MQTT

1、MQTT应用实例 2

2、消息代理软件 3

3、客户端库 3

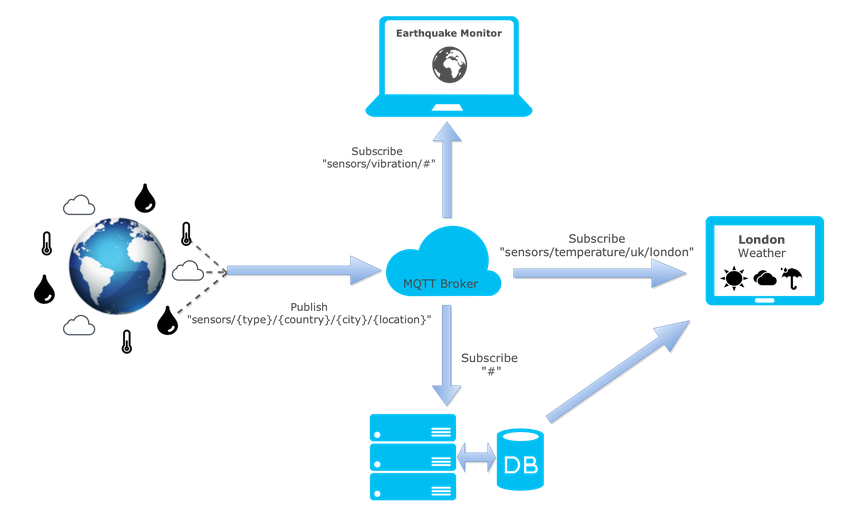
4、MQTT基础 3

5、MQTT服务器（broker：mosquitto-->实现了MQTT3.1协议的代理服务器） 6

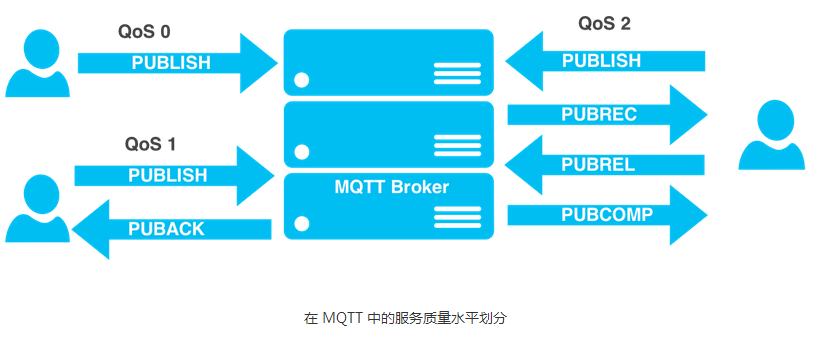
5.4.1、log输出功能文件（mosquitto\_root/src/logging.c） 7

5.4.2、配置参数管理 7

# 1、MQTT应用实例（messagesight.demos.ibm.com）



1.1 MQTT拓扑



1.2 MQTT服务质量等级

# 2、消息代理软件

2.1、[Mosquitto](http://mosquitto.org/" \t "http://geek.csdn.net/news/detail/_blank) —— 这是最早在生产环境中可用的消息代理之一，以 C 语言编写，提供多种配置与高性能。

2.2、[Mosca](https://github.com/mcollina/mosca" \t "http://geek.csdn.net/news/detail/_blank) —— 以 Node.js 编写，可嵌入 Node 应用或以独立可执行文件的形式运行。由于配置简单并具有可扩展性，它也是我们最喜欢的消息代理，具有高性能的优点。

2.3、[RSMB](https://www.ibm.com/developerworks/community/groups/service/html/communityview?communityUuid=d5bedadd-e46f-4c97-af89-22d65ffee070" \t "http://geek.csdn.net/news/detail/_blank) —— IBM 对 MQTT 协议的实现，也是最不常用的选项之一，不过它是一个用C语言编写的成熟系统。

2.4、[HiveMQ](http://www.hivemq.com/" \t "http://geek.csdn.net/news/detail/_blank) —— HiveMQ 是一种相对较新的消息代理，面向企业环境，在博客上有很多关于 MQTT 不错的信息。

# 3、客户端库

几乎包含了所有流行语言的客户端库，想要具体了解的话，Paho 项目会是你的最佳选择。这个项目隶属于 Eclipse，旨在提供各种语言尽可能多样化的 MQTT 客户端实现参考。这是个很好的资源，包含以C、Java、Python、Javascript等语言编写的可用客户端。

# 4、MQTT基础

4.1、发布/订阅模式（**最大订阅数：10--->供参考**）

发布/订阅模式即PUB/SUB，发布者和订阅者之间存在第三方经纪人（Broker），所有的发布者和接收者都要连接到消息经纪人，消息经济人会过滤所有到来的消息，并根据需要分发这些消息。

4.2、发布/订阅模式实现了发布者和订阅者之间的解耦

4.2.1、空间解耦：发布者和订阅者无需知道彼此的存在（比如ip地址和端口号）

4.2.2、时间解耦：发布者和订阅者无需同时运行

4.2.3、同步解耦：在发布和接收期间，双方组件的操作不会暂停

总结：broker（经纪人）通过对消息的过滤实现了只有某些客户端才能接收到相应的消息

4.3、增加了可扩展性

可以对broker节点进行集群来实现，以便通过负载均衡把负载分配到更多的服务器节点来实现

4.4、消息过滤

**中心**：经纪人broker怎样过滤所有的消息，以及怎样让每个订阅者只获得他感兴趣的消息

4.4.1、基于主题的过滤

过滤是根据标题（subject）或主题（Topic）的，而标题或主题是每一个消息的组成部分之一，**主题(TOPIC)是一个具有层次结构的字符串，可以基于有限数量的表达式进行过滤。**

4.4.2、基于内容的过滤

缺点：消息不能加密，而且消息事先知道

4.4.3、基于类型的过滤

4.5、协议介绍

4.5.1、发布者和订阅者只需要知道broker的ip和port，以实现发布/订阅消息，broker要能够存储信息，方便那些不在线的客户端订阅。（满足条件：客户端已经连接一次，它的回话是持久的，并且客户端订阅了服务质量Qos大于0的主题）**大多数的发布/订阅系统的实现逻辑通常都是在Broker端。**

4.5.2、MQTT使用了基于主题的过滤方式，因此，每一个消息都包含了一个主题Topic。客户端订阅信息时会指定主题，所以broker就可以根据主题来决定给订阅了主题的客户端发送那些信息，**认真的设计主题数（topic tree）会使得MQTT越来越好用**（问题：没有被订阅的主题可以输出日志到数据库供后续分析）

4.5.3、MQTT与消息队列的区别

1》消息队列会一直保存消息，直到消息被消费

2》消息队列中一条消息只能由一个客户端消费，MQTT对于每一个订阅者，只要订阅了某个主题，他们都可以得到该主题的消息。

3》队列的创建远不及主题的创建灵活。

4.6、MQTT数据传输

1》MQTT最小的数据包只有2bytes，且无应用消息头

2》client行为

连接到droker

发布消息以供其他client订阅

订阅其他client的消息

取消订阅

与droker断开连接

3》server行为

接收client的连接请求

接收client发布的消息

处理各client的订阅/取消订阅请求

推送与订阅相匹配的消息到对应的client

4》通过MQTT协议传输的消息数据都有一个与之关联的服务质量等级和主题名

5》subscription（订阅）

订阅由topic filter和maximum Qos组成

**一个session可以包含多个subscription**

**同一个session中的每一个subscription都有不同的topic filter**

6》主题名

消息的标签

Server推送消息到client的时候用topic name进行匹配推送

7》主题过滤

主题过滤是一个表达式，指出一个或多个感兴趣的话题

一个主题过滤可以包含多个通配符

8》session

Client与server之间的一个有状态的连接交互（会话）

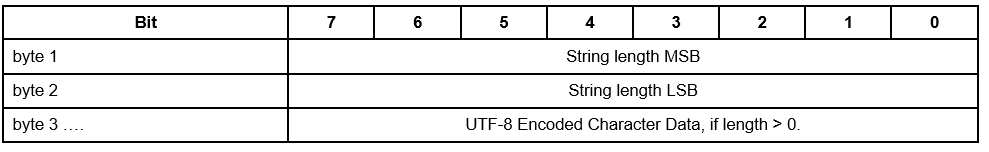
某些session存活时间与网路连接有关，一旦断开连接，session失效；某些session能跨越多个网络连接

9》数据格式

一个16位的祖父在网络传输中呈现的格式为MSB+LSB

所有的UTF-8编码字符串长度范围：0~65535 bytes

Utf-8编码结构：



10》MQTT控制包格式

固定头+可变头（不一定有）+负载（不一定有）

1》固定头（2字节）

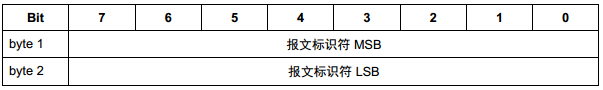


0~3 bits：指定控制报文类型的标志位

4~7 bits：MQTT控制报文的类型

**8~15 bits：剩余长度，包括可变报头和负载数据，不包含剩余长度字段本身，使用变长编码方案**

2》可变报头



固定头部仅定义了消息类型和一些标志位

一些消息的元数据，需要放入可变头部中，可变头部的内容根据packet type的不同而不同，一般包含：协议名，协议等级，连接标志，保持连接

某些情况下，可变头部包含了数据包ID

3》负载（payload）

消息体主要是为了配合固定/可变头部命令（比如connect可变头部user name标记若为1，则需要在消息体重附加用户名字符串）而存在。

CONNECT/SUBSCRIBE/SUBACK等消息有消息体。PUBLISH的消息体以二进制形式表示

# 5、MQTT服务器（broker：mosquitto-->实现了MQTT3.1协议的代理服务器）

Mosquitto在ubuntu上的安装和卸载：

安装：make-->make install

卸载：make uninstall-->make clean

## 5.1、mosquitto命令参数说明

-c：选择mosquitto启动时读取的配置文件

如果不跟默认选择/etc/mosquitto/mosquitto.conf

-d：以后台方式启动mosquitto

-p：指定监听端口（默认为1883，支持8883,8080,8081,8884）

-v：监控日志，类似于在配置文件中把log\_type设置为all

## 5.2、Mosquitto\_pub命令参数说明

-d：打印debug信息

-f：指定文件的内容作为发送消息的内容

-h：指定要连接的域名，默认为localhost

**-i：指定要给哪个clientId的用户发送消息**

-I：指定要哪个clientId前缀的用户发送消息

-m：消息内容

-n：发送一个空（null）消息

-p：连接端口号

-q：指定Qos等级的值（0,1,2）

-t：指定topic

-u：指定broker访问用户

-P：指定broker访问密码

-V：指定MQTT协议版本

--will-payload  指定一个消息，该消息当客户端与broker意外断开连接时发出。该参数需要与--will-topic一起使用

--will-qos  Will的QoS值。该参数需要与--will-topic一起使用

--will-retain 指定Will消息被当做一个retain消息（即消息被广播后，该消息被保留起来）。该参数需要与--will-topic一起使用

--will-topic  用户发送Will消息的topic

## 5.3、Mosquitto\_sub参数说明

-c：设定‘clean session’为无效状态，这样一直保持订阅状态，即便是断开连接再连接仍能接收断开时的数据

-d：打印debug信息

-h：指定要连接的域名，默认为localhost

-i：指定clientId

**-k：keepalive每隔一段时间，发PING消息通知broker，仍处于连接状态，默认为60秒**

-q：指定希望接收到Qos为什么的消息，默认为0

-R：不显示陈旧的消息

-t：订阅topic

-v：打印消息

--will-payload：指定一个消息，该消息当客户端与broker意外断开连接时发出，需要与--wil-topic一起使用

--will-qos：Will的Qos值，该参数需要与--will-topic一起使用

--will-retain：指定Will消息被当做一个retain消息（即消息被广播以后保留起来），一定要与--will-topic一起使用

--will-topic：用户发送Will消息的topic

目录结构：其中src和lib目录下主要放置mosquitto的实现代码以及部分底层与网络相关的操作，client目录主要为两个客户端程序的实现源码。

## 5.4、mosquitto辅助功能

### 5.4.1、log输出功能文件（mosquitto\_root/src/logging.c）

Log模块功能：日志分等级输出、日志的多平台输出、日志对各种参数的格式化输出

int mqtt3\_log\_init(struct mqtt3\_config \*config)：log初始化模块

### 5.4.2、配置参数管理

2.2.1、实现文件：mosquitto\_root/src/conf.c

2.2.2、Mosquitto对外提供的操作接口均以“mqtt3\_config\_”开头，共有以下四个：

(1)mqtt3\_config\_init

该函数的声明为：void mqtt3\_config\_init(struct mqtt3\_config \*config)

它主要完成对参数config的各结构体成员的初始化工作。

(2)mqtt3\_config\_cleanup

该函数的声明为：void mqtt3\_config\_cleanup(struct mqtt3\_config \*config)

它主要完成对参数config的各成员的清除、释放等操作。

(3)mqtt3\_config\_read

该函数的声明为：int mqtt3\_config\_read(struct mqtt3\_config \*config, bool reload)

它主要完成从配置文件中读取各配置参数到结构体config中，具体的配置文件读取工作由函数\_config\_read\_file完成。

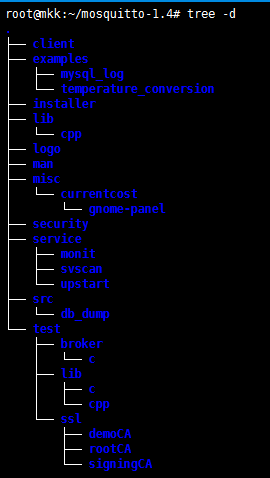
(4)mqtt3\_config\_parse\_args

该函数的声明为：int mqtt3\_config\_parse\_args(struct mqtt3\_config \*config, intargc, char \*argv[])

它主要完成对用户输出参数的解析工作。

2.2.3、内存操作封装

## 5.5、mosquitto目录树



## 5.6、订阅树机制的优缺点分析

Mosquito程序采用订阅树形式维护客户端之间的订阅与发布消息，这种方式优点是逻辑清晰，便于开发和维护。缺点是其遍历过程效率较低。同时，程序中存在很多对订阅树的遍历过程：订阅、发布消息、取消订阅等，在客户端数量增加时，该功能对效率的影响将更为明显。

因此，在mosquitto的实际应用中很难支持5万以上的客户端，尤其在客户端网络状态不好时，其断开重练操作将非常频繁，这样也造成大量对订阅树的遍历操作，从而严重影响mosquitto的效率。

## 5.7、mosquitto收发消息机制

Mosquito最主要的功能就是对消息的正确接收、维护和转发，因此其消息的接、收功能非常重要。Mosquitto的消息收发主要通过poll机制来完成。