TSR - Segon Parcial. 2025-01-13

Aquest examen consta de 22 qüestions, amb una puntuació total de 10 punts. Cada qüestió té 4 alternatives, de les quals únicament una és certa. La nota es calcula de la següent manera: després de descartar les dues pitjors qüestions, cada encert suma 0.5 punts, i cada error descompta 1/6 punts. Has de contestar en el full de respostes.

A

1. ***Considereu el programa JavaScript següent:***

```
const z = require("zeromq")
const r = z.socket("router")
r.bindSync("tcp://127.0.0.1:8999")
r.on("message", (x, y, z) => {
  console.log(x+" "+y+" "+z)
  r.send([x, parseInt(y+"")*parseInt(z+"")])
})
```

Indiqueu quina afirmació descriu el comportament d'aquest programa.

A. Si la cinquena línia mostra la cadena `C1 4 5` en rebre un missatge, aleshores qui va emetre aquest missatge podria haver utilitzat un `s.send([4,5])` per a enviar-ho.

B. Si la cinquena línia mostra la cadena `C1 4 5` en rebre un missatge, aleshores qui va emetre aquest missatge podria haver utilitzat un `s.send(["C1", 4,5])` per a enviar-ho.

C. Si la cinquena línia mostra la cadena `C1 4 5` en rebre un missatge, aleshores qui va emetre aquest missatge va poder haver utilitzat un `s.send("C1 4 5")` per a enviar-ho.

D. Cap de les altres afirmacions és correcta.

2. ***Considereu el següent programa JavaScript, utilitzat per executar un procés servidor `S` amb el qual interactuarà algun procés client `C`:***

```
const z = require("zeromq")
const r = z.socket("router")
r.bindSync("tcp://127.0.0.1:8999")
r.on("message", (x, y, z) => {
  console.log(x+" "+y+" "+z)
  r.send([x, parseInt(y+"")*parseInt(z+"")])
})
```

Seleccioneu l'afirmació correcta:

A. `C` ha pogut utilitzar dos sockets (p.ex., de tipus PUSH i PULL) per a enviar les peticions i rebre les respostes de `S`.

B. No hi pot haver més d'un procés client `C` connectat amb `S`, ja que `S` només ha definit un listener per a l'esdeveniment `message`.

C. El paràmetre `x` definit a la quarta línia d'aquest programa rebrà la identitat del socket utilitzat a `C` per connectar-se amb `r`.

D. `C` ha pogut utilitzar un socket de tipus PUSH per rebre les respostes de `S`.

3. ***En enviar missatges mitjançant sockets de tipus DEALER...:***

A. Totes les altres afirmacions són falses.

B. Se segueix una política d'enviament de torn circular

C. Es pot triar, mitjançant el primer segment del missatge, a quin altre socket connectat amb el DEALER s'està enviant el missatge.

D. El socket sempre afegeix automàticament un primer segment addicional amb la identitat del socket emissor.

4. ***Té sentit que en un Dockerfile no hi haja cap línia `CMD` ni `ENTRYPOINT`?***

A. No, perquè en aquest cas l'ordre `docker build` produeix un error i no construeix cap imatge

B. Sí, perquè la instrucció `WORKDIR` té un objectiu similar i les pot substituir

C. Sí, ja que la imatge a generar podria utilitzar-se com a base per a altres Dockerfile que sí utilitzaran `CMD` o `ENTRYPOINT`

D. No, perquè si aquestes instruccions no hi estan, les instruccions `RUN` no tenen cap efecte

(Les preguntes 5 a 11 es refereixen a aquesta mateixa descripció) S'ha dissenyat un sistema amb tres

components (A,B,C) el codi dels quals es mostra seguidament i que es connecten segons indica la figura. Per a no perdre cap missatge publicat per A, s'ha introduit un retard abans de l'enviament: suposarem que sempre arranquen totes les instàncies dins d'eixe interval.

A.js

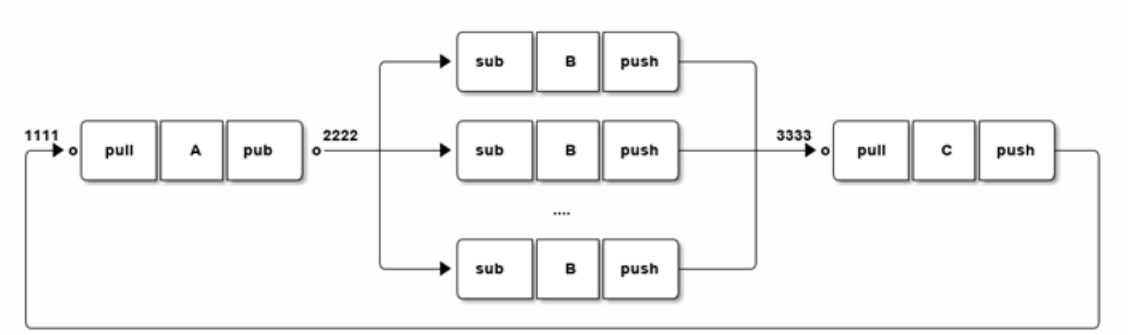
```
const zmq = require('zeromq')
let sin = zmq.socket('pull')
let sout = zmq.socket('pub')
sin.bind('tcp://*:1111')
sout.bind('tcp://*:2222')
sin.on('message', msg =>
  console.log('Recibido '+msg)
)
setTimeout(() => {
  console.log('empezamos')
  sout.send('hola')
}, 5000)
```

B.js

```
const zmq = require('zeromq')
let sin = zmq.socket('sub')
let sout = zmq.socket('push')
sin.connect(process.argv[2])
sout.connect(process.argv[3])
sin.subscribe('')

sin.on('message', msg => {
  console.log('recibido '+msg)
  sout.send(msg)
})
```

C.js

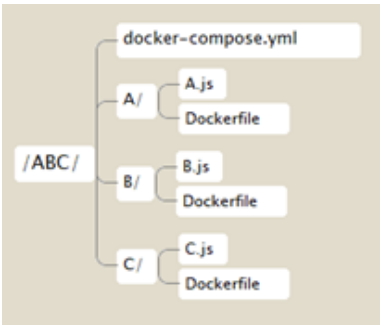
```
const zmq = require('zeromq')
let sin = zmq.socket('pull')
let sout = zmq.socket('push')
sin.bind('tcp://*:3333')
sout.connect(process.argv[2])

sin.on('message', msg => {
  console.log('recibido '+msg)
  sout.send(msg)
})
```



Per a fer les proves es llancen múltiples terminals per a les diferents instàncies. Per example:

```
node A.js
node B.js tcp://localhost:2222 tcp://localhost:3333
node B.js tcp://localhost:2222 tcp://localhost:3333
node B.js tcp://localhost:2222 tcp://localhost:3333
node C.js tcp://localhost:1111
```

Observem que el resultat és correcte independentment de la quantitat d'instàncies de B, i de l'ordre en què s'inicien les instàncies. Volem automatitzar el desplegament usant docker (totes les instàncies en contenidors al mateix amfitrió):



- Assumim creada la imatge `tsr-zmq` (base per a les imatges d'A, B i C)
- Al component C, assumirem que el port del seu socket PULL és el 3333, i rep per la línia d'ordres la URL del socket PULL d'A
- El codi de B rep per línia d'ordre les URLs del socket PUB d'A i del socket PULL de C

S'ha organitzat el codi seguint l'estructura de directoris que es mostra a la dreta. Respon les qüestions 5 a 11 relatives al desplegament usant docker.

5. ***El fitxer `/ABC/B/Dockerfile` ha de contenir la línia següent:***
   - **A.** `CMD node B.js tcp://localhost:1111 tcp://localhost:3333`
   - **B.** `CMD node $URL_A $URL_B`
   - **C.** Cap de les altres afirmacions és correcta.
   - **D.** `CMD node B.js`

6. ***Desitgem llançar una execució amb 1 instància d'A, 5 instàncies de B, i una instància de C. Per a fer això hem d'executar l'ordre:***
   - **A.** `docker build B 5`
   - **B.** `docker compose up --scale B=5`
   - **C.** `docker compose up`
   - **D.** `docker run --scale B=5`

7. ***Si estem situats al directori `/ABC/`, indica quina de les ordres següents no s'executa correctament***
   - **A.** `docker build -t a .`
   - **B.** `docker compose up`
   - **C.** `docker images`
   - **D.** `docker ps`

8. ***En relació amb el contingut del fitxer `docker-compose.yml`***
   - **A.** A la secció corresponent al component A ha d'aparèixer una secció `links`
   - **B.** A la secció corresponent al component C ha d'aparèixer una secció `links`
   - **C.** A la secció corresponent al component A ha d'aparèixer una secció `environment`
   - **D.** A la secció corresponent al component B ha d'aparèixer una secció `expose`

9. ***Suposant que hem llançat un contenidor que executa una instància d'A, si volem esbrinar la seua IP ...***
   - **A.** Hem d'executar `docker ps` per esbrinar la identitat del contenidor seguit de `docker inspect identitat`
   - **B.** En executar el contenidor mitjançant `docker run A` obtenim com a resultat la IP associada a ell
   - **C.** Hem d'executar `docker images` per esbrinar la identitat de la imatge d'A, seguit de `docker rmi identitat`
   - **D.** És una informació interna a la qual no podem accedir

10. ***L'ordre de desplegament d'aquests contenidors haurà de ser:***
    - **A.** Primer A, després C i després B
    - **B.** Primer C, després A i després B
    - **C.** Primer A, la resta en qualsevol ordre
    - **D.** Primer C, la resta en qualsevol ordre

11. ***Sobre els Dockerfile respectius...***
    - **A.** Totes les opcions són correctes
    - **B.** Al d'A ha d'aparèixer `EXPOSE 1111 2222`
    - **C.** Al de B ha d'aparèixer `EXPOSE 1111 2222`
    - **D.** Al de C ha d'aparèixer `EXPOSE 1111 2222`

12. ***Suposem que afegim un socket ROUTER associat al port 3333 del component A de l'esquema anterior, amb els canvis adequats al codi d'A.js, i desitgem que siga accessible des de l'exterior mitjançant el port 99 de l'amfitrió.*** **Després de generar la nova imatge, haurem de...**
    - **A.** No podem desplegar conjuntament el nou sistema perquè necessitem iniciar manualment A amb `docker run -p 99:3333 ...`
    - **B.** Modificar `docker-compose.yml` per posar una entrada `ports` adequada al servei A
    - **C.** Generar les imatges de la resta de components B i C
    - **D.** No funcionarà perquè l'ús d'un únic port és incompatible amb un socket de múltiples cues

13. ***Selecciona l'afirmació correcta sobre la replicació passiva:***
    - **A.** Permet que les rèpliques servidores executen localment cada petició dels clients sense necessitat d'enviar les modificacions generades a les altres rèpliques
    - **B.** Permet que una sola rèplica execute l'operació formulada i difonga les modificacions a les altres
    - **C.** Generalment tolera fallades arbitràries
    - **D.** Exigeix que cada petició dels clients arribe a totes les rèpliques servidores abans que alguna inicie el processament d'aquesta petició

14. ***Per desplegar el sistema CBWL s'utilitza un fitxer `docker-compose.yml` on, dins de la secció corresponent al component `bro`, apareixen aquestes línies:***

```
environment:
  - LOGGER_HOST=log
  - LOGGER_PORT=9995
```

Seleccioneu l'opció correcta sobre les implicacions que té aquesta subsecció `environment`:

- **A.** Els components `wor` i `cli` han de ser iniciats abans que `bro`, ja que no se'ls arriba a esmentar en aquesta subsecció
- **B.** Al Dockerfile de la imatge usada per generar el component `bro`, s'utilitzen les variables d'entorn `LOGGER_HOST` i `LOGGER_PORT`
- **C.** El component `bro` ha de ser iniciat durant el desplegament abans que el component `log`
- **D.** El component `log` s'ha de connectar a un port, que no pot ser el `9995`, del component `bro`

15. ***L'ordre `docker commit` es pot utilitzar per a:***
   A. Totes les altres opcions són incorrectes
   B. Construir una imatge docker a partir d'un fitxer Dockerfile
   C. Generar una imatge docker a partir de l'estat actual d'un contenidor determinat
   D. Iniciar l'execució d'un contenidor utilitzant la imatge indicada als seus arguments

16. ***Si una execució respecta la consistència causal, aleshores també respectarà la consistència...***
   A. Estricta
   B. FIFO
   C. Cache
   D. Seqüencial

17. ***Suposeu un sistema format per tres processos P1, P2 i P3, on s'ha donat la següent execució:*** `W1(x)2, W2(x)1, R1(x)1, R2(x)2, R3(x)2, R3(x)1`. ***Aquesta execució respecta, entre altres, les consistències:***
   A. FIFO i causal
   B. FIFO i cache
   C. Només la consistència cache
   D. Estricta i seqüencial

18. ***En considerar el Teorema CAP per a un servei escalable se sol recomanar la renúncia a una consistència forta. En base a això, seleccioneu quin model de consistència no es podria suportar quan hi haja problemes de connectivitat que generen una partició a la xarxa:***
   A. De manera general, qualsevol model "ràpid"
   B. FIFO
   C. Causal
   D. Seqüencial

19. ***Seleccioneu quina característica es respecta en la replicació multimàster:***
   A. Generalment ofereix consistència seqüencial
   B. Totes les rèpliques executen la seqüència d'instruccions de cada operació demanades pels processos clients
   C. Pot tolerar fallades arbitràries
   D. Sol ser més eficient que els models de replicació actiu i passiu

20. ***La imatge `tsr-zmq`, esmentada al llarg de la pràctica 3, s'ha de construir mitjançant el Dockerfile proporcionat al mateix enunciat de la pràctica. Indica quina de les següents afirmacions sobre aquest Dockerfile és certa.***
   A. No inclou la biblioteca `tsr.js` a la imatge produïda, però sí que incorpora la de ZeroMQ
   B. Es basa en una imatge Ubuntu
   C. Afegeix suport per a NodeJS
   D. Totes les respostes són correctes

21. ***El desplegament realitzat a la primera sessió de la pràctica 3 va ser...***
   A. Un sistema de xat amb patrons PUB/SUB i PUSH/PULL
   B. Un sistema client-broker-worker amb doble broker
   C. Cap de les altres opcions és correcta
   D. Un sistema client-broker-worker amb tolerància a fallades del broker

22. ***Selecciona l'afirmació correcta sobre el component `logger` que afegim al patró CBW per generar el patró CBWL:***
   A. Afegeix cada missatge en un fitxer que forma part d'un volum accessible des de la màquina amfitriona
   B. Manté en fitxer únicament el darrer missatge rebut (cada missatge reemplaça el contingut anterior)
   C. Aboca els missatges que rep a través del socket PULL a un fitxer intern, accessible únicament des del propi contenidor que executa la instància del `logger`
   D. Difon els missatges que li arriben del broker a la resta de components