Implementation of GHS in Erlang

Team 7:
Rakesh Mukkara
K.L.Bhanu

Professor: Kishore Kothapalli

Introduction

- GHS is a distributed minimum spanning tree algorithm which involves the construction of a minimum spanning tree in a network where nodes communicate by passing messages to each other.
- Graph should be connected undirected.
- Each Node Should know its neighbours initially.
- FIFO Message delivery order.
- Messages should be exchanged bidirectionally and the arrival time should be finite without errors.
- Weights should be unique.

Local Storage of a Process

- Local Storage of the process has FragmentID(FN), LevelNumber(LN), Edges(Neighbour Edges), Status of Node, best-edge, best-wt, TestEdge, InBranch, FindCount.
- FragmentID -> Representation of a fragment.
- LevelNumber -> Initially 0 and increases while combining fragments(helps in connect).
- Edges -> Edges contain weights, Status of Edges and PID's of neighbours.(Status(Edge) can be basic, branch or Rejected)
- Status of Node(SN) -> SN can be find, found, sleep).
- best-edge and best-wt -> best-edge is the best outgoing edge found by the process until then.

Local Storage of a Process

- TestEdge -> a basic edge which will be used for checking the validity of the edge using the Test Message.
- InBranch -> InBranch is the parent of the node.
- FindCount -> FindCount is number of unanswered edges about their status these will be altered when the report come to the node and while SN = Find.

Status of the Node:

Status of the node will be **sleep** at the start of the algorithm, **find** when it is trying to find the MOE, found at all other time.

Status of the Node:

Status of edge is **basic** if not yet decided or initial stage, **branch** if it is part of the MST, **rejected otherwise.**

Types of Messages

Wakeup

Node in a sleeping state can be awoken by using wakeup message, This can be done manually or the algorithm eventually wakes up every node if atleast 1 node is manually awoken.

Connect

Connect is sent from the Minimum Outgoing Edge(MOE) for joining the fragments, joining of the fragments depend on the Levels of both nodes sent and received(LN(sent) <= LN(Received)) otherwise will place the message at the end of the queue to wait for level increase.

Initiate

Initiate is like a broadcast message which helps in changing the FragmentID, Level Number for connecting fragments and it also takes care of finding the MOE when Staus = find and also send Test message across the minimum basic edge.

Types of messages

Test

Test checks the validity of edge based on LevelNumbers and FragmentID, based on that it will send accept and reject messages.

Accept

If a node receives Accept it will check the edge with the best edge it found and replaces if new edge has low weight and executes procedure report.

Reject

Reject sets the status of the edge to reject and then send the Test message to minimum weight basic edge again.

Types of Messages

Report

Report messages are like convergecast messages, they carry the best edge to the parent and depending on the weight and status of the node it can report to its parent, it can stop.

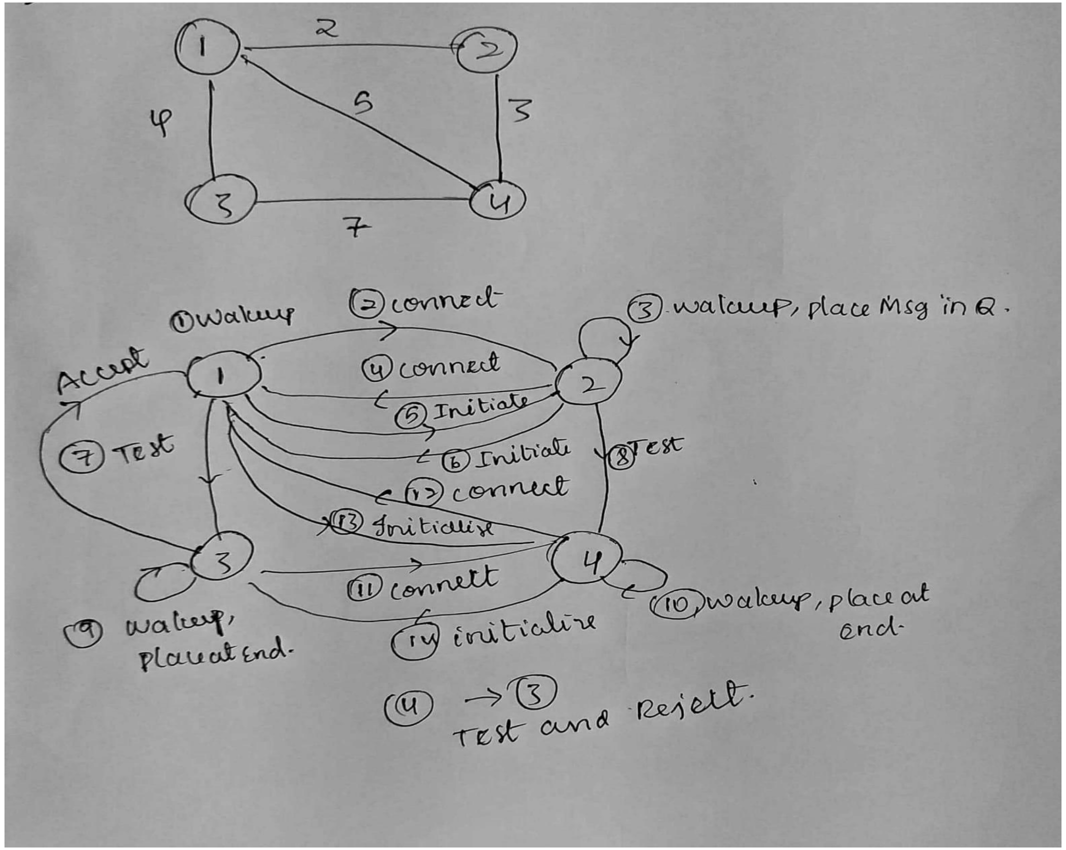
ChangeRoot

When the MOE is found from the report message changeRoot message will be sent through the best-edges and it will send a connect at MOE to other fragment.

End Condition

In the report messages if the best weight it got is infinite from branches after FindCount becoming 0 then the node can exit as there are no MOE’s present.

Example


1
2
3
4
2
3
4
5
7
① wakeup
② connect
③ wakeup, place Msg in Q.
④ connect
⑤ Initiate
⑥ Initiate
⑦ Test
⑧ Test
⑨ wakeup, place at End.
⑩ wakeup, place at end.
⑪ connect
⑫ connect
⑬ Initialise
⑭ initialize
Accept
④ → ③ Test and Reject.

Time and Space Complexity

Local Complexity -> O(E) depends on number of edges.

Maximum number of levels -> O(log N) N is the number of nodes.

Time Complexity -> O(d + l logN) d is the diameter of graph and l is the local diameter.