Implementation of Asynchronous GHS Algorithm to find Minimum Spanning Tree in a Distributed Environment

Introduction

This project aims at developing a working program to find a minimum spanning tree in an asynchronous distributed environment. To achieve this we followed the AsynchGHS algorithm. The algorithm proposed by Gallager, Humblet, and Spira, is one of the best known distributed algorithm.

Problem Statement

We assume that the underlying graph G = (V, E) is connected and undirected, and we assume that the edges have associated weights. We want the processes to cooperate to construct a minimum-weight spanning tree (MST) for the graph G, that is, a tree spanning the vertices of G whose total edge weight is less than or equal to that of every other spanning tree for G. We assume that processes have UIDs and that the weight of each edge is known to the processes associated with the incident vertices [1].

Approach

- Each thread represents a process. These threads will be spawned by a master thread
- Threads communicate with each other using **Blocking Queues**
- Each message between threads has a delay factor. Delay will be determined randomly between 0 and 20
- If thread – 1 wants to send a message to thread - 2, thread - 1 places the message in the thread - 2's blocking queue. Thread – 2 reads the data from its blocking queue after the delay of the message becomes zero.
- Master thread will send a "**tick**" message every 10ms. During each tick message, the delay factor of messages in all the blocking queues decrements by 1.
- Messages from the thread are ordered in a FIFO manner. **FIFO is implemented by adding a message number** to each message. If there are two messages in the blocking queues from the same sender, then the message with the small message number will be processed first
- Each component is represented using a triplet. **The first element of the triplet is the higher process ID of the core edge, second element is the length of the core edge and third element is the lower process ID of the core edge**
- Algorithm terminates when the leader of the component finds that no new MWOE could be found

References

1. Lynch, Nancy A. (1996-04-16). Distributed Algorithms (The Morgan Kaufmann Series in Data Management Systems). Elsevier Science. Kindle Edition.