Architecture et Système 2023-2024

Département Informatique, ENS Paris Saclay

TP 13 : Leader election and network

1 Ring election - an introduction

An important problem in concurrent programming is choosing a leader, e.g. for organizing and coordinating a network. We start with a number n of computers or processes (we will call them *nodes*) identical in principle and function, with the exception of a unique identifier. By following the same protocol, all the nodes will agree on the choice of a leader.

There is a wide variety of protocols leading to this result. We are going in this TP to study and implement one : the Dolev, Klawe, and Rodeh protocol which is distinguished by the low number of messages traded for election : $\mathcal{O}(n \log n)$. By comparison, simple approaches tend to use $\mathcal{O}(n^2)$ messages. The explanatory slides are available in the directory.

The protocol assumes that the nodes are organized in a ring, each node sends messages to his neighbor on the right. The TP directory contains a skeleton that will create n processes (the nodes) for n given ; these processes will already be equipped with pipes to communicate. Your task is to complete the program by simply implementing the protocol function, following these steps :

— Initially, all nodes are active and have a unique identifier.

— An active node waits for a small random period (the `delay()` function does this), then sends its identifier to its neighbor on the right. Then, it waits for the credentials of its two closest active neighbors at its left. It will then decide whether to stay active, become passive, or declare itself a leader. The conditions are specified in the presentation. If it remains active, it repeats the behavior shown above.

— A passive (inactive) node simply forwards all messages received from the left to its neighbor on the right. In addition, if the message declares a leader, it displays a corresponding message on the screen.

— There are three types of messages exchanged by the nodes, all in the format (type, identifier). - “neighbor” (`v`) : to send his identifier to the neighbor on the right. - “next” (`p`) : a node which has received the identifier of its left neighbor sends it to its neighbor of right. - “winner” (`g`) : a node declares itself a leader.

2 Implementation on your own computer

1. Complete the `protocol` function of the `ring-pipe.c` file which implements the above-described protocol.
2. Test your code with multiple values of n. See if your program always terminates, and if there is always exactly one leader.

3 Netowrk implementaton

The network version of the program, `ring-net.c` instantiates a single node per run. You can copy the `protocol` function of your own `ring-pipe.c` The program takes 3 arguments : the port listening node, the name of the neighbor node (machine name), the connection port to the neighbor node.

The program creates a server in a thread and waits for a neighbor to connect. In parallel, in a another thread, it tries a connection on its neighbor (machine name and port passed as argument).

— Write a script that implements the execution of your code on 5 machines in the department. Start by testing the execution of your code locally (localhost machine using different ports). Gradually expand the size of the ring.
— Create a ring with your neighbors and extend it to the whole lab room.

4 Chat

We want to create an application to chat with each other on the terminal. So we need a program that can both receive and send messages. To do this, we'll either create two threads, or two processes. One will act as a server, the other as a client. You can use the functions available in the files packets.c and packets.h.

How can I receive messages from several people and send messages to several people ? Implement.