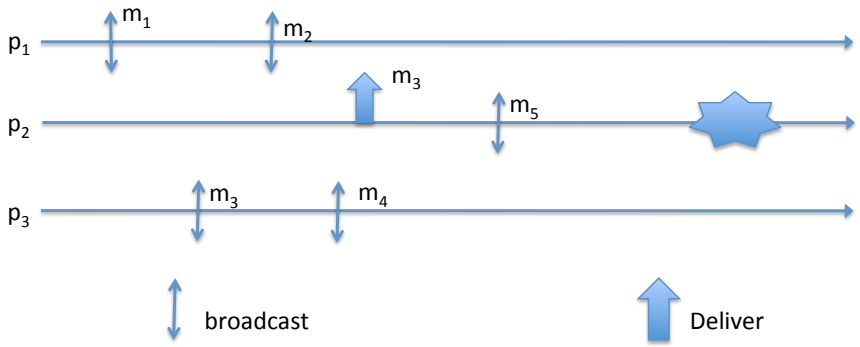
**Dependable Distributed Systems**
**Master of Science in Engineering in Computer Science**

**AA 2022/2023**

**Lecture 11 – Exercises**
**October 20th, 2022**

**Ex 1:** Consider the partial execution depicted in the following figure:




1. Complete the execution in order to obtain a run satisfying *Best Effort Broadcast* but *not Reliable Broadcast.*
2. Complete the execution in order to obtain a run satisfying *Regular Reliable Broadcast* but *not Uniform Reliable Broadcast.*
3. Complete the execution in order to obtain a run satisfying *Uniform Reliable Broadcast.*

**Ex 2:** Consider a distributed system composed by n processes $\{p_1,p_2,\ldots,p_n\}$. Each process is connected to all the others trough fair-loss point-to-point links and has access to a perfect failure detector.

Write the pseudo-code of an algorithm implementing a Uniform Reliable Broadcast primitive.

Additionally, answer to the following questions:

1. Is it possible to provide a quiescent implementation of the Uniform Reliable Broadcast primitive?
2. Given the system model described here, is it possible to provide an implementation that uses only data structure with finite size?

**Ex 3:** Consider a distributed system composed by N servers $\{s_1, s_2, \ldots s_n\}$ and M clients $\{c_1, c_2, \ldots c_m\}$.
Each client $c_i$ runs its algorithm and it can request to servers the execution of a particular task $T_i$. Servers will execute the task $T_i$ and, after that, a notification will be sent to $c_i$ that $T_i$ has been completed.
The Figure shows the code executed by a generic client $c_i$.

| Operation executeTask ($T_i$) | Upon pp2pdeliver (TASK_COMPLETED, $T_i$) from $s_j$ |
|---|---|
| 1. For each $s_i \in \{s_1, s_2, \ldots s_n\}$<br>2. pp2psend (TASK_REQ, $T_i$, $c_i$) to $s_i$; | 1. **trigger** completedTask ($T_i$); |

Write the pseudo-code of an algorithm, executed by servers, able to allocate tasks assuming that:

- Once clients ask for a task execution, they remain blocked until the task is not terminated.
- Any two clients $c_i$ and $c_j$ can concurrently require the execution of two different tasks $T_i$ and $T_j$;
- Each task is univocally identified by the pair ($T_i$, $c_i$);
- Each server can manage at most one task at every time;
- At most N-1 servers can crash;
- Servers can use a uniform consensus primitive;
- Servers can use a failure detector P;
- Servers communicate through a uniform reliable broadcast primitive.

Note that, if a server crashes while executing a task, such task needs to be re-allocated and re-processed by a different server.

**Ex 4:** Consider a distributed system formed by n processes $p_1$, $p_2$, …, $p_n$ connected along a ring i.e., a process $p_i$ is initially connected to a process $p_{(i+1) \bmod n}$ through a unidirectional perfect point-to-point link.
Write the pseudo-code of a distributed algorithm implementing a consensus primitive.