TSR - Pràctica 2: **0MQ**

Curs 2019/20

Contents

1. Introducció

1.1. Objectius

- Afermar els conceptes teòrics introduïts en el tema 3
- Experimentar amb diferents patrons de disseny (patrons bàsics de comunicació) i tipus de sockets
- Aprofundir en el patró broker (proxy invers)

1.2. Proposta per a l'organització del temps

- Sessió 1.- Publicador rotatori, prova aplicació Xat
- Sessió 2.- Parametrització broker, estadístiques broker
- Sessió 3.- Broker per a clients + broker per a workers
- Sessió 4.- Estudi possibles ampliacions broker, Prova patró broker tolerant a fallades

1.3. Mètode de treball

- Per a simplificar les proves, llancem els diferents components d'una aplicació en la mateixa màquina (utilitzant 'localhost' com IP del servidor). No obstant això, tot el codi subministrat podria provar-se sobre màquines diferents interconnectades sense fer canvis
- Els números de port que apareixen en els exemples s'han de modificar segons les instruccions indicades en el butlletí de la primera pràctica
- Tots els fragments de codi proporcionats haurien de provar-se en el laboratori. Per brevetat, els fragments de codi inclouen valors fixos per a indicar port, host, missatge a escriure, etc: en codi real haurien de llegir-se aqueixos valors des de la línia d'ordres

- El fitxer `RefZMQ.pdf` conté material de consulta/estudi/referència necessari per al desenvolupament de les tasques
- El fitxer `fonts.zip` conté tots els exemples descrits en `RefZMQ.pdf`

2. Tasques

2.1. Publicador rotatori

Utilitzant el patró `pub/sub` (difusió) descrit en `RefZMQ.pdf`, desenvolupa un programa `publicador.js` que s'invoca com

```
node publicador port numMissatges tema1 tema2 ...
```

on:

- `port` és el port al qual han de connectar-se els subscriptors
- `numMissatges` és el nombre total de missatges a emetre (un missatge per segon), després de la qual cosa acaba
- `tema1 tema2 ...` és un nombre variable de temes que es recorren segons un torn rotatori

Ex. després de la invocació `node publicador 8888 5 Política Futbol Cultura` el publicador ha d'emetre

- Després d'1 segon: `1: Política 1`
- Després de 2 segons: `2: Futbol 1`
- Després de 3 segons: `3: Cultura 1`
- Després de 4 segons: `4: Política 2`
- Després de 5 segons: `5: Futbol 2`

És a dir, un primer valor corresponent al segon, el tema, i un segon número que correspon a la quantitat de missatges d'aquest tipus que s'han emès.

2.2. Prova aplicació xat

En `RefZMQ.pdf` es dissenya i implementa una aplicació xat rudimentària. Analitza el codi per a comprendre el seu funcionament, i comprova el seu funcionament.

2.3. Parametrització broker

Assumint el patró broker implementat amb sockets `router/router` descrit en `RefZMQ.pdf`, modifica el codi per a acceptar els següents paràmetres en línia d'ordres:

- `node broker.js portFrontend portBackend`
- n invocacions `node worker.js urlBackend nickWorker txtResposta`
- m invocacions `node client.js urlFrontend nickClient txtPetició`

Els valors dels arguments han de ser coherents entre si. Per exemple:

- `node broker.js 8000 8001`
- `node worker.js tcp://localhost:8001 W1 Resp1`
- `node worker.js tcp://localhost:8001 W2 Resp2`
- `node client.js tcp://localhost:8000 C1 Hello`
- `node client.js tcp://localhost:8000 C2 Hola`
- `node client.js tcp://localhost:8000 C3 Hi`

Quan un client llança una petició (per exemple, missatge "txtPetició") ha de rebre com a resposta "txtResposta n", on n és un valor numèric que indica la quantitat de respostes que s'han generat fins al moment entre tots els workers.

Si es vol fer una prova amb diversos clients i/o workers, és convenient tenir en compte el següent:

- En lloc d'obrir un terminal per a cada ordre, es poden llançar diverses instàncies amb una única ordre:
  - `node client.js & node client.js & ...`
  - o mitjançant un shell-script (al qual es passa com a argument la quantitat d'instàncies)
- En aqueix cas no resulta tan fàcil finalitzar l'execució de cadascun (no tenim un terminal dedicat on prémer ctrl-C)
  - Una opció és usar l'ordre `kill pid1 pid2 ...`, on `pidX` és el número de procés que s'observa en llançar ordres en background
  - Una altra solució és limitar la duració de cada client/treballador introduint en el programa la sentència `setTimeout(()=>{process.exit(0)}, ms)`, on `ms` indica el temps de vida del programa en mil · lisegons

2.4. Estadístiques broker

Assumint el patró broker implementat amb sockets `router/router` descrit en `RefZMQ.pdf`, afig el codi necessari en el broker per a mantenir el total de peticions ateses i el nombre de peticions ateses per cada worker. Aquesta informació ha de mostrar-se en pantalla cada 5 segons

2.5. Broker per a clients + Broker per a workers

Assumint el patró broker implementat amb sockets `router/router` descrit en `RefZMQ.pdf`, es tracta de dividir el broker actual en 2 (broker1 i broker2):

- Broker1 es responsabilitza dels clients, i manté la cua de peticions pendents
- Broker2 es responsabilitza dels workers (alta, baixa, equilibrat de càrrega…)

El funcionament és el següent:

- Els clients envien les peticions a Broker1, que les passa a Broker2, i aquest al worker que corresponga.
- La resposta del worker arriba a Broker2, que la passa a Broker1, i aquest al client.

La solució triada ha de mantenir les mateixes característiques externes que observaven workers i clients.

2.6. Prova del patró broker tolerant a fallades

`RefZMQ.pdf` descriu diferents implementacions del patró broker. Compara el funcionament del broker `router/router` i el broker tolerant a fallades dels workers.

1. Llancem el broker `router/router` i tres processos worker, eliminem el primer dels workers mitjançant l'ordre `kill`, i llancem 3 clients
   - `$ node broker & node worker & node worker & node worker &`
     `[1] 10300 [2] 10301 [3] 10302 [4] 10303`

```
$ kill 10301
$ node client & node client & node client &
```

- Anotem quantes respostes s'obtenen, quins treballadors les han enviades, i si queden clients esperant
- Eliminem tots els processos: `killall node`

2. Repetim la mateixa prova, però utilitzant el broker que tolera fallades dels workers