Concurrence, Parallelism and Distributed Systems (CPDS)
Module III: Distributed Systems
Facultat d'Informàtica de Barcelona
Final Exam
January 16th 2018

Answer the questions concisely and precisely
Answer in the same sheet
Closed-book exam
Duration: 2 hours

Name and Surname:

---

1. (5 pts) Select the answer (only one) that you consider to be correct on every section. Every correct answer computes 1/2 points. Every wrong answer deducts 1/6.

   (a) Which of the following characteristics of distributed systems is false?

   - Nodes are connected to the network, and for that reason, they are always available
   - Every node has an independent clock, and for that reason, they may be desynchronized
   - Nodes can fail independently, and for that reason, if some nodes fail, the others can continue working
   - Nodes communicate by sending messages through the network, which introduces some communication delay

   (b) Which of the following types of failure would occur in our Erlang programs if a process has an unexpected behavior as a result of the reception of a message that it does not recognize?

   - Crash failure
   - Omission failure
   - Value failure
   - State transition failure

   (c) In a distributed system, the maximum clock drift rate from the perfect time is 10^{-6}. i) What could be the maximum difference between the readings of any two clocks 24 hours after they have been synchronized? ii) What should be the maximum resynchronization interval, so that their clock skew does not exceed 3 ms?

   - i) 172.8 ms ii) 3000 s
   - i) 172.8 ms ii) 1500 s
   - i) 86.4 ms ii) 3000 s
   - i) 86.4 ms ii) 1500 s

   (d) Let a be the sending of a message and b be the reception of that message, which of the following statements is false?

   - A cut that includes both a and b is consistent
   - A cut that includes a but not b is consistent
   - A cut that includes b but not a is consistent
   - A cut that includes neither a nor b is consistent

(e) How does *basic reliable multicast* ensure that every message is delivered at most once?
- As it cannot detect duplicates, messages can be delivered several times
- Messages are discarded when its sequence number is lower than the next one the receiver is expecting
- Messages are discarded when its sequence number is greater than the next one the receiver is expecting
- As messages are sent only once, nothing is needed to avoid duplicates

(f) Which of the following statements is true when a group of processes elect a leader by means of the *Chang and Robert's algorithm* but false when they use the *enhanced ring algorithm*?
- Processes are organized in a logical unidirectional ring
- The process initiating the algorithm sends an `election` message containing its identifier to its successor
- At all times, the `election` message only includes the highest process identifier detected so far
- When a process identifies what process has the highest identifier, it sends a `coordinator` message to its successor

(g) Which of the following statements about the *Paxos consensus algorithm* is false?
- Paxos can tolerate less than a majority of acceptors failing
- Paxos can tolerate network partitions
- Paxos guarantees safety
- Paxos guarantees liveness

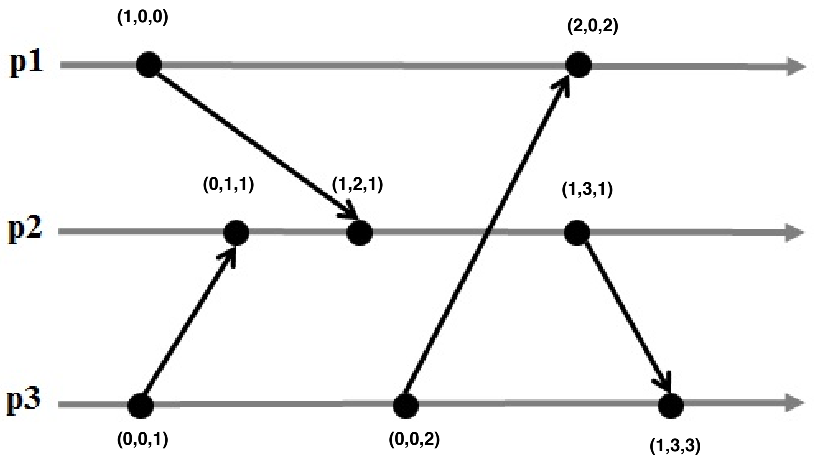
(h) The *Dolev and Strong algorithm* aims to solve consensus in a scenario with ...
- Faulty processes and reliable communication in an asynchronous system
- Crash-faulty processes and reliable communication in a synchronous system
- Byzantine-faulty processes and reliable communication in a synchronous system
- Non-faulty processes and unreliable communication

(i) Which of the following statements is true for both *strict two-phase locking* and *timestamp ordering*?
- A conflicting access to an object is detected as the object is accessed
- Serialization order of a transaction is decided when it starts
- A conflicting access to an object is solved aborting the transaction
- Provides better performance for transactions with predominantly read operations

(j) What is the main characteristic of *relaxed consistency models*?
- They provide a consistent view of the data for each client with a granularity of individual operations
- They provide a system-wide consistent view of the data with a granularity of individual operations on data stores with concurrent writes
- They provide a system-wide consistent view of the data with a granularity of individual operations on data stores without concurrent writes
- They provide a system-wide consistent view of the data with a granularity of groups of operations on data stores with concurrent writes

Name and Surname:

2. (1 pts) Given the following events performed by P1, P2, and P3:



a) Tag each event with its corresponding **vector** logical clock.

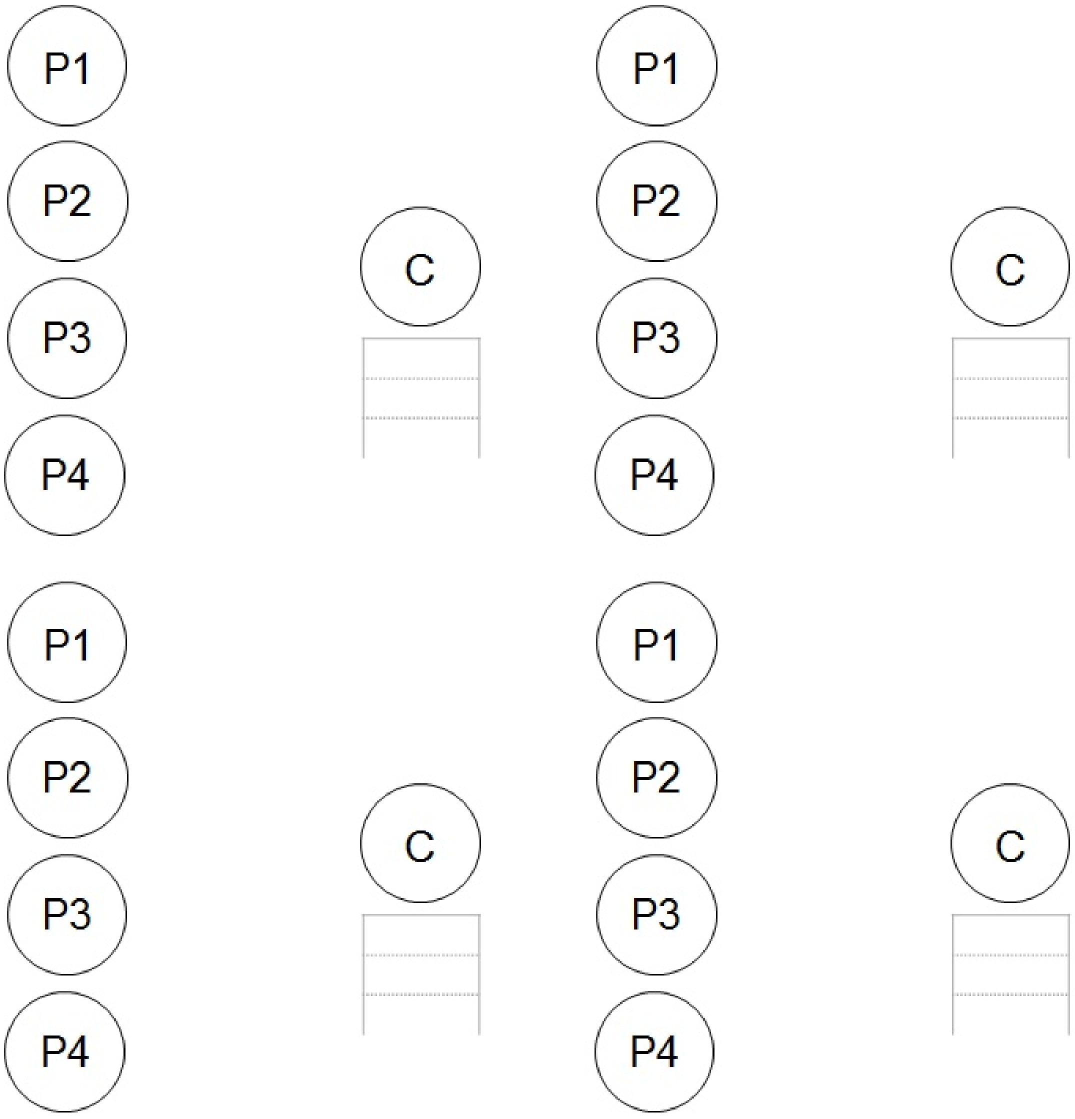
b) Draw the lattice with the reachability relation between the resulting consistent global states.

S_000

S_100 S_010 S_001

3. (1 pts) Given the following set of four processes that coordinate their accesses to a critical section by using the *centralized algorithm*, being process C the coordinator:

   a) If P1, P3, and P4 want to access the critical section concurrently, indicate in the following pictures the sequence of messages sent until the three processes have been granted access to the critical section. Network latency between P1 and C is 8 ms, between P3 and C is 2 ms, and between P4 and C is 5 ms.

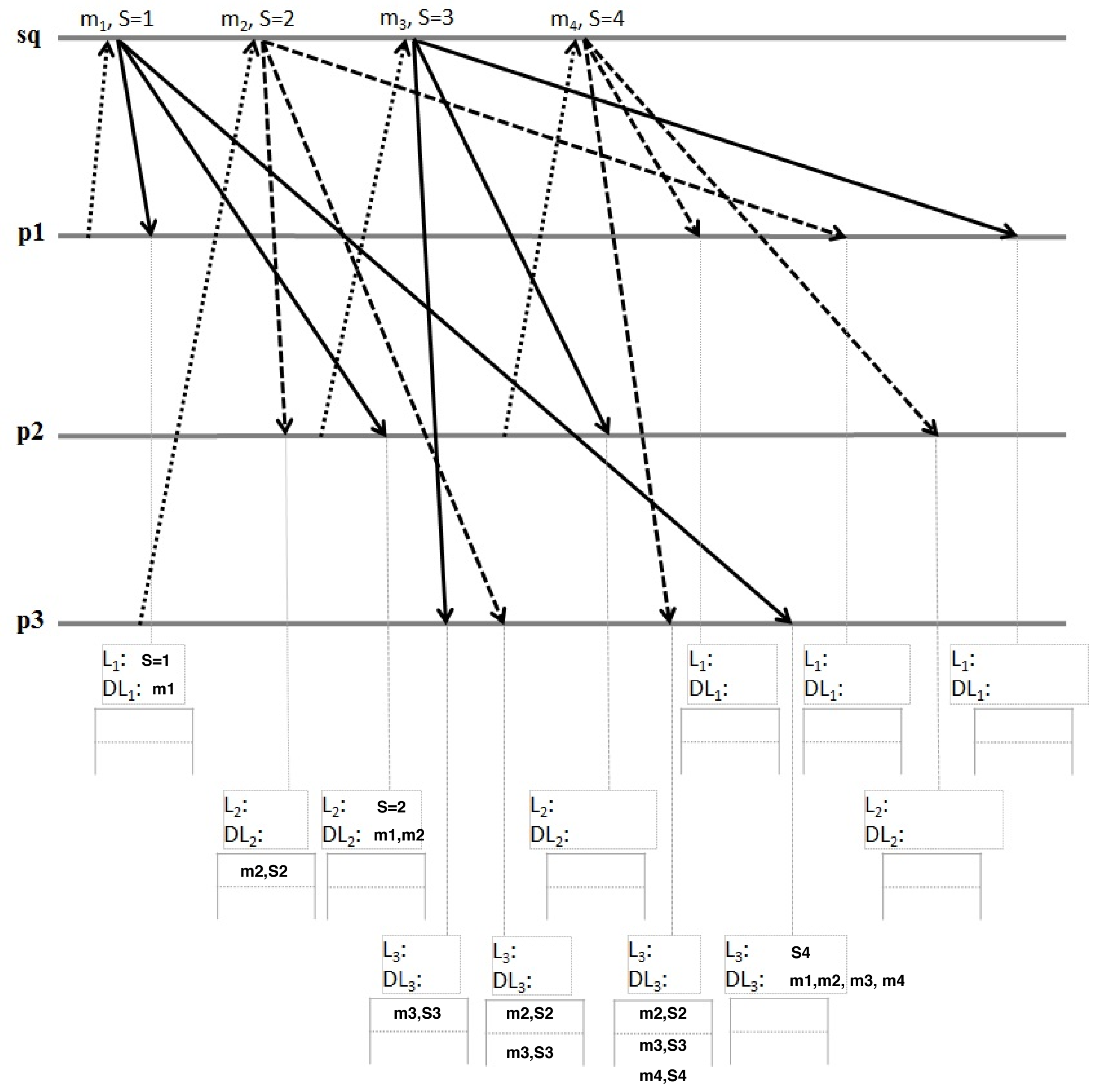


   b) If the system is synchronous and communication is reliable, and a process does not receive any response to a request to access the critical section before a given maximum time, can it assure that the coordinator got a crash failure? If not, explain how the algorithm could be modified to allow distinguishing this situation.

Name and Surname:

4. (1 pts) Given a group of processes that communicate through *totally-ordered multicast* implemented by means of a sequencer process (sq).

   a) Complete the following figure indicating for each process i: the sequence number of the last message it has delivered (L_i), the messages stored at the hold-back queue pending to be delivered, and the list of messages delivered up to now (DL_i).



   b) If the system supports *totally-ordered view-synchronous atomic multicast*, indicate what actions will be carried out to respond to a crash failure of the sequencer process.

5. (1 pts) We have two transactions T and U operating concurrently on the same set of objects in a distributed system implementing *optimistic concurrency control.* T and U are defined as follows:

   - T: `R(a)`; `R(d)`; `W(b)`
   - U: `W(a)`; `R(c)`; `W(b)`

   a) For each transaction X, indicate the objects on its read and write sets, the list of transactions that X validates against and the sets that are compared (i.e. read/write set of transaction X with read/write set of transaction Y), the outcome of this validation (i.e. success/failure), and what transaction is aborted (if any) in each of the following cases:

      i) T's request to commit comes first and backward validation is used
         - Transaction T:
         - Transaction U:

      ii) U's request to commit comes first and backward validation is used
         - Transaction U:
         - Transaction T:

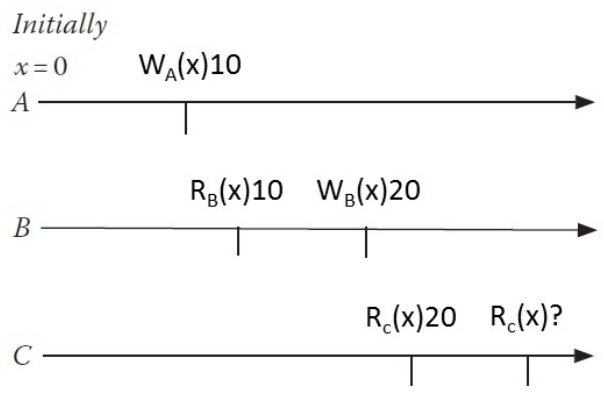
      iii) T's request to commit comes first and forward validation is used
         - Transaction T:
         - Transaction U:

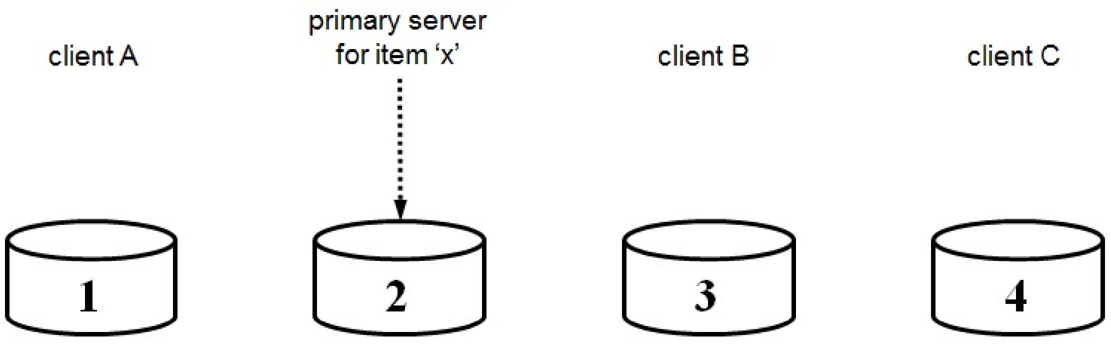
      iv) U's request to commit comes first and forward validation is used
         - Transaction U:
         - Transaction T:

   b) Explain briefly how the backward validation of those transactions would proceed if objects 'a' and 'b' are stored in server 'X', and objects 'c' and 'd' are stored in server 'Y'.

6. (1 pts) Given the following data store consisting of 4 replicas that uses a *primary-backup remote-write protocol* with blocking write:

   a) Indicate in the figure the sequence of actions (explaining them briefly below) that will take place if clients A, B, and C execute the operations as shown in the following picture, assuming that the data store provides only *sequential consistency* as consistency model.





   b) Is it possible that the operation $R_C(x)$? returns value 10? Justify your answer.

7. (SEMINARS) GEQ. **Paxy**.

a) Complete the following code excerpt corresponding to the *acceptor* process:

```
acceptor(Name, Promised, Voted, Value) ->
  receive
    {accept, Proposer, Round, Proposal} ->
        case order:goe(  ...  ,  ...  ) of
            true ->
                  ...  ! {vote,  ...  },
                case order:goe(  ...  ,  ...  ) of
                    true ->
                        acceptor(Name,  ...  ,  ...  ,  ...  );
                    false ->
                        acceptor(Name,  ...  ,  ...  ,  ...  )
                end;
            false ->
                  ...  ! {sorry, {accept,  ...  }},
                acceptor(Name, Promised, Voted, Value)
        end;
  end.
```

b) The following code excerpt intends to create the proposers in a remote Erlang instance (named `paxy-pro` and running in host `cpds.fib.upc.edu`) and connect them correctly to the acceptors, which are executing in a different Erlang instance (named `paxy-acc` and running in host `cpds.fib.upc.edu`), but it does not work. Make the needed adaptations to fix the error. Note that there are not syntax errors and the acceptors must use locally registered names.

```
start_proposers() ->
    PropInfo = [{p1, ?RED}, {p2, ?GREEN}, {p3, ?BLUE}],
    AccRegs = [a, b, c],
    lists:foreach(fun({PropName, PropCol}) ->
                  spawn('paxy-pro@cpds.fib.upc.edu',
                  fun() -> proposer:start(PropName, PropCol, AccRegs) end)
                  end, PropInfo).
```

c) Justify the rationale of the optimization based on `sorry` messages in the *proposer* process (how it works, why it works, and what benefit it provides).

Name and Surname:

8. (SEMINARS) IEQ. **Opty**.

   a) Complete the following code excerpt corresponding to the *handler* process:

```
handler(Client, Validator, Store, Reads, Writes) ->
    receive
        {read, Ref, N} ->
            case lists:keyfind(  ...  , 1,  ...  ) of
                {N, _, Value} ->
                      ...  ! {value, Ref, Value},
                    handler(Client, Validator, Store, Reads, Writes);
                false ->
                    Entry = store:lookup(  ...  , Store),
                      ...  ! {read, Ref, self()},
                    handler(Client, Validator, Store, Reads, Writes)
            end;
        {Ref, Entry, Value, Time} ->
            Client ! {value, Ref,  ...  },
            handler(Client, Validator, Store, [{Entry,  ...  }|  ...  ], Writes);
        {write, N, Value} ->
            Entry = store:lookup(N, Store),
            Added = lists:keystore(N, 1,  ...  , {N, Entry,  ...  }),
            handler(Client, Validator, Store,  ...  ,  ...  );
        {commit, Ref} ->
            Validator ! {validate, Ref,  ...  ,  ...  , Client}
    end.
```

   b) Reason what the impact on the percentage of committed transactions with respect to the total is when we increase the number of entries in the store.

   c) Given the following code excerpt corresponding to the *validator* process, explain briefly what changes would be needed if we change our concurrency control technique from backward to forward validation.

```
validator() ->
    receive
        {validate, Ref, Reads, Writes, Client} ->
            Tag = make_ref(),
            send_read_checks(Reads, Tag),
            case check_reads(length(Reads), Tag) of
                ok ->
                    update(Writes),
                    Client ! {Ref, ok};
                abort ->
                    Client ! {Ref, abort}
            end,
            validator()
    end.
```