Sistemes Distribuïts en Xarxa (SDX)
Facultat d'Informàtica de Barcelona
Examen Final (2ª part). 22 de Juny 2017

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: 85 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) La cadena *CN=Jordi Guitart,OU=Arquitectura de Computadors,O=UPC,C=Spain*
   - És un *flat name*
   - És un *structured name*
   - És un *attribute-based name*
   - No és un nom

   (b) Quina semàntica proporciona NFS a l'hora de compartir fitxers si NO permetem als clients fer caching?
   - UNIX semantics
   - Immutable files
   - Session semantics
   - Transactional semantics

   (c) Quina de les següents característiques del model *upload/download* per sistemes de fitxers distribuïts és falsa?
   - La major part de la feina es fa a la banda del client
   - La consistència de les dades és més difícil de mantenir
   - Té un alt cost de comunicació per obrir fitxers, però baix per operar en ells
   - Té poca tolerància a fallades del servidor i a particions de xarxa

   (d) Quina de les següents alternatives té un rol anàleg en els Web Services al que té el IDL en les RPCs?
   - UDDI
   - WSDL
   - SOAP
   - HTML

   (e) Quina de les següents afirmacions referents al sistema *Google Docs* és falsa?
   - Cada document es guarda en el servidor com una llista cronològica dels canvis realitzats
   - Quan un usuari fa un canvi, l'envia al servidor, i mentre no en rep confirmació, guarda els nous canvis en una llista de canvis pendents de ser enviats al servidor
   - Quan el servidor rep un canvi, actualitza el document, confirma el canvi a l'emissor i notifica el canvi a la resta d'usuaris del document
   - Quan un usuari rep un canvi del servidor, l'aplica directament sobre la seva versió local del document

(f) En què consisteix la tècnica de *Random walk* per controlar el *flooding* a Gnutella?

- Limitar a un el nombre de nodes veïns als quals es reenvia cada missatge de Query
- Anar incrementant el nombre màxim de vegades que un missatge Query pot ser reenviat fins que es troba el contingut cercat
- Limitar el nombre màxim de vegades que un missatge Query pot ser reenviat
- Guardar els missatges Query enviats recentment per tal d'evitar reenviar el mateix missatge a un mateix node

(g) Quina de les següents afirmacions referents al sistema P2P *Kademlia* és falsa?

- Si usem claus de 160 bits, cada node guardarà 160 k-buckets
- Els k-buckets no s'actualitzen quan es reben missatges d'altres nodes, només quan el node fa un *refresh* explícit
- Els k-buckets es mantenen ordenats, de manera que el node que s'ha vist més recentment es troba al final
- Cada k-bucket i del node P conté fins a k nodes que es troben a distància $[2^i, 2^{i+1})$ de P

(h) Quina de les següents afirmacions és certa per sistemes de computació Grid però falsa per sistemes de computació voluntària?

- Agreguen recursos heterogenis
- Agreguen recursos geogràficament distribuïts
- Agreguen recursos pertanyents a diferents organitzacions/usuaris
- Agreguen recursos fortament acoblats per permetre la col·laboració entre institucions

(i) Quin tipus de virtualització ofereixen els contenidors *Docker*?

- Virtualització completa de tipus I (*bare-metal*)
- Virtualització completa de tipus II (*hosted*)
- Virtualització de sistema operatiu
- No ofereixen cap virtualització

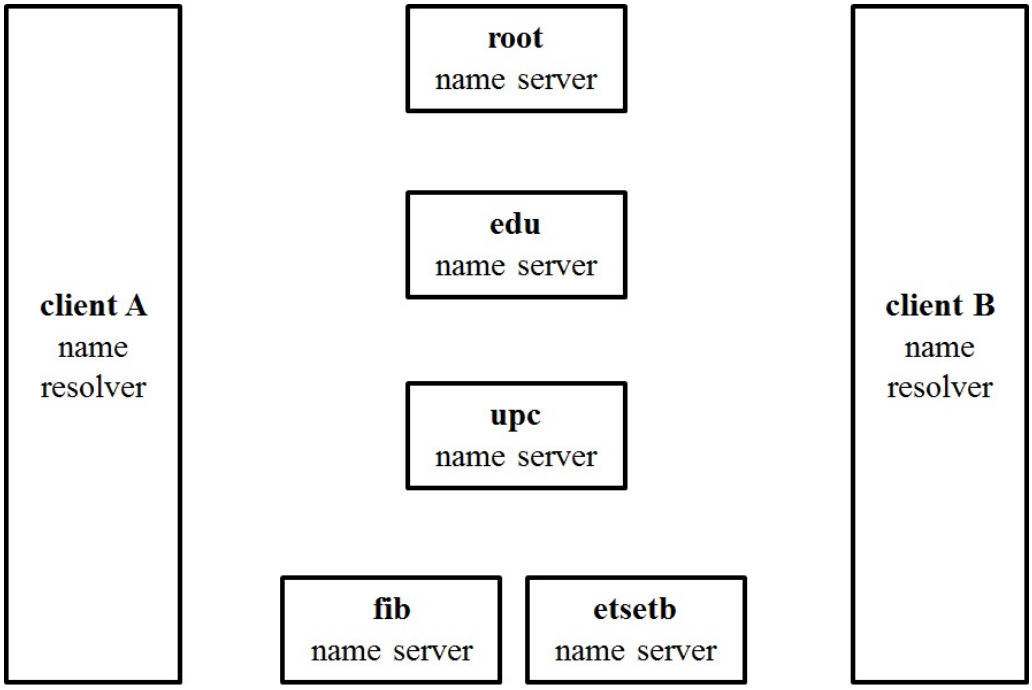
(j) Quina és la funcionalitat de les *wireless sensor networks* que permet l'encaminament de missatges en entorns amb connectivitat discontínua entre nodes que impedeixen la definició de rutes extrem-a-extrem estables?

- Disruption-tolerant networking
- Directed diffusion
- In-network processing
- Multihop communication

Nom i Cognoms:

2. (1,25 punts) Donat el següent servei de noms que utilitza resolució **iterativa** i on els resolvers utilitzen caching **cooperatiu** entre ells, indica la seqüència d'accions de cada resolver (incloent accessos a la cache i missatges que s'intercanvien amb els diferents servidors de noms) per resoldre seqüencialment els següents dominis. Indica el contingut de cada missatge (quina part del domini es demana resoldre i quins resultats s'obtenen), el resultat dels accessos a la cache (HIT or MISS), i les modificacions sobre el contingut de la cache (nota: els elements de la cache no expiren mai).

   - client A: *[www,fib,upc,edu]*
   - client A: *[ftp,fib,upc,edu]*
   - client B: *[www,etsetb,upc,edu]*
   - client B: *[ftp,fib,upc,edu]*



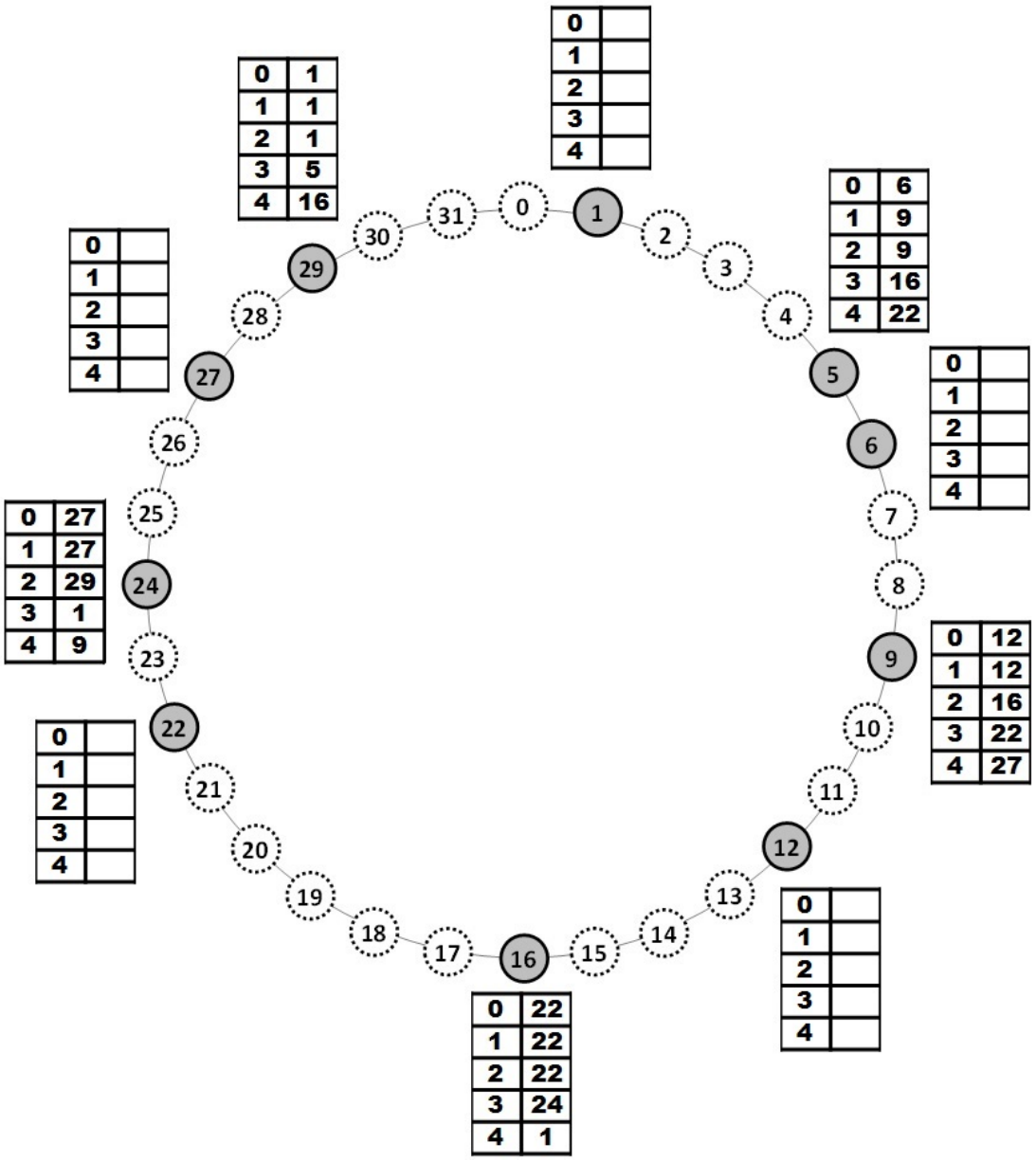
3. (1,5 punts) Donat el següent sistema P2P *Chord*, on els nodes ombrejats són aquells que formen part del sistema en aquest moment:

   a) Completa les finger tables dels nodes 1, 6, 12, 22 i 27.

   b) Indica a quins nodes es guardarien les keys 5, 14, 18, 25 i 31.

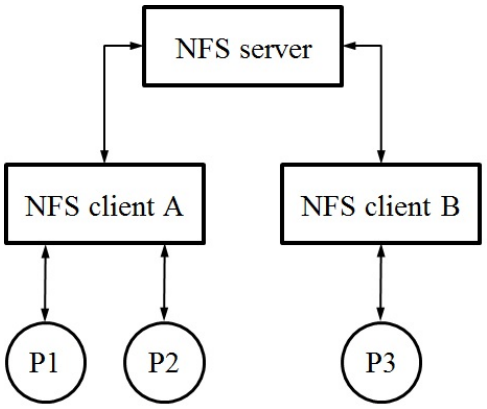
   c) Indica la interacció de missatges quan i) el node 5 fa un lookup de la key 25, ii) el node 29 fa un lookup de la key 14.

   d) Si el node 19 fa join al sistema, indica el contingut de la seva finger table, les modificacions que s'haurien de fer a les finger tables dels altres nodes, i si caldria moure alguna de les keys de l'apartat b).



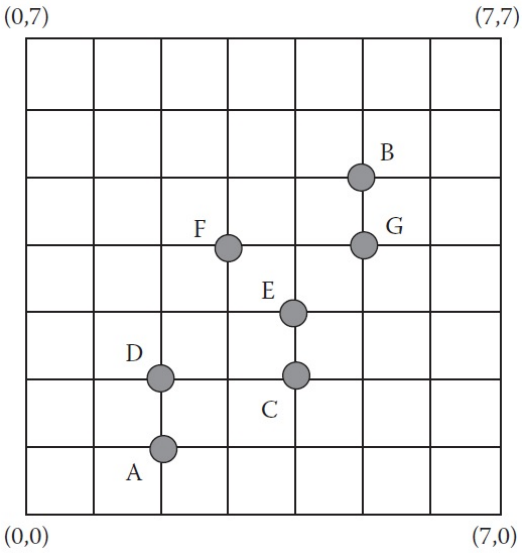
Nom i Cognoms:

4. (1 punts) Tres processos organitzats tal com es mostra a la següent figura executen les següents operacions *open* en un sistema de fitxers distribuït NFS amb *share reservations* i *open delegations*. Els paràmetres de la funció *open* indiquen el nom del fitxer, l'accés demanat, i el tipus d'accés que s'ha de denegar als altres. Indica per cada operació *open* dels processos P2 i P3: el seu resultat (justificant els casos d'error), i si la operació s'ha de demanar al servidor o es pot resoldre localment al client (nota: el temps s'incrementa cap abaix).



P1	P2	P3
`open('fileA', RDWR, READ)`		
`Client A is awarded a write delegation for 'fileA'`		
	`open('fileA', READ, NONE)`	
	`open('fileA', WRITE, WRITE)`	
		`open('fileA', WRITE, NONE)`
`open('fileB', READ, NONE)`		
`Client A is awarded a read delegation for 'fileB'`		
	`open('fileB', READ, NONE)`	
		`open('fileB', READ, NONE)`
	`open('fileB', WRITE, NONE)`	

5. (1,25 punts) Donada una *wireless sensor network* on els nodes estan situats tal com es mostra a la figura següent, si l'energia necessària per enviar dades a una distància d és $E_d = d^3$, determina el camí òptim per enviar dades des del node A al node B de manera que l'energia total necessària sigui mínima (pista: et pot ajudar omplir la taula següent per calcular la distància (i la energia) entre dos punts P i Q qualssevol, i després fer servir aquestes dades per avaluar combinacions de punts que permetin fer la ruta entre A i B).



P	Q	$\sqrt{(Q_x - P_x)^2 + (Q_y - P_y)^2}$	d	E_d
A	B	$\sqrt{(5-2)^2 + (5-1)^2}$	5	125
A	C			
A	D			
A	E			
A	F			
A	G			
B	C			
B	D			
B	E			
B	F			
B	G			
C	D			
C	E			
C	F			
C	G			
D	E			
D	F			
D	G			
E	F			
E	G			
F	G			

Nom i Cognoms:

6. (SEMINARIS) GEQ. **Groupy**.

   a) Completa el següent extracte de codi corresponent al procés *leader* en la implementació *gms3*.

```
leader(Name, Master, N, Slaves) ->
  receive
    {mcast, Msg} ->
      lists:foreach(fun(Node) ->
                    Node ! {msg,    ...   ,    ...   }
                    end, Slaves),
         ...    ! {deliver,    ...   },
      leader(Name, Master,    ...   ,    ...   );
    {join, Peer} ->
      NewSlaves = lists:append(   ...   , [Peer]),
      lists:foreach(fun(Node) ->
                    Node ! {   ...   ,    ...   , self(),    ...   }
                    end, NewSlaves),
      leader(Name, Master,    ...   ,    ...   );
  end.
```

   b) Adapta el següent extracte de codi de manera que cada worker es creï i s'executi en instàncies Erlang remotes diferents. Concretament, el worker P1 s'ha d'executar a la instància `n1@127.0.0.1`, el worker P2 a la instància `n2@127.0.0.1`, i el worker P3 a la instància `n3@127.0.0.1`.

```
start(Module, Sleep) ->
  register(w1, worker:start("P1", Module, Sleep)),
  register(w2, worker:start("P2", Module, w1, Sleep)),
  register(w3, worker:start("P3", Module, w2, Sleep)).
```

   c) Justifica per què els workers es desincronitzen en la implementació *gms2* quan fem que el procés emissor pugui fer crash amb una certa probabilitat durant l'enviament del multicast.

7. (SEMINARIS) IEQ.

a) **Chordy**. Completa el següent extracte de codi corresponent a l'operació d'inserció en el store en la implementació del *node4* (la versió que suporta replicació).

```
% APIs: key:between(Key, From, To)
%       storage:add(Key, Value, Store)
node(MyKey, Predecessor, Successor, Next, Store, Replica) ->
  receive
    {add, Key, Value, Qref, Client} ->
      Added = add(Key, Value, Qref, Client, MyKey, Predecessor, Successor, Store),
      node(MyKey, Predecessor, Successor, Next,    ...   ,    ...   );
    {replicate, Key, Value} ->
      Added = storage:add(Key, Value,    ...   ),
      node(MyKey, Predecessor, Successor, Next,    ...   ,    ...   )
  end.

add(Key, Value, Qref, Client, MyKey, {Pkey, _, _}, {_, _, Spid}, Store) ->
  case key:between(Key,    ...   ,    ...   ) of
    true ->
      Added = storage:add(Key,    ...   ,    ...   ),
         ...    ! {   ...   ,    ...   , Value},
      Client ! {Qref, ok},
      Added;
    false ->
         ...    ! {   ...   , Key, Value, Qref, Client},
      Store
  end.
```

b) **Namy**: Donat el següent codi que implementa els servidors de noms a Namy, modifica'l com calgui per tal que quan el servidor sigui aturat, s'esborri la seva entrada en el servidor pare.

```
init() ->
    server([], 0).

init(Domain, Parent) ->
    Parent ! {register, Domain, {domain, self()}},
    server([], 0).

server(Entries, TTL) ->
    receive
        {request, From, Req}->
            Reply = entry:lookup(Req, Entries),
            From ! {reply, Reply, TTL},
            server(Entries, TTL);
        {register, Name, Entry} ->
            Updated = entry:add(Name, Entry, Entries),
            server(Updated, TTL);
        {deregister, Name} ->
            Updated = entry:remove(Name, Entries),
            server(Updated, TTL);
        stop ->
            ok
    end.
```

Nom i Cognoms:

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

   i) Explica el concepte de IXFR (*Incremental Zone Transfer*) tal com es descriu a l'article *Vixie07*.

   ii) Explica en què consisteix la política de *Endgame Mode* que usa BitTorrent tal com es descriu l'article *Cohen03*.

   iii) Resumeix el problema de *Bugs in Large-Scale Distributed Systems* tal com es descriu a l'article *Armbrust10* i indica quina solució es proposa.