Master 2 Ubinet / SI5 – Exam, 3 hours
An algorithmic approach to distributed systems

F. Baude

18 November 2019

1 Exercise on message based computing (6 pts)

Assume a network organized as a uni-directional link. Associated to each link, there is a weight (an integer value, corresponding to the cost of transmitting a message on this link). See following ring, on Figure 1. The problem is, for each process to learn how much it would cost it to send a

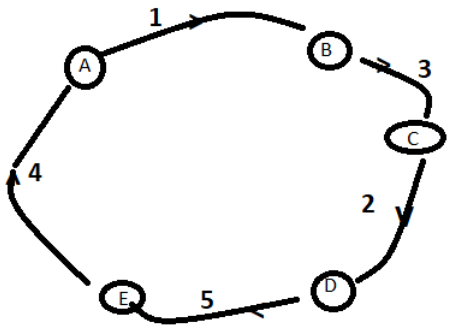

Figure 1: The ring with weighted links

message at a distance d. See for instance on Figure 1 where d has been set to value 3.

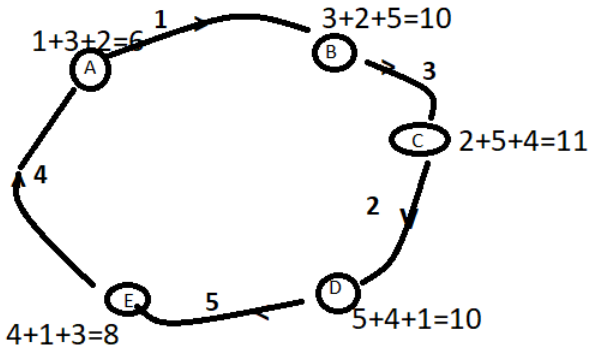

Figure 2: How much it costs for each process to send one message forward on the ring at a distance = 3

Question 1: a weighted sum computation algorithm

The goal is to end up with a **simple** algorithm on a uni-directional ring, that allows for each process to compute the sum of weight of links forward, up to a distance d. Each process should get back this sum. Somehow, each process creates a message (a token), that has to visit each process, accumulating the needed values, before coming back to it, holding now the computed sum.

You can assume that each process has a local integer variable, named *count*, initialized to 0. Spontaneously, when *count* is equal to 0, each process will be able to start its computation to compute that sum at distance d. To simplify, assume that d is $\leq$ the total number of processes on the ring. Assume that each process has a local function that gives the weight of the link that connects it to the next process in the ring.

Make sure that: each process will ending up with the requested sum, meaning, all processes have to get a chance to start this computation.

Question 2: Termination detection to be added

Given the algorithm you have given in Question 1, consider now the problem of termination, say another way, how you can handle the termination in a proper way. It may depend of your solution, but, you must end up with a solution that when a process leaves this application this means that 1) it has got the sum that it expected to be computed, and 2) its participation is not needed for the other processes of the ring to also get back their expected sum value.

If your discussion shows that you need a detection of termination algorithm, sketch it.

2 Group communication (5 pts)

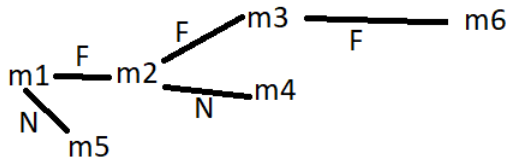

Figure 3: Ordering relation of messages to broadcast among the 2 processes forming a group of processes

The figure above depicts the causality relation between some **broadcast** messages. Moreover, the label on each arrow means the following: "F" is a FIFO ordering relation between the two messages, "N" is a Network ordering relation between the two messages, and, notice that we do not need to show transitivity relations as it can easily be deduced from the causality diagram.

The exercise is about applying the CBCAST (Causal Order Broadcast) algorithm, using Vector Clocks, to the time diagram presented on figure 4.

The causal order broadcast algorithm relies on a specific use of vector clocks. The goal of this exercise is to play with this algorithm on a simple example, exhibit some various scenarios and discuss about the meaning of the vector clock values.

Question 1

Build up a scenario of an execution of the CBCAST algo studied in the course and used in the homework for that course, where you show

- at least two "hold" situations

- in which you do not deliver the 6 messages in the same order on the two processes.

Say another way, you have to complete this time diagram (see Figure 4) and show all vector clocks' values. You must also clearly mention when a message is immediately delivered, or, when it has to be hold once received before the process can deliver it.

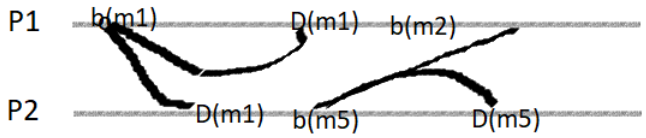

Figure 4: Time diagram showing the broadcast of some messages among the 2 processes : you have to continue this scenario

Question 2

According to the figure showing relations between the 6 messages (see Fig 3), is it possible that a process receives two different messages that are timestamped with a same vector clock value ?

Question 3

Generalize: why at the end, all the processes hold the same value in their Vector clocks? What do these vector clock values mean?

3 Uniform versus non-Uniform Consensus (4 pts)

This exercise is about to compare algorithms to solve the consensus problem. We have studied two algorithms in the course, and you had to study a third one as a homework.

Question 1

Recall what it means for a consensus algorithm to be classified as uniform versus non uniform.

Question 2

Discuss briefly the consensus problem in the specific case of deciding a transaction commit or abort in a distributed database. In this context, what is the consensus about, and what processes do once they learn the decision, and once they recover after a possible crash. In this context, is it allowed to have the algorithm that implements the consensus or commit protocol to feature non uniformity ?

Question 3

In the general case, the model of faults of your distributed system can be either crash-recovery or crash-fault. In these two different cases, what do you think about the need or not the need of uniformity in solving consensus.

Question 4

Given your understanding about how the 3 studied algorithms work, compare them by discussing the following aspects:
Q4a) Does the algorithm allow "crash event of process Pi" be received by still correct processes

after Pi has already decided a value ? Illustrate this by giving one example (and the associated lines of code) in each of the studied algorithms .
Q4b) Conclude: What is the key mechanism (or idea), for making sure a consensus algorithm features uniformity ?

4 Election algorithms (2 pts)

In the algorithm due to Franklin on a bi-directional ring, process identifiers are a priori not sorted in any way on the ring. In which case the algorithm features the smallest parallel execution time ? Explain. Give also the total number of messages that need to be exchanged (excluding the proclamation phase). And give the parallel time complexity of this best case situation.

5 Perfect links (3 pts)

In the course about group communication, the basic operation to send a message from one process to another has been studied. Three possible algorithms featuring different properties were considered: fair-loss link (flp2p), stubborn link, perfect point-to-point link (pp2p).

The goal of this exercise is to implement pp2p, but, not relying on the use of the stub-born link, and only on the use of fair-loss link algorithm. For sure, you are allowed to implement some of the ideas of a stub born link, however you must not try to just simulate it. On the contrary end up with an algorithm that is simply considering you use the fair loss link algorithm, and that your algorithm does what is needed so to ensure the properties of a pp2p primitive: PL1, PL2, PL3.

You can consider each message has a unique ID. Also, you can consider that there is a timeout value, known in advance, that is fine tuned to represent the time it takes normally (without loss) for a message to reach any process of your distributed system.

Once you have written your algorithm, discuss how PL1, PL2, PL3, the usual properties one can expect from a communication link algorithm, are ensured.