Master Program in Computer Science and Networking

High Performance Computing

2012-13

Homework 3

Submit the written answer. Deadline: lecture of October 8, or send an *e*-mail. The work has to be discussed at Question Time.

Question 1

a) The multicast communication (one-to-many communication in which a process P sends the same message to a known set of processes Q_0, .., Q_{n-1}) is not primitive in LC.

i. Emulate the multicast communication in LC: write a LC code and explain it.

ii. Derive the multicast cost model, i.e., evaluate the communication latency of the multicast implementation designed in *i). Optional:* try to improve the multicast implementation in order to minimize its latency.

b) A LC computation is composed of processes M_0, M_1 and P_0, P_1, P_2, P_3.

Every M_i (i = 1, 2) is an infinite cycle: during each iteration a value *Val* is computed by a given function *G*, then *Val* is multicasted to P_0, P_1, P_2, P_3.

Every P_j (j = 0 … 3) is an infinite cycle: during each iteration a given function *F* (the *same* for all P_j) is applied to the value received *non-deterministically* by M_0 or by M_1.

Write and explain the generic M_i and P_j with the following requirement: *the sequence of F results is exactly the same for P_0, P_1, P_2, P_3.*

No assumption can be made about the relative service times of M_0, M_1, as well as of and P_0, P_1, P_2, P_3. No assumption can be made on *F* and *G* functions.

Note: This question has to be studied with the following decreasing priorities:

- *a-i);*
- *a-ii)* without optional part;
- *b)* without LC code: only explain *what is the problem to be solved*, i.e. what is the reason for which, in absence of a proper solution, in general the sequence of *F* results is different for P_0, P_1, P_2, P_3;
- *b)*: solution to the problem, explained by words;
- *b)*: complete solution with LC code;
- *a-ii)* optional part.

Question 2

Consider the run-time support of a *send* primitive, according to any version chosen by the student (generic, peer, zero-copy, …): explain which data structures are referred to by *shared pointers*, and how these shared pointers are implemented.