weights.

Every process initially knows the weight of all its incident edges and knows the total number of nodes (n) in the network

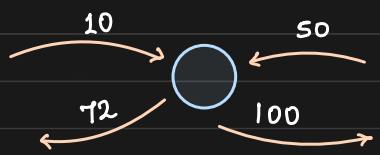
Most interesting thing about Distributed Systems, No one has the complete information

This algorithm is synchronous i.e. it proceeds further synchronously in "rounds"

All nodes move from one round to another in sync via message passing or clocks.

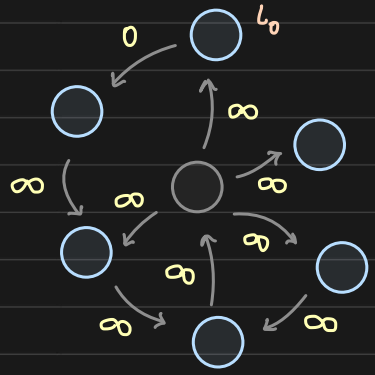
Every node in the network

- knows total nodes n
- knows weight of the edge incident on it



Every node keeps track of dist, that represents and holds the shortest distance to it from i_0

Initially, $dist_{i_0} = 0$ and $dist_i = \infty \; \forall \; i \neq i_0$



At each round, every node sends its dist to all of its neighbours.

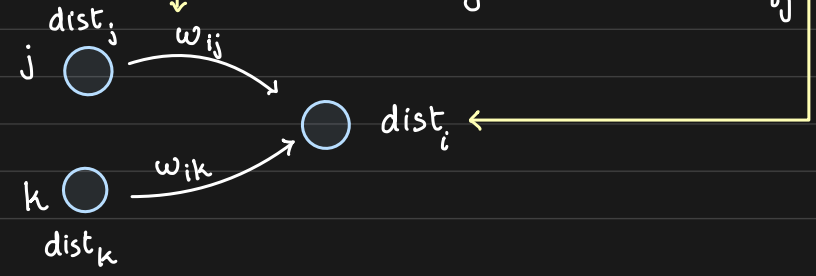
Round 1 → Outgoing

Node i_0 sends dist = 0 to its neighbours

Every other node sends dist = ∞ to theirs

Each node i, upon receiving an incoming dist from j, compares

1. its own dist
2. Incoming dist + weight $_{ij}$



After comparing, it updates its own dist with the minimum of the compared value

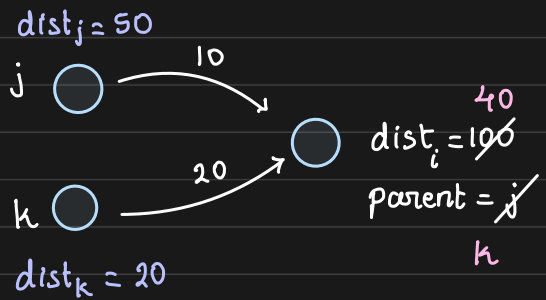
and this signifies the minimum distance (until now) from i_0

$$dist_i = MIN\left(dist_i \ , \ dist_j + w_{ij} \ \forall \text{ neighbour } j \right)$$

↑ Classic relaxation step in Bellman Ford algorithm

Given that, we also need to know the shortest path, every node keeps track of the parent.

if the $dist_i$ relaxes, the parent of the node is updated



i receives dist from j and k and it relaxes dist from 100 to 40 (incoming from k + wt)

Hence, i changes its parent to k

Round n - 1

After n-1 rounds, dist at every node will contain the shortest distance from i_0 and parent will contain its parent in shortest path

The algorithm we discussed is just a distributed variant of the classic BellmanFord algorithm.

Complexity Analysis

Because we require 'n-1' rounds to complete the algorithm, the time complexity is $O(n)$ where n: # nodes in the network.

Because at every round, every node sends message (dist) to all its neighbours, the communication complexity is $O(n|E|)$ where $|E|$ are # edges in the network.