**分散式系統**

**Lab: MQTT**

**112753138 資碩工一 周宸宇**

**Basic Information**

(Type names, student IDs, and your department name below)

**安裝MQTT伺服器mosquitto**

1. 若您使用Windows請到<https://mosquitto.org/download/> 下傳並安裝適合的版本，系統預設安裝到C:\program files\mosquitto\下。安裝過程中，建議不要勾選install as service。
2. 若您使用Mac，請先確定您有安裝homebrew (參考https://brew.sh/ )

安裝完brew後，請開啟終端機，輸入brew install mosquitto

1. 如果無法安裝者，可使用公用測試伺服器mqtt://test.mosquitto.org:1883，但可能會額外收到奇怪訊息或是連線速度較慢。

**啟動與測試MQTT 伺服器**

1. 啟動MQTT伺服器

Windows: 開啟命令列視窗(cmd)，切換目錄至C:\Program Files\mosquitto之後，執行mosquitto.exe (雙擊滑鼠)

Mac: 開啟命令列視窗，請參考這裡的Step 2.: <https://oranwind.org/post-post-2/>或依次執行以下

brew services list

brew link mosquitto

brew services start mosquitto -d

brew services restart mosquitto

1. 測試Listener: 新增一個mqtt listener來訂閱「EVENT」這個topic

Windows: 在命令列視窗(cmd) C:\Program Files\mosquitto下執行mosquitto\_sub –t EVENT

Mac: 開啟終端機，執行mosquitto\_sub -t EVENT

1. 測試Publisher: 新增一個mqtt message publisher，發佈訊息到EVENT這個topic。

Windows: 在命令列視窗(cmd) C:\Program Files\mosquitto下執行mosquitto\_pub -t EVENT -m hello。此時，mqtt listener中應該會收到hello訊息。

Mac: 在終端機執行mosquitto\_pub -t EVENT -m hello。此時，mqtt listener中應該會收到hello訊息。

**寫作MQTT Listener**

1. 建立一個名為「mqttlab」的空專案，新增一個package.json 檔案
2. package.json對於JavaScript專案來說是專案設定檔，可以在其中載明所使用的library，之後可透過npm指令自動下傳。package.json內容如下:

{

"name": "mqttlab",

"version": "1.0.0",

"dependencies": {

"mqtt": "\*"

}

}

1. 請使用npm install安裝所需要的函式庫。
2. 將cep-template.js更名為cep.js，參考程式中的註解與下面說明的範例程式，完成一個簡單的MQTT CEP Listener。

由EVENT中接收輸入的事件，事件共四種: DO(DOOR ON), DF(DOOR OFF), CO(CAB ON), CF(CAB OFF)。偵測連續的「CO->CF->DO->DF」的sequence，當這四個事件先後發生時，就往EVENT送出一個「LEAVE」的事件。注意下列事項:

* 這四個事件只需要依次須序出現，不一定要連續出現，中間若插入任何事件時，應該予以忽略。例如: 「CO->AA->CF->BB->DO->DF」在DF偵測到之後，也需要輸出「LEAVE」。
* 匹配完最後的DF，並輸出「LEAVE」後，系統應該要回到從CO開始匹配「CO->CF->DO->DF」的sequence。
* 提示:單一事件匹配到的時候，利用JavaScript的shift方法，將第一個元素移除。全部匹配完成時，eventSeq.length 為0 (陣列中元素都匹配完了)。
* 可使用mosquitto\_sub/mosquito\_pub進行測試，例如mosquitto\_pub -t EVENT -m CO

程式樣板請見附檔: cep-template.js

|  |
| --- |
| 將您的cep.js所有**程式碼**貼在這裡:  const mqtt = require('mqtt');  const client = mqtt.connect();  client.subscribe('EVENT'); //請填入適當的值  let eventSeq = ['CO', 'CF', 'DO', 'DF'];  client.on('message', function (topic, message) {  let item = eventSeq[0]; // 取得eventSeq的第一個元素  // 除錯用  // message是Buffer型別，要用message.toString()才能取得字串值  console.log('input: ' + message.toString()); //message.toString()是 -m 後面的訊息  if (message.toString() != 'LEAVE'){  console.log('expected: ' + item);  console.log('-----------------');  }  //if (…) …; // 如果message和item匹配，將匹配元素從陣列移出 (使用eventSeq.shift())  // message是Buffer型別，要用message.toString()才能取得字串值  if (message.toString() == item) {  eventSeq.shift();  if (eventSeq.length != 0){  console.log('Seq ' + eventSeq);  }  else {  console.log('Seq empty');  }  console.log('-----------------');  }  //if (…) { // 如果全部匹配  // 送出LEAVE到EVENT topic並回復eventSeq的內容  if (eventSeq.length == 0) {  client.publish('EVENT', 'LEAVE');  eventSeq = ['CO', 'CF', 'DO', 'DF'];  }  }); |