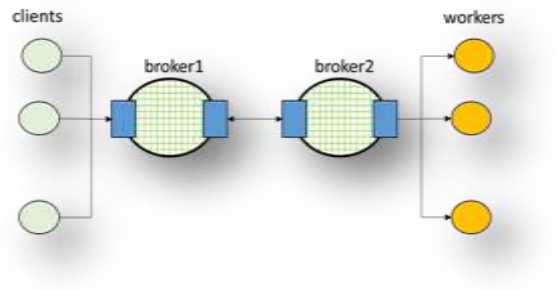
TSR – EXAMEN LABORATORI 2, 02-12-2021

Pregunta 1 (4 punts) Els dos programes següents (**broker1** i **broker2**) representen un dels possibles intents de solució al problema de dividir el broker en dos, tal com es proposa en la pràctica 2

broker1.js

```
1:  const zmq = require('zeromq')
2:  let nw=0, cli=[], msg=[]
3:  let sc = zmq.socket('router')
4:  let sb = zmq.socket('*****')
5:  sc.bind('tcp://*:9990')
6:  sb.bind('tcp://*:9991')
7:
8:  function dispatch(c,m) {
9:     nw--
10:    sb.send([c,'',m])
11: }
12:
13: sc.on('message',(c,sep,m) => {
14:    if (nw!=0) dispatch(c,m)
15:    else {cli.push(c); msg.push(m)}
16: })
17:
18: sb.on('message',(c,sep,r) => {
19:    nw++
20:    if (c!='') sc.send([c,'',r])
21:    //
22: })
```

NOTA.- assumeix que clients i workers són parametrizables amb el seu id i url on connectar (d'altra banda són els mateixos usats en pràctiques)



broker2.js

```
1:  const zmq = require('zeromq')
2:  let workers=[]
3:  let sb = zmq.socket('*****')
4:  let sw = zmq.socket('router')
5:  sb.connect('tcp://localhost:9991')
6:  sw.bind('tcp://*:9992')
7:
8:  sw.on('message', (w,sep,c,sep2,r) => {
9:     workers.push(w)
10:    sb.send([c,'',r])
11:
12: })
13:
14: sb.on('message', (c,sep,m) => {
15:    sw.send([workers.shift(),'',c,'',m])
16: })
```

Respon <u>de forma raonada</u> a les següents qüestions:

a) Indica el tipus de socket que podem usar per al socket `sb` de broker1 i per al socket `sb` de broker2 (línies 4 de broker1 i 3 de broker2). Justifica la resposta.
b) Justifica si el codi proposat funcionarà correctament en aquelles situacions relacionades amb la disponibilitat de workers.
c) Justifica si caldrà afegir codi addicional a partir de la línia 21 de broker1 per arribar a un funcionament correcte del doble broker. Si faltara codi, escriu-lo.

Pregunta 2 (1 punt) La versió del **broker amb tolerància a fallades** inclosa en la pràctica 2 permet experimentar amb certs escenaris d'error. Has de contestar a les següents preguntes relacionades amb aqueix sistema:

a) Explica què ocorre si el broker falla. Has d'esmentar l'efecte sobre la resta de components, les possibles peticions en curs i el sistema complet.
b) Explica què ocorre si falla un worker. Has d'incloure les diferències entre els casos en els quals, en el moment de la fallada, el worker estiguera processant una sol·licitud o a l'espera de rebre'n alguna.
c) Explica què ocorre si un worker, que el broker va considerar *avariat*, retorna amb retard al broker una resposta `r` a la sol·licitud `m` d'un client `c`.

Pregunta 3 (4 punts) Disposem d'un **sistema de xat** idèntic al descrit en l'apartat 5 de la pràctica 2, del qual es mostra el codi d'un client.

```
1:  const zmq = require('zeromq')
2:  const nick='Ana' //Assume it's random.
3:  let sub = zmq.socket('sub')
4:  let psh = zmq.socket('push')
5:  sub.connect('tcp://127.0.0.1:9998')
6:  psh.connect('tcp://127.0.0.1:9999')
7:  sub.subscribe('')
8:  sub.on('message', (nick,m) => {
9:     console.log('['+nick+']'+m)
10: })
11: process.stdin.resume()
12: process.stdin.setEncoding('utf8')
13: process.stdin.on('data'  ,(str) => {
14:    psh.send([nick, str.slice(0,-1)])
15: })
16: process.stdin.on('end',() => {
17:    psh.send([nick, 'BYE'])
18:    sub.close(); psh.close()
19: })
20: process.on('SIGINT',() => {
21:    process.stdin.end()
22: })
23: psh.send([nick,'HI'])
```

Un dels participants en aquest xat (el "*mafiós*") ha ideat una manera d'abusar del sistema per mitjà de clients del xat ficticis ("*esbirros*") que obeeixen les seues ordres. La seua operativa és…

- Quan el mafiós llig un missatge del xat (*missatge_original*) procedent d'un client concret (el denominem *objectiu*), reaccionarà ordenant que cada *esbirro* envie un missatge al xat amb contingut "**No m'agrada el missatge d'*objectiu* : *missatge_original***".
- Tant el *mafiós* com els *esbirros* seran versions modificades del client genèric de xat. És extremadament convenient conèixer que **la seua implementació només afig instruccions a l'original**, mantenint una separació neta entre la part *client de xat* i la part d'interacció *mafiós-esbirros*.
    - L'única excepció és el moment que provoca la reacció del *mafiós*.
- El *mafiós* i els *esbirros* reben des de la línia d'ordres, en la seua variable ***port***, el número de port a utilitzar per a intercomunicar-se. El mafiós rep també en la seua variable ***target*** l'identificador de l'usuari a molestar. No és necessari escriure codi per a assignar valor a aqueixes variables en les qüestions c) i d).

Dissenya el codi de mafiós i esbirros per a contestar les següents qüestions:

a) Tria el o els tipus de socket ZeroMQ per a comunicar mafiós i esbirros. Argumenta l'elecció.
b) El codi del mafiós es basa en el client de xat. Quines instruccions inseriries en aqueix codi per a detectar el missatge de l'objectiu i iniciar la reacció? Indica on, referenciant els números (o número) de línia del codi del client de xat.
c) Seguint amb *el mafiós*, escriu les instruccions que afegiries al codi del client de xat per a construir el programa `mafiós`.
d) Centrant-nos en els *esbirros*, escriu les instruccions que afegiries al codi del client de xat per a construir el programa `esbirro`.

Pregunta 4 (1 punt) En un dels apartats de la pràctica 2 es demana visualitzar periòdicament **estadístiques sobre les peticions ateses**, però en aquesta pregunta no ens interessa el número total. Explica quines estructures de dades has dissenyat per a mantenir la informació necessària **per a cada worker**, i com accedeixes a aquestes. Il·lustra-ho implementant la funció que, segons l'enunciat, hauria de mostrar aqueixa informació cada 5 segons en pantalla (per exemple, una funció `visualize()` invocada mitjançant `setInterval(visualize, 5000)`)