Erlang MQTT Broker Documentation

Release 1.0

Feng Lee

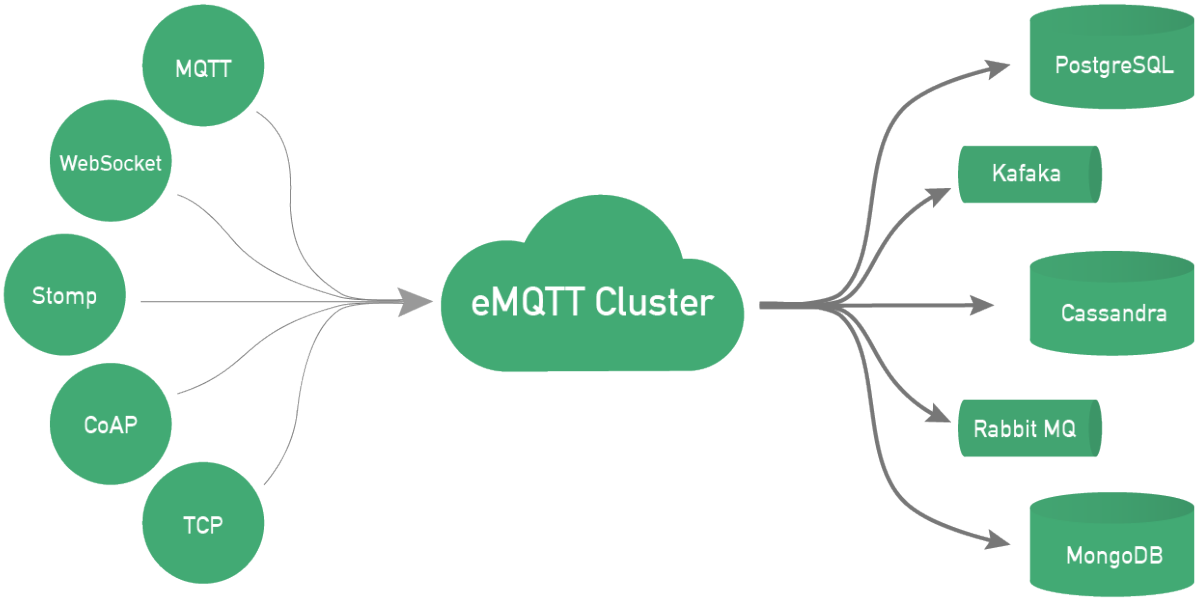
August 19, 2016

Contents

emqttd(Erlang MQTT Broker) is a massively scalable and clusterable MQTT V3.1/V3.1.1 broker written in Erlang/OTP.

emqttd is fully open source and licensed under the Apache Version 2.0. emqttd implements both MQTT V3.1 and V3.1.1 protocol specifications, and supports WebSocket, STOMP, SockJS, CoAP and MQTT-SN at the same time.

Latest release of the emqttd broker is scaling to 1.3 million MQTT connections on a 12 Core, 32G CentOS server.



The emqttd project provides a scalable, enterprise grade, extensible open-source MQTT broker for IoT, M2M, Smart Hardware, Mobile Messaging and HTML5 Web Messaging Applications.

Sensors, Mobiles, Web Browsers and Application Servers could be connected by emqttd brokers with asynchronous PUB/SUB MQTT messages.

Homepage:	http://emqtt.io
Downloads:	http://emqtt.io/downloads
GitHub:	https://github.com/emqtt
Twitter:	@emqtt
Forum:	https://groups.google.com/d/forum/emqtt
Mailing List:	emqtt@googlegroups.com
Author:	Feng Lee <feng@emqtt.io>

Note: MQTT-SNCoAP Protocols are planned to 1.x release.

Contents:

CHAPTER 1

Get Started

1.1 Overview

emqttd(Erlang MQTT Broker) is an open source MQTT broker written in Erlang/OTP. Erlang/OTP is a concurrent, fault-tolerant, soft-realtime and distributed programming platform. MQTT is an extremely lightweight publish/subscribe messaging protocol powering IoT, M2M and Mobile applications.

The emqttd project is aimed to implement a scalable, distributed, extensible open-source MQTT broker for IoT, M2M and Mobile applications that hope to handle millions of concurrent MQTT clients.

Highlights of the emqttd broker:

- Full MQTT V3.1/3.1.1 Protocol Specifications Support
- Easy to Install - Quick Install on Linux, FreeBSD, Mac and Windows
- Massively scalable - Scaling to 1 million connections on a single server
- Cluster and Bridge Support
- Easy to extend - Hooks and plugins to customize or extend the broker
- Pluggable Authentication - LDAP, MySQL, PostgreSQL, Redis Authentication Plugins

1.2 Features

- Full MQTT V3.1/V3.1.1 protocol specification support
- QoS0, QoS1, QoS2 Publish and Subscribe
- Session Management and Offline Messages
- Retained Message
- Last Will Message
- TCP/SSL Connection
- MQTT Over WebSocket(SSL)
- HTTP Publish API
- STOMP protocol
- STOMP over SockJS
- $SYS/# Topics

- ClientID Authentication
- IpAddress Authentication
- Username and Password Authentication
- Access control based on IpAddress, ClientID, Username
- Authentication with LDAP, Redis, MySQL, PostgreSQL and HTTP API
- Cluster brokers on several servers
- Bridge brokers locally or remotely
- mosquitto, RSMB bridge
- Extensible architecture with Hooks, Modules and Plugins
- Passed eclipse paho interoperability tests

1.3 Quick Start

1.3.1 Download and Install

The emqttd broker is cross-platform, which could be deployed on Linux, FreeBSD, Mac, Windows and even Raspberry Pi.

Download binary package from: http://emqtt.io/downloads.

Installing on Mac, for example:

```
unzip emqttd-macosx-1.1-beta-20160601.zip && cd emqttd

# Start emqttd
./bin/emqttd start

# Check Status
./bin/emqttd_ctl status

# Stop emqttd
./bin/emqttd stop
```

1.3.2 Installing from Source

Note: emqttd broker requires Erlang R18+ to build since 1.1 release.

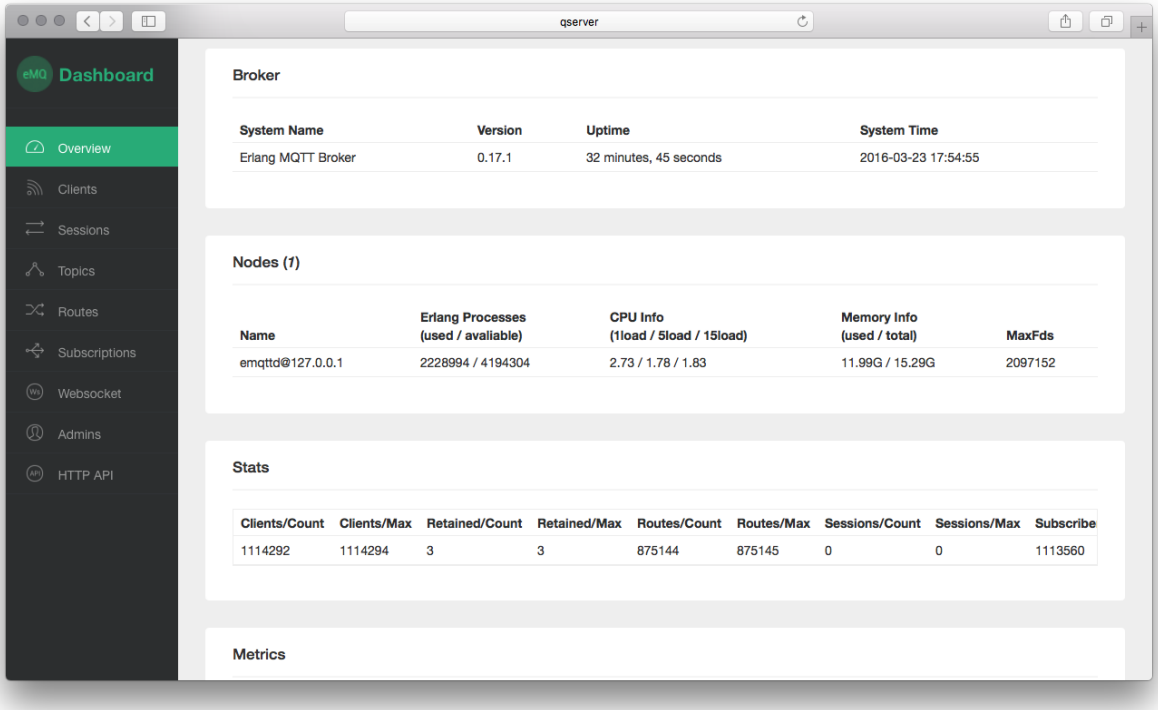
```
git clone https://github.com/emqtt/emqttd.git

cd emqttd && make && make dist

cd rel/emqttd && ./bin/emqttd console
```

1.4 Web Dashboard

A Web Dashboard will be loaded when the emqttd broker is started successfully.

The Dashboard helps check running status of the broker, monitor statistics and metrics of MQTT packets, query clients, sessions, topics and subscriptions.

Default Address	http://localhost:18083
Default User	admin
Default Password	public



1.5 Modules and Plugins

The Authentication and Authorization(ACL) are usually implemented by a Module or Plugin.

1.5.1 Modules

emqttd_auth_clientid	Authentication with ClientId
emqttd_auth_username	Authentication with Username and Password
emqttd_auth_ldap	Authentication with LDAP
emqttd_mod_presence	Publish presence message to $SYS topics when client connected or disconnected
emqttd_mod_subscription	Subscribe topics automatically when client connected
emqttd_mod_rewrite	Topics rewrite like HTTP rewrite module

Configure the 'auth', 'module' paragraph in 'etc/emqttd.config' to enable a module.

Enable 'emqttd_auth_username' module:

```
{access, [
    %% Authetication. Anonymous Default
    {auth, [
        %% Authentication with username, password
        {username, []},

        ...
```

Enable 'emqttd_mod_presence' module:

```
{modules, [
    %% Client presence management module.
    %% Publish messages when client connected or disconnected
    {presence, [{qos, 0}]}
```

1.5.2 Plugins

A plugin is an Erlang application to extend the emqttd broker.

emqttd_plugin_template	Plugin template and demo
emqttd_dashboard	Web Dashboard
emqttd_auth_http	Authentication/ACL with HTTP API
emqttd_plugin_mysql	Authentication with MySQL
emqttd_plugin_pgsql	Authentication with PostgreSQL
emqttd_plugin_redis	Authentication with Redis
emqttd_plugin_mongo	Authentication with MongoDB
emqttd_stomp	STOMP Protocol Plugin
emqttd_sockjs	SockJS(Stomp) Plugin
emqttd_recon	Recon Plugin

A plugin could be enabled by 'bin/emqttd_ctl plugins load' command.

For example, enable 'emqttd_plugin_pgsql' plugin:

```
./bin/emqttd_ctl plugins load emqttd_plugin_pgsql
```

1.6 One Million Connections

Latest release of emqttd broker is scaling to 1.3 million MQTT connections on a 12 Core, 32G CentOS server.

Note: The emqttd broker only allows 512 concurrent connections by default, for 'ulimit -n' limit is 1024 on most platform.

We need tune the OS Kernel, TCP Stack, Erlang VM and emqttd broker for one million connections benchmark.

1.6.1 Linux Kernel Parameters

```
# 2M:
sysctl -w fs.file-max=2097152
sysctl -w fs.nr_open=2097152
echo 2097152 > /proc/sys/fs/nr_open
```

```
# 1M:
ulimit -n 1048576
```

1.6.2 TCP Stack Parameters

```
# backlog
sysctl -w net.core.somaxconn=65536
```

1.6.3 Erlang VM

emqttd/etc/vm.args:

```
## max process numbers
+P 2097152

## Sets the maximum number of simultaneously existing ports for this system
+Q 1048576

## Increase number of concurrent ports/sockets
-env ERL_MAX_PORTS 1048576

-env ERTS_MAX_PORTS 1048576
```

1.6.4 emqttd broker

emqttd/etc/emqttd.config:

```
{mqtt, 1883, [
    %% Size of acceptor pool
    {acceptors, 64},

    %% Maximum number of concurrent clients
    {max_clients, 1000000},

    %% Socket Access Control
    {access, [{allow, all}]},

    %% Connection Options
    {connopts, [
        %% Rate Limit. Format is 'burst, rate', Unit is KB/Sec
        %% {rate_limit, "100,10"} %% 100K burst, 10K rate
    ]},
    ...
```

1.6.5 Test Client

```
sysctl -w net.ipv4.ip_local_port_range="500 65535"
echo 1000000 > /proc/sys/fs/nr_open
ulimit -n 100000
```

1.7 MQTT Client Libraries

GitHub: https://github.com/emqtt

emqttc	Erlang MQTT Client
emqtt_benchmark	MQTT benchmark Tool
CocoaMQTT	Swift MQTT Client
QMQTT	QT MQTT Client

CHAPTER 2

Installation

The emqttd broker is cross-platform, which could be deployed on Linux, FreeBSD, Mac, Windows and even Raspberry Pi.

Note: Linux, FreeBSD Recommended.

2.1 Download Packages

Download binary packages from: http://emqtt.io/downloads

Debian	http://emqtt.io/downloads/latest/debian
Ubuntu	http://emqtt.io/downloads/latest/ubuntu
CentOS	http://emqtt.io/downloads/latest/centos
FreeBSD	http://emqtt.io/downloads/latest/freebsd
Mac OS X	http://emqtt.io/downloads/latest/macosx
Windows	http://emqtt.io/downloads/latest/windows

The package name consists of platform, version and release time.

For example: emqttd-centos64-1.1-beta-20160601.zip

2.2 Installing on Linux

Download CentOS Package from: http://emqtt.io/downloads/latest/centos, and then unzip:

```
unzip emqttd-centos64-1.1-beta-20160601.zip
```

Start the broker in console mode:

```
cd emqttd && ./bin/emqttd console
```

If the broker is started successfully, console will print:

```
starting emqttd on node 'emqttd@127.0.0.1'
emqttd ctl is starting...[done]
emqttd trace is starting...[done]
emqttd pubsub is starting...[done]
emqttd stats is starting...[done]
emqttd metrics is starting...[done]
```

```
emqttd retainer is starting...[done]
emqttd pooler is starting...[done]
emqttd client manager is starting...[done]
emqttd session manager is starting...[done]
emqttd session supervisor is starting...[done]
emqttd broker is starting...[done]
emqttd alarm is starting...[done]
emqttd mod supervisor is starting...[done]
emqttd bridge supervisor is starting...[done]
emqttd access control is starting...[done]
emqttd system monitor is starting...[done]
http listen on 0.0.0.0:18083 with 4 acceptors.
mqtt listen on 0.0.0.0:1883 with 16 acceptors.
mqtts listen on 0.0.0.0:8883 with 4 acceptors.
http listen on 0.0.0.0:8083 with 4 acceptors.
Erlang MQTT Broker 1.1 is running now
Eshell V6.4  (abort with ^G)
(emqttd@127.0.0.1)1>
```

CTRL+C to close the console and stop the broker.

Start the broker in daemon mode:

```
./bin/emqttd start
```

The boot logs in log/emqttd_sasl.log file.

Check the running status of the broker:

```
$ ./bin/emqttd_ctl status
Node 'emqttd@127.0.0.1' is started
emqttd 1.1 is running
```

Or check the status by URL:

```
http://localhost:8083/status
```

Stop the broker:

```
./bin/emqttd stop
```

2.3 Installing on FreeBSD

Download FreeBSD Package from: http://emqtt.io/downloads/latest/freebsd

The installing process is same to Linux.

2.4 Installing on Mac OS X

We could install the broker on Mac OS X to develop and debug MQTT applications.

Download Mac Package from: http://emqtt.io/downloads/latest/macosx

Configure 'lager' log level in 'etc/emqttd.config', all MQTT messages recevied/sent will be printed on console:

```
{lager, [
    ...
    {handlers, [
        {lager_console_backend, info},
        ...
    ]}
]},
```

The install and boot process on Mac are same to Linux.

2.5 Installing on Windows

Download Package from: http://emqtt.io/downloads/latest/windows.

Unzip the package to install folder. Open the command line window and 'cd' to the folder.

Start the broker in console mode:

```
.\bin\emqttd console
```

If the broker started successfully, a Erlang console window will popup.

Close the console window and stop the emqttd broker. Prepare to register emqttd as window service.

Install emqttd serivce:

```
.\bin\emqttd install
```

Start emqttd serivce:

```
.\bin\emqttd start
```

Stop emqttd serivce:

```
.\bin\emqttd stop
```

Uninstall emqttd service:

```
.\bin\emqttd uninstall
```

Warning: './bin/emqttd_ctl' command line cannot work on Windows.

2.6 Installing From Source

The emqttd broker requires Erlang/OTP R17+ and git client to build:

Install Erlang: http://www.erlang.org/

Install Git Client: http://www.git-scm.com/

Could use apt-get on Ubuntu, yum on CentOS/RedHat and brew on Mac to install Erlang and Git.

When all dependencies are ready, clone the emqttd project from github.com and build:

```
git clone https://github.com/emqtt/emqttd.git

cd emqttd

make && make dist
```

The binary package output in folder:

```
rel/emqttd
```

2.7 TCP Ports Used

1883	MQTT Port
8883	MQTT Over SSL Port
8083	MQTT(WebSocket), HTTP API Port
18083	Dashboard Port

The TCP ports used can be configured in etc/emqttd.config:

```
{listeners, [
    {mqtt, 1883, [
        ...
    ]},

    {mqtts, 8883, [
        ...
    ]},
    %% HTTP and WebSocket Listener
    {http, 8083, [
        ...
    ]}
]},
```

The 18083 port is used by Web Dashboard of the broker. Default login: admin, Password: public

2.8 Quick Setup

Two main configuration files of the emqttd broker:

etc/vm.args	Erlang VM Arguments
etc/emqttd.config	emqttd broker Config

Two important parameters in etc/vm.args:

+P	Max number of Erlang proccesses. A MQTT client consumes two proccesses. The value should be larger than max_clients * 2
+Q	Max number of Erlang Ports. A MQTT client consumes one port. The value should be larger than max_clients.

Note: +Q > maximum number of allowed concurrent clients +P > maximum number of allowed concurrent clients * 2

The maximum number of allowed MQTT clients:

```
{listeners, [
    {mqtt, 1883, [
        %% TCP Acceptor Pool
        {acceptors, 16},

        %% Maximum number of concurrent MQTT clients
        {max_clients, 8192},

        ...

    ]},
```

2.9 /etc/init.d/emqttd

```
#!/bin/sh
#
# emqttd        Startup script for emqttd.
#
# chkconfig: 2345 90 10
# description: emqttd is mqtt broker.

# source function library
. /etc/rc.d/init.d/functions

# export HOME=/root

start() {
    echo "starting emqttd..."
    cd /opt/emqttd && ./bin/emqttd start
}

stop() {
    echo "stopping emqttd..."
    cd /opt/emqttd && ./bin/emqttd stop
}

restart() {
    stop
    start
}

case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart)
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop}"
        RETVAL=2
esac
```

chkconfig:

```
chmod +x /etc/init.d/emqttd
chkconfig --add emqttd
chkconfig --list
```

boot test:

```
service emqttd start
```

Note: ## erlexec: HOME must be set uncomment '# export HOME=/root' if "HOME must be set" error.

CHAPTER 3

Configuration

Configuration files of the broker are under 'etc/' folder, including:

File	Description
etc/vm.args	Erlang VM Arguments
etc/emqttd.config	emqttd broker Config
etc/acl.config	ACL Config
etc/clients.config	ClientId Authentication
etc/rewrite.config	Rewrite Rules
etc/ssl/*	SSL certificate and key files

3.1 etc/vm.args

Configure and Optimize Erlang VM:

```
##-------------------------------------------------------------------------
## Name of the node: Name@Host
##-------------------------------------------------------------------------
-name emqttd@127.0.0.1

# or
#-name emqttd@localhost.

## Cookie for distributed erlang
-setcookie emqttdsecretcookie

##-------------------------------------------------------------------------
## Flags
##-------------------------------------------------------------------------

## Heartbeat management; auto-restarts VM if it dies or becomes unresponsive
## (Disabled by default..use with caution!)
##-heart
-smp true

## Enable kernel poll and a few async threads
+K true

## 12 threads/core.
+A 48
```

```
## max process numbers
+P 8192

## Sets the maximum number of simultaneously existing ports for this system
+Q 8192

## max atom number
## +t

## Set the distribution buffer busy limit (dist_buf_busy_limit) in kilobytes.
## Valid range is 1-2097151. Default is 1024.
## +zdbbl 8192

## CPU Schedulers
## +sbt db

##------------------------------------------------------------------------
## Env
##------------------------------------------------------------------------

## Increase number of concurrent ports/sockets, deprecated in R17
-env ERL_MAX_PORTS 8192

-env ERTS_MAX_PORTS 8192

-env ERL_MAX_ETS_TABLES 1024

## Tweak GC to run more often
-env ERL_FULLSWEEP_AFTER 1000
```

The two most important parameters in etc/vm.args:

+P	Max number of Erlang proccesses. A MQTT client consumes two proccesses. The value should be larger than max_clients * 2
+Q	Max number of Erlang Ports. A MQTT client consumes one port. The value should be larger than max_clients.

The name and cookie of Erlang Node should be configured when clustering:

```
-name emqttd@host_or_ip

## Cookie for distributed erlang
-setcookie emqttdsecretcookie
```

3.2 etc/emqttd.config

This is the main emqttd broker configuration file.

3.2.1 File Syntax

The file use the standard Erlang config syntax and consists of a list of erlang applications and their environments.

```
[{kernel, [
    {start_timer, true},
    {start_pg2, true}
```

```
 ]},
 {sasl, [
    {sasl_error_logger, {file, "log/emqttd_sasl.log"}}
 ]},

 ...

 {emqttd, [
    ...
 ]}
].
```

The file adopts Erlang Term Syntax:

1. [ ]: List, seperated by comma
2. { }: Tuple, Usually {Env, Value}
3. % : comment

3.2.2 Log Level and File

Logger of emqttd broker is implemented by 'lager' application:

```
{lager, [
  ...
]},
```

Configure log handlers:

```
{handlers, [
    {lager_console_backend, info},

    {lager_file_backend, [
        {formatter_config, [time, " ", pid, " [",severity,"] ", message, "\n"]},
        {file, "log/emqttd_info.log"},
        {level, info},
        {size, 104857600},
        {date, "$D0"},
        {count, 30}
    ]},

    {lager_file_backend, [
        {formatter_config, [time, " ", pid, " [",severity,"] ", message, "\n"]},
        {file, "log/emqttd_error.log"},
        {level, error},
        {size, 104857600},
        {date, "$D0"},
        {count, 30}
    ]}
]}
```

3.2.3 emqttd Application

The MQTT broker is implemented by erlang 'emqttd' application:

```
{emqttd, [
   %% Authentication and Authorization
   {access, [
       ...
   ]},
   %% MQTT Protocol Options
   {mqtt, [
       ...
   ]},
   %% Broker Options
   {broker, [
       ...
   ]},
   %% Modules
   {modules, [
       ...
   ]},
   %% Plugins
   {plugins, [
       ...
   ]},

   %% Listeners
   {listeners, [
       ...
   ]},

   %% Erlang System Monitor
   {sysmon, [
   ]}
]}
```

3.2.4 Pluggable Authentication

The emqttd broker supports pluggable authentication mechanism with a list of modules and plugins.

The broker provides Username, ClientId, LDAP and anonymous authentication modules by default:

```
%% Authetication. Anonymous Default
{auth, [
    %% Authentication with username, password
    %% Add users: ./bin/emqttd_ctl users add Username Password
    %% {username, [{"test", "public"}]},

    %% Authentication with clientid
    % {clientid, [{password, no}, {file, "etc/clients.config"}]},

    %% Authentication with LDAP
    % {ldap, [
    %    {servers, ["localhost"]},
    %    {port, 389},
    %    {timeout, 30},
    %    {user_dn, "uid=$u,ou=People,dc=example,dc=com"},
    %    {ssl, fasle},
    %    {sslopts, [
    %        {"certfile", "ssl.crt"},
    %        {"keyfile", "ssl.key"}]}
```

```
    % ]},

    %% Allow all
    {anonymous, []}
]},
```

The modules enabled at the same time compose an authentication chain:

```
            ----------------           ----------------           -------------
Client --> |   Username    | -ignore-> |   ClientID    | -ignore-> | Anonymous |
            ----------------           ----------------           -------------
                   |                          |                         |
                  \|/                        \|/                       \|/
             allow | deny               allow | deny               allow | deny
```

Note: There are also MySQLPostgreSQLRedisMongoDB Authentication Plugins.

Username Authentication

```
{username, [{client1, "passwd1"}, {client2, "passwd2"}]},
```

Two ways to configure users:

1. Configure username and plain password directly:

```
    {username, [{client1, "passwd1"}, {client2, "passwd2"}]},
```

2. Add user by './bin/emqttd_ctl users' command:

```
    $ ./bin/emqttd_ctl users add <Username> <Password>
```

ClientID Authentication

```
{clientid, [{password, no}, {file, "etc/clients.config"}]},
```

Configure ClientIDs in etc/clients.config:

```
testclientid0
testclientid1 127.0.0.1
testclientid2 192.168.0.1/24
```

LDAP Authentication

```
{ldap, [
   {servers, ["localhost"]},
   {port, 389},
   {timeout, 30},
   {user_dn, "uid=$u,ou=People,dc=example,dc=com"},
   {ssl, fasle},
   {sslopts, [
       {"certfile", "ssl.crt"},
       {"keyfile", "ssl.key"}]}
]},
```

Anonymous Authentication

Allow any client to connect to the broker:

```
{anonymous, []}
```

3.2.5 ACL

Enable the default ACL module:

```
{acl, [
    %% Internal ACL module
    {internal,  [{file, "etc/acl.config"}, {nomatch, allow}]}
]}
```

3.2.6 MQTT Packet and ClientID

```
{packet, [

    %% Max ClientId Length Allowed
    {max_clientid_len, 1024},

    %% Max Packet Size Allowed, 64K default
    {max_packet_size,  65536}
]},
```

3.2.7 MQTT Client Idle Timeout

```
{client, [
    %% Socket is connected, but no 'CONNECT' packet received
    {idle_timeout, 10}
]},
```

3.2.8 MQTT Session

```
{session, [
    %% Max number of QoS 1 and 2 messages that can be "in flight" at one time.
    %% 0 means no limit
    {max_inflight, 100},

    %% Retry interval for unacked QoS1/2 messages.
    {unack_retry_interval, 20},

    %% Awaiting PUBREL Timeout
    {await_rel_timeout, 20},

    %% Max Packets that Awaiting PUBREL, 0 means no limit
    {max_awaiting_rel, 0},

    %% Interval of Statistics Collection(seconds)
    {collect_interval, 20},
```

```
    %% Expired after 2 day (unit: minute)
    {expired_after, 2880}

]},
```

Session parameters:

max_inflight	Max number of QoS1/2 messages that can be delivered in the same time
unack_retry_interval	Retry interval for unacked QoS1/2 messages.
await_rel_timeout	Awaiting PUBREL Timeout
max_awaiting_rel	Max number of Packets that Awaiting PUBREL
collect_interval	Interval of Statistics Collection
expired_after	Expired after (unit: minute)

3.2.9 MQTT Message Queue

The message queue of session stores:

1. Offline messages for persistent session.
2. Pending messages for inflight window is full

Queue parameters:

```
{queue, [
    %% simple | priority
    {type, simple},

    %% Topic Priority: 0~255, Default is 0
    %% {priority, [{"topic/1", 10}, {"topic/2", 8}]},

    %% Max queue length. Enqueued messages when persistent client disconnected,
    %% or inflight window is full.
    {max_length, infinity},

    %% Low-water mark of queued messages
    {low_watermark, 0.2},

    %% High-water mark of queued messages
    {high_watermark, 0.6},

    %% Queue Qos0 messages?
    {queue_qos0, true}
]}
```

type	Queue type: simple or priority
priority	Topic priority
max_length	Max Queue size, infinity means no limit
low_watermark	Low watermark
high_watermark	High watermark
queue_qos0	If Qos0 message queued?

3.2.10 Sys Interval of Broker

```
%% System interval of publishing $SYS messages
{sys_interval, 60},
```

3.2.11 Retained messages

```
{retained, [
    %% Expired after seconds, never expired if 0
    {expired_after, 0},

    %% Maximum number of retained messages
    {max_message_num, 100000},

    %% Max Payload Size of retained message
    {max_playload_size, 65536}
]},
```

3.2.12 PubSub and Router

```
{pubsub, [
    %% PubSub Pool
    {pool_size, 8},

    %% Subscription: true | false
    {subscription, true},

    %% Route aging time(seconds)
    {route_aging, 5}
]},
```

3.2.13 Bridge Parameters

```
{bridge, [
    %% Bridge Queue Size
    {max_queue_len, 10000},

    %% Ping Interval of bridge node
    {ping_down_interval, 1}
]}
```

3.2.14 Enable Modules

'presence' module will publish presence message to $SYS topic when a client connected or disconnected:

```
{presence, [{qos, 0}]},
```

'subscription' module forces the client to subscribe some topics when connected to the broker:

```
%% Subscribe topics automatically when client connected
{subscription, [
    %% Subscription from stored table
    stored,

    %% $u will be replaced with username
    {"$Q/username/$u", 1},

    %% $c will be replaced with clientid
```

```
    {"$Q/client/$c", 1}
]}
```

'rewrite' module supports to rewrite the topic path:

```
%% Rewrite rules
{rewrite, [{file, "etc/rewrite.config"}]}
```

3.2.15 Plugins Folder

```
{plugins, [
    %% Plugin App Library Dir
    {plugins_dir, "./plugins"},

    %% File to store loaded plugin names.
    {loaded_file, "./data/loaded_plugins"}
]},
```

3.2.16 TCP Listeners

Configure the TCP listeners for MQTT, MQTT(SSL) and HTTP Protocols.

The most important parameter is 'max_clients' - max concurrent clients allowed.

The TCP Ports occupied by emqttd broker by default:

1883	MQTT Port
8883	MQTT(SSL) Port
8083	MQTT(WebSocket), HTTP API Port

```
{listeners, [

    {mqtt, 1883, [
        %% Size of acceptor pool
        {acceptors, 16},

        %% Maximum number of concurrent clients
        {max_clients, 8192},

        %% Socket Access Control
        {access, [{allow, all}]},

        %% Connection Options
        {connopts, [
            %% Rate Limit. Format is 'burst, rate', Unit is KB/Sec
            %% {rate_limit, "100,10"} %% 100K burst, 10K rate
        ]},

        %% Socket Options
        {sockopts, [
            %Set buffer if hight thoughtput
            %{recbuf, 4096},
            %{sndbuf, 4096},
            %{buffer, 4096},
            %{nodelay, true},
            {backlog, 1024}
```

```
        ]}
    ]},

    {mqtts, 8883, [
        %% Size of acceptor pool
        {acceptors, 4},

        %% Maximum number of concurrent clients
        {max_clients, 512},

        %% Socket Access Control
        {access, [{allow, all}]},

        %% SSL certificate and key files
        {ssl, [{certfile, "etc/ssl/ssl.crt"},
               {keyfile,  "etc/ssl/ssl.key"}]},

        %% Socket Options
        {sockopts, [
            {backlog, 1024}
            %{buffer, 4096},
        ]}
    ]},
    %% WebSocket over HTTPS Listener
    %% {https, 8083, [
    %%  %% Size of acceptor pool
    %%  {acceptors, 4},
    %%  %% Maximum number of concurrent clients
    %%  {max_clients, 512},
    %%  %% Socket Access Control
    %%  {access, [{allow, all}]},
    %%  %% SSL certificate and key files
    %%  {ssl, [{certfile, "etc/ssl/ssl.crt"},
    %%         {keyfile,  "etc/ssl/ssl.key"}]},
    %%  %% Socket Options
    %%  {sockopts, [
    %%      %{buffer, 4096},
    %%      {backlog, 1024}
    %%  ]}
    %%]},

    %% HTTP and WebSocket Listener
    {http, 8083, [
        %% Size of acceptor pool
        {acceptors, 4},
        %% Maximum number of concurrent clients
        {max_clients, 64},
        %% Socket Access Control
        {access, [{allow, all}]},
        %% Socket Options
        {sockopts, [
            {backlog, 1024}
            %{buffer, 4096},
        ]}
    ]}
]},
```

Listener Parameters:

acceptors	TCP Acceptor Pool
max_clients	Maximum number of concurrent TCP connections allowed
access	Access Control by IP, for example: [{allow, "192.168.1.0/24"}]
connopts	Rate Limit Control, for example: {rate_limit, "100,10"}
sockopts	TCP Socket parameters

3.3 etc/acl.config

The 'etc/acl.config' is the default ACL config for emqttd broker. The rules by default:

```
%% Allow 'dashboard' to subscribe '$SYS/#'
{allow, {user, "dashboard"}, subscribe, ["$SYS/#"]}.

%% Allow clients from localhost to subscribe any topics
{allow, {ipaddr, "127.0.0.1"}, pubsub, ["$SYS/#", "#"]}.

%% Deny clients to subscribe '$SYS#' and '#'
{deny, all, subscribe, ["$SYS/#", {eq, "#"}]}.

%% Allow all by default
{allow, all}.
```

An ACL rule is an Erlang tuple. The Access control module of emqttd broker matches the rule one by one from top to bottom:

```
          ---------               ---------               ---------
Client -> | Rule1 | --nomatch--> | Rule2 | --nomatch--> | Rule3 | --> Default
          ---------               ---------               ---------
              |                       |                       |
            match                   match                   match
              \|/                     \|/                     \|/
        allow | deny            allow | deny            allow | deny
```

3.4 etc/clients.config

Enable ClientId Authentication in 'etc/emqttd.config':

```
{auth, [
    %% Authentication with clientid
    {clientid, [{password, no}, {file, "etc/clients.config"}]}
]},
```

Configure all allowed ClientIDs, IP Addresses in etc/clients.config:

```
testclientid0
testclientid1 127.0.0.1
testclientid2 192.168.0.1/24
```

3.5 etc/rewrite.config

The Rewrite Rules for emqttd_mod_rewrite:

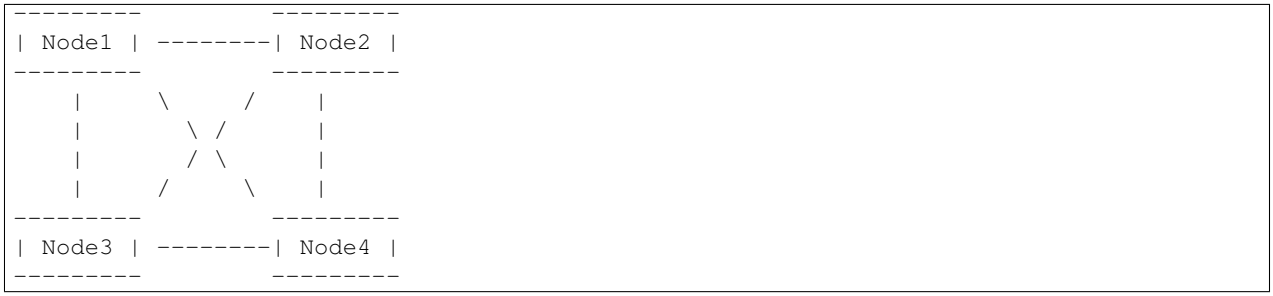
```
{topic, "x/#", [
    {rewrite, "^x/y/(.+)$", "z/y/$1"},
    {rewrite, "^x/(.+)$", "y/$1"}
]}.

{topic, "y/+/z/#", [
    {rewrite, "^y/(.+)/z/(.+)$", "y/z/$2"}
]}.
```

CHAPTER 4

Clustering

4.1 Distributed Erlang/OTP

Erlang/OTP is a concurrent, fault-tolerant, distributed programming platform. A distributed Erlang/OTP system consists of a number of Erlang runtime systems called 'node'. Nodes connect to each other with TCP/IP sockets and communite by Message Passing.



4.1.1 Node

An erlang runtime system called 'node' is identified by a unique name like email addreass. Erlang nodes communicate with each other by the name.

Suppose we start four Erlang nodes on localhost:

```
erl -name node1@127.0.0.1
erl -name node2@127.0.0.1
erl -name node3@127.0.0.1
erl -name node4@127.0.0.1
```

connect all the nodes:

```
(node1@127.0.0.1)1> net_kernel:connect_node('node2@127.0.0.1').
true
(node1@127.0.0.1)2> net_kernel:connect_node('node3@127.0.0.1').
true
(node1@127.0.0.1)3> net_kernel:connect_node('node4@127.0.0.1').
true
(node1@127.0.0.1)4> nodes().
['node2@127.0.0.1','node3@127.0.0.1','node4@127.0.0.1']
```

4.1.2 epmd

epmd(Erlang Port Mapper Daemon) is a daemon service that is responsible for mapping node names to machine addresses(TCP sockets). The daemon is started automatically on every host where an Erlang node started.

```
(node1@127.0.0.1)6> net_adm:names().
{ok,[{"node1",62740},
     {"node2",62746},
     {"node3",62877},
     {"node4",62895}]}
```

4.1.3 Cookie

Erlang nodes authenticate each other by a magic cookie when communicating. The cookie could be configured by:

```
1. $HOME/.erlang.cookie

2. erl -setcookie <Cookie>
```

Note: Content of this chapter is from: http://erlang.org/doc/reference_manual/distributed.html

4.2 Cluster Design

The cluster architecture of emqttd broker is based on distrubuted Erlang/OTP and Mnesia database.

The cluster design could be summarized by the following two rules:

1. When a MQTT client SUBSCRIBE a Topic on a node, the node will tell all the other nodes in the cluster: I subscribed a Topic.
2. When a MQTT Client PUBLISH a message to a node, the node will lookup the Topic table and forward the message to nodes that subscribed the Topic.

Finally there will be a global route table(Topic -> Node) that replicated to all nodes in the cluster:

```
topic1 -> node1, node2
topic2 -> node3
topic3 -> node2, node4
```

4.2.1 Topic Trie and Route Table

Every node in the cluster will store a topic trie and route table in mnesia database.

Suppose that we create subscriptions:

Client	Node	Topics
client1	node1	t/+/x, t/+/y
client2	node2	t/#
client3	node3	t/+/x, t/a

Finally the topic trie and route table in the cluster:

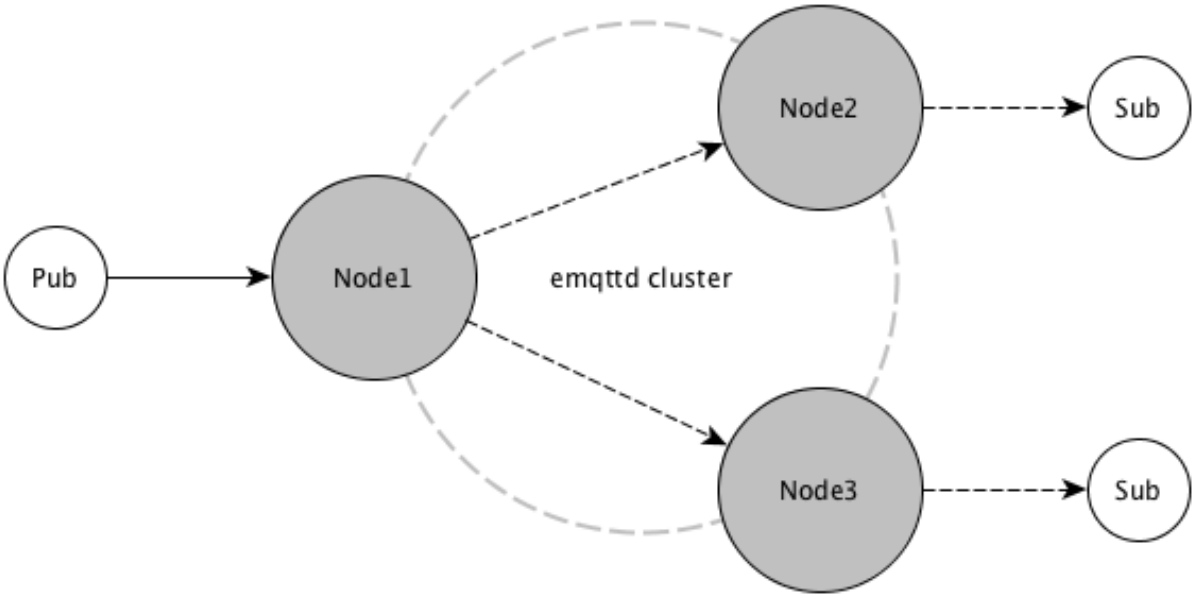
```
-------------------------
|           t           |
|          / \          |
|         +   #         |
|       /  \            |
|     x       y         |
-------------------------
| t/+/x -> node1, node3 |
| t/+/y -> node1        |
| t/#   -> node2        |
| t/a   -> node3        |
-------------------------
```

4.2.2 Message Route and Deliver

The brokers in the cluster route messages by topic trie and route table, deliver messages to MQTT clients by subscriptions. Subscriptions are mapping from topic to subscribers, are stored only in the local node, will not be replicated to other nodes.

Suppose client1 PUBLISH a message to the topic 't/a', the message Route and Deliver process:

```
title: Message Route and Deliver

client1->node1: Publish[t/a]
node1-->node2: Route[t/#]
node1-->node3: Route[t/a]
node2-->client2: Deliver[t/#]
node3-->client3: Deliver[t/a]
```



4.3 Cluster Setup

Suppose we deploy two nodes cluster on s1.emqtt.io, s2.emqtt.io:

Node	Host(FQDN)	IP and Port
emqttd@s1.emqtt.io or emqttd@192.168.0.10	s1.emqtt.io	192.168.0.10:1883
emqttd@s2.emqtt.io or emqttd@192.168.0.20	s2.emqtt.io	192.168.0.20:1883

Warning: The node name is Name@Host, where Host is IP address or the fully qualified host name.

4.3.1 emqttd@s1.emqtt.io setting

emqttd/etc/vm.args:

```
-name emqttd@s1.emqtt.io

or

-name emqttd@192.168.0.10
```

Warning: The name cannot be changed after node joined the cluster.

4.3.2 emqttd@s2.emqtt.io setting

emqttd/etc/vm.args:

```
-name emqttd@s2.emqtt.io

or

-name emqttd@192.168.0.20
```

4.3.3 Join the cluster

Start the two broker nodes, and 'cluster join ' on emqttd@s2.emqtt.io:

```
$ ./bin/emqttd_ctl cluster join emqttd@s1.emqtt.io

Join the cluster successfully.
Cluster status: [{running_nodes,['emqttd@s1.emqtt.io','emqttd@s2.emqtt.io']}]
```

Or 'cluster join' on emqttd@s1.emqtt.io:

```
$ ./bin/emqttd_ctl cluster join emqttd@s2.emqtt.io

Join the cluster successfully.
Cluster status: [{running_nodes,['emqttd@s1.emqtt.io','emqttd@s2.emqtt.io']}]
```

Query the cluster status:

```
$ ./bin/emqttd_ctl cluster status

Cluster status: [{running_nodes,['emqttd@s1.emqtt.io','emqttd@s2.emqtt.io']}]
```

4.3.4 Leave the cluster

Two ways to leave the cluster:

1. leave: this node leaves the cluster
2. remove: remove other nodes from the cluster

emqttd@s2.emqtt.io node tries to leave the cluster:

```
$ ./bin/emqttd_ctl cluster leave
```

Or remove emqttd@s2.emqtt.io node from the cluster on emqttd@s1.emqtt.io:

```
$ ./bin/emqttd_ctl cluster remove emqttd@s2.emqtt.io
```

4.4 Session across Nodes

The persistent MQTT sessions (clean session = false) are across nodes in the cluster.

If a persistent MQTT client connected to node1 first, then disconnected and connects to node2, the MQTT connection and session will be located on different nodes:

```
                                 node1
                               -----------
                            |-->| session |
                            |   -----------
             node2          |
          ---------------   |
client-->| connection |<--|
          ---------------
```

4.5 The Firewall

If there is a firewall between clustered nodes, the cluster requires to open 4369 port used by epmd daemon, and a port segment for nodes' communication.

Configure the port segment in etc/emqttd.config, for example:

```
[{kernel, [
    ...
    {inet_dist_listen_min, 20000},
    {inet_dist_listen_max, 21000}
 ]},
 ...
```

4.6 Network Partitions

The emqttd 1.0 cluster requires reliable network to avoid network partitions. The cluster will not recover from a network partition automatically.

If a network partition occures, there will be critical logs in log/emqttd_error.log:

```
Mnesia inconsistent_database event: running_partitioned_network, emqttd@host
```

To recover from a network partition, you have to stop the nodes in a partition, clean the 'data/mneisa' of these nodes and reboot to join the cluster again.

4.7 Consistent Hash and DHT

Consistent Hash and DHT are popular in the design of NoSQL databases. Cluster of emqttd broker could support 10 million size of global routing table now. We could use the Consistent Hash or DHT to partition the routing table, and evolve the cluster to larger size.

CHAPTER 5

Bridge

5.1 emqttd Bridge

Two or more emqttd brokers could be bridged together. Bridges forward MQTT messages from one broker node to another:

```
                ---------                          ---------                          ---------
Publisher --> | node1 | --Bridge Forward--> | node2 | --Bridge Forward--> | node3 | --> Subscriber
                ---------                          ---------                          ---------
```

5.1.1 Configure Bridge

Suppose that we create two emqttd brokers on localhost:

Name	Node	MQTT Port
emqttd1	emqttd1@127.0.0.1	1883
emqttd2	emqttd2@127.0.0.1	2883

Create a bridge that forwards all the 'sensor/#' messages from emqttd1 to emqttd2.

1. Start Brokers

```
cd emqttd1/ && ./bin/emqttd start
cd emqttd2/ && ./bin/emqttd start
```

2. Create bridge: emqttd1–sensor/#–>emqttd2

```
$ cd emqttd1 && ./bin/emqttd_ctl bridges start emqttd2@127.0.0.1 sensor/#

bridge is started.

$ ./bin/emqttd_ctl bridges list

bridge: emqttd1@127.0.0.1--sensor/#-->emqttd2@127.0.0.1
```

3. Test the bridge

```
#emqttd2
mosquitto_sub -t sensor/# -p 2883 -d

#emqttd1
mosquitto_pub -t sensor/1/temperature -m "37.5" -d
```

4. Delete the bridge

```
./bin/emqttd_ctl bridges stop emqttd2@127.0.0.1 sensor/#
```

5.2 emqttd Bridge CLI

```
#query bridges
./bin/emqttd_ctl bridges list

#start bridge
./bin/emqttd_ctl bridges start <Node> <Topic>

#start bridge with options
./bin/emqttd_ctl bridges start <Node> <Topic> <Options>

#stop bridge
./bin/emqttd_ctl bridges stop <Node> <Topic>
```

5.3 mosquitto Bridge

Bridge mosquitto to emqttd broker:

5.3.1 mosquitto.conf

Suppose that we start an emqttd broker on localost:2883, and mosquitto on localhost:1883.

A bridge configured in mosquitto.conf:

```
connection emqttd
address 127.0.0.1:2883
topic sensor/# out 2

# Set the version of the MQTT protocol to use with for this bridge. Can be one
# of mqttv31 or mqttv311. Defaults to mqttv31.
bridge_protocol_version mqttv311
```

5.4 rsmb Bridge

Bridge RSMB to emqttd broker, same settings as mosquitto.

broker.cfg:

```
connection emqttd
addresses 127.0.0.1:2883
topic sensor/#
```

CHAPTER 6

User Guide

6.1 Authentication

The emqttd broker supports to authenticate MQTT clients with ClientID, Username/Password, IpAddress and even HTTP Cookies.

The authentication is provided by a list of extended modules, or MySQL, PostgreSQL and Redis Plugins.

Enable an authentication module in etc/emqttd.config:

```
%% Authentication and Authorization
{access, [
    %% Authetication. Anonymous Default
    {auth, [
        %% Authentication with username, password
        %{username, []},

        %% Authentication with clientid
        %{clientid, [{password, no}, {file, "etc/clients.config"}]},

        %% Authentication with LDAP
        % {ldap, [
        %    {servers, ["localhost"]},
        %    {port, 389},
        %    {timeout, 30},
        %    {user_dn, "uid=$u,ou=People,dc=example,dc=com"},
        %    {ssl, fasle},
        %    {sslopts, [
        %        {"certfile", "ssl.crt"},
        %        {"keyfile", "ssl.key"}]}
        % ]},

        %% Allow all
        {anonymous, []}
    ]},
```

Note: "%" comments the line.

If we enable several modules at the same time, the authentication process:

```
           ----------------          -----------------          -------------
Client --> |   Username    | -ignore-> |   ClientID    | -ignore-> | Anonymous |
```

```
        ----------------          ----------------          -------------
               |                         |                         |
              \|/                       \|/                       \|/
         allow | deny              allow | deny              allow | deny
```

The authentication plugins developed by emqttd:

Plugin	Description
emqttd_plugin_mysql	MySQL Auth/ACL Plugin
emqttd_plugin_pgsql	PostgreSQL Auth/ACL Plugin
emqttd_plugin_redis	Redis Auth/ACL Plugin

Note: If we load an authentication plugin, the authentication modules will be disabled.

6.1.1 Username

Authenticate MQTT client with Username/Password:

```
{username, [{client1, "passwd1"}, {client1, "passwd2"}]},
```

Two ways to add users:

1. Configure username and plain password directly:

```
    {username, [{client1, "passwd1"}, {client1, "passwd2"}]},
```

2. Add user by './bin/emqttd_ctl users' command:

```
    $ ./bin/emqttd_ctl users add <Username> <Password>
```

6.1.2 ClientId

```
{clientid, [{password, no}, {file, "etc/clients.config"}]},
```

Configure ClientIDs in etc/clients.config:

```
testclientid0
testclientid1 127.0.0.1
testclientid2 192.168.0.1/24
```

6.1.3 LDAP

```
{ldap, [
   {servers, ["localhost"]},
   {port, 389},
   {timeout, 30},
   {user_dn, "uid=$u,ou=People,dc=example,dc=com"},
   {ssl, fasle},
   {sslopts, [
       {"certfile", "ssl.crt"},
       {"keyfile", "ssl.key"}]}
]},
```

6.1.4 Anonymous

Allow any client to connect to the broker:

```
{anonymous, []}
```

6.1.5 MySQL

Authenticate against MySQL database. Support we create a mqtt_user table:

```
CREATE TABLE `mqtt_user` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `username` varchar(100) DEFAULT NULL,
  `password` varchar(100) DEFAULT NULL,
  `salt` varchar(20) DEFAULT NULL,
  `created` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `mqtt_username` (`username`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Configure the 'authquery' and 'password_hash' in emqttd_plugin_mysql/etc/plugin.config:

```
[

{emqttd_plugin_mysql, [

    ...

    %% select password only
    {authquery, "select password from mqtt_user where username = '%u' limit 1"},

    %% hash algorithm: md5, sha, sha256, pbkdf2?
    {password_hash, sha256},

    ...

]}
].
```

Load the plugin:

```
./bin/emqttd_ctl plugins load emqttd_plugin_mysql
```

6.1.6 PostgreSQL

Authenticate against PostgreSQL database. Create a mqtt_user table:

```
CREATE TABLE mqtt_user (
  id SERIAL primary key,
  username character varying(100),
  password character varying(100),
  salt character varying(40)
);
```

Configure the 'authquery' and 'password_hash' in emqttd_plugin_pgsql/etc/plugin.config:

```
[
  {emqttd_plugin_pgsql, [

    ...

    %% select password only
    {authquery, "select password from mqtt_user where username = '%u' limit 1"},

    %% hash algorithm: md5, sha, sha256, pbkdf2?
    {password_hash, sha256},

    ...

  ]}
].
```

Load the plugin:

```
./bin/emqttd_ctl plugins load emqttd_plugin_pgsql
```

6.1.7 Redis

Authenticate against Redis. MQTT users could be stored in redis HASH, the key is "mqtt_user:<Username>".

Configure 'authcmd' and 'password_hash' in emqttd_plugin_redis/etc/plugin.config:

```
[
  {emqttd_plugin_redis, [

    ...

    %% HMGET mqtt_user:%u password
    {authcmd, ["HGET", "mqtt_user:%u", "password"]},

    %% Password hash algorithm: plain, md5, sha, sha256, pbkdf2?
    {password_hash, sha256},

    ...

  ]}
].
```

Load the plugin:

```
./bin/emqttd_ctl plugins load emqttd_plugin_redis
```

6.2 ACL

The ACL of emqttd broker is responsbile for authorizing MQTT clients to publish/subscribe topics.

The ACL rules define:

```
Allow|Deny Who Publish|Subscribe Topics
```

Access Control Module of emqttd broker will match the rules one by one:

```
          ---------               ---------               ---------
Client -> | Rule1 | --nomatch--> | Rule2 | --nomatch--> | Rule3 | --> Default
          ---------               ---------               ---------
              |                       |                       |
            match                   match                   match
              \|/                     \|/                     \|/
         allow | deny            allow | deny            allow | deny
```

6.2.1 Internal

The default ACL of emqttd broker is implemented by an 'internal' module.

Enable the 'internal' ACL module in etc/emqttd.config:

```
{acl, [
    %% Internal ACL module
    {internal,  [{file, "etc/acl.config"}, {nomatch, allow}]}
]}
```

The ACL rules of 'internal' module are defined in 'etc/acl.config' file:

```
%% Allow 'dashboard' to subscribe '$SYS/#'
{allow, {user, "dashboard"}, subscribe, ["$SYS/#"]}.

%% Allow clients from localhost to subscribe any topics
{allow, {ipaddr, "127.0.0.1"}, pubsub, ["$SYS/#", "#"]}.

%% Deny clients to subscribe '$SYS#' and '#'
{deny, all, subscribe, ["$SYS/#", {eq, "#"}]}.

%% Allow all by default
{allow, all}.
```

6.2.2 MySQL

ACL against MySQL database. The mqtt_acl table and default data:

```
CREATE TABLE `mqtt_acl` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `allow` int(1) DEFAULT NULL COMMENT '0: deny, 1: allow',
  `ipaddr` varchar(60) DEFAULT NULL COMMENT 'IpAddress',
  `username` varchar(100) DEFAULT NULL COMMENT 'Username',
  `clientid` varchar(100) DEFAULT NULL COMMENT 'ClientId',
  `access` int(2) NOT NULL COMMENT '1: subscribe, 2: publish, 3: pubsub',
  `topic` varchar(100) NOT NULL DEFAULT '' COMMENT 'Topic Filter',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO mqtt_acl (id, allow, ipaddr, username, clientid, access, topic)
VALUES
    (1,1,NULL,'$all',NULL,2,'#'),
    (2,0,NULL,'$all',NULL,1,'$SYS/#'),
    (3,0,NULL,'$all',NULL,1,'eq #'),
    (5,1,'127.0.0.1',NULL,NULL,2,'$SYS/#'),
    (6,1,'127.0.0.1',NULL,NULL,2,'#'),
    (7,1,NULL,'dashboard',NULL,1,'$SYS/#');
```

Configure 'aclquery' and 'acl_nomatch' in emqttd_plugin_mysql/etc/plugin.config:

```
[

  {emqttd_plugin_mysql, [

    ...

    %% comment this query, the acl will be disabled
    {aclquery, "select allow, ipaddr, username, clientid, access, topic from mqtt_acl where ipaddr =

    %% If no rules matched, return...
    {acl_nomatch, allow}

  ]}
].
```

6.2.3 PostgreSQL

ACL against PostgreSQL database. The mqtt_acl table and default data:

```
CREATE TABLE mqtt_acl (
  id SERIAL primary key,
  allow integer,
  ipaddr character varying(60),
  username character varying(100),
  clientid character varying(100),
  access  integer,
  topic character varying(100)
);

INSERT INTO mqtt_acl (id, allow, ipaddr, username, clientid, access, topic)
VALUES
    (1,1,NULL,'$all',NULL,2,'#'),
    (2,0,NULL,'$all',NULL,1,'$SYS/#'),
    (3,0,NULL,'$all',NULL,1,'eq #'),
    (5,1,'127.0.0.1',NULL,NULL,2,'$SYS/#'),
    (6,1,'127.0.0.1',NULL,NULL,2,'#'),
    (7,1,NULL,'dashboard',NULL,1,'$SYS/#');
```

Configure 'aclquery' and 'acl_nomatch' in emqttd_plugin_pgsql/etc/plugin.config:

```
[

  {emqttd_plugin_pgsql, [

    ...

    %% Comment this query, the acl will be disabled. Notice: don't edit this query!
    {aclquery, "select allow, ipaddr, username, clientid, access, topic from mqtt_acl
                  where ipaddr = '%a' or username = '%u' or username = '$all' or clientid = '%c'"},

    %% If no rules matched, return...
    {acl_nomatch, allow}

    ...
```

```
  ]}
].
```

6.2.4 Redis

ACL against Redis. We store ACL rules for each MQTT client in a Redis List by defualt. The key is "mqtt_acl:<Username>", the value is a list of "publish <Topic>", "subscribe <Topic>" or "pubsub <Topic>".

Configure 'aclcmd' and 'acl_nomatch' in emqttd_plugin_redis/etc/plugin.config:

```
[
  {emqttd_plugin_redis, [

    ...

    %% SMEMBERS mqtt_acl:%u
    {aclcmd, ["SMEMBERS", "mqtt_acl:%u"]},

    %% If no rules matched, return...
    {acl_nomatch, deny},

    ...

  ]}
].
```

6.3 MQTT Publish/Subscribe

MQTT is a an extremely lightweight publish/subscribe messaging protocol desgined for IoT, M2M and Mobile applications.

Install and start the emqttd broker, and then any MQTT client could connect to the broker, subscribe topics and publish messages.

MQTT Client Libraries: https://github.com/mqtt/mqtt.github.io/wiki/libraries

For example, we use mosquitto_sub/pub commands:

```
mosquitto_sub -t topic -q 2
mosquitto_pub -t topic -q 1 -m "Hello, MQTT!"
```

MQTT V3.1.1 Protocol Specification: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

MQTT Listener of emqttd broker is configured in etc/emqttd.config:

```
{mqtt, 1883, [
    %% Size of acceptor pool
    {acceptors, 16},

    %% Maximum number of concurrent clients
    {max_clients, 512},

    %% Socket Access Control
    {access, [{allow, all}]},

    %% Connection Options
    {connopts, [
        %% Rate Limit. Format is 'burst, rate', Unit is KB/Sec
        %% {rate_limit, "100,10"} %% 100K burst, 10K rate
    ]},

    %% Socket Options
    {sockopts, [
        %Set buffer if hight thoughtput
        %{recbuf, 4096},
        %{sndbuf, 4096},
        %{buffer, 4096},
        %{nodelay, true},
        {backlog, 512}
    ]}
]},
```

MQTT(SSL) Listener, Default Port is 8883:

```
{mqtts, 8883, [
    %% Size of acceptor pool
    {acceptors, 4},

    %% Maximum number of concurrent clients
    {max_clients, 512},

    %% Socket Access Control
    {access, [{allow, all}]},

    %% SSL certificate and key files
    {ssl, [{certfile, "etc/ssl/ssl.crt"},
           {keyfile,  "etc/ssl/ssl.key"}]},

    %% Socket Options
    {sockopts, [
        {backlog, 1024}
        %{buffer, 4096},
    ]}
]},
```

6.4 HTTP Publish API

The emqttd broker provides a HTTP API to help application servers publish messages to MQTT clients.

HTTP API: POST http://host:8083/mqtt/publish

Web servers such as PHP, Java, Python, NodeJS and Ruby on Rails could use HTTP POST to publish MQTT messages to the broker:

```
curl -v --basic -u user:passwd -d "qos=1&retain=0&topic=/a/b/c&message=hello from http..." -k http:/
```

Parameters of the HTTP API:

Name	Description
client	clientid
qos	QoS(0, 1, 2)
retain	Retain(0, 1)
topic	Topic
message	Payload

Note: The API uses HTTP Basic Authentication.

6.5 MQTT Over WebSocket

Web browsers could connect to the emqttd broker directly by MQTT Over WebSocket.

WebSocket URI:	ws(s)://host:8083/mqtt
Sec-WebSocket-Protocol:	'mqttv3.1' or 'mqttv3.1.1'

The Dashboard plugin provides a test page for WebSocket:

```
http://127.0.0.1:18083/websocket.html
```

Listener of WebSocket and HTTP Publish API is configured in etc/emqttd.config:

```
%% HTTP and WebSocket Listener
{http, 8083, [
    %% Size of acceptor pool
    {acceptors, 4},
    %% Maximum number of concurrent clients
    {max_clients, 64},
    %% Socket Access Control
    {access, [{allow, all}]},
    %% Socket Options
    {sockopts, [
        {backlog, 1024}
        %{buffer, 4096},
    ]}
]}
```

6.6 $SYS Topics

The emqttd broker periodically publishes internal status, MQTT statistics, metrics and client online/offline status to $SYS/# topics.

For emqttd broker is clustered, the $SYS topic path is started with:

```
$SYS/brokers/${node}/
```

'${node}' is the erlang node name of emqttd broker. For example:

```
$SYS/brokers/emqttd@127.0.0.1/version

$SYS/brokers/emqttd@host2/uptime
```

Note: The broker only allows clients from localhost to subscribe $SYS topics by default.

Sys Interval of publishing $SYS messages, could be configured in etc/emqttd.config:

```
{broker, [
    %% System interval of publishing broker $SYS messages
    {sys_interval, 60},
```

6.6.1 Broker Version, Uptime and Description

Topic	Description
$SYS/brokers	Broker nodes
$SYS/brokers/${node}/version	Broker Version
$SYS/brokers/${node}/uptime	Broker Uptime
$SYS/brokers/${node}/datetime	Broker DateTime
$SYS/brokers/${node}/sysdescr	Broker Description

6.6.2 Online/Offline Status of MQTT Client

The topic path started with: $SYS/brokers/${node}/clients/

Topic	Payload(JSON)	Description
${clientid}/connected	**{ipaddress: "127.0.0.1", username: "test",** session: false, version: 3, connack: 0, ts: 1432648482}	Publish when a client connected
${clientid}/disconnected	**{reason: "keepalive_timeout",** ts: 1432749431}	Publish when a client disconnected

Properties of 'connected' Payload:

```
ipaddress: "127.0.0.1",
username:  "test",
session:   false,
protocol:  3,
connack:   0,
ts:        1432648482
```

Properties of 'disconnected' Payload:

```
reason: normal,
ts:     1432648486
```

6.6.3 Broker Statistics

Topic path started with: $SYS/brokers/${node}/stats/

Clients

Topic	Description
clients/count	Count of current connected clients
clients/max	Max number of cocurrent connected clients

Sessions

Topic	Description
sessions/count	Count of current sessions
sessions/max	Max number of sessions

Subscriptions

Topic	Description
subscriptions/count	Count of current subscriptions
subscriptions/max	Max number of subscriptions

Topics

Topic	Description
topics/count	Count of current topics
topics/max	Max number of topics

6.6.4 Broker Metrics

Topic path started with: $SYS/brokers/${node}/metrics/

Bytes Sent/Received

Topic	Description
bytes/received	MQTT Bytes Received since broker started
bytes/sent	MQTT Bytes Sent since the broker started

Packets Sent/Received

Topic	Description
packets/received	MQTT Packets received
packets/sent	MQTT Packets sent
packets/connect	MQTT CONNECT Packet received
packets/connack	MQTT CONNACK Packet sent
packets/publish/received	MQTT PUBLISH packets received
packets/publish/sent	MQTT PUBLISH packets sent
packets/subscribe	MQTT SUBSCRIBE Packets received
packets/suback	MQTT SUBACK packets sent
packets/unsubscribe	MQTT UNSUBSCRIBE Packets received
packets/unsuback	MQTT UNSUBACK Packets sent
packets/pingreq	MQTT PINGREQ packets received
packets/pingresp	MQTT PINGRESP Packets sent
packets/disconnect	MQTT DISCONNECT Packets received

Messages Sent/Received

Topic	Description
messages/received	Messages Received
messages/sent	Messages Sent
messages/retained	Messages Retained
messages/stored	TODO: Messages Stored
messages/dropped	Messages Dropped

6.6.5 Broker Alarms

Topic path started with: $SYS/brokers/${node}/alarms/

Topic	Description
${alarmId}/alert	New Alarm
${alarmId}/clear	Clear Alarm

6.6.6 Broker Sysmon

Topic path started with: '$SYS/brokers/${node}/sysmon/'

Topic	Description
long_gc	Long GC Warning
long_schedule	Long Schedule
large_heap	Large Heap Warning
busy_port	Busy Port Warning
busy_dist_port	Busy Dist Port

6.7 Trace

The emqttd broker supports to trace MQTT packets received/sent from/to a client, or trace MQTT messages published to a topic.

Trace a client:

```
./bin/emqttd_ctl trace client "clientid" "trace_clientid.log"
```

Trace a topic:

```
./bin/emqttd_ctl trace topic "topic" "trace_topic.log"
```

Lookup Traces:

```
./bin/emqttd_ctl trace list
```

Stop a Trace:

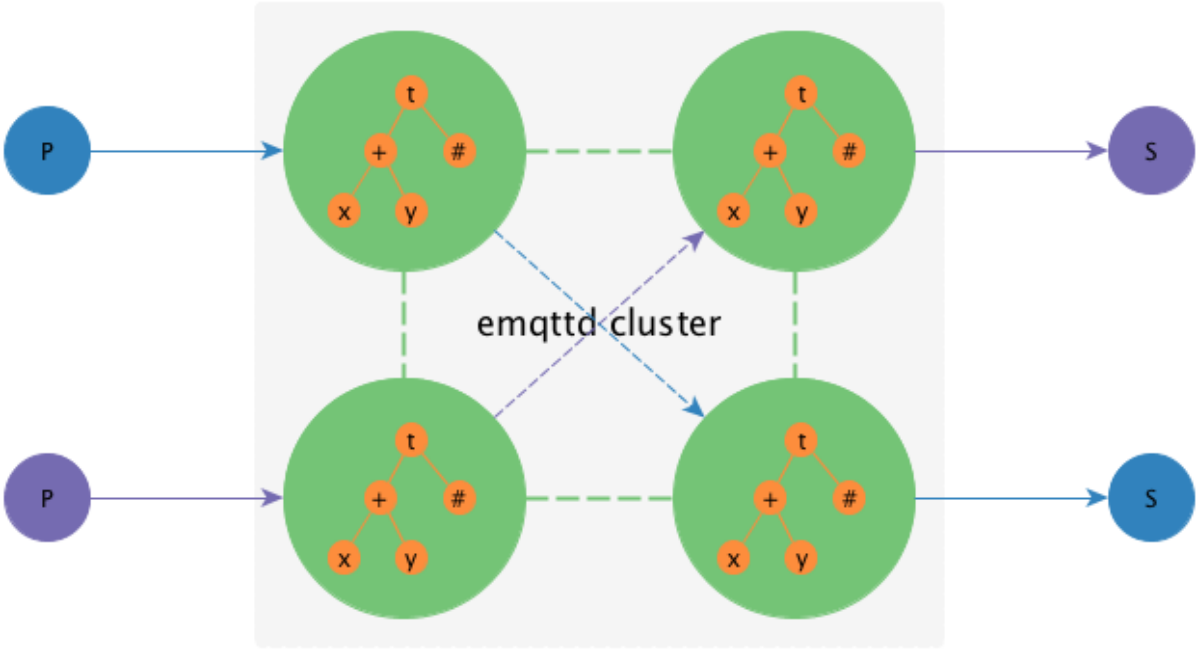
```
./bin/emqttd_ctl trace client "clientid" off

./bin/emqttd_ctl trace topic "topic" off
```

CHAPTER 7

Design

7.1 Architecture

The emqttd broker 1.0 is more like a network Switch or Router, not a traditional enterprise message queue. Compared to a network router that routes packets based on IP or MPLS label, the emqttd broker routes MQTT messages based on topic trie.



7.1.1 Design Philosophy

1. Focus on handling millions of MQTT connections and routing MQTT messages between clustered nodes.
2. Embrace Erlang/OTP, The Soft-Realtime, Low-Latency, Concurrent and Fault-Tolerant Platform.
3. Layered Design: Connection, Session, PubSub and Router Layers.
4. Separate the Message Flow Plane and the Control/Management Plane.
5. Stream MQTT messages to various backends including MQ or databases.

7.1.2 System Layers

1. Connection Layer

   Handle TCP and WebSocket connections, encode/decode MQTT packets.

2. Session Layer

   Process MQTT PUBLISH/SUBSCRIBE Packets received from client, and deliver MQTT messages to client.

3. PubSub Layer

   Dispatch MQTT messages to subscribers in a node.

4. Routing(Distributed) Layer

   Route MQTT messages among clustered nodes.

7.2 Connection Layer

This layer is built on the eSockd library which is a general Non-blocking TCP/SSL Socket Server:

- Acceptor Pool and Asynchronous TCP Accept
- Parameterized Connection Module
- Max connections management
- Allow/Deny by peer address or CIDR
- Keepalive Support
- Rate Limit based on The Leaky Bucket Algorithm
- Fully Asynchronous TCP RECV/SEND

This layer is also responsible for encoding/decoding MQTT frames:

1. Parse MQTT frames received from client
2. Serialize MQTT frames sent to client
3. MQTT Connection Keepalive

Main erlang modules of this layer:

Module	Description
emqttd_client	TCP Client
emqttd_ws_client	WebSocket Client
emqttd_protocol	MQTT Protocol Handler
emqttd_parser	MQTT Frame Parser
emqttd_serializer	MQTT Frame Serializer

7.3 Session Layer

The session layer processes MQTT packets received from client and delivers PUBLISH packets to client.

A MQTT session will store the subscriptions and inflight messages in memory:

1. The Client's subscriptions.

2. Inflight qos1/2 messages sent to the client but unacked, QoS 2 messages which have been sent to the Client, but have not been completely acknowledged.
3. Inflight qos2 messages received from client and waiting for PUBREL. QoS 2 messages which have been received from the Client, but have not been completely acknowledged.
4. All qos1, qos2 messages published to when client is disconnected.

7.3.1 MQueue and Inflight Window

Concept of Message Queue and Inflight Window:

```
      |<----------------- Max Len ----------------->|
      ------------------------------------------------
IN -> |     Messages Queue    |  Inflight Window    | -> Out
      ------------------------------------------------
                              |<---   Win Size  --->|
```

1. Inflight Window to store the messages delivered and await for PUBACK.
2. Enqueue messages when the inflight window is full.
3. If the queue is full, drop qos0 messages if store_qos0 is true, otherwise drop the oldest one.

The larger the inflight window size is, the higher the throughput is. The smaller the window size is, the more strict the message order is.

7.3.2 PacketId and MessageId

The 16-bit PacketId is defined by MQTT Protocol Specification, used by client/server to PUBLISH/PUBACK packets. A GUID(128-bit globally unique Id) will be generated by the broker and assigned to a MQTT message.

Format of the globally unique message id:

```
------------------------------------------------------
|        Timestamp       |  NodeID + PID  |  Sequence  |
|<------- 64bits ------->|<--- 48bits --->|<- 16bits ->|
------------------------------------------------------
```

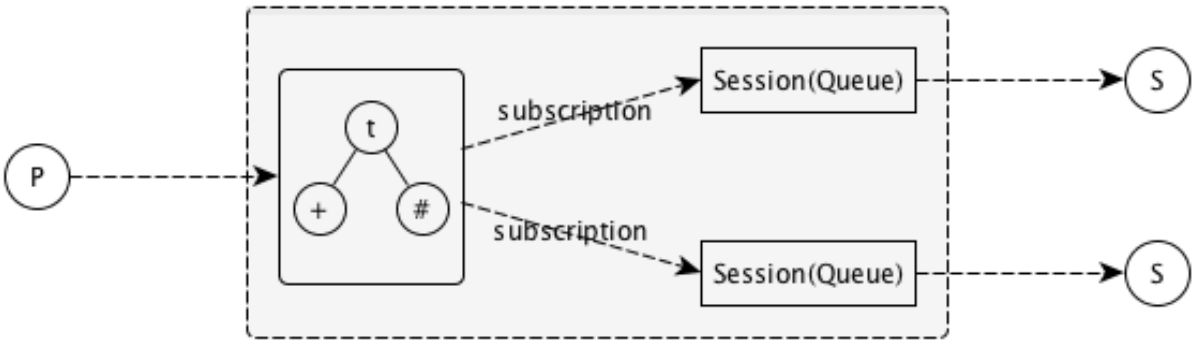
1. Timestamp: erlang:system_time if Erlang >= R18, otherwise os:timestamp
2. NodeId: encode node() to 2 bytes integer
3. Pid: encode pid to 4 bytes integer
4. Sequence: 2 bytes sequence in one process

The PacketId and MessageId in a End-to-End Message PubSub Sequence:

```
PktId <-- Session --> MsgId <-- Router --> MsgId <-- Session --> PktId
```

7.4 PubSub Layer

The PubSub layer maintains a subscription table and is responsible to dispatch MQTT messages to subscribers.



MQTT messages will be dispatched to the subscriber's session, which finally delivers the messages to client.

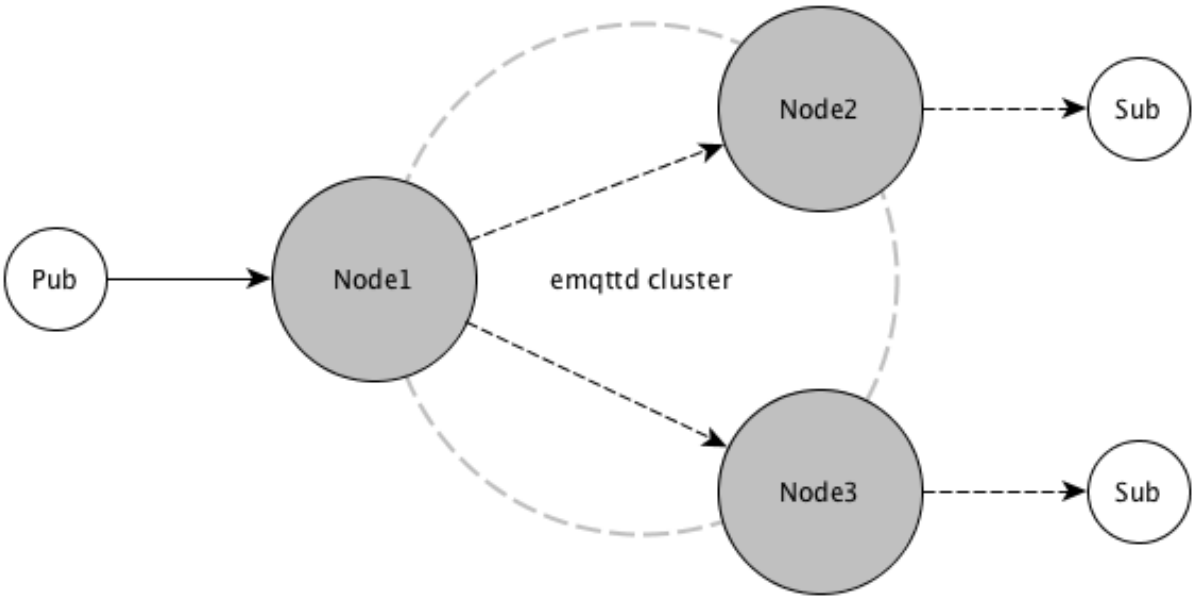
7.5 Routing Layer

The routing(distributed) layer maintains and replicates the global Topic Trie and Routing Table. The topic tire is composed of wildcard topics created by subscribers. The Routing Table maps a topic to nodes in the cluster.

For example, if node1 subscribed 't/+/x' and 't/+/y', node2 subscribed 't/#' and node3 subscribed 't/a', there will be a topic trie and route table:

```
-------------------------
|             t           |
|            / \          |
|           +   #         |
|         /  \            |
|       x      y          |
-------------------------
| t/+/x -> node1, node3 |
| t/+/y -> node1        |
| t/#   -> node2        |
| t/a   -> node3        |
-------------------------
```

The routing layer would route MQTT messages among clustered nodes by topic trie match and routing table lookup:



The routing design follows two rules:

1. A message only gets forwarded to other cluster nodes if a cluster node is interested in it. This reduces the network traffic tremendously, because it prevents nodes from forwarding unnecessary messages.
2. As soon as a client on a node subscribes to a topic it becomes known within the cluster. If one of the clients somewhere in the cluster is publishing to this topic, the message will be delivered to its subscriber no matter to which cluster node it is connected.

7.6 Authentication and ACL

The emqttd broker supports an extensible authentication/ACL mechanism, which is implemented by emqttd_access_control, emqttd_auth_mod and emqttd_acl_mod modules.

emqttd_access_control module provides two APIs that help register/unregister auth or ACL module:

```
register_mod(auth | acl, atom(), list()) -> ok | {error, any()}.

register_mod(auth | acl, atom(), list(), non_neg_integer()) -> ok | {error, any()}.
```

7.6.1 Authentication Bahaviour

The emqttd_auth_mod defines an Erlang behaviour for authentication module:

```
-module(emqttd_auth_mod).

-ifdef(use_specs).

-callback init(AuthOpts :: list()) -> {ok, State :: any()}.

-callback check(Client, Password, State) -> ok | ignore | {error, string()} when
    Client     :: mqtt_client(),
    Password   :: binary(),
```

```
    State      :: any().

-callback description() -> string().

-else.

-export([behaviour_info/1]).

behaviour_info(callbacks) ->
    [{init, 1}, {check, 3}, {description, 0}];
behaviour_info(_Other) ->
    undefined.

-endif.
```

The authentication modules implemented by default:

Module	Authentication
emqttd_auth_username	Username and Password
emqttd_auth_clientid	ClientID
emqttd_auth_ldap	LDAP
emqttd_auth_anonymous	Anonymous

7.6.2 Authorization(ACL)

The emqttd_acl_mod defines an Erlang behavihour for ACL module:

```
-module(emqttd_acl_mod).

-include("emqttd.hrl").

-ifdef(use_specs).

-callback init(AclOpts :: list()) -> {ok, State :: any()}.

-callback check_acl({Client, PubSub, Topic}, State :: any()) -> allow | deny | ignore when
    Client    :: mqtt_client(),
    PubSub    :: pubsub(),
    Topic     :: binary().

-callback reload_acl(State :: any()) -> ok | {error, any()}.

-callback description() -> string().

-else.

-export([behaviour_info/1]).

behaviour_info(callbacks) ->
    [{init, 1}, {check_acl, 2}, {reload_acl, 1}, {description, 0}];
behaviour_info(_Other) ->
    undefined.

-endif.
```

emqttd_acl_internal implements the default ACL based on etc/acl.config file:

```
%%%-----------------------------------------------------------------------------
%%%
%%% -type who() :: all | binary() |
%%%                {ipaddr, esockd_access:cidr()} |
%%%                {client, binary()} |
%%%                {user, binary()}.
%%%
%%% -type access() :: subscribe | publish | pubsub.
%%%
%%% -type topic() :: binary().
%%%
%%% -type rule() :: {allow, all} |
%%%                 {allow, who(), access(), list(topic())} |
%%%                 {deny, all} |
%%%                 {deny, who(), access(), list(topic())}.
%%%
%%%-----------------------------------------------------------------------------

{allow, {user, "dashboard"}, subscribe, ["$SYS/#"]}.

{allow, {ipaddr, "127.0.0.1"}, pubsub, ["$SYS/#", "#"]}.

{deny, all, subscribe, ["$SYS/#", {eq, "#"}]}.

{allow, all}.
```

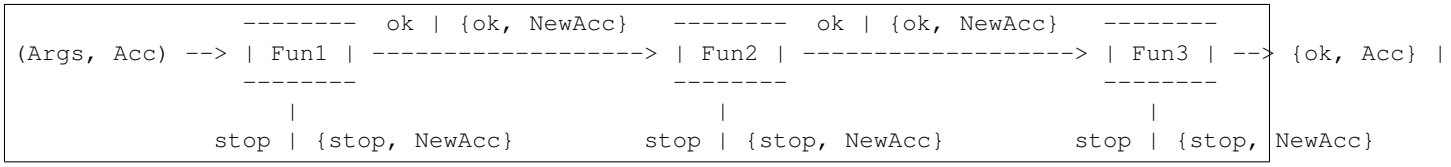
7.7 Hooks Design

The emqttd broker implements a simple but powerful hooks mechanism to help users develop plugin. The broker would run the hooks when a client is connected/disconnected, a topic is subscribed/unsubscribed or a MQTT message is published/delivered/acked.

Hooks defined by the emqttd 1.0 broker:

Hook	Description
client.connected	Run when client connected to the broker successfully
client.subscribe	Run before client subscribes topics
client.subscribe.after	Run After client subscribed topics
client.unsubscribe	Run when client unsubscribes topics
message.publish	Run when a MQTT message is published
message.delivered	Run when a MQTT message is delivered
message.acked	Run when a MQTT message is acked
client.disconnected	Run when client disconnected from broker

The emqttd broker uses the Chain-of-responsibility_pattern to implement hook mechanism. The callback functions registered to hook will be executed one by one:



The callback function for a hook should return:

Return	Description
ok	Continue
{ok, NewAcc}	Return Acc and Continue
stop	Break
{stop, NewAcc}	Return Acc and Break

The input arguments for a callback function are depending on the types of hook. Clone the emqttd_plugin_template project to check the argument in detail.

7.7.1 Hook Implementation

The hook APIs defined in emqttd module:

```
-module(emqttd).

%% Hooks API
-export([hook/4, hook/3, unhook/2, run_hooks/3]).
hook(Hook :: atom(), Callback :: function(), InitArgs :: list(any())) -> ok | {error, any()}.

hook(Hook :: atom(), Callback :: function(), InitArgs :: list(any()), Priority :: integer()) -> ok |

unhook(Hook :: atom(), Callback :: function()) -> ok | {error, any()}.

run_hooks(Hook :: atom(), Args :: list(any()), Acc :: any()) -> {ok | stop, any()}.
```

And implemented in emqttd_hook module:

```
-module(emqttd_hook).

%% Hooks API
-export([add/3, add/4, delete/2, run/3, lookup/1]).

add(HookPoint :: atom(), Callback :: function(), InitArgs :: list(any())) -> ok.

add(HookPoint :: atom(), Callback :: function(), InitArgs :: list(any()), Priority :: integer()) -> o

delete(HookPoint :: atom(), Callback :: function()) -> ok.

run(HookPoint :: atom(), Args :: list(any()), Acc :: any()) -> any().

lookup(HookPoint :: atom()) -> [#callback{}].
```

7.7.2 Hook Usage

The emqttd_plugin_template project provides the examples for hook usage:

```
-module(emqttd_plugin_template).

-export([load/1, unload/0]).

-export([on_message_publish/2, on_message_delivered/3, on_message_acked/3]).

load(Env) ->
    emqttd:hook('message.publish', fun ?MODULE:on_message_publish/2, [Env]),
    emqttd:hook('message.delivered', fun ?MODULE:on_message_delivered/3, [Env]),
    emqttd:hook('message.acked', fun ?MODULE:on_message_acked/3, [Env]).
```

```
on_message_publish(Message, _Env) ->
    io:format("publish ~s~n", [emqttd_message:format(Message)]),
    {ok, Message}.

on_message_delivered(ClientId, Message, _Env) ->
    io:format("delivered to client ~s: ~s~n", [ClientId, emqttd_message:format(Message)]),
    {ok, Message}.

on_message_acked(ClientId, Message, _Env) ->
    io:format("client ~s acked: ~s~n", [ClientId, emqttd_message:format(Message)]),
    {ok, Message}.

unload() ->
    emqttd:unhook('message.publish', fun ?MODULE:on_message_publish/2),
    emqttd:unhook('message.acked', fun ?MODULE:on_message_acked/3),
    emqttd:unhook('message.delivered', fun ?MODULE:on_message_delivered/3).
```

7.8 Plugin Design

Plugin is a normal erlang application that can be started/stopped dynamically by a running emqttd broker.

7.8.1 emqttd_plugins Module

The plugin mechanism is implemented by emqttd_plugins module:

```
-module(emqttd_plugins).

-export([load/1, unload/1]).

%% @doc Load a Plugin
load(PluginName :: atom()) -> ok | {error, any()}.

%% @doc UnLoad a Plugin
unload(PluginName :: atom()) -> ok | {error, any()}.
```

7.8.2 Load a Plugin

Use './bin/emqttd_ctl' CLI to load/unload a plugin:

```
./bin/emqttd_ctl plugins load emqttd_plugin_redis

./bin/emqttd_ctl plugins unload emqttd_plugin_redis
```

7.8.3 Plugin Template

http://github.com/emqtt/emqttd_plugin_template

CHAPTER 8

Commands

The './bin/emqttd_ctl' command line could be used to query and administrate emqttd broker.

Warning: Cannot work on Windows

8.1 status

Show running status of the broker:

```
$ ./bin/emqttd_ctl status

Node 'emqttd@127.0.0.1' is started
emqttd 1.1 is running
```

8.2 broker

Query basic information, statistics and metrics of the broker.

broker	Show version, description, uptime of the broker
broker pubsub	Show status of the core pubsub process
broker stats	Show statistics of client, session, topic, subscription and route of the broker
broker metrics	Show metrics of MQTT bytes, packets, messages sent/received.

Query version, description and uptime of the broker:

```
$ ./bin/emqttd_ctl broker

sysdescr  : Erlang MQTT Broker
version   : 0.15.0
uptime    : 1 hours, 25 minutes, 24 seconds
datetime  : 2016-01-16 13:17:32
```

8.2.1 broker stats

Query statistics of MQTT Client, Session, Topic, Subscription and Route:

```
$ ./bin/emqttd_ctl broker stats

clients/count       : 1
clients/max         : 1
queues/count        : 0
queues/max          : 0
retained/count      : 2
retained/max        : 2
routes/count        : 2
routes/reverse      : 2
sessions/count      : 0
sessions/max        : 0
subscriptions/count : 1
subscriptions/max   : 1
topics/count        : 54
topics/max          : 54
```

8.2.2 broker metrics

Query metrics of Bytes, MQTT Packets and Messages(sent/received):

```
$ ./bin/emqttd_ctl broker metrics

bytes/received          : 297
bytes/sent              : 40
messages/dropped        : 348
messages/qos0/received  : 0
messages/qos0/sent      : 0
messages/qos1/received  : 0
messages/qos1/sent      : 0
messages/qos2/received  : 0
messages/qos2/sent      : 0
messages/received       : 0
messages/retained       : 2
messages/sent           : 0
packets/connack         : 5
packets/connect         : 5
packets/disconnect      : 0
packets/pingreq         : 0
packets/pingresp        : 0
packets/puback/received : 0
packets/puback/sent     : 0
packets/pubcomp/received: 0
packets/pubcomp/sent    : 0
packets/publish/received: 0
packets/publish/sent    : 0
packets/pubrec/received : 0
packets/pubrec/sent     : 0
packets/pubrel/received : 0
packets/pubrel/sent     : 0
packets/received        : 9
packets/sent            : 9
packets/suback          : 4
packets/subscribe       : 4
packets/unsuback        : 0
packets/unsubscribe     : 0
```

8.3 cluster

Cluster two or more emqttd brokers.

cluster join <Node>	Join the cluster
cluster leave	Leave the cluster
cluster remove <Node>	Remove a node from the cluster
cluster status	Query cluster status and nodes

Suppose we create two emqttd nodes on localhost and cluster them:

Folder	Node	MQTT Port
emqttd1	emqttd1@127.0.0.1	1883
emqttd2	emqttd2@127.0.0.1	2883

Start emqttd1 node:

```
cd emqttd1 && ./bin/emqttd start
```

Start emqttd2 node:

```
cd emqttd2 && ./bin/emqttd start
```

Under emqttd2 folder:

```
$ ./bin/emqttd_ctl cluster join emqttd1@127.0.0.1

Join the cluster successfully.
Cluster status: [{running_nodes,['emqttd1@127.0.0.1','emqttd2@127.0.0.1']}]
```

Query cluster status:

```
$ ./bin/emqttd_ctl cluster status

Cluster status: [{running_nodes,['emqttd2@127.0.0.1','emqttd1@127.0.0.1']}]
```

Message Route between nodes:

```
# Subscribe topic 'x' on emqttd1 node
mosquitto_sub -t x -q 1 -p 1883

# Publish to topic 'x' on emqttd2 node
mosquitto_pub -t x -q 1 -p 2883 -m hello
```

emqttd2 leaves the cluster:

```
cd emqttd2 && ./bin/emqttd_ctl cluster leave
```

Or remove emqttd2 from the cluster on emqttd1 node:

```
cd emqttd1 && ./bin/emqttd_ctl cluster remove emqttd2@127.0.0.1
```

8.4 clients

Query MQTT clients connected to the broker:

clients list	List all MQTT clients
clients show <ClientId>	Show a MQTT Client
clients kick <ClientId>	Kick out a MQTT client

8.4.1 clients lists

Query All MQTT clients connected to the broker:

```
$ ./bin/emqttd_ctl clients list

Client(mosqsub/43832-airlee.lo, clean_sess=true, username=test, peername=127.0.0.1:64896, connected_a
Client(mosqsub/44011-airlee.lo, clean_sess=true, username=test, peername=127.0.0.1:64961, connected_a
...
```

Properties of the Client:

clean_sess	Clean Session Flag
username	Username of the client
peername	Peername of the TCP connection
connected_at	The timestamp when client connected to the broker

8.4.2 clients show <ClientId>

Show a specific MQTT Client:

```
./bin/emqttd_ctl clients show "mosqsub/43832-airlee.lo"

Client(mosqsub/43832-airlee.lo, clean_sess=true, username=test, peername=127.0.0.1:64896, connected_a
```

8.4.3 clients kick <ClientId>

Kick out a MQTT Client:

```
./bin/emqttd_ctl clients kick "clientid"
```

8.5 sessions

Query all MQTT sessions. The broker will create a session for each MQTT client. Persistent Session if clean_session flag is true, transient session otherwise.

sessions list	List all Sessions
sessions list persistent	Query all persistent Sessions
sessions list transient	Query all transient Sessions
sessions show <ClientId>	Show a session

8.5.1 sessions list

Query all sessions:

```
$ ./bin/emqttd_ctl sessions list

Session(clientid, clean_sess=false, max_inflight=100, inflight_queue=0, message_queue=0, message_drop
Session(mosqsub/44101-airlee.lo, clean_sess=true, max_inflight=100, inflight_queue=0, message_queue=0
```

Properties of Session:

TODO:??

clean_sess	clean sess flag. false: persistent, true: transient
max_inflight	Inflight window (Max number of messages delivering)
inflight_queue	Inflight Queue Size
message_queue	Message Queue Size
message_dropped	Number of Messages Dropped for queue is full
awaiting_rel	The number of QoS2 messages received and waiting for PUBREL
awaiting_ack	The number of QoS1/2 messages delivered and waiting for PUBACK
awaiting_comp	The number of QoS2 messages delivered and waiting for PUBCOMP
created_at	Timestamp when the session is created

8.5.2 sessions list persistent

Query all persistent sessions:

```
$ ./bin/emqttd_ctl sessions list persistent

Session(clientid, clean_sess=false, max_inflight=100, inflight_queue=0, message_queue=0, message_drop
```

8.5.3 sessions list transient

Query all transient sessions:

```
$ ./bin/emqttd_ctl sessions list transient

Session(mosqsub/44101-airlee.lo, clean_sess=true, max_inflight=100, inflight_queue=0, message_queue=0
```

8.5.4 sessions show <ClientId>

Show a session:

```
$ ./bin/emqttd_ctl sessions show clientid

Session(clientid, clean_sess=false, max_inflight=100, inflight_queue=0, message_queue=0, message_drop
```

8.6 routes

Show routing table of the broker.

8.6.1 routes list

List all routes:

```
$ ./bin/emqttd_ctl routes list

t2/# -> emqttd2@127.0.0.1
t/+/x -> emqttd2@127.0.0.1,emqttd@127.0.0.1
```

8.6.2 routes show <Topic>

Show a route:

```
$ ./bin/emqttd_ctl routes show t/+/x

t/+/x -> emqttd2@127.0.0.1,emqttd@127.0.0.1
```

8.7 topics

Query topic table of the broker.

8.7.1 topics list

Query all the topics:

```
$ ./bin/emqttd_ctl topics list

$SYS/brokers/emqttd@127.0.0.1/metrics/packets/subscribe: static
$SYS/brokers/emqttd@127.0.0.1/stats/subscriptions/max: static
$SYS/brokers/emqttd2@127.0.0.1/stats/subscriptions/count: static
...
```

8.7.2 topics show <Topic>

Show a topic:

```
$ ./bin/emqttd_ctl topics show '$SYS/brokers'

$SYS/brokers: static
```

8.8 subscriptions

Query the subscription table of the broker:

subscriptions list	List all subscriptions
subscriptions show <ClientId>	Show a subscription
subscriptions add <ClientId> <Topic> <Qos>	Add a static subscription manually
subscriptions del <ClientId> <Topic>	Remove a static subscription manually

8.8.1 subscriptions list

Query all subscriptions:

```
$ ./bin/emqttd_ctl subscriptions list

mosqsub/91042-airlee.lo -> t/y:1
mosqsub/90475-airlee.lo -> t/+/x:2
```

8.8.2 subscriptions list static

List all static subscriptions:

```
$ ./bin/emqttd_ctl subscriptions list static

clientid -> new_topic:1
```

8.8.3 subscriptions show <ClientId>

Show the subscriptions of a MQTT client:

```
$ ./bin/emqttd_ctl subscriptions show clientid

clientid: [{<<"x">>,1},{<<"topic2">>,1},{<<"topic3">>,1}]
```

8.8.4 subscriptions add <ClientId> <Topic> <QoS>

Add a static subscription manually:

```
$ ./bin/emqttd_ctl subscriptions add clientid new_topic 1
ok
```

8.8.5 subscriptions del <ClientId> <Topic>

Remove a static subscription manually:

```
$ ./bin/emqttd_ctl subscriptions del clientid new_topic
ok
```

8.9 plugins

List, load or unload plugins of emqttd broker.

plugins list	List all plugins
plugins load <Plugin>	Load Plugin
plugins unload <Plugin>	Unload (Plugin)

8.9.1 plugins list

List all plugins:

```
$ ./bin/emqttd_ctl plugins list

Plugin(emqttd_dashboard, version=0.16.0, description=emqttd web dashboard, active=true)
Plugin(emqttd_plugin_mysql, version=0.16.0, description=emqttd Authentication/ACL with MySQL, active=
Plugin(emqttd_plugin_pgsql, version=0.16.0, description=emqttd PostgreSQL Plugin, active=false)
Plugin(emqttd_plugin_redis, version=0.16.0, description=emqttd Redis Plugin, active=false)
Plugin(emqttd_plugin_template, version=0.16.0, description=emqttd plugin template, active=false)
Plugin(emqttd_recon, version=0.16.0, description=emqttd recon plugin, active=false)
Plugin(emqttd_stomp, version=0.16.0, description=Stomp Protocol Plugin for emqttd broker, active=fals
```

Properties of a plugin:

version	Plugin Version
description	Plugin Description
active	If the plugin is Loaded

8.9.2 load <Plugin>

Load a Plugin:

```
$ ./bin/emqttd_ctl plugins load emqttd_recon

Start apps: [recon,emqttd_recon]
Plugin emqttd_recon loaded successfully.
```

8.9.3 unload <Plugin>

Unload a Plugin:

```
$ ./bin/emqttd_ctl plugins unload emqttd_recon

Plugin emqttd_recon unloaded successfully.
```

8.10 bridges

Bridge two or more emqttd brokers:

```
                ---------                          ---------
Publisher --> | node1 | --Bridge Forward--> | node2 | --> Subscriber
                ---------                          ---------
```

commands for bridge:

bridges list	List all bridges
bridges options	Show bridge options
bridges start <Node> <Topic>	Create a bridge
bridges start <Node> <Topic> <Options>	Create a bridge with options
bridges stop <Node> <Topic>	Delete a bridge

Suppose we create a bridge between emqttd1 and emqttd2 on localhost:

Name	Node	MQTT Port
emqttd1	emqttd1@127.0.0.1	1883
emqttd2	emqttd2@127.0.0.1	2883

The bridge will forward all the the 'sensor/#' messages from emqttd1 to emqttd2:

```
$ ./bin/emqttd_ctl bridges start emqttd2@127.0.0.1 sensor/#

bridge is started.

$ ./bin/emqttd_ctl bridges list

bridge: emqttd1@127.0.0.1--sensor/#-->emqttd2@127.0.0.1
```

The the 'emqttd1–sensor/#–>emqttd2' bridge:

```
#emqttd2 node

mosquitto_sub -t sensor/# -p 2883 -d

#emqttd1

mosquitto_pub -t sensor/1/temperature -m "37.5" -d
```

8.10.1 bridges options

Show bridge options:

```
$ ./bin/emqttd_ctl bridges options

Options:
  qos     = 0 | 1 | 2
  prefix  = string
  suffix  = string
  queue   = integer
Example:
  qos=2,prefix=abc/,suffix=/yxz,queue=1000
```

8.10.2 bridges stop <Node> <Topic>

Delete the emqttd1–sensor/#–>emqttd2 bridge:

```
$ ./bin/emqttd_ctl bridges stop emqttd2@127.0.0.1 sensor/#

bridge is stopped.
```

8.11 vm

Query the load, cpu, memory, processes and IO information of the Erlang VM.

vm all	Query all
vm load	Query VM Load
vm memory	Query Memory Usage
vm process	Query Number of Erlang Processes
vm io	Query Max Fds of VM

8.11.1 vm load

Query load:

```
$ ./bin/emqttd_ctl vm load

cpu/load1               : 2.21
cpu/load5               : 2.60
cpu/load15              : 2.36
```

8.11.2 vm memory

Query memory:

```
$ ./bin/emqttd_ctl vm memory

memory/total            : 23967736
memory/processes        : 3594216
memory/processes_used   : 3593112
memory/system           : 20373520
memory/atom             : 512601
memory/atom_used        : 491955
memory/binary           : 51432
memory/code             : 13401565
memory/ets              : 1082848
```

8.11.3 vm process

Query number of erlang processes:

```
$ ./bin/emqttd_ctl vm process

process/limit           : 8192
process/count           : 221
```

8.11.4 vm io

Query max, active file descriptors of IO:

```
$ ./bin/emqttd_ctl vm io

io/max_fds              : 2560
io/active_fds           : 1
```

8.12 trace

Trace MQTT packets, messages(sent/received) by ClientId or Topic.

trace list	List all the traces
trace client <ClientId> <LogFile>	Trace a client
trace client <ClientId> off	Stop tracing the client
trace topic <Topic> <LogFile>	Trace a topic
trace topic <Topic> off	Stop tracing the topic

8.12.1 trace client <ClientId> <LogFile>

Start to trace a client:

```
$ ./bin/emqttd_ctl trace client clientid log/clientid_trace.log

trace client clientid successfully.
```

8.12.2 trace client <ClientId> off

Stop tracing the client:

```
$ ./bin/emqttd_ctl trace client clientid off

stop tracing client clientid successfully.
```

8.12.3 trace topic <Topic> <LogFile>

Start to trace a topic:

```
$ ./bin/emqttd_ctl trace topic topic log/topic_trace.log

trace topic topic successfully.
```

8.12.4 trace topic <Topic> off

Stop tracing the topic:

```
$ ./bin/emqttd_ctl trace topic topic off

stop tracing topic topic successfully.
```

8.12.5 trace list

List all traces:

```
$ ./bin/emqttd_ctl trace list

trace client clientid -> log/clientid_trace.log
trace topic topic -> log/topic_trace.log
```

8.13 listeners

Show all the TCP listeners:

```
$ ./bin/emqttd_ctl listeners

listener on http:8083
  acceptors       : 4
  max_clients     : 64
  current_clients : 0
  shutdown_count  : []
listener on mqtts:8883
  acceptors       : 4
  max_clients     : 512
  current_clients : 0
  shutdown_count  : []
listener on mqtt:1883
  acceptors       : 16
  max_clients     : 8192
```

```
  current_clients : 1
  shutdown_count  : [{closed,1}]
listener on http:18083
  acceptors       : 4
  max_clients     : 512
  current_clients : 0
  shutdown_count  : []
```

listener parameters:

acceptors	TCP Acceptor Pool
max_clients	Max number of clients
current_clients	Count of current clients
shutdown_count | Statistics of client shutdown reason	

8.14 mnesia

Show system_info of mnesia database.

8.15 admins

The 'admins' CLI is used to add/del admin account, which is registered by the dashboard plugin.

admins add <Username> <Password>	Add admin account
admins passwd <Username> <Password>	Reset admin password
admins del <Username>	Delete admin account

8.15.1 admins add

Add admin account:

```
$ ./bin/emqttd_ctl admins add root public
ok
```

8.15.2 admins passwd

Reset password:

```
$ ./bin/emqttd_ctl admins passwd root private
ok
```

8.15.3 admins del

Delete admin account:

```
$ ./bin/emqttd_ctl admins del root
ok
```

CHAPTER 9

Plugins

The emqttd broker could be extended by plugins. Users could develop plugins to customize authentication, ACL and functions of the broker, or integrate the broker with other systems.

The plugins that emqtt project released:

Plugin	Description
emqttd_plugin_template	Template Plugin
emqttd_dashboard	Web Dashboard
emqttd_auth_http	HTTP Auth/ACL Plugin
emqttd_plugin_mysql	MySQL Auth/ACL Plugin
emqttd_plugin_pgsql	PostgreSQL Auth/ACL Plugin
emqttd_plugin_redis	Redis Auth/ACL Plugin
emqttd_plugin_mongo	MongoDB Auth/ACL Plugin
emqttd_stomp	STOMP Protocol Plugin
emqttd_sockjs	STOMP over SockJS Plugin
emqttd_recon	Recon Plugin
emqttd_reloader	Reloader Plugin

9.1 emqttd_plugin_template - Template Plugin

A plugin is just a normal Erlang application under the 'emqttd/plugins' folder. Each plugin has e configuration file: 'etc/plugin.config'.

plugins/emqttd_plugin_template is a demo plugin. The folder structure:

File	Description
etc/plugin.config	Plugin config file
ebin/	Erlang program files

9.1.1 Load, unload Plugin

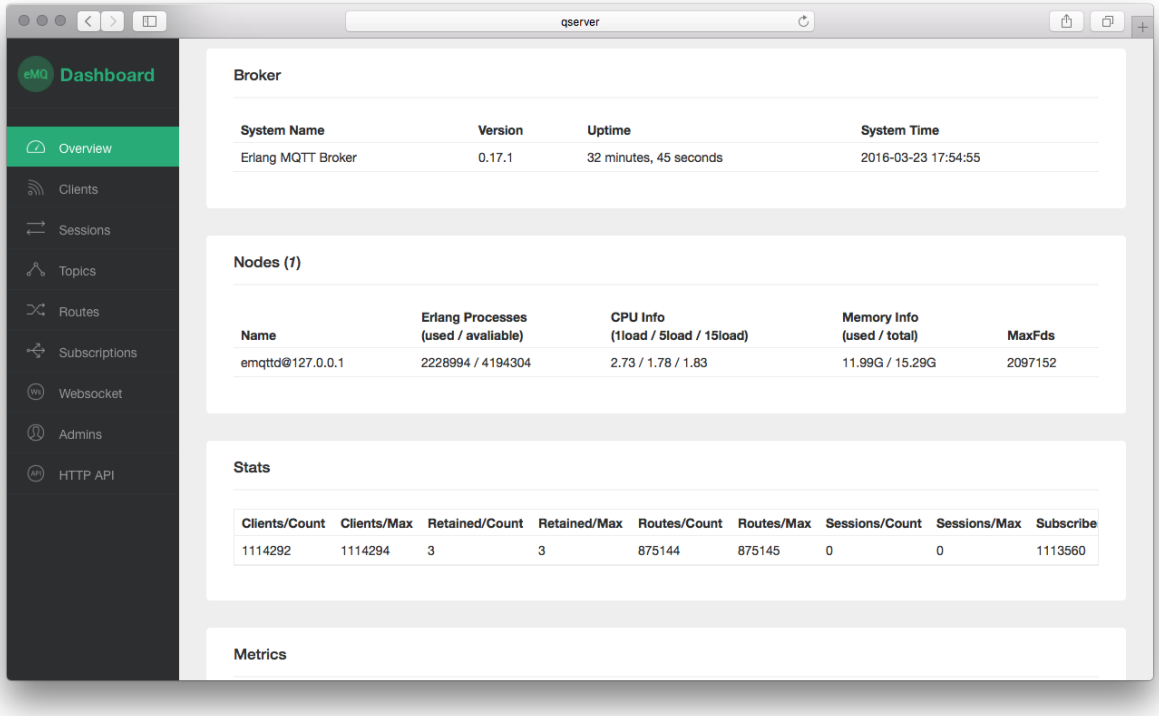
Use 'bin/emqttd_ctl plugins' CLI to load, unload a plugin:

```
./bin/emqttd_ctl plugins load <PluginName>

./bin/emqttd_ctl plugins unload <PluginName>

./bin/emqttd_ctl plugins list
```

9.2 emqttd_dashboard - Dashboard Plugin

The Web Dashboard for emqttd broker. The plugin will be loaded automatically when the broker started successfully.

Address	http://localhost:18083
Default User	admin
Default Password	public



9.2.1 Configure Dashboard

emqttd_dashboard/etc/plugin.config:

```
[
  {emqttd_dashboard, [
    {listener,
      {emqttd_dashboard, 18083, [
        {acceptors, 4},
        {max_clients, 512}]}
    }
  ]}
].
```

9.3 emqttd_auth_http - HTTP Auth/ACL Plugin

MQTT Authentication/ACL with HTTP API: https://github.com/emqtt/emqttd_auth_http

Note: Supported in 1.1 release

9.3.1 Configure emqttd_auth_http/etc/plugin.config

```
[

  {emqttd_auth_http, [

    %% Variables: %u = username, %c = clientid, %a = ipaddress, %t = topic

    {super_req, [
      {method, post},
      {url, "http://localhost:8080/mqtt/superuser"},
      {params, [
        {username, "%u"},
        {clientid, "%c"}
      ]}
    ]},

    {auth_req, [
      {method, post},
      {url, "http://localhost:8080/mqtt/auth"},
      {params, [
        {clientid, "%c"},
        {username, "%u"},
        {password, "%P"}
      ]}
    ]},

    %% 'access' parameter: sub = 1, pub = 2

    {acl_req, [
      {method, post},
      {url, "http://localhost:8080/mqtt/acl"},
      {params, [
        {access,   "%A"},
        {username, "%u"},
        {clientid, "%c"},
        {ipaddr,   "%a"},
        {topic,    "%t"}
      ]}
    ]}
  ]}

].
```

9.3.2 HTTP API

Return 200 if ok

Return 4xx if unauthorized

9.3.3 Load emqttd_auth_http plugin

```
./bin/emqttd_ctl plugins load emqttd_auth_http
```

9.4 emqttd_plugin_mysql - MySQL Auth/ACL Plugin

MQTT Authentication, ACL with MySQL database.

9.4.1 MQTT User Table

```
CREATE TABLE `mqtt_user` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `username` varchar(100) DEFAULT NULL,
  `password` varchar(100) DEFAULT NULL,
  `salt` varchar(20) DEFAULT NULL,
  `is_superuser` tinyint(1) DEFAULT 0,
  `created` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `mqtt_username` (`username`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

9.4.2 MQTT ACL Table

```
CREATE TABLE `mqtt_acl` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `allow` int(1) DEFAULT NULL COMMENT '0: deny, 1: allow',
  `ipaddr` varchar(60) DEFAULT NULL COMMENT 'IpAddress',
  `username` varchar(100) DEFAULT NULL COMMENT 'Username',
  `clientid` varchar(100) DEFAULT NULL COMMENT 'ClientId',
  `access` int(2) NOT NULL COMMENT '1: subscribe, 2: publish, 3: pubsub',
  `topic` varchar(100) NOT NULL DEFAULT '' COMMENT 'Topic Filter',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `mqtt_acl` (`id`, `allow`, `ipaddr`, `username`, `clientid`, `access`, `topic`)
VALUES
    (1,1,NULL,'$all',NULL,2,'#'),
    (2,0,NULL,'$all',NULL,1,'$SYS/#'),
    (3,0,NULL,'$all',NULL,1,'eq #'),
    (5,1,'127.0.0.1',NULL,NULL,2,'$SYS/#'),
    (6,1,'127.0.0.1',NULL,NULL,2,'#'),
    (7,1,NULL,'dashboard',NULL,1,'$SYS/#');
```

9.4.3 Configure emqttd_plugin_mysql/etc/plugin.config

Configure MySQL host, username, password and database:

```
[

  {emqttd_plugin_mysql, [
```

```
    {mysql_pool, [
        %% ecpool options
        {pool_size, 8},
        {auto_reconnect, 3},

        %% mysql options
        {host,     "localhost"},
        {port,     3306},
        {user,     ""},
        {password, ""},
        {database, "mqtt"},
        {encoding, utf8}
    ]},

    %% Variables: %u = username, %c = clientid, %a = ipaddress

    %% Superuser Query
    {superquery, "select is_superuser from mqtt_user where username = '%u' limit 1"},

    %% Authentication Query: select password only
    {authquery, "select password from mqtt_user where username = '%u' limit 1"},

    %% hash algorithm: plain, md5, sha, sha256, pbkdf2?
    {password_hash, sha256},

    %% select password with salt
    %% {authquery, "select password, salt from mqtt_user where username = '%u'"},

    %% sha256 with salt prefix
    %% {password_hash, {salt, sha256}},

    %% sha256 with salt suffix
    %% {password_hash, {sha256, salt}},

    %% '%a' = ipaddress, '%u' = username, '%c' = clientid
    %% Comment this query, the acl will be disabled
    {aclquery, "select allow, ipaddr, username, clientid, access, topic from mqtt_acl where ipaddr =

    %% If no ACL rules matched, return...
    {acl_nomatch, allow}

  ]}

].
```

9.4.4 Load emqttd_plugin_mysql plugin

```
./bin/emqttd_ctl plugins load emqttd_plugin_mysql
```

9.5 emqttd_plugin_pgsql - PostgreSQL Auth/ACL Plugin

MQTT Authentication, ACL with PostgreSQL Database.

9.5.1 MQTT User Table

```
CREATE TABLE mqtt_user (
  id SERIAL primary key,
  is_superuser boolean,
  username character varying(100),
  password character varying(100),
  salt character varying(40)
);
```

9.5.2 MQTT ACL Table

```
CREATE TABLE mqtt_acl (
  id SERIAL primary key,
  allow integer,
  ipaddr character varying(60),
  username character varying(100),
  clientid character varying(100),
  access  integer,
  topic character varying(100)
);

INSERT INTO mqtt_acl (id, allow, ipaddr, username, clientid, access, topic)
VALUES
    (1,1,NULL,'$all',NULL,2,'#'),
    (2,0,NULL,'$all',NULL,1,'$SYS/#'),
    (3,0,NULL,'$all',NULL,1,'eq #'),
    (5,1,'127.0.0.1',NULL,NULL,2,'$SYS/#'),
    (6,1,'127.0.0.1',NULL,NULL,2,'#'),
    (7,1,NULL,'dashboard',NULL,1,'$SYS/#');
```

9.5.3 Configure emqttd_plugin_pgsql/etc/plugin.config

Configure host, username, password and database of PostgreSQL:

```
[

  {emqttd_plugin_pgsql, [

    {pgsql_pool, [
        %% ecpool options
        {pool_size, 8},
        {auto_reconnect, 3},

        %% pgsql options
        {host, "localhost"},
        {port, 5432},
        {ssl, false},
        {username, "feng"},
        {password, ""},
        {database, "mqtt"},
        {encoding,  utf8}
    ]},

    %% Variables: %u = username, %c = clientid, %a = ipaddress
```

```
    %% Superuser Query
    {superquery, "select is_superuser from mqtt_user where username = '%u' limit 1"},

    %% Authentication Query: select password only
    {authquery, "select password from mqtt_user where username = '%u' limit 1"},

    %% hash algorithm: plain, md5, sha, sha256, pbkdf2?
    {password_hash, sha256},

    %% select password with salt
    %% {authquery, "select password, salt from mqtt_user where username = '%u'"},

    %% sha256 with salt prefix
    %% {password_hash, {salt, sha256}},

    %% sha256 with salt suffix
    %% {password_hash, {sha256, salt}},

    %% Comment this query, the acl will be disabled. Notice: don't edit this query!
    {aclquery, "select allow, ipaddr, username, clientid, access, topic from mqtt_acl
                where ipaddr = '%a' or username = '%u' or username = '$all' or clientid = '%c'"},

    %% If no rules matched, return...
    {acl_nomatch, allow}
  ]}
].
```

9.5.4 Load emqttd_plugin_pgsql Plugin

```
./bin/emqttd_ctl plugins load emqttd_plugin_pgsql
```

9.6 emqttd_plugin_redis - Redis Auth/ACL Plugin

MQTT Authentication, ACL with Redis: https://github.com/emqtt/emqttd_plugin_redis

9.6.1 Configure emqttd_plugin_redis/etc/plugin.config

```
[
  {emqttd_plugin_redis, [

    {eredis_pool, [
      %% ecpool options
      {pool_size, 8},
      {auto_reconnect, 2},

      %% eredis options
      {host, "127.0.0.1"},
      {port, 6379},
      {database, 0},
      {password, ""}
    ]},
```

```
    %% Variables: %u = username, %c = clientid

    %% HMGET mqtt_user:%u is_superuser
    {supercmd, ["HGET", "mqtt_user:%u", "is_superuser"]},

    %% HMGET mqtt_user:%u password
    {authcmd, ["HGET", "mqtt_user:%u", "password"]},

    %% Password hash algorithm: plain, md5, sha, sha256, pbkdf2?
    {password_hash, sha256},

    %% SMEMBERS mqtt_acl:%u
    {aclcmd, ["SMEMBERS", "mqtt_acl:%u"]},

    %% If no rules matched, return...
    {acl_nomatch, deny},

    %% Load Subscriptions form Redis when client connected.
    {subcmd, ["HGETALL", "mqtt_subs:%u"]}
  ]}
].
```

9.6.2 User HASH

Set a 'user' hash with 'password' field, for example:

```
HSET mqtt_user:<username> is_superuser 1
HSET mqtt_user:<username> password "passwd"
```

9.6.3 ACL Rule SET

The plugin uses a redis SET to store ACL rules:

```
SADD mqtt_acl:<username> "publish topic1"
SADD mqtt_acl:<username> "subscribe topic2"
SADD mqtt_acl:<username> "pubsub topic3"
```

9.6.4 Subscription HASH

The plugin can store static subscriptions in a redis Hash:

```
HSET mqtt_subs:<username> topic1 0
HSET mqtt_subs:<username> topic2 1
HSET mqtt_subs:<username> topic3 2
```

9.6.5 Load emqttd_plugin_redis Plugin

```
./bin/emqttd_ctl plugins load emqttd_plugin_redis
```

9.7 emqttd_plugin_mongo - MongoDB Auth/ACL Plugin

MQTT Authentication, ACL with MongoDB: https://github.com/emqtt/emqttd_plugin_mongo

9.7.1 Configure emqttd_plugin_mongo/etc/plugin.config

```
[
  {emqttd_plugin_mongo, [

    {mongo_pool, [
      {pool_size, 8},
      {auto_reconnect, 3},

      %% Mongodb Driver Opts
      {host, "localhost"},
      {port, 27017},
      %% {login, ""},
      %% {password, ""},
      {database, "mqtt"}
    ]},

    %% Variables: %u = username, %c = clientid

    %% Superuser Query
    {superquery, [
      {collection, "mqtt_user"},
      {super_field, "is_superuser"},
      {selector, {"username", "%u"}}
    ]},

    %% Authentication Query
    {authquery, [
      {collection, "mqtt_user"},
      {password_field, "password"},
      %% Hash Algorithm: plain, md5, sha, sha256, pbkdf2?
      {password_hash, sha256},
      {selector, {"username", "%u"}}
    ]},

    %% ACL Query: "%u" = username, "%c" = clientid
    {aclquery, [
      {collection, "mqtt_acl"},
      {selector, {"username", "%u"}}
    ]},

    %% If no ACL rules matched, return...
    {acl_nomatch, deny}

  ]}
].
```

9.7.2 MongoDB Database

9.7.3 User Collection

```
{
    username: "user",
    password: "password hash",
    is_superuser: boolean (true, false),
    created: "datetime"
}
```

For example:

```
db.mqtt_user.insert({username: "test", password: "password hash", is_superuser: false})
db.mqtt_user:insert({username: "root", is_superuser: true})
```

9.7.4 ACL Collection

```
{
    username: "username",
    clientid: "clientid",
    publish: ["topic1", "topic2", ...],
    subscribe: ["subtop1", "subtop2", ...],
    pubsub: ["topic/#", "topic1", ...]
}
```

For example:

```
db.mqtt_acl.insert({username: "test", publish: ["t/1", "t/2"], subscribe: ["user/%u", "client/%c"]})
db.mqtt_acl.insert({username: "admin", pubsub: ["#"]})
```

9.7.5 Load emqttd_plugin_mongo Plugin

```
./bin/emqttd_ctl plugins load emqttd_plugin_mongo
```

9.8 emqttd_stomp - STOMP Protocol

Support STOMP 1.0/1.1/1.2 clients to connect to emqttd broker and communicate with MQTT Clients.

9.8.1 Configure emqttd_stomp/etc/plugin.config

Note: Default Port for STOMP Protocol: 61613

```
[
  {emqttd_stomp, [

    {default_user, [
        {login,    "guest"},
        {passcode, "guest"}
```

```
    ]},

    {allow_anonymous, true},

    %%TODO: unused...
    {frame, [
      {max_headers,       10},
      {max_header_length, 1024},
      {max_body_length,   8192}
    ]},

    {listeners, [
      {emqttd_stomp, 61613, [
        {acceptors,   4},
        {max_clients, 512}
      ]}
    ]}

  ]}
].
```

9.8.2 Load emqttd_stomp Plugin

```
./bin/emqttd_ctl plugins load emqttd_stomp
```

9.9 emqttd_sockjs - STOMP/SockJS Plugin

emqttd_sockjs plugin enables web browser to connect to emqttd broker and communicate with MQTT clients.

Note: Default TCP Port: 61616

9.9.1 Configure emqttd_sockjs

```
[
  {emqttd_sockjs, [

    {sockjs, []},

    {cowboy_listener, {stomp_sockjs, 61616, 4}},

  ]}
].
```

9.9.2 Load emqttd_sockjs Plugin

Note: emqttd_stomp Plugin required.

```
./bin/emqttd_ctl plugins load emqttd_stomp

./bin/emqttd_ctl plugins load emqttd_sockjs
```

9.9.3 SockJS Demo Page

http://localhost:61616/index.html

9.10 emqttd_recon - Recon Plugin

The plugin loads recon library on a running emqttd broker. Recon libray helps debug and optimize an Erlang application.

9.10.1 Load emqttd_recon Plugin

```
./bin/emqttd_ctl plugins load emqttd_recon
```

9.10.2 Recon CLI

```
./bin/emqttd_ctl recon

recon memory                #recon_alloc:memory/2
recon allocated             #recon_alloc:memory(allocated_types, current|max)
recon bin_leak              #recon:bin_leak(100)
recon node_stats            #recon:node_stats(10, 1000)
recon remote_load Mod       #recon:remote_load(Mod)
```

9.11 emqttd_reloader - Reloader Plugin

Erlang Module Reloader for Development

Note: Don't load the plugin in production!

9.11.1 Load emqttd_reloader Plugin

```
./bin/emqttd_ctl plugins load emqttd_reloader
```

9.11.2 reload CLI

```
./bin/emqttd_ctl reload

reload <Module>             # Reload a Module
```

9.12 Plugin Development Guide

9.12.1 Create a Plugin Project

Clone emqttd source from github.com:

```
git clone https://github.com/emqtt/emqttd.git
```

Create a plugin project under 'plugins' folder:

```
cd plugins && mkdir emqttd_my_plugin

cd emqttd_my_plugin && rebar create-app appid=emqttd_my_plugin
```

Template Plugin: https://github.com/emqtt/emqttd_plugin_template

9.12.2 Register Auth/ACL Modules

emqttd_auth_demo.erl - demo authentication module:

```
-module(emqttd_auth_demo).

-behaviour(emqttd_auth_mod).

-include("../../../include/emqttd.hrl").

-export([init/1, check/3, description/0]).

init(Opts) -> {ok, Opts}.

check(#mqtt_client{client_id = ClientId, username = Username}, Password, _Opts) ->
    io:format("Auth Demo: clientId=~p, username=~p, password=~p~n",
              [ClientId, Username, Password]),
    ok.

description() -> "Demo Auth Module".
```

emqttd_acl_demo.erl - demo ACL module:

```
-module(emqttd_acl_demo).

-include("../../../include/emqttd.hrl").

%% ACL callbacks
-export([init/1, check_acl/2, reload_acl/1, description/0]).

init(Opts) ->
    {ok, Opts}.

check_acl({Client, PubSub, Topic}, Opts) ->
    io:format("ACL Demo: ~p ~p ~p~n", [Client, PubSub, Topic]),
    allow.

reload_acl(_Opts) ->
    ok.

description() -> "ACL Module Demo".
```

emqttd_plugin_template_app.erl - Register the auth/ACL modules:

```
ok = emqttd_access_control:register_mod(auth, emqttd_auth_demo, []),
ok = emqttd_access_control:register_mod(acl, emqttd_acl_demo, []),
```

9.12.3 Register Callbacks for Hooks

The plugin could register callbacks for hooks. The hooks will be run by the broker when a client connected/disconnected, a topic subscribed/unsubscribed or a message published/delivered:

Name	Description
client.connected	Run when a client connected to the broker successfully
client.subscribe	Run before a client subscribes topics
client.subscribe.after	Run after a client subscribed topics
client.unsubscribe	Run when a client unsubscribes topics
message.publish	Run when a message is published
message.delivered	Run when a message is delivered
message.acked	Run when a message(qos1/2) is acked
client.disconnected	Run when a client is disconnnected

emqttd_plugin_template.erl for example:

```
%% Called when the plugin application start
load(Env) ->
    emqttd:hook('client.connected', fun ?MODULE:on_client_connected/3, [Env]),
    emqttd:hook('client.disconnected', fun ?MODULE:on_client_disconnected/3, [Env]),
    emqttd:hook('client.subscribe', fun ?MODULE:on_client_subscribe/3, [Env]),
    emqttd:hook('client.subscribe.after', fun ?MODULE:on_client_subscribe_after/3, [Env]),
    emqttd:hook('client.unsubscribe', fun ?MODULE:on_client_unsubscribe/3, [Env]),
    emqttd:hook('message.publish', fun ?MODULE:on_message_publish/2, [Env]),
    emqttd:hook('message.delivered', fun ?MODULE:on_message_delivered/3, [Env]),
    emqttd:hook('message.acked', fun ?MODULE:on_message_acked/3, [Env]).
```

9.12.4 Register CLI Modules

emqttd_cli_demo.erl:

```
-module(emqttd_cli_demo).

-include("../../../include/emqttd_cli.hrl").

-export([cmd/1]).

cmd(["arg1", "arg2"]) ->
    ?PRINT_MSG("ok");

cmd(_) ->
    ?USAGE([{"cmd arg1 arg2", "cmd demo"}]).
```

emqttd_plugin_template_app.erl - register the CLI module to emqttd broker:

```
emqttd_ctl:register_cmd(cmd, {emqttd_cli_demo, cmd}, []).
```

There will be a new CLI after the plugin loaded:

```
./bin/emqttd_ctl cmd arg1 arg2
```

CHAPTER 10

Tuning Guide

Tuning the Linux Kernel, Networking, Erlang VM and emqttd broker for one million concurrent MQTT connections.

10.1 Linux Kernel Tuning

The system-wide limit on max opened file handles:

```
# 2 million system-wide
sysctl -w fs.file-max=2097152
sysctl -w fs.nr_open=2097152
echo 2097152 > /proc/sys/fs/nr_open
```

The limit on opened file handles for current session:

```
ulimit -n 1048576
```

10.1.1 /etc/sysctl.conf

Add the 'fs.file-max' to /etc/sysctl.conf, make the changes permanent:

```
fs.file-max = 1048576
```

10.1.2 /etc/security/limits.conf

Persist the limits on opened file handles for users in /etc/security/limits.conf:

```
*      soft   nofile      1048576
*      hard   nofile      1048576
```

10.2 Network Tuning

Increase number of incoming connections backlog:

```
sysctl -w net.core.somaxconn=32768
sysctl -w net.ipv4.tcp_max_syn_backlog=16384
sysctl -w net.core.netdev_max_backlog=16384
```

Local Port Range:

```
sysctl -w net.ipv4.ip_local_port_range="1000 65535"
```

Read/Write Buffer for TCP connections:

```
sysctl -w net.core.rmem_default=262144
sysctl -w net.core.wmem_default=262144
sysctl -w net.core.rmem_max=16777216
sysctl -w net.core.wmem_max=16777216
sysctl -w net.core.optmem_max=16777216

#sysctl -w net.ipv4.tcp_mem='16777216 16777216 16777216'
sysctl -w net.ipv4.tcp_rmem='1024 4096 16777216'
sysctl -w net.ipv4.tcp_wmem='1024 4096 16777216'
```

Connection Tracking:

```
sysctl -w net.nf_conntrack_max=1000000
sysctl -w net.netfilter.nf_conntrack_max=1000000
sysctl -w net.netfilter.nf_conntrack_tcp_timeout_time_wait=30
```

The TIME-WAIT Buckets Pool, Recycling and Reuse:

```
net.ipv4.tcp_max_tw_buckets=1048576

# Enable fast recycling of TIME_WAIT sockets.  Enabling this
# option is not recommended for devices communicating with the
# general Internet or using NAT (Network Address Translation).
# Since some NAT gateways pass through IP timestamp values, one
# IP can appear to have non-increasing timestamps.
# net.ipv4.tcp_tw_recycle = 1
# net.ipv4.tcp_tw_reuse = 1
```

Timeout for FIN-WAIT-2 sockets:

```
net.ipv4.tcp_fin_timeout = 15
```

10.3 Erlang VM Tuning

Tuning and optimize the Erlang VM in etc/vm.args file:

```
## max number of erlang processes
+P 2097152

## Sets the maximum number of simultaneously existing ports for this system
+Q 1048576

## Increase number of concurrent ports/sockets, deprecated in R17
-env ERL_MAX_PORTS 1048576

-env ERTS_MAX_PORTS 1048576

## Mnesia and SSL will create temporary ets tables.
-env ERL_MAX_ETS_TABLES 1024

## Tweak GC to run more often
-env ERL_FULLSWEEP_AFTER 1000
```

10.4 emqttd broker

Tune the acceptor pool, max_clients limit and sockopts for TCP listener in etc/emqttd.config:

```
{mqtt, 1883, [
    %% Size of acceptor pool
    {acceptors, 64},

    %% Maximum number of concurrent clients
    {max_clients, 1000000},

    %% Socket Access Control
    {access, [{allow, all}]},

    %% Connection Options
    {connopts, [
        %% Rate Limit. Format is 'burst, rate', Unit is KB/Sec
        %% {rate_limit, "100,10"} %% 100K burst, 10K rate
    ]},
    ...
```

10.5 Client Machine

Tune the client machine to benchmark emqttd broker:

```
sysctl -w net.ipv4.ip_local_port_range="500 65535"
sysctl -w fs.file-max=1000000
echo 1000000 > /proc/sys/fs/nr_open
ulimit -n 100000
```

10.6 emqtt_benchmark

Test tool for concurrent connections: http://github.com/emqtt/emqtt_benchmark

CHAPTER 11

Changes

11.1 Version 1.1.3

Release Date: 2016-08-19

Support './bin/emqttd_ctl users list' CLI (#621)

Cannot publish payloads with a size of the order 64K using WebSockets (#643)

Optimize the procedures that retrieve the Broker version and Borker description in the tick timer (PR#627)

Fix SSL certfile, keyfile config (#651)

11.2 Version 1.1.2

11.3 Version 1.1.2

Release Date: 2016-06-30

Upgrade mysql-otp driver to 1.2.0 (#564, #523, #586, #596)

Fix WebSocket Client Leak (PR #612)

java.io.EOFException using paho java client (#551)

Send message from paho java client to javascript client (#552)

Compatible with the Qos0 PUBREL packet (#575)

Empty clientId with non-clean session accepted (#599)

Update docs to fix typos (#601, #607)

11.4 Version 1.1.1

Release Date: 2016-06-04

Compatible with the Qos0 PUBREL packet (#575)

phpMqtt Client Compatibility (#572)

java.io.EOFException using paho java client (#551)

11.5 Version 1.1

Release Date: 2016-06-01

11.5.1 Highlights

Upgrade eSockd library to 4.0 and Support IPv6

Support to listen on specific IP Address:

```
{mqtt, {"192.168.1.20", 1883}, [
    ...
]},
```

Add MongoDB, HTTP Authentication/ACL Plugins

Upgrade MySQL, PostgreSQL, Redis Plugins to support superuser authentication and avoid SQL Injection

11.5.2 Enhancements

Allow human-friendly IP addresses (PR#395)

File operation error: emfile (#445)

emqttd_plugin_mongo not found in emqttd (#489)

emqttd_plugin_mongo Error While Loading in emqttd (#505)

Feature request: HTTP Authentication (#541)

Compatible with the Qos0 PUBREL packet (#575)

11.5.3 Bugfix

Bugfix: function_clause exception occurs when registering a duplicated authentication module (#542)

Bugfix: ./emqttd_top msg_q result: {“init terminating in do_boot”,{undef,[{etop,start,[],[]},{init,start_it,1,[]},{init,start_em,1,[]}]}} (#557)

11.5.4 Tests

111 common test cases.

11.5.5 Dashboard Plugin

WebSocket Page: Support ‘Clean Session’, Qos, Retained parameters (emqttd_dashboard#52)

Upgrade eSockd library to 4.0, Show OTP Release on Overview Page (emqttd_dashboard#61)

Changing dashboard credentials for username authentication (emqttd_dashboard#56)

Add ‘./bin/emqttd_ctl admins’ CLIsupport to add/delete admins

11.5.6 HTTP Auth Plugin

Authentication/ACL by HTTP API: https://github.com/emqtt/emqttd_auth_http

11.5.7 MongoDB Plugin

Upgrade Erlang MongoDB driver to v1.0.0

Support superuser authentication

Support ACL (emqttd_plugin_mongo#3)

11.5.8 MySQL Plugin

Support superuser authentication

Use parameterized query to avoid SQL Injection

11.5.9 Postgre Plugin

Support superuser authentication

Use parameterized query to avoid SQL Injection

11.5.10 Redis Plugin

Support superuser authentication

Support ClientId authentication by '%c' variable

11.5.11 Reloader Plugin

Reload modified modules during development automatically.

11.6 Version 1.0.3

Release Date: 2016-05-23

eSockd 3.2

MochiWeb 4.0.1

11.7 Version 1.0.2

Release Date: 2016-05-04

Issue#534 - './bin/emqttd_ctl vm' - add 'port/count', 'port/limit' statistics

Issue#535 - emqttd_client should be terminated properly even if exception happened when sending data

PR#519 - The erlang '-name' requires the fully qualified host name

emqttd_reloader plugin - help reload modified modules during development.

11.8 Version 1.0.1

Release Date: 2016-04-16

PR#515 - Fix '$queue' pubsub, add 'pubsub_queue' test and update docs

11.9 Version 1.0 (The Seven Mile Journey)

Release Date: 2016-04-13

Release Name: The Seven Mile Journey

We finally released Version 1.0 (The Seven Mile Journey) with full documentation after two years' development and more than fifty iterations.

The emqttd 1.0 implements a fully-featured, scalable, distributed and extensible open-source MQTT broker for IoT, M2M and Mobile applications:

1. Full MQTT V3.1/3.1.1 Protocol Specifications Support
2. Massively scalable - Scaling to 1 million connections on a single server
3. Distributed - Route MQTT Messages among clustered or bridged broker nodes
4. Extensible - LDAP, MySQL, PostgreSQL, Redis Authentication/ACL Plugins

11.9.1 Bugfix and Enhancements

Possible race condition using emqttd_cm (#486)

Improve the design of retained message expiration (#503)

Do not expire the retained messages from $SYS/# topics (#500)

11.9.2 Documentation

http://emqtt.io/docs

http://docs.emqtt.com/

11.9.3 Thanks

Thank Ericsson for the Great Erlang/OTP Platform (http://erlang.org/)!

Contributors on GitHub: @callbay @lsxredrain @hejin1026 @desoulter @turtleDeng @Hades32 @huangdan @phanimahesh @dvliman @Prots @joaohf

Partners: EACG (http://eacg.de/)

Favorite Band: The Seven Mile Journey (http://www.thesevenmilejourney.dk/)

11.10 Version 0.17.1-beta

Release Date: 2016-03-22

11.10.1 Enhancements

Time unit of session 'expired_after' changed to minute. (#479)

11.10.2 Dashboard

Code Review and improve the design of Dashboard.

11.11 Version 0.17.0-beta

Release Date: 2016-03-15

11.11.1 Highlights

Installation and Configuration Guide released on http://docs.emqtt.com

Improve and Consolidate the design of Hook, Server, PubSub and Router

Upgrade the [Web Dashboard](https://github.com/emqtt/emqttd_dashboard) to support pagination

Bridge emqttd broker to another emqttd broker & emqttd to mosquitto bridge (#438)

11.11.2 Enhancements

emqttd_ctl: better error message (#450)

./bin/emqttd_ctl: add 'routes' command:

```
routes list             # List all routes
routes show <Topic>     # Show a route
```

Add 'backend_subscription' table and support static subscriptions (emqttd_backend)

Add 'retained_message' table and refactor emqttd_retainer module (emqttd_backend)

A New Hook and Callback Design (emqttd_hook)

Add PubSub, Hooks APIs to emqttd module (emqttd)

Move start_listeners/0, stop_listeners/0 APIs to emqttd_app module (emqttd_app)

11.11.3 Tests

Add 100+ common test cases.

11.11.4 Plugins

Upgrade Dashboard, Redis, Stomp and Template Plugins

11.12 Version 0.16.0-beta

Release Date: 2016-02-16

11.12.1 Highlights

Licensed under the Apache License, Version 2.0 Now.

Improve the design of cluster, support to join or leave the cluster (#449):

```
$ ./bin/emqttd_ctl cluster
cluster join <Node>                     #Join the cluster
cluster leave                           #Leave the cluster
cluster remove <Node>                   #Remove the node from cluster
cluster status                          #Cluster status
```

Improve the design of Trie and Route, only the wildcard topics stored in Trie.

Common Test to replace EUnit.

11.12.2 Enhancements

mqtt_message record: add 'sender' field (#440)

refactor the emqttd, emqttd_time, emqttd_opts, emqttd_node modules.

11.12.3 Bugfix

noproc error when call to gen_server2:call(false, {add_route,Topic,<0.685.0>}, infinity) (#446)

11.12.4 Plugins

Changed the license of all plugins.

11.13 Version 0.15.0-beta

Release Date: 2016-01-31

11.13.1 Highlights

Optimize for Push Application, 500K+ Subscribers to a Topic.

Optimization for Route ETS insertion (#427)

Priority Message Queue for Persistent Session (#432)

Add Redis, MongoDB Plugins (#417)

11.13.2 Enhancements

Username/Password Authentication: Support to configure default users (#428)

Improve CLI Commands: pubsub, bridges, trace (#429)

emqttd_mod_subscription: fix client_connected/3

emqttd_auth_mod: add passwd_hash/2 function

priority_queue: add plen/2, out/2 functions

11.13.3 Bugfix

Fix dequeue/1 of emqttd_bridge...

Add emqttd:seed_now/0 function

11.13.4 Plugins

emqttd_plubin_mysql: Changed mysql driver to mysql-otp

emqttd_plugin_pgsql: Integrate with ecpool

emqttd_plugin_redis: First release

emqttd_plugin_mongo: First release

11.14 Version 0.14.1-beta

Release Date: 2015-12-28

Bugfix: emqttd_ws_client.erl: Unexpected Info: {'EXIT',<0.27792.18>,{shutdown,destroy}} (#413)

Improve: fix spec errors found by dialyzer

11.15 Version 0.14.0-beta

Release Date: 2015-12-18

11.15.1 Highlights

Scaling to 1.3 Million Concurrent MQTT Connections on a 12 Core, 32G CentOS server.

New PubSub, Router Design (#402). Prepare for scaling to 10 millions on one cluster.

11.15.2 Enhancements

Improve the gproc_pool usage with a general emqttd_pool_sup

Improve the design of emqttd_pubsub, add a new emqttd_router module

Improve the design of the whole supervisor tree

Route aging mechanism to remove the topics that have no subscriptions

Improve the dashboard, mysql, pgsql, stomp, sockjs plugins

Add 'topics', 'subscriptions' admin commands

Avoid using mnesia table index and mnesia:index_read API to lower CPU usage

Subscribe timeout exception (#366)

Long Delay on Multiple Topic Subscription (#365)

Subscriptions persistence (#344)

emqttd_ctl: 'subscriptions' command to force clients to subscribe some topics (#361)

11.15.3 Bugfix

emqttd_sm: spec of lookup_session/1 is not right BUG (#411)

Observer application should be removed from reltool.config for 'wx' app is not available (#410)

11.15.4 Benchmark

1.3 million concurrent MQTT connections on a 12 Core, 32G CentOS Server, consume about 15G Memory and 200% CPU.

11.16 Version 0.13.1-beta

Release Date: 2015-11-28

Bugfix: Plugin pathes error under windows (#387)

Improve: Too many error logs "[error] Session ..... Unexpected EXIT: client_pid=<0.14137.35>, exit_pid=<0.30829.22>, reason=nop..." (#383)

Improve: Define QOS0/1/2, Pooler Error (PR#382)

Improve: High CPU load when 400K unstable mobile connections (#377)

BugFix: emqttd_plugin_pgsql - error using same query with latest update plugin (pgsql#5)

11.17 Version 0.13.0-beta

Release Date: 2015-11-08

11.17.1 Highlights

Rate Limiting based on [Token Bucket](https://en.wikipedia.org/wiki/Token_bucket) and [Leaky Bucket](https://en.wikipedia.org/wiki/Leaky_bucket#The_Leaky_Bucket_Algorithm_as_a_Meter) Algorithm

Upgrade eSockd and MochiWeb libraries to support Parameterized Connection Module

Improve emqttd_client to support fully asynchronous socket networking

11.17.2 Enhancements

Protocol Compliant - Session Present Flag (#163)

Compilation fails if repo is cloned with a different name (#348)

emqttd_client: replace gen_tcp:send with port_command (#358)

TCP sndbuf, recbuf, buffer tuning (#359)

emqttd_client.erl to handle 'inet_async', 'inet_reply' properly (#360)

Refator the [client/session management design](https://github.com/emqtt/emqttd/blob/master/doc/design/ClientSession.md)

11.17.3 Bugfix

Cannot kick transient client out when clientId collision (#357)

Fix the order of emqttd_app:start_server/1 (#367)

emqttd_session:subscribe/2 will crash (#374)

11.17.4 Benchmark

[benchmark for 0.13.0 release](https://github.com/emqtt/emqttd/wiki/benchmark-for-0.13.0-release)

3.1G memory and 50+ CPU/core:

```
Connections: 250K
Subscribers: 250K
Topics:      50K
Qos1 Messages/Sec In:  4K
Qos1 Messages/Sec Out: 20K
Traffic In(bps):  12M+
Traffic Out(bps): 56M+
```

11.18 Version 0.12.3-beta

Release Date: 2015-10-22

Bugfix: emqttd_sysmon crasher for 'undefined' process_info (#350)

Bugfix: emqttd_client: catch parser exception (#353)

11.19 Version 0.12.2-beta

Release Date: 2015-10-16

Bugfix: Retained messages should not be expired if 'broker.retained.expired_after = 0' (#346)

11.20 Version 0.12.1-beta

Release Date: 2015-10-15

Highlight: Release for Bugfix and Code Refactor.

Feature: Retained message expiration (#182)

Improve: '$SYS/#' publish will not match '#' or '+/#' (#68)

Improve: Add more metrics and ignore '$SYS/#' publish (#266)

Improve: emqttd_sm should be optimized for clustered nodes may be crashed (#282)

Improve: Refactor emqttd_sysmon and suppress 'monitor' messages (#328)

Task: benchmark for 0.12.0 release (#225)

Benchmark: About 900K concurrent connections established on a 20Core, 32G CentOS server.

11.21 Version 0.12.0-beta

Release Date: 2015-10-08

11.21.1 Highlights

Enhance the **emqttd_ctl** module to allow plugins to register new commands (#256)

Add [emqttd_recon plugin](https://github.com/emqtt/emqttd_recon) to debug/optimize the broker (#235)

Add **'./bin/emqttd_ctl broker pubsub'** command to check the status of core pubsub processes

Add **'./bin/emqttd_top'** command(like etop) to show the top 'msg_q', 'reductions', 'memory' or 'runtime' processes

'rel/files/emqttd.config.production' for production deployment(default)

'rel/files/emqttd.config.development' for development deployment

11.21.2 Enhancements

Qos1/2 messages will not be dropped under unstable mobile network (#264)

emqttd_session:subscribe/2, emqttd_session:unsubscribe/2 APIs should be asynchronous (#292)

etc/emqttd.config: 'idle_timeout' option to close the idle client(socket connected but no 'CONNECT' frame received)

etc/emqttd.config: 'unack_retry_interval' option for redelivering Qos1/2 messages

How to monitor large 'message_queue_len' (#283)

11.21.3 Bugfix

Behaviour emqttd_auth_mod is missing init callback (#318)

11.21.4 Benchmark

Write a new [benchmark tool](https://github.com/emqtt/emqtt_benchmark) to benchmark this release

Hw requirements - 5K users, 25-50 msgs/sec, QoS=1 (#209)

Supported Number of Connections Greatly Reduced When Clients are Subscribing (#324)

11.22 Version 0.11.0-beta

Release Date: 2015-09-25

Highlight: Rebar to manage plugin dependencies.

Highlight: [Stomp](https://github.com/emqtt/emqttd_stomp) and [SockJS](https://github.com/emqtt/emqttd_sockjs) Plugins!

Improve: add rel/files/emqttd.config.development|production.

Improve: rel/reltool.config.script to release deps of plugin.

Improve: persist mnesia schema on slave nodes.

Improve: use timer:seconds/1 api.

Improve: The binary release will be compiled with R18.1 now.

Bugfix: issue#306 - emqttd_cm should unregister the duplicated client

Bugfix: issue#310 - usage of emqttd_ctl error: 'session list' should be 'sessions list'

Bugfix: issue#311 - './bin/emqttd_ctl sessions list' error

Bugfix: issue#312 - unsubcribe will lead to crash if emqttd_plugin_template plugin loaded

11.23 Version 0.10.4-beta

Release Date: 2015-09-18

Optimize session management and upgrade eSockd library to 2.7.1

[Benchmark for 0.10.4 release](https://github.com/emqtt/emqttd/wiki/benchmark-for-0.10.4-release)

Improve: issue#294 - [error] failed to start connection on 0.0.0.0:1883 - enotconn

Improve: issue#297 - How do I allow user with some pattern to access topic with some pattern?

Bugfix: issue#291 - "./bin/emqttd attach ..." cannot work

Bugfix: issue#284 - Should not use erlang:list_to_atom/1 in emqttd_vm.erl

11.24 Version 0.10.3-beta

Release Date: 2015-08-30

Bugfix: issue#271 - add emqttd_ws_client:subscribe/2 function

Bugfix: issue#269 - bin/emqttd Syntax error on ubuntu

Improve: issue#265 - client under unstable mobile network generate a lot of logs

11.25 Version 0.10.2-beta

Release Date: 2015-08-26

Improve: issue#257 - After the node name changed, the broker cannot restart for mnesia schema error.

11.26 Version 0.10.1-beta

Release Date: 2015-08-25

Bugfix: issue#259 - when clustered the emqttd_dashboard port is close, and the 'emqttd' application cannot stop normally.

Feature: issue#262 - Add 'http://host:8083/mqtt/status' Page for health check

11.27 Version 0.10.0-beta

Release Date: 2015-08-20

[Web Dashboard](https://github.com/emqtt/emqttd_dashboard) and [MySQL](https://github.com/emqtt/emqttd_plugin_mysql), [PostgreSQL](https://github.com/emqtt/emqttd_plugin_pgsql) Authentication/ACL Plugins!

Highlight: Web Dashboard to monitor Statistics, Metrics, Clients, Sessions and Topics of the broker.

Highlight: JSON/HTTP API to query all clients connected to broker.

Highlight: A new [Plugin Design](https://github.com/emqtt/emqttd/wiki/Plugin%20Design) and a [Template project](https://github.com/emqtt/emqttd_plugin_template) for plugin development.

Highlight: Authentication/ACL with MySQL, PostreSQl databases (#194, #172)

Feature: Session Statistics including inflight_queue, message_queue, message_dropped, awaiting_rel, awaiting_ack, awaiting_comp (#213)

Feature: Cookie based authentication for MQTT over websocket connections (#231)

Feature: Get all clients connected to the broker (#228, #230, #148, #129)

Feature: "./bin/emqttd_ctl clients show ClientId" to query client status (#226)

Feature: "./bin/emqttd_ctl clients kick ClientId" to kick out a client

Feature: "./bin/emqttd_ctl sessions list" to show all sessions

Feature: "./bin/emqttd_ctl sessions show ClientId" to show a session

Feature: Erlang VM metrics monitor with Web Dashboard (#59)

Improve: Too many "inflight queue is full!" log when session is overloaded (#247)

Improve: There are two many "MQueue(~s) drop ~s" logs if the message queue of session is small (#244)

Improve: gen_server2(from RabbitMQ) to improve emqttd_session, emqttd_pubsub

Improve: Makefile to build plugins

Bugfix: emqttd_broker:unhook/2 cannot work (#238)

Bugfix: emqttd plugin cannot include_lib("emqttd/include/emqttd.hrl") (#233)

Bugfix: Too many 'Session ~s cannot find PUBACK' logs (#212)

Bugfix: emqttd_pooler cannot work

11.28 Version 0.9.3-alpha

Release Date: 2015-07-25

Wiki: [Bridge](https://github.com/emqtt/emqttd/wiki/Bridge)

Improve: emqttd_protocol.hrl to define 'QOS_I'

Improve: emqttd_pubsub to add subscribe/2 API

Improve: ./bin/emqttd_ctl to support new bridges command

Bugfix: issue #206 - Cannot bridge two nodes

11.29 Version 0.9.2-alpha

Release Date: 2015-07-18

Improve: issue #196 - Add New Hook 'client.subscribe.after'

11.30 Version 0.9.1-alpha

Release Date: 2015-07-10

Bugfix: issue #189 - MQTT over WebSocket(SSL) cannot work?

Bugfix: issue #193 - 'client.ack' hook should be renamed to 'message.acked', and called by emqttd_broker:foreach_hooks

11.31 Version 0.9.0-alpha

Release Date: 2015-07-09

[Session, Queue, Inflight Window, Hooks, Global MessageId and More Protocol Compliant](https://github.com/emqtt/emqttd/releases/tag/0.9.0-alpha) Now!

Feature: Session/Queue/Inflight Window Design (#145).

Feature: Support to resume a persistent session on other clustered node.

Feature: Support alarm management.

Feature: emqttd_guid to generate global unique message id.

Feature: Hooks for message pub/ack.

Feature: Protocol compliant - message ordering, timeout and retry.

Improve: Every client will start_link a session process, whether or not the client is persistent.

Improve: etc/emqttd.config to support more session, queue configuration.

Improve: issue #179 - Max offline message queue {max_queue, 100} meaning.

Improve: issue #180 - Should change project structure for other projects maybe depend on 'emqttd'. Merge emqtt, emqttd apps.

Improve: issue #185 - PacketId and MessageId: the broker should generate global unique message id.

Improve: issue #187 - etc/emqttd.config to support https listener

Improve: issue #186 - emqttd_cm to store client details

Improve: issue #174 - add 'from' field to mqtt_message record.

Improve: issue #170 - $SYS Topics should support alarms.

Improve: issue #169 - Add More [Hooks](https://github.com/emqtt/emqttd/wiki/Hooks-Design)

Improve: issue #167 - Inflight window to assure message ordering.

Improve: issue #166 - Message delivery timeout and retry.

Improve: issue #143 - Qos1, Qos2 PubSub message timeout.

Improve: issue #122 - Labeling message with unique id. emqttd_guid module to generate global unique msgid.

Improve: emqttd_bridge to support pending message queue, and fix the wrong Qos design.

Improve: mqtt_message record to add 'msgid', 'from' and 'sys' fields.

Change: Add emqttd_mqueue, emqttd_guid, emqttd_alarm modules.

Bugfix: issue #184 - emqttd_stats:setstats is not right.

Bugfix: Closed issues #181, #119.

Tests: fix the parser, acl test cases.

11.32 Version 0.8.6-beta

Release Date: 2015-06-17

Bugfix: issue #175 - publish Will message when websocket is closed without 'DISCONNECT' packet

11.33 Version 0.8.5-beta

Release Date: 2015-06-10

Bugfix: issue #53 - client will receive duplicate messages when overlapping subscription

11.34 Version 0.8.4-beta

Release Date: 2015-06-08

Bugfix: issue #165 - duplicated message when publish 'retained' message to persistent client

11.35 Version 0.8.3-beta

Release Date: 2015-06-05

Bugfix: issue #158 - should queue:in new message after old one dropped

Bugfix: issue #155 - emqtt_parser.erl: parse_topics/3 should reverse topics

Bugfix: issue #149 - Forget to merge plugins/emqttd_auth_mysql from 'dev' branch to 'master' in 0.8.x release

11.36 Version 0.8.2-alpha

Release Date: 2015-06-01

Bugfix: issue #147 - WebSocket client cannot subscribe queue '$Q/queue/${clientId}'

Bugfix: issue #146 - emqttd_auth_ldap: fill(Username, UserDn) is not right

11.37 Version 0.8.1-alpha

Release Date: 2015-05-28

Client [Presence](https://github.com/emqtt/emqttd/wiki/Presence) Support and [$SYS Topics](https://github.com/emqtt/emqttd/wiki/$SYS-Topics) Redesigned!

Bugfix: issue #138 - when client disconnected normally, broker will not publish disconnected $SYS message

Bugfix: fix websocket url in emqttd/priv/www/websocket.html

Improve: etc/emqttd.config to allow websocket connections from any hosts

Improve: rel/reltool.config to exclude unnecessary apps.

11.38 Version 0.8.0-alpha

Release Date: 2015-05-25

[Hooks](https://github.com/emqtt/emqttd/wiki/Hooks%20Design), Modules and [Plugins](https://github.com/emqtt/emqttd/wiki/Plugin%20Design) to extend the broker Now!

Plugin: emqttd_auth_mysql - MySQL authentication plugin (issues #116, #120)

Plugin: emqttd_auth_ldap - LDAP authentication plugin

Feature: emqttd_broker to support Hooks API

Feature: issue #111 - Support 'Forced Subscriptions' by emqttd_mod_autosub module

Feature: issue #126 - Support 'Rewrite rules' by emqttd_mod_rewrite module

Improve: Support hooks, modules to extend the broker

Improve: issue #76 - dialyzer check

Improve: 'Get Started', 'User Guide', 'Developer Guide' Wiki

Improve: emqtt_topic to add join/1, feed_var/3, is_queue/1

Improve: emqttd_pooler to execute common tasks

Improve: add emqttd_sm_sup module, and use 'hash' gproc_pool to manage sessions

Tests: add more test cases for 'emqttd' app

11.39 Version 0.7.1-alpha

Release Date: 2015-05-04

Add doc/design/* and merge doc/* to github Wiki

Bugfix: issue #121 - emqttd cluster issuse

Bugfix: issue #123 - emqttd:unload_all_plugins/0 cannot unload any plugin

Bugfix: fix errors found by dialyzer

11.40 Version 0.7.0-alpha

Release Date: 2015-05-02

[MQTT over WebSocket(SSL)](https://github.com/emqtt/emqttd/wiki/MQTT-Over-WebSocket) Now!

[Plugin Achitecture](https://github.com/emqtt/emqttd/wiki/Plugin%20Design) based on OTP application

[Trace MQTT Packets or Messages](https://github.com/emqtt/emqttd/wiki/Trace%20Design) to log files

Feature: issue #40, #115 - WebSocket/SSL Support

Feature: issue #49, #105 - Plugin Architecture Support

Feature: issue #93 - Trace API Design

Improve: issue #109 - emqttd_broker should add subscribe, notify API

Improve: update README.md to add 'Goals', 'Contributors' chapters

Change: rename etc/app.config to etc/emqttd.config

Change: etc/emqttd.config changed

Bugfix: critical issue #54 - error when resume session!

Bugfix: issue #118 - error report when UNSUBSCRIBE with no topics

Bugfix: issue #117 - sys_interval = 0 config cannot work

Bugfix: issue #112 - Makefile to support build plugins

Bugfix: issue #96 - "make clean" cannot work

11.41 Version 0.6.2-alpha

Release Date: 2015-04-24

Bugfix: critical issue #54, #104, #106 - error when resume session

Improve: add emqttd_cm_sup module, and use 'hash' gproc_pool to register/unregister client ids

Improve: kick old client out when session is duplicated.

Improve: move mnesia dir config from etc/app.config to etc/vm.args

11.42 Version 0.6.1-alpha

Release Date: 2015-04-20

Integrate with [gproc library](https://github.com/uwiger/gproc) to support pool

Feature: issues#91 - should use worker_pool to handle some async work?

Feature: issues#95 - Topic filters in ACL rule should support 'eq' tag

Improve: issues#84 - emqttd_pubsub is redesigned again to protect mnesia transaction

Improve: issues#74 - ACL Support and update [ACL Design Wiki](https://github.com/emqtt/emqttd/wiki/ACL-Design)

11.43 Version 0.6.0-alpha

Release Date: 2015-04-17

ACL Support Now: [ACL-Design Wiki](https://github.com/emqtt/emqttd/wiki/ACL-Design)

Authentication with username, clientid Now: [Authentication Wiki](https://github.com/emqtt/emqttd/wiki/Authentication)

Seperate common MQTT library to 'emqtt' application

Redesign message pubsub, route and retain modules

Redesign mnesia database cluster

Feature: issues#47 - authentication, authorization support

Feature: issues#92 - merge emqttd_acl and emqttd_auth to emqttd_access_control

Feature: emqttd_acl_mod, emqttd_auth_mod behaviour to extend ACL, authentication

Feature: issues#85 - lager:info to log subscribe, unsubscribe actions

Feature: issues#77 - authentication with clientid, ipaddress

Improve: issues#90 - fix lager_file_backend log format, and rotate 10 log files

Improve: issues#88 - use '-mneisa_create', '-mnesia_replicate' attributes to init mneisa

Improve: issues#87 - record mqtt_user and mqtt_client is duplicated

Improve: issues#81 - redesign nodes cluster to support disc_copies mnesia tables

Improve: issues#80 - redesign emqttd_cm to handle more concurrent connections

Improve: issues#70 - how to handle connection flood? Now could support 2K+ CONNECT/sec

Change: redesign mnesia tables: message, topic, subscriber, trie, trie_node

Bugfix: issues#83 - emqttd_broker stats cannot work

Bugfix: issues#75 - careless about function name when emqttd_pubsub handle getstats message

11.44 Version 0.5.5-beta

Release Date: 2015-04-09

Bugfix: issue #75 - careless about function name when emqttd_pubsub handle getstats message.

Bugfix: issue #79 - cannot find topic_subscriber table after cluster with other nodes.

11.45 Version 0.5.4-alpha

Release Date: 2015-03-22

Benchmark this release on a ubuntu/14.04 server with 8 cores, 32G memory from QingCloud.com:

```
200K Connections,
30K Messages/Sec,
20Mbps In/Out Traffic,
200K Topics,
200K Subscribers,

Consumed 7G memory, 40% CPU/core
```

Benchmark code: https://github.com/emqtt/emqttd_benchmark

Change: rewrite emqttd_pubsub to handle more concurrent subscribe requests.

Change: ./bin/emqttd_ctl add 'stats', 'metrics' commands.

Bugfix: issue #71, #72

11.46 Version 0.5.3-alpha

Release Date: 2015-03-19

Bugfix: issues#72 - emqttd_cm, emqtt_sm ets:match_delete/2 with wrong pattern

11.47 Version 0.5.2-alpha

Release Date: 2015-03-18

Change: upgrade esockd to 2.1.0-alpha, do not tune socket buffer for mqtt connection.

11.48 Version 0.5.1-alpha

Release Date: 2015-03-13

Change: upgrade esockd to v1.2.0-beta, rename 'acceptor_pool' to 'acceptors'

11.49 Version 0.5.0-alpha

Release Date: 2015-03-12

RENAME 'emqtt' to 'emqttd'!

Support [Broker Bridge](https://github.com/emqtt/emqttd/wiki/Bridge-Design) Now!

Change: rename project from 'emqtt' to 'emqttd'

Change: lager:debug to dump RECV/SENT packets

Feature: emqttd_bridge, emqttd_bridge_sup to support broker bridge

Feature: emqtt_event to publish client connected/disconnected message to $SYS topics

Feature: ./bin/emqttd_ctl add more commands: listeners, broker, bridges, start_bridge, stop_bridge...

Feature: issue#57 - support to configure max packet size

Feature: issue#68 - if sys_interval = 0, emqttd_broker will not publish messages to $SYS/brokers/#

Bugfix: issue#67 - subscribe '#' to receive all messages

Bugfix: issue#64 - emqtt_app start/2: should wait_for_databases

Test: emqttd_topic_tests add more '_match_test'

11.50 Version 0.4.0-alpha

Release Date: 2015-03-10

Support [$SYS Topics of Broker](https://github.com/emqtt/emqttd/wiki/$SYS-Topics-of-Broker) Now!

Feature: emqtt_broker to publish version, uptime, datetime to $SYS/brokers/# topics

Feature: emqtt_broker to publish count of clients, sessions, suscribers to $SYS/brokers/# topics

Feature: emqtt_metrics to publish bytes, packets, messages metrics to $SYS/brokers/# topics

Feature: add include/emqtt_systop.hrl

Change: emqtt_cm to count current clients

Change: emqtt_sm to count current sessions

Change: emqtt_pubsub to count current topics and suscribers

Change: emqtt_pubsub to add create/1 API

Change: emqtt_pubsub dispatch/2 to return number of subscribers

Change: emqtt_pubsub to count 'dropped' messages

Change: emqtt_opts to add merge/2 function

Test: add emqtt_serialiser_tests.erl

11.51 Version 0.3.4-beta

Release Date: 2015-03-08

Bugfix: emqtt_serialiser.erl cannot serialise UNSUBACK packets

11.52 Version 0.3.3-beta

Release Date: 2015-03-07

Bugfix: emqtt_serialiser.erl cannot serialise PINGRESP issue#60

11.53 Version 0.3.2-beta

Release Date: 2015-03-05

Improve: merge emqttc serialiser, parser, packet

Add: emqtt_opts to merge socket options

11.54 Version 0.3.1-beta

Release Date: 2015-03-02

Feature: SSL Socket Support

Feature: issue#44 HTTP API should add Qos parameter

Bugfix: issue#52 emqtt_session crash

Bugfix: issue#53 sslsocket keepalive error

Upgrade: esockd to v0.2.0

Upgrade: mochiweb to v3.0.0

11.55 Version 0.3.0-beta

Release Date: 2015-01-19

Feature: HTTP POST API to support 'qos', 'retain' parameters

Feature: $SYS system topics support

Change: Rewrite emqtt_topic.erl, use '', '#', '+' to replace <<"">>, <<"#">>, <<"+">>

Change: fix emqtt_pubsub.erl to match '#', '+'

Tests: emqtt_topic_tests.erl add more test cases

11.56 Version 0.3.0-alpha

Release Date: 2015-01-08

NOTICE: Full MQTT 3.1.1 support now!

Feature: Passed org.eclipse.paho.mqtt.testing/interoperability tests

Feature: Qos0, Qos1 and Qos2 publish and suscribe

Feature: session(clean_sess=false) management and offline messages

Feature: redeliver awaiting puback/pubrec messages(doc: Chapter 4.4)

Feature: retain messages, add emqtt_server module

Feature: MQTT 3.1.1 null client_id support

Bugfix: keepalive timeout to send will message

Improve: overlapping subscription support

Improve: add emqtt_packet:dump to dump packets

Test: passed org.eclipse.paho.mqtt.testing/interoperability

Test: simple cluster test

Closed Issues: #22, #24, #27, #28, #29, #30, #31, #32, #33, #34, #36, #37, #38, #39, #41, #42, #43

11.57 Version 0.2.1-beta

Release Date: 2015-01-08

pull request 26: Use binaries for topic paths and fix wildcard topics

emqtt_pubsub.erl: fix wildcard topic match bug caused by binary topic in 0.2.0

Makefile: deps -> get-deps

rebar.config: fix mochiweb git url

tag emqtt release accoding to [Semantic Versioning](http://semver.org/)

max clientId length is 1024 now.

11.58 Version 0.2.0

Release Date: 2014-12-07

rewrite the project, integrate with esockd, mochiweb

support MQTT 3.1.1

support HTTP to publish message

11.59 Version 0.1.5

Release Date: 2013-01-05

Bugfix: remove QOS_1 match when handle PUBREL request

Bugfix: reverse word in emqtt_topic:words/1 function

11.60 Version 0.1.4

Release Date: 2013-01-04

Bugfix: fix “mosquitto_sub -q 2 .....” bug

Bugfix: fix keep alive bug

11.61 Version 0.1.3

Release Date: 2013-01-04

Feature: Support QOS2 PUBREC, PUBREL, PUBCOMP messages

Bugfix: fix emqtt_frame to encode/decoe PUBREC/PUBREL messages

11.62 Version 0.1.2

Release Date: 2012-12-27

Feature: release support like riak

Bugfix: use ?INFO/?ERROR to print log in tcp_listener.erl

11.63 Version 0.1.1

Release Date: 2012-09-24

Feature: use rebar to generate release

Feature: support retained messages

Bugfix: send will msg when network error

11.64 Version 0.1.0

Release Date: 2012-09-21

The first public release.

CHAPTER 12

Upgrade

12.1 Upgrade to 1.1.2

Note: 1.0+ releases can be upgraded to 1.1.2 smoothly

Steps:

1. Download and install emqttd-1.1.2 to the new directory, for example:

```
Old installation: /opt/emqttd_1_0_0/

New installation: /opt/emqttd_1_1_2/
```

2. Copy the 'etc/' and 'data/' from the old installation:

```
cp -R /opt/emqttd_1_0_0/etc/* /opt/emqttd_1_1_2/etc/

cp -R /opt/emqttd_1_0_0/data/* /opt/emqttd_1_1_2/data/
```

3. Copy the plugins/{plugin}/etc/* from the old installation if you loaded plugins.
4. Stop the old emqttd, and start the new one.

CHAPTER 13

License

Apache License Version 2.0