Sistemes Distribuïts en Xarxa (SDX)
Facultat d'Informàtica de Barcelona
Prova parcial. 20 d'Abril 2016

Contesteu a les preguntes de manera concisa i precisa
Contesteu al mateix full
No es poden consultar apunts
(només els VOSTRES informes de les lectures)
Durada: 75 minuts

Nom i Cognoms:

---

1. (5 punts) Seleccioneu la resposta (una només) que considereu correcta en cadascun dels apartats. Cada resposta correcta val 1/2 punts. Cada resposta incorrecta resta 1/6.
   (a) Quina de les següents NO és una característica dels sistemes distribuïts?
      - Cada node té un rellotge independent, i per tant, poden estar desincronitzats
      - Les fallades afecten sempre tots els nodes del sistema distribuït, i per tant, deixen tot el sistema inutilitzable
      - Els nodes es comuniquen enviant missatges per la xarxa, i això introdueix un retard en la comunicació
      - Cada node té el seu espai d'adreces, i per tant, no comparteixen memòria

   (b) Quin dels següents tipus de fallada es produeix quan un motor de cerca retorna resultats que no estan relacionats amb cap dels termes de cerca que l'usuari ha especificat?
      - Response failure
      - Crash failure
      - Omission failure
      - Timing failure

   (c) En quin dels següents tipus de comunicació l'emissor es bloqueja fins que rep una confirmació de que el receptor ha rebut la seva petició?
      - Asynchronous communication
      - Submission-based synchronous communication
      - Delivery-based synchronous communication
      - Response-based synchronous communication

   (d) Com es pot aconseguir una semàntica *exactly-once* en l'execució d'una RPC?
      - No és possible garantir aquesta semàntica
      - Només cal que el client reintenti la RPC tantes vegades com calgui fins a rebre una resposta
      - Només cal que el client desisteixi de fer la RPC si no rep una resposta del servidor
      - Cal que el client reintenti la RPC tantes vegades com calgui fins a rebre una resposta i que el servidor recordi les RPCs que ha executat de manera que pugui refusar executar una RPC per segona vegada

(e) Quina de les següents operacions de l'API estàndard dels sistemes *publish-subscribe* no la invoca ni l'emissor ni el receptor sinó el middleware?

- subscribe
- unsubscribe
- notify
- publish

(f) Si volem sincronitzar un rellotge físic, quins serien els efectes col·laterals d'ajustar-lo cap enrere modificant-lo directament amb el seu nou valor?

- Cap, el valor dels rellotges físics es pot ajustar directament sense problema
- Es violaria la propietat de monotonicitat del temps, i events futurs podrien aparèixer com anteriors a events passats
- Es perdrien alguns instants temporals, cosa que pot ser un problema si hi havia events planificats durant aquest interval
- Els efectes col·laterals es podrien evitar incrementant la freqüència del rellotge físic fins a assolir l'ajustament desitjat

(g) Quina de les següents afirmacions referents a l'algorisme *Chang & Roberts* per l'elecció de líder és falsa?

- El nombre de missatges que necessita l'algorisme per finalitzar és el mateix independentment de la posició en l'anell del procés que inicia l'elecció
- Pot violar la propietat de *liveness* si el procés amb l'identificador més gran falla amb una elecció en curs
- El missatge d'elecció conté l'identificador del procés més gran detectat fins el moment en el recorregut per l'anell
- Pot haver-hi múltiples eleccions en curs de manera concurrent, però els missatges d'elecció redundants són eliminats

(h) Quina de les següents afirmacions és certa quan un grup de processos es comuniquen mitjançant multicast fiable bàsic (*basic reliable multicast*) però falsa quan usen multicast fiable escalable (*scalable reliable multicast*)?

- L'emissor assigna un número de seqüència a cada missatge enviat i cada procés guarda el número de seqüència del darrer missatge que ha lliurat procedent de cadascun dels altres processos
- Si un procés rep un missatge amb un número de seqüència igual que el següent que està esperant, llavors lliura el missatge i confirma la recepció a l'emissor
- Si un procés rep un missatge amb un número de seqüència més gran que el següent que està esperant, llavors guarda el missatge en una cua (*hold-back queue*) i es demana a l'emissor la retransmissió dels missatges que falten
- Si un procés rep un missatge que ja ha lliurat amb anterioritat, llavors descarta el missatge

(i) Quin model de consistència garanteix que tots els processos veuen la mateixa seqüència d'operacions, que no ha de respectar necessàriament el seu ordre temporal global però sí l'ordre local de cada procés?

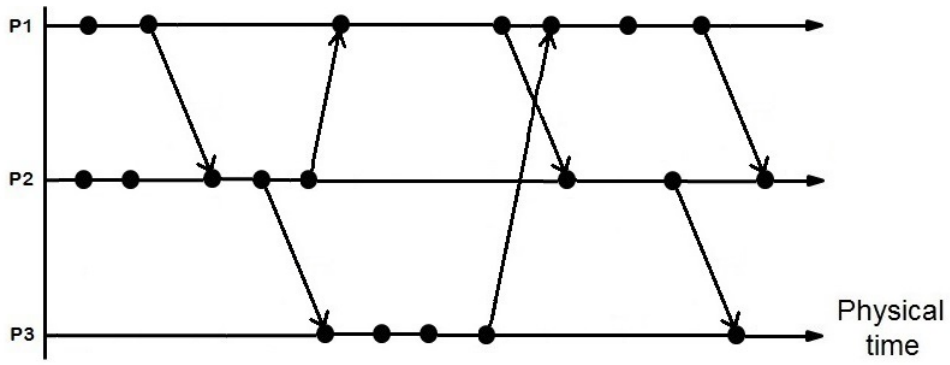
- Causal consistency
- FIFO consistency
- Linearizability
- Sequential consistency

(j) Quin dels següents protocols estableix la següent seqüencia de missatges per propagar les modificacions en un *data store* replicat? i) el servidor notifica al client que les dades han estat modificades, ii) el client demana al servidor les dades actualitzades, iii) el servidor envia les dades actualitzades al client.

- push-based protocol amb invalidació
- push-based protocol amb actualització
- pull-based protocol
- lease-based protocol amb el *lease* caducat

Nom i Cognoms:

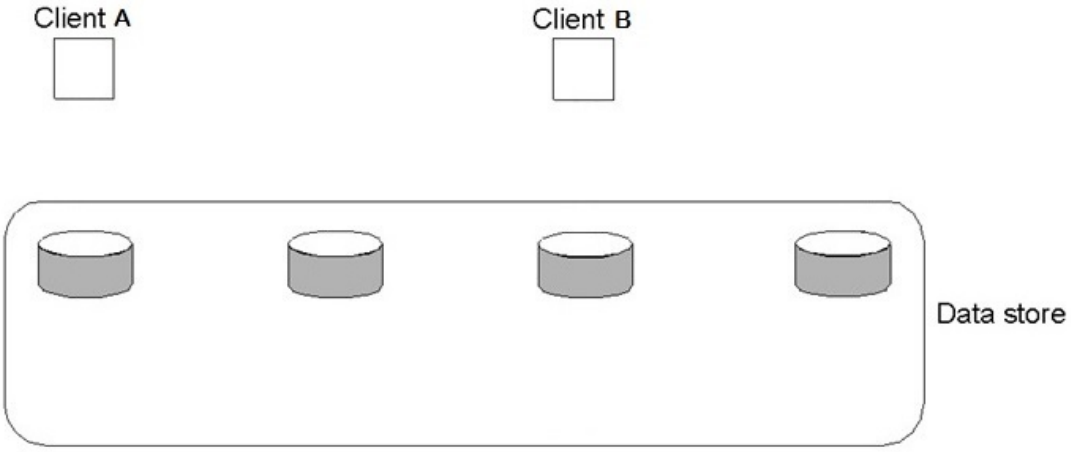
2. (1,25 punts) Donada la següent seqüència d'events executada pels processos P1, P2 i P3:



   a) Etiqueta cada event amb el valor del seu rellotge lògic escalar (i.e. rellotge de Lamport).
   b) Etiqueta cada event amb el valor del seu rellotge lògic vectorial.

3. (1,25 punts) Donat el següent *data store* de 4 rèpliques que utilitza un protocol d'escriptura replicada (*replicated-write protocol*) basat en replicació activa (*active replication*):

   a) Indica en la figura la seqüència d'accions (i enumera a sota cadascuna d'elles) que es duran a terme si s'executen de manera concurrent una escriptura de la dada 'x' per part del 'Client A', i una lectura de la mateixa dada per part del 'Client B', tenint en compte que el *data store* proporciona només consistència seqüencial.

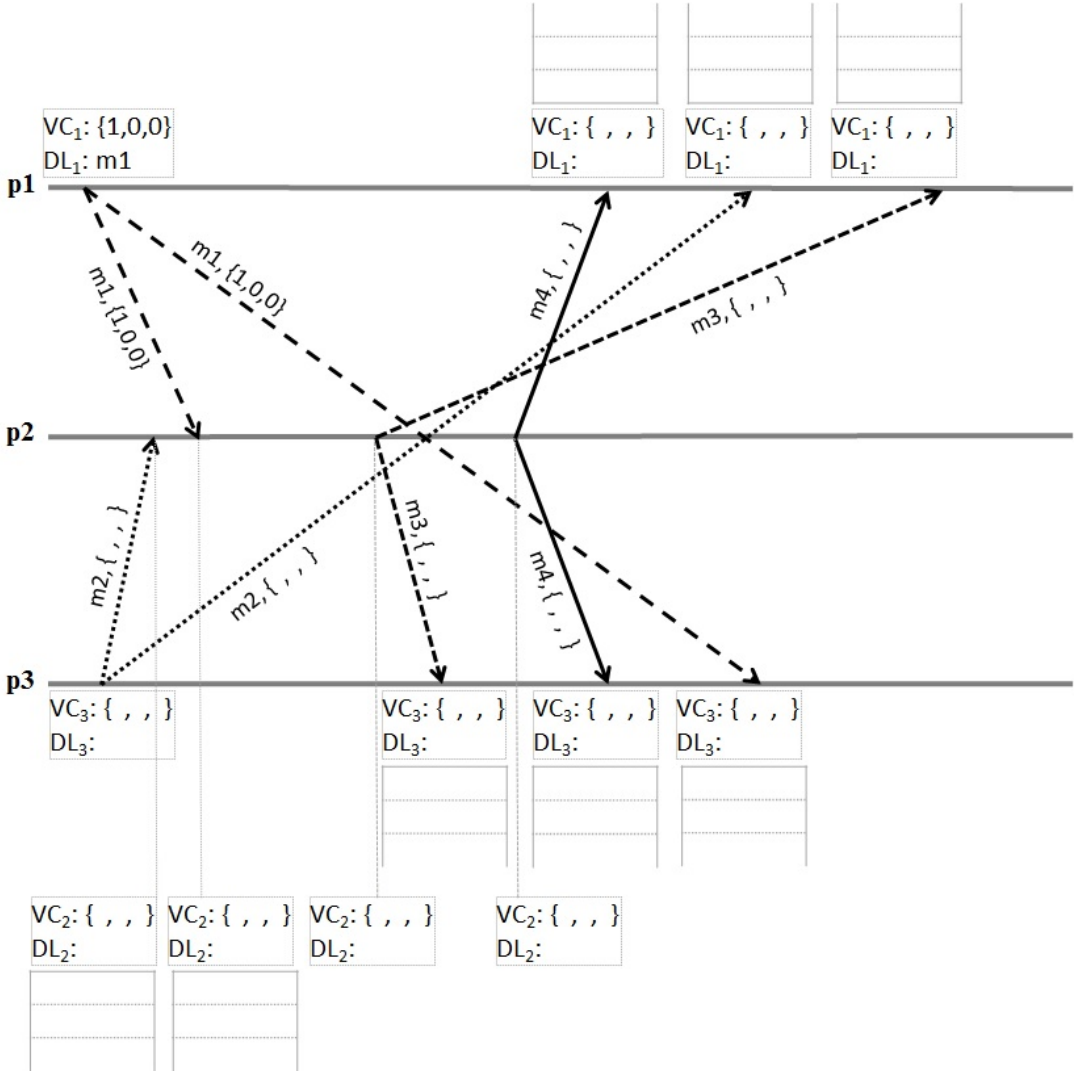


   b) Pot suportar aquest *data store* fallades Bizantines durant l'execució d'operacions de lectura? En cas negatiu, quina modificació caldria fer per suportar-les? Amb la modificació, quantes fallades d'aquest tipus es podrien suportar tenint en compte la mida del *data store*?

4. (1,25 punts) Donat un grup de processos que es comuniquen mitjançant multicast fiable casualment ordenat (*causally-ordered reliable multicast*) basat en vector clocks.

   a) Indica quines dues condicions ha de complir el vector clock d'un missatge m (VC_m) enviat per un procés P_i per què un procés P_j el pugui lliurar.

   b) Completa la figura indicant per cada procés el seu vector clock (VC_i), els missatges guardats a la *hold-back queue* pendents de ser lliurats, la llista de missatges lliurats fins el moment (DL_i), a més del vector clock dels missatges enviats.

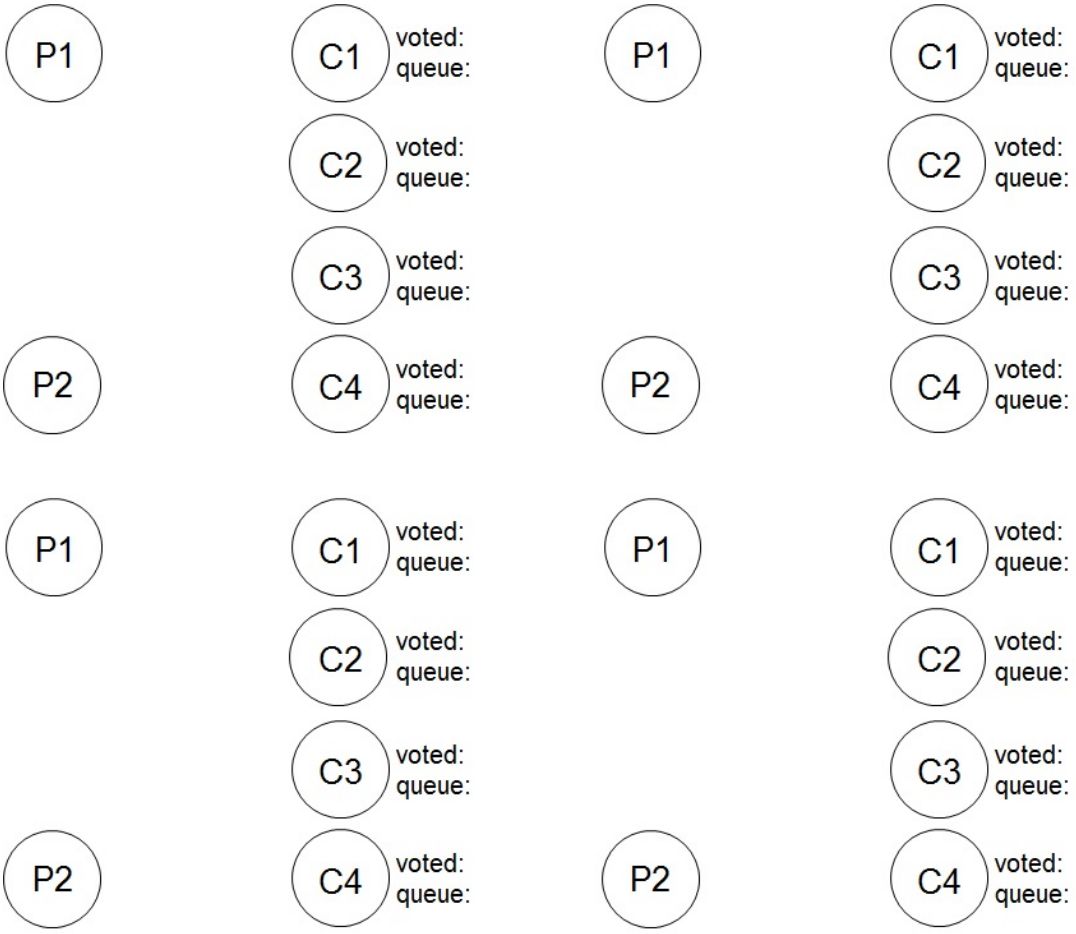


   c) Justifica si la seqüència de missatges lliurats per cada procés es conforme a les ordenacions FIFO i total.

Nom i Cognoms:

5. (1,25 punts) Els processos P1 i P2 coordinen els seus accessos a una regió crítica mitjançant l'*algorisme de votació de Lin*, que està configurat amb quatre processos coordinadors (C1 a C4) i amb missatges *Yield*.

   a) Si tant P1 com P2 volen accedir la regió crítica de manera concurrent i el temps lògic de P1 és 5 i el temps lògic de P2 és 8, indica en els següents quatre gràfics la seqüència de missatges enviats i l'estat de cada coordinador (a quin procés ha votat i els processos a la seva cua d'espera) fins que un procés aconsegueixi l'accés a la regió crítica, assumint que inicialment C1 i C3 voten per P2, i C2 i C4 voten per P1.



   b) Justica si l'*algorisme de votació de Lin* proporciona ordre *happened-before* (i.e. si una petició es produeix abans que una altra, llavors l'accés a la regió crítica es produeix en aquest ordre).

6. (SEMINARIS) GEQ: **Muty**.

   a) Completa el següent extracte de codi corresponent a la implementació del mecanisme d'exclusió mútua de Ricart & Agrawala utilitzant prioritats (i.e. lock2).

```
% MyId: Priority of this process
% ReqId: Priority of the process sending request message
wait(MyId, Nodes, Master, Refs, Waiting, TakeRef) ->
  receive
    {request, From, Ref, ReqId} ->
      if
         ...  <  ...  ->
            From ! {ok, Ref},
            R = make_ref(),
            From !   ...
            wait(MyId, Nodes, Master,   ...   , Waiting, TakeRef);
        true ->
            wait(MyId, Nodes, Master, Refs,   ...   , TakeRef)
      end
  end.
```

   b) Completa el següent extracte de codi corresponent a la implementació del mecanisme d'exclusió mútua de Ricart & Agrawala utilitzant rellotges lògics de Lamport (i.e. *lock3*).

```
% MyClock: Logical clock of this process
open(MyId, Nodes, MyClock) ->
  receive
    {take, Master, Ref} ->
      MyNewClock =    ...
      Refs = lists:map(fun(P) ->
        R = make_ref(),
        P ! {request, self(), R, MyId,  ...  },
        R
      end, Nodes),
      wait(MyId, Nodes, Master, Refs, [],   ...   ,   ...   , Ref);
    {request, From,  Ref, _, ReqTime} ->
      MyNewClock =   ...
      From ! {ok, Ref},
      open(MyId, Nodes,   ...   )
  end.
```

   c) Quin és el principal inconvenient del *lock1*?

Nom i Cognoms:

7. (SEMINARIS) IEQ.

   a) **Ordy**. Completa el següent extracte de codi corresponent a la implementació de multicast totalment ordenat.

```
server(Master, MaxPrp, MaxAgr, Nodes, Cast, Queue, Jitter) ->
  receive
    {send, Msg} ->
        Ref = make_ref(),
        Self = self(),
        lists:foreach(fun(Node) ->
                        Node ! {request,  ...  ,  ...  ,  ...  }
                    end, Nodes),
        NewCast = cast(Ref, Nodes, Cast),
        server(Master,  ...  , MaxAgr, Nodes,  ...  , Queue, Jitter);
    {request, From, Ref, Msg} ->
        NewMaxPrp = seq:  ...  (seq:maxfirst(  ...  ,  ...  )),
        From ! {proposal, Ref,  ...  },
        NewQueue = insert(Ref, Msg, NewMaxPrp, Queue),
        server(Master,  ...  ,  ...  , Nodes, Cast,  ...  , Jitter);
    ⋮
```

   b) **Ordy**. Assumint que tenim N processos worker, quants missatges es generen en la implementació de multicast totalment ordenat des de que un worker envia un missatge multicast fins que tots els workers l'han lliurat?

   c) **Chatty**. És possible que un client rebi una resposta a un missatge per part d'un altre client abans de rebre el missatge original enviat per un tercer client? Justifica la resposta en funció de si tots els clients estan connectats a un únic servidor o cadascun a un servidor diferent.

8. (READINGS) Contesta les següents preguntes referents als articles llegits a l'assignatura.

   i) Quina de les assumpcions errònies que els arquitectes de sistemes distribuïts tendeixen a fer segons l'article *Rotem06* fa referència a la necessitat d'interoperabilitat entre dispositius de diferents tipus?

   ii) Quins són els quatre timestamps que utilitza el protocol NTP per calcular el *round-trip delay* i l'*offset* segons l'article *Corcoran13*?

   iii) Explica en què consisteix el model de consistència *Bounded Staleness* tal com es descriu a l'article *Terry13*.