TSR

Examen de recuperació de la Pràctica 2 (30 de gener de 2025)

Aquesta prova es compon de dues preguntes. Requereix obtenir el mínim indicat a la guia docent (3 sobre 10), i contribueix amb 3 punts a la nota final.

---

1. (**5 punts**) (Contesta en paper separat) Donat el codi del **publicador** de la pràctica 2:

```
 1: const {zmq, error, lineaOrdenes, traza, adios, creaPuntoConexion} =
    require('../tsr')
 2: lineaOrdenes("port tema1 tema2 tema3")
 3: let temas = [tema1,tema2,tema3]
 4: let pub = zmq.socket('pub')
 5: creaPuntoConexion(pub, port)
 6:
 7: function envia(tema, numMensaje, ronda) {
 8:   traza('envia','tema numMensaje ronda',[tema, numMensaje, ronda])
 9:   pub.send([tema, numMensaje, ronda])
10: }
11: function publica(i) {
12:   return () => {
13:     envia(temas[i%3], i, Math.trunc(i/3))
14:     if (i==10) adios([pub],"No me queda nada que publicar. Adios")()
15:     else setTimeout(publica(i+1),1000)
16:   }
17: }
18: setTimeout(publica(0), 1000)
19: pub.on('error', (msg) => {error(`${msg}`)})
20: process.on('SIGINT', adios([pub],"abortado con CTRL-C"))
```

Modifiqueu aquest programa de manera que respecte totes aquestes condicions simultàniament:

a) Emetrà un missatge **periòdicament**, alternant cíclicament entre tots els temes especificats als arguments rebuts, sense fi. (**30%**)

b) El nombre de temes a utilitzar el decidirà l'usuari en cada execució, facilitant els arguments necessaris en la línia d'ordres. (**30%**)

c) Els missatges es difondran cada mig segon. (**10%**)

d) No s'ha d'utilitzar `setTimeout` i tampoc una variable global per donar el valor de `numMissatges` quan s'invoqu2 la funció `envia`. (**30%**)

(**5 punts**) Contesta a la pàgina següent) Donat el codi del broker del sistema tolerant a fallades utilitzat en la darrera sessió de la pràctica 2:

```
 1:  const {zmq,lineaOrdenes,traza,error,adios,creaPuntoConexion} = require('../tsr')
 2:  const ans_interval = 2000
 3:  lineaOrdenes("frontendPort backendPort")
 4:  let failed   = {}
 5:  let working  = {}
 6:  let ready    = []
 7:  let pending  = []
 8:  let frontend = zmq.socket('router')
 9:  let backend  = zmq.socket('router')
10:  function dispatch(client, message) {
11:    traza('dispatch','client message',[client,message])
12:    if (ready.length) new_task(ready.shift(), client, message)
13:    else pending.push([client,message])
14:  }
15:  function new_task(worker, client, msg) {
16:    traza('new_task','client message',[client,msg])
17:    working[worker]=setTimeout(()=>{failure(worker,client,msg)}, ans_interval)
18:    backend.send([worker,'', client,'', msg])
19:  }
20:  function failure(worker, client, message) {
21:    traza('failure','client message',[client,message])
22:    failed[worker] = true
23:    dispatch(client, message)
24:  }
25:  function frontend_message(client, sep, message) {
26:    traza('frontend_message','client sep message',[client,sep,message])
27:    dispatch(client, message)
28:  }
29:  function backend_message(worker, sep1, client, sep2, message) {
30:    traza('backend_message','worker sep1 client sep2 message',
31:           [worker,sep1,client,sep2,message])
32:    if (failed[worker]) return
33:    if (worker in working) {
34:      clearTimeout(working[worker])
35:      delete(working[worker])
36:    }
37:    if (pending.length) new_task(worker, ...pending.shift())
38:    else ready.push(worker)
39:    if (client) frontend.send([client,'',message])
40:  }
41:  frontend.on('message', frontend_message)
42:   backend.on('message',  backend_message)
43:  frontend.on('error'  , (msg) => {error(`${msg}`)})
44:   backend.on('error'  , (msg) => {error(`${msg}`)})
45:   process.on('SIGINT' , adios([frontend, backend],"abortado con CTRL-C"))
46:  creaPuntoConexion(frontend, frontendPort)
47:  creaPuntoConexion( backend,  backendPort)
```

(Les qüestions estan a la pàgina següent)

ALUMNA/E

TORN LABORATORI

Cert programador ha analitzat el codi d'aquest broker i ha suggerit que permet gestionar adequadament els escenaris següents:

a) Reenviament d'una petició cap al primer treballador disponible, ja que n'hi ha algun.
b) Es fica a la cua una petició si no hi ha treballadors disponibles.
c) Reenviament d'una resposta cap al seu client.
d) Reenviament d'una petició cap a un altre treballador, quan el treballador inicialment assignat falla.
e) Es fica a la cua una petició després d'haver fallat el treballador cap al qual va ser reenviada inicialment, si no hi ha altres treballadors disponibles.
f) Descart d'una resposta tardana enviada per un treballador excessivament lent.
g) Admissió d'un missatge inicial de registre enviat per un nou treballador.
h) Arribada, dins del termini previst, d'una resposta emesa per un treballador.
i) Reenviament d'una petició que estava a la cua cap a un treballador que acaba de quedar lliure.

Identifiqueu (marcant a la taula) quin escenari o escenaris, d'entre els que acabem de llistar, podria ocasionar que les condicions utilitzades en les línies següents arribaren a complir-se i s'executaren les seues instruccions associades:

i. Línia 12: `if (ready.length) new_task(ready.shift(), client, message)`
ii. Línia 32: `if (failed[worker]) return`
iii. Línia 33: `if (worker in working) { ... }`
iv. Línia 37: `if (pending.length) new_task(worker, ...pending.shift())`
v. Línia 39: `if (client) frontend.send ([client,'',message])`

(contesta en aquesta mateixa taula amb SÍ o NO a cada cel·la)

	a	b	c	d	e	f	g	h	i
i									
ii									
iii									
iv									
v									

NO OBLIDES ENTREGAR AQUEST FULL