1 mqtt broker

1.1 retained消息

retained 消息是指在 PUBLISH 数据包中 Retain 标识设为 1 的消息，broker 收到这样的 PUBLISH 包以后，将保存这个消息，当有一个新的订阅者订阅相应主题的时候，broker 会马上将这个消息发送给订阅者。有以下这些特点：

- 一个Topic只能有一条retained消息，发布新的retained 消息将覆盖老的 retained 消息（所以想删除一个 retained 消息也很简单，只要向这个主题发布一个 payload 长度为 0 的 retained 消息就可以了，**这样订阅端再登录就收不到消息了**），比如以下所示：

  ```
  mosquitto_pub -t hello -m "" -r
  ```

- 如果订阅者使用通配符订阅主题，它会收到**所有匹配的主题上的 retained 消息**；
- 只有新的订阅者才会收到 retained 消息，**如果订阅者重复订阅一个主题，也会被当做新的订阅者，然后收到 retained 消息**；

  ```
  如果client_id不为空，clean_sesson为false，则在成功订阅一次后，对应的client_id再次登录的时候可以不再调用mosquitto_subscribe
  mosquitto_subscribe(mosq, NULL, TOPIC_NAME, QOS);
  ```

- broker 收到 retained 消息后，会单独保存一份，再向当前的订阅者发送一份普通的消息（retained 标识为 0）。当有新订阅者的时候， broker 会把保存的这条消息发给新订阅者（retained 标识为 1）。

启动代理服务：mosquitto -v

- -v 详细模式 打印调试信息
- 默认占用：1883端口

发布内容：mosquitto_pub -t hello -m world -r

- -t 指定订阅的主题，主题为：hello
- -m 指定发布的消息的内容
- -r retain
- -q 默认为0，可以指定

订阅主题：mosquitto_sub -v -t hello -i id_123456 -c

- -t 指定订阅的主题，主题为：hello
- -v 详细模式 打印调试信息

- -i 对应client_id

更多看emqx_sub.cpp

1.2 cleansession

此项配置是client端在初始化时传入的参数（主要针对sub订阅端）

```
bool clean_session = false;
mosq = mosquitto_new(CLIENT_ID, clean_session, NULL);
```

每个库的名字可能不一样.

- broker端其实并不是靠clientId区别client端的，**实际上还是靠session**，broker端如果没有持久化到数据库中，**重新启动以后session会全部丢失。**
  - 此配置项如果改为true，client端每次重连都会重新申请session，**broker端就不能够判断出此设备是以前已经连接过的设备了。**
  - 如果配置为false，client端在上线并订阅主题后，broker会查找设备在离线时没有接收到的消息，一股脑全发出去，而这些消息很可能早就没有价值了。
- 当你刚启动client端就收到一堆消息,并且并没有pub消息时，很有可能就是此项配置的问题，所以sub端需要根据需要设置正确的参数。

retained消息和持久性会话的区别：

- retained消息是broker为每一个Topic单独存储的；
- 持久性会话是broker为每一个Client单独存储的

实验一：可以通过修改emqx_sub.cpp的clean_session字段为true、false进行测试。

实验二：使用mosquitto （其实不需要进一步设置配置文件就能收到历史消息）

```
# 启动服务
mosquitto -v

# 通过-q 1设置qos为1，确保支持离线消息。实测qos为2也可以。
mosquitto_pub -h localhost -p 1883 -i "pub1" -t "/alarm/" -m "date" -q 1



# 通过-c -q 1支持离线消息。实测qos为2也可以。
# 不加-c参数则不会打印离线消息
# 注册后，可以先断开sub端， 然后 pub多个消息，然后再重新启动sub端，观察收到的数据情况
mosquitto_sub -h localhost -p 1883 -i "sub1" -t "/alarm/" -c -q 1
```

实验三

需要mosquitto -v -c /etc/mosquitto/mosquitto.conf启动

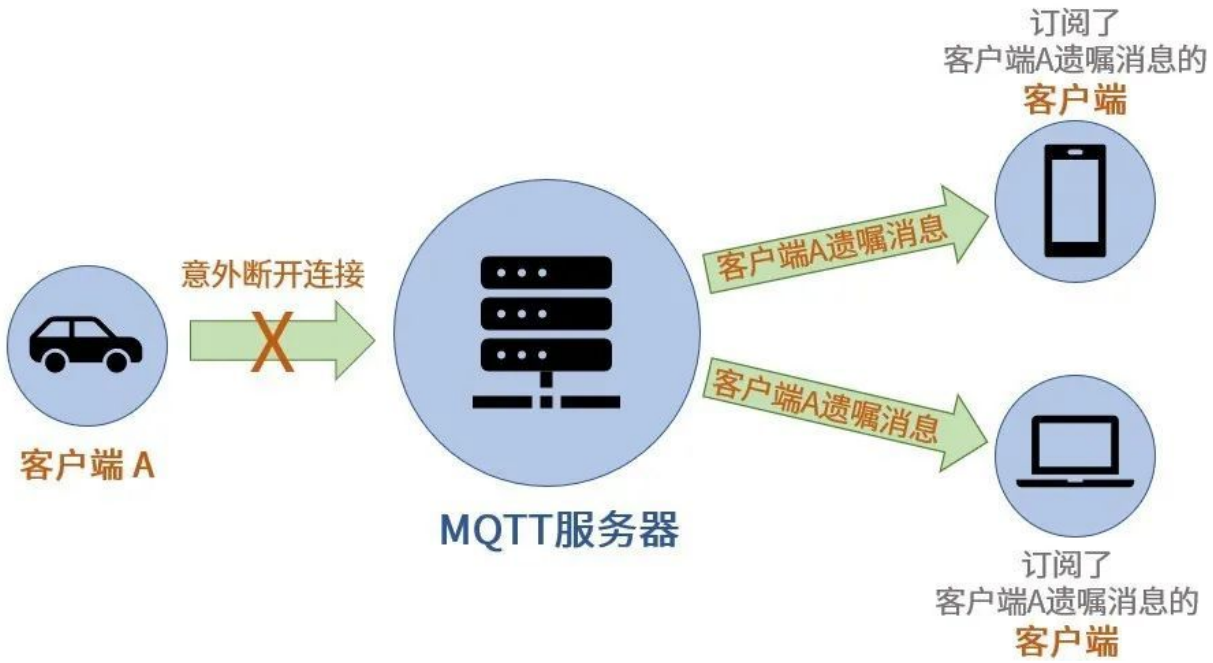
sub.cpp 订阅端和发布端我们把client_id区别开来，不要设置成一样的，设置成一样的就变成了使用同一个session （课上演示离线消息不成功不是配置文件的，而是因为sub、pub 的client id一致，导致在pub的时候此时client并不是离线的，对于sub就没有所谓离线的说法，所以sub、pub在做实验的时候两者的client id需要有区别）

pub.cpp

1.3 mqtt的遗嘱机制

MQTT的心跳机制可以让服务端随时掌握客户端连接情况。当客户端“心跳”正常时，服务端即知道客户端仍然在线（活着）。当心跳一旦停止，服务端就会发现该客户端已经断线（死亡）。
为了让客户端可以更好的发挥作用，便于服务端管理，MQTT协议允许客户端在“活着”的时候就写好遗嘱，这样一旦客户端意外断线，服务端就可以将客户端的遗嘱公之于众。

遗嘱消息可以看作是一个简化版的 PUBLISH 消息，他也包含 Topic, payload, QoS 等字段。遗嘱消息会在设备与服务端连接时，通过 CONNECT 报文指定，然后在设备意外断线时由服务端将该遗嘱消息发布到连接时指定的遗嘱主题（Will Topic）上。这也意味着服务端必须在回复 CONNACK 之前完成遗嘱消息的存储，以确保之后任一时刻发生意外断线的情况，服务端都能保证遗嘱消息被发布。



那么什么是意外断线呢?

- 正常断开：当客户端正常断开连接时，会向服务端发送DISCONNECT报文，服务端接收到该报文后，就知道，客户端是正常断开连接，而并非意外断开连接。
- 意外断线：当服务端在没有收到DISCONNECT报文的情况下，发现客户端“心跳”停止了，这时服务端就知道客户端是意外断线了。
    1. 因网络故障或网络波动，设备在保持连接周期内未能通讯，连接被服务端关闭
    2. 设备意外掉电
    3. 设备尝试进行不被允许的操作而被服务端关闭连接，例如订阅自身权限以外的主题等

以下为遗嘱消息在 MQTT 5.0 和 MQTT 3.1 & 3.1.1 的差异:

	MQTT 5.0	**MQTT 3.1 & 3.1.1**
Will Retain	Yes	Yes
Will QoS	Yes	Yes
Will Flag	Yes	Yes
Will Properties	Yes	No
Will Topic	Yes	Yes
Will payload	Yes	Yes

Will Retain、Will QoS、Will Topic 和 Will payload 的用处与普通 PUBLISH 报文基本一致

唯一值得一提的是 Will Retain 的使用场景，它是保留消息与遗嘱消息的结合。如果订阅该遗嘱主题（Will Topic）的客户端不能保证遗嘱消息发布时在线，那么建议为遗嘱消息设置 Will Retain，

```
int qos = 1;
bool retain = true;
mosquitto_will_set(mosq, "topic/on/unexpected/disconnect", strlen("will message"),
"will message", qos, retain);
```

避免订阅端错过遗嘱消息。

遗嘱操作示例

broker： mosquitto -v

订阅端：mosquitto_sub -t topic/on/unexpected/disconnect

发布端：01-will-set.cpp （代码在git@gitlab.0voice.com:2304_vip/mqtt_0voice.git）

1.4 操作连接emqx云平台

代码

云平台申请：https://cloud.emqx.com/console/deployments/new
emqx_sub.cpp 订阅 （代码在：git@gitlab.0voice.com:2304_vip/mqtt_0voice.git）
emqx_pub.cpp 发布

云平台文档

云平台申请：https://cloud.emqx.com/console/deployments/new

选择专有版本免费试用14天，按具体提示操作即可。我选择的是阿里云，其他云操作应该也是一样的。

连接 MQTTX：https://docs.emqx.com/zh/cloud/latest/connect_to_deployments/mqttx.html
创建专有版部署：https://docs.emqx.com/zh/cloud/latest/create/dedicated.html
认证鉴权：先用用户名+密码方式
https://docs.emqx.com/zh/cloud/latest/deployments/auth_dedicated.html
监控指标：https://docs.emqx.com/zh/cloud/latest/deployments/operation_overview.html

2 发布订阅的过滤器

订阅

- mqttx_4299c767/home/+
- mqttx_4299c767/home/PM2_5

发布

- mqttx_4299c767/home/PM2_5
- mqttx_4299c767/home/temperature

观察订阅端收到的信息情况。

3 mqtt的日志实时监控

云平台实时监控

见课堂 emqx 云平台操作。
参考文档：https://docs.emqx.com/zh/cloud/latest/deployments/operation_overview.html

4 进阶学习

基于 EMQX 的 MQTT 使用案例 | EMQX

MQTT 最全教程：从入门到精通 | EMQ (emqx.com)

设计与实现 | EMQX 5.1 文档

访问控制 | EMQX 5.1 文档

MQTT over QUIC 多流支持 | EMQ (emqx.com)

MQTT Version 3.1.1 (oasis-open.org)

MQTT Version 5.0 (oasis-open.org)

5 docker 部署mqtt

有商业应用需要的朋友可以参考：http://www.steves-internet-guide.com/running-the-mosquitto-mqtt-broker-in-docker-beginners-guide/#more-17845

这个需要docker的基础才能理解，该网站也有大量和Mosquitto相关的文章

6 参考

MQTT 遗嘱消息（Will Message）的使用
Mosquitto感知客户端上下线的方法
EMQX vs Mosquitto | 2023 MQTT broker 对比
mqtt协议qos,retain,dup,cleansession,will的简单理解和应用