CPSC 479 Project 1: Introduction to HPC - Electing Two Leaders in a Ring Topology

Prof. Doina Bein, CSU Fullerton

dbein@fullerton.edu

Introduction

In this project you will design and implement one algorithm related to leader election on a ring topology. You will design one algorithm, describe the algorithm using clear pseudocode and implement your algorithm using MPI primitives, compile, test it, and submit BOTH the report (as a PDF file) and the program. Let N be the size of the distributed system, i.e. run your program with N identical copies. The value of N must be greater than 5. You can choose the value of N to be between 6 and 20. **But each run of a program must terminate within 60 minutes if N=10**.

The even-odd-counting problem on a ring

Given a distributed system with N processes arranged in a ringlike topology, each process goes through a finite number of events. The events could be internal or send or receive.

Consider the ring communication shown in class, when the process of rank 0 sends a message to the process of rank 1, and that message is forwarded to the process of rank 2, etc. until it reaches back the process of rank 0: we can simplify that by saying that

- the process of rank k (k>0) receives the message from the process of rank (k-1) modulo n and
- the process of rank k ($k \geq 0$) sends a message to the process of rank (k+1) modulo n.

Each process generates a single random integer value and does some computations on it before sending it:

- The value needs to be positive, so if the value is negative, then take the absolute value of it.
- The value needs to be greater than 10 and less than 100, so if the value is less than 10, then add 10 to it; if the value is greater than 100, then take modulo 100.
- The value 1 gets concatenated with the random value and then the resulting value gets concatenated with the process rank, such that two random values that are equal become distinct. For example, if the process of rank 0 generates the random value 15, then the new value is 1150 (1 concatenated with 15 concatenated with 0). Another example: if the process of rank 2 generates the random value of 78 then the new value is 1782 (1 concatenated with 78 concatenated with 2).

Leader Election Problem in a Ring

The leader election problem[1] is the process of designating a single node in a distributed network of autonomous nodes as the organizer of some task. Each node needs to agree on whom among them will be the leader node, and the leader needs to announce itself. So there are two parts: leader selection and leader

[1] https://en.wikipedia.org/wiki/Leader_election

announcement. For the leader selection, we consider a simplified algorithm that will prevent deadlocks in sending/receiving messages between neighboring nodes.

In the leader selection, process 0 sends its computed value to process 1. On receiving the value, process 1 compares it with its own and if it is larger than its own, it sends it further to process 2. If it is the same or smaller than its own, it sends instead its own value to process 2. Process 2 does the same check. The value received finally at process 0 after it traverses the entire ring will be the largest generated value in the ring and will be declared by process 0 as the leader.

Our Leader Election Problem in a Ring (to be implemented)

Our algorithm selects two leaders, which we can call them as the president and the vice-president. The president will be the largest odd value and the vice president will be the largest even value. The two elections can run concurrently (by sending/receiving two values in the same MPI send) or separately (sending one message for president-odd value and another message for vice-president-even value).

We will describe next how the concurrent two leader election algorithm works, followed by a sequential version. You can choose to implement either of them.

Concurrent two leader election algorithm

Process 0 sends its computed value to the left neighbor (clockwise). That value can be even or odd. If the value is odd, then process 0 considers a second value of 0; if the value is even, then process 0 considers a second value of 1. The two values are sent to process 1.

On receiving the value, process 1 compares:

- its value with the odd one if it has an odd value or
- its value with the even one, if it has an even value.

The largest value and the other unchanged value are sent further to process 2. Process 2 does the same check. Eventually, the two values received at process 0 will be the largest odd and even generated values in the ring and will be declared by process 0 as the two leaders.

Sequential two leader election algorithm

We will first have the election of the odd leader (president) followed by the election of the even leader (vice-president).

Electing the odd leader: If process 0 has an odd computed value, it sends it to process 1; if process 0 has an even computed value then it sends the value of 1 to process 1. On receiving the value, if process 1 has:

- also an odd value, it compares the received value with its own and the largest among the two is sent further to process 2

- an even value, then the value is sent further to process 2.

Process 2 does the same comparison and sends the result to process 3, etc.. The largest value received at process 0 is declared as the odd leader.

<u>Electing the even leader</u>: If process 0 has an even computed value, it sends it to process 1; if process 0 has an odd computed value then it sends the value of 0 to process 1. On receiving the value, if process 1 has:

- also an even value, it compares the received value with its own and the smallest among the two is sent further to process 2

- an odd value, then the value is sent further to process 2.

Process 2 does the same comparison and sends the result to process 3, etc. The largest value received at process 0 is declared as the even leader.

Implementation

You are provided with the following files.

1. README.md contains a brief description of the project, and a place to write the names and CSUF email addresses of the group members. You need to modify this file to identify your group members.
2. LICENSE contains a description of the MIT license.

What to Do

Add your group members' names to README.md and remove the existing ones. Write the pseudocode for either algorithm, describe what parameters are needed to execute the program and add it to the report. Implement your algorithm in C/C++ using MPI commands to solve the problem described. Add to the report what is the command to run the program(s). Submit your report as a PDF by committing it to your GitHub repository along with your code. Your report must include the following:

1. Your names, CSUF-supplied email address(es), and an indication that the submission is for project 1.
2. A full-screen screenshot with your group member names shown clearly. One way to make your names appear in Atom is to simply open your README.md.
3. The pseudocode for the chosen algorithm
4. A brief description on how to run the code.
5. Two snapshots of code executing for some two distinct values of N.

Grading rubric

The total grade is 40 points. Your grade will be comprised of three parts, Form, Function, and Report:

- Function refers to whether your code works properly (30 points).
- Form refers to the design, organization, and presentation of your code. The instructor will read your code and evaluate these aspects of your submission (3 points):
    - README.md completed clearly (1 points)
    - Style (whitespace, variable names, comments, helper functions, etc.) (2 points)
- Report (7 points) divided as follows:
    - Summary of report document (2 points)
    - Pseudocode of the chosen algorithm (2 points)
    - Three screenshots: one for the group members and two snapshots of code executing for some two distinct values of N (1 point each, total 3 points)

Obtaining and Submitting Code

This document explains how to obtain and submit your work:

GitHub Education / Tuffix Instructions

Here is the invitation link for this project:

https://classroom.github.com/g/hakUMbmX