CSC0056 – Data Communication

Homework 2

NTUST\_M11207521陳俊博

1. **Written assignment (40 points)**

Write your answer to the following questions in a PDF file and name it hw2-part1.pdf. Label your answer clearly.

1. **The epoll event notification facility (30 points)**
2. (20points) Does our Mosquitto version use the edge-trigger or level-trigger option of epoll? Give your reasoning and evidence.

**Ans:**

1. 我們的Mosquitto 版本使用的是level-trigger。
2. 在Mosquitto目錄中使用grep -r "epoll\_ctl" .，可得出以下內容：

m11207521@csc0056:~/mosquitto$ grep -r "epoll\_ctl" .

Binary file ./src/mux\_epoll.o matches

./src/mux\_epoll.c: if (epoll\_ctl(db.epollfd, EPOLL\_CTL\_ADD, listensock[i].sock, &ev) == -1) {

./src/mux\_epoll.c: if(epoll\_ctl(db.epollfd, EPOLL\_CTL\_ADD, context->sock, &ev) == -1) {

./src/mux\_epoll.c: if((errno != EEXIST)||(epoll\_ctl(db.epollfd, EPOLL\_CTL\_MOD, context->sock, &ev) == -1)) {

./src/mux\_epoll.c: if(epoll\_ctl(db.epollfd, EPOLL\_CTL\_ADD, context->sock, &ev) == -1) {

./src/mux\_epoll.c: if((errno != EEXIST)||(epoll\_ctl(db.epollfd, EPOLL\_CTL\_MOD, context->sock, &ev) == -1)) {

./src/mux\_epoll.c: if (epoll\_ctl(db.epollfd, EPOLL\_CTL\_ADD, context->sock, &ev) == -1) {

./src/mux\_epoll.c: if(epoll\_ctl(db.epollfd, EPOLL\_CTL\_DEL, context->sock, &ev) == -1){

開啟顯示紫色區塊的檔案，執行vim ./src/mux\_epoll.c。接著按下/輸入ev.events查詢，可以分別在第97行、116行、135行以及152行看到以下資訊：

97 ev.events = EPOLLIN;

116 ev.events = EPOLLIN | EPOLLOUT;

135 ev.events = EPOLLIN;

152 ev.events = EPOLLIN;

由此可知，並沒有發現有關EPOLLET的相關資訊。因為如果要啟用edge-trigger，就要在ev.events中設定EPOLLET。所以沒有設定EPOLLET就代表預設使用level-trigger。

一張含有 文字, 螢幕擷取畫面 的圖片

自動產生的描述

圖1 查詢過程

1. (10points) Independent from Mosquitto, use your own words to illustrate a run-time scenario where edge-trigger is better than level-trigger, in the context of using epoll.

**Ans:** 例如有一個購物平台上，有大量使用者同時在下訂單，伺服器需要處理很多的訂單請求，比如說檢查庫存、寫入訂單的資料等等。如果使用的是level-trigger，伺服器會一直被提醒說「可以處理囉」，但伺服器暫時還沒有辦法處理，這就會造成很多重複的通知導致浪費伺服器資源。但如果使用的是edge-trigger的話只會在狀態變化時通知一次。當有資源準備好時再去處理，不會重複地收到通知。尤其是在高峰的情況下，edge-trigger可以減少重複或不必要的通知，讓伺服器運作得更有效率。

1. **The broker-side implementation (10 points)**
2. (10 points) Give your own reasoning about why the Mosquitto broker only implements a reactive server design and not those more advanced ones (review the Week-2 slides for server designs)? Think about some practical scenarios: for example, what assumptions would make implementing a reactive server more appealing than implementing one that runs the leader/followers pattern? Your reason can be both very simple and correct. A good reason needs not to be complicate.

**Ans:**

1. Mosquitto用反應式設計是因為它簡單能處理大量小型訊息，不需要額外的複雜設計，已經夠用了。
2. 在一個場景當中有很多IoT裝置(像溫溼度感測器等等)連接到了Mosquitto broker，這些裝置只傳小量的資料量。用反應式設計就很夠了，因為系統只需要在有請求時做事，既省事又省資源。而像領導者/追隨者模式這種複雜設計，對簡單的需求反而太過頭了，不用搞得這麼麻煩。