IN.5022: Concurrent and Distributed Computing - Serie 1

Antoine Tissot

September 25, 2023

1 Exercise 1

A distributed system must check if a given integer N is a prime number. The system has a fixed number of processes. Initially only a specific process, known as the initiator, knows N. The final answer must be available to the initiator. Imagine a distributed algorithm that could solve this problem in an "efficient" manner. Describe what each process will do and what interprocess messages will be exchanged. Assume that each process is able to check whether a number is divisible by another number (i.e., you don't need to describe that part of the algorithm).

To determine if an integer N is prime, the range from 2 to $\sqrt{N}$ is inspected for divisors. The work is distributed among all the processes.

Algorithm:

Initiator:

1. Broadcasts the number N to all other processes.
2. Divides the range 2 to $\sqrt{N}$ into equal parts based on the number of processes. If there are P processes, each process will handle approximately $\frac{\sqrt{N}}{P}$ numbers.

Every Process:

1. Receives the number N and the range subset it should check.
2. Checks for divisors of N within its assigned range.
3. Sends a message back to the initiator: "Not Prime" if a divisor is found, otherwise "No Divisor Found".

Initiator:

1. Waits for responses from all processes.
2. Concludes N is not prime if any process reports "Not Prime".
3. Concludes N is prime if all processes report "No Divisor Found".

2 Exercise 2

To ensure the robots do not collide, we can use a simple protocol based on their (x, y) positions. Given the robots move synchronously, we can ensure that if they're about to occupy the same space in the next step, one of the robots should wait.

Robot A (Moving Horizontally):

1. Check its own position (x_A, y_A).
2. Check the position of Robot B (xB, yB).
3. If $x_A == x_B$ and $y_A == y_B$, then both robots are at the same position, which shouldn't happen initially but if it does, Robot A waits for a step.
4. If $x_A + 1 == x_B$ (means Robot A is about to move to Robot B's current position in the next step) and $y_A == y_B$, then Robot A waits for a step.
5. Otherwise, Robot A moves one step to the right.

Robot B (Moving Vertically):

1. Check its own position (x_B, y_B).
2. Check the position of Robot A (x_A, y_A).
3. If $x_B == x_A$ and $y_B == y_A$, then both robots are at the same position, which shouldn't happen initially but if it does, Robot B waits for a step.
4. If $y_B + 1 == y_A$ (means Robot B is about to move to Robot A's current position in the next step) and $x_B == x_A$, then Robot B waits for a step.
5. Otherwise, Robot B moves one step upward.

This method ensures that at each step, the robots check for potential collisions and one of them stops moving if a collision is imminent, while the other proceeds.

3 Exercise 3

How might the clocks in two computers that are linked by a local network be synchronized without reference to an external time source? What factors limit the accuracy of the procedure you have described?

1. **Request-Reply Mechanism:** Computer A sends a time request to Computer B. On receipt, Computer B replies with its current time. Then, Computer A calculates the round-trip time, divides it by two to estimate one-way delay, and adjusts its clock accordingly.
2. **Accuracy limits:**
   - Network delays
   - Processing delays: Time taken to process the message can differ between requests.
   - Clock resolution: Different granularities of clocks between the two computers.

How could the clocks in a large number of computers connected by the Internet be synchronized? Discuss the accuracy of that procedure.

1. **Network Time Protocol (NTP):** Servers in a hierarchical stratum system disseminate time. Computers synchronize with these servers using request-reply mechanisms. The system ensures that computers pick the best server to get the most accurate time.
   Stratum refers to a level or layer in the hierarchy of time servers. Each stratum level represents how many steps away a server is from a reference clock source.

2. **Accuracy:** NTP can typically synchronize clocks to within 10 milliseconds on the Internet. On local networks, accuracy can be as good as 1 millisecond.

   - Network delays: Internet traffic can cause inconsistent delays.
   - Stratum levels: Lower stratum servers are more accurate; higher levels add cumulative delay.
   - Malicious attacks: Potential targeting of NTP servers can affect accuracy.

4 Exercise 4

Consider the example presented in the course that converts a non-FIFO channel to a FIFO-channel. This example uses a buffer (queue) of infinite size. Describe the changes that must be made so that it works with a buffer of size w. The size w is known from both the sender and the receiver process. There is no upper bound on the propagation delay of messages, so this delay can be arbitrarily large.

Sender:

1. **Sequence Numbers**: Attach a unique sequence number to each message.
2. **Window Size**: Send up to w messages without waiting for acknowledgment.

Receiver:

1. **Expected Sequence**: Track the next expected sequence number.
2. **Buffer Storage**: Store out-of-order messages in the buffer until the expected one arrives or the buffer is full.
3. **Acknowledge**: After processing a message, send an acknowledgment.
4. **Buffer Full**: If the buffer is full and a new out-of-order message arrives, discard the earliest buffered message and notify the sender.

5 Exercise 5

Docker is up and running.