

Firehose: a Unified Message Bus for Infra Services

Matthew Treinish
mtreinisch@kortar.org
mtreinisch on Freenode

Jeremy Stanley
fungi@yuggoth.org
fungi on Freenode

May 11, 2017

<https://github.com/mtreinisch/firehose/tree/boston-summit>

OpenStack Infrastructure

Imagine a big, complicated diagram here.

Firehose

- ▶ An MQTT broker for the OpenStack community infrastructure
- ▶ Has anonymous, read-only access via MQTT on 1883/tcp
- ▶ SSL/TLS MQTT also available on 8883/tcp
- ▶ Websockets supported (but temporarily disabled)



MQTT

- ▶ Pub/sub messaging protocol
- ▶ Formerly MQ Telemetry Transport
- ▶ ISO/IEC 20922
- ▶ Protocol dates back to 1999
- ▶ Lightweight design, low bandwidth

MQTT Topics

- ▶ Topics are generated dynamically
- ▶ Topics are hierarchical
- ▶ Support wildcarding

Topic Examples

sensors/*HOSTNAME*/temperature/*HDD_NAME*

- ▶ sensors/sinanju/temperature/nvme0n1p1
- ▶ sensors/+/temperature/+
- ▶ sensors/sinanju/temperature/+
- ▶ sensors/sinanju/#

QoS

- ▶ 0: The broker/client will deliver the message once, with no confirmation.
- ▶ 1: The broker/client will deliver the message at least once, with confirmation required.
- ▶ 2: The broker/client will deliver the message exactly once by using a four step handshake.

MQTT Brokers

- ▶ Centralized broker
- ▶ Many different options: <https://github.com/mqtt/mqtt.github.io/wiki/servers>

MQTT Clients

- ▶ Bindings available for most languages
- ▶ <https://github.com/mqtt/mqtt.github.io/wiki/libraries>
- ▶ Eclipse Paho project provides similar interfaces across multiple languages

The Firehose

Mosquitto

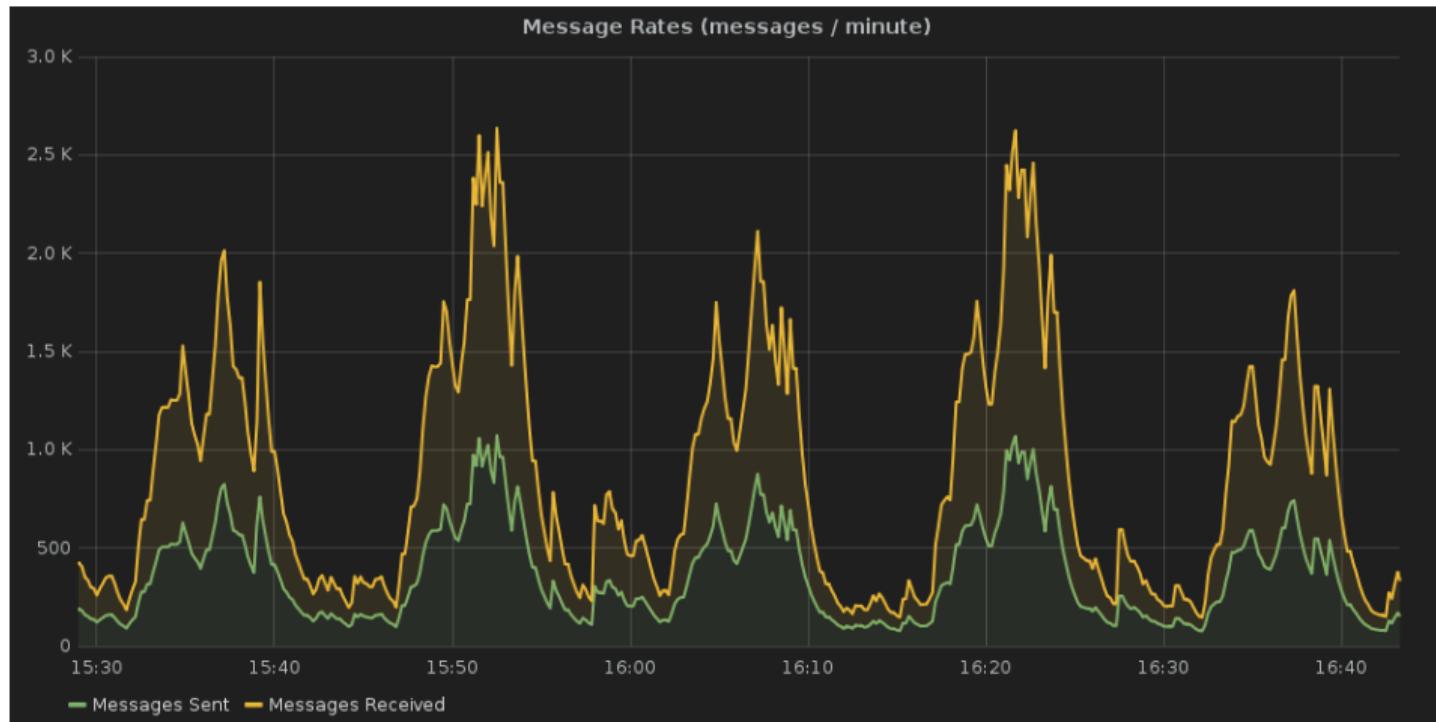
- ▶ MQTT broker implemented in C

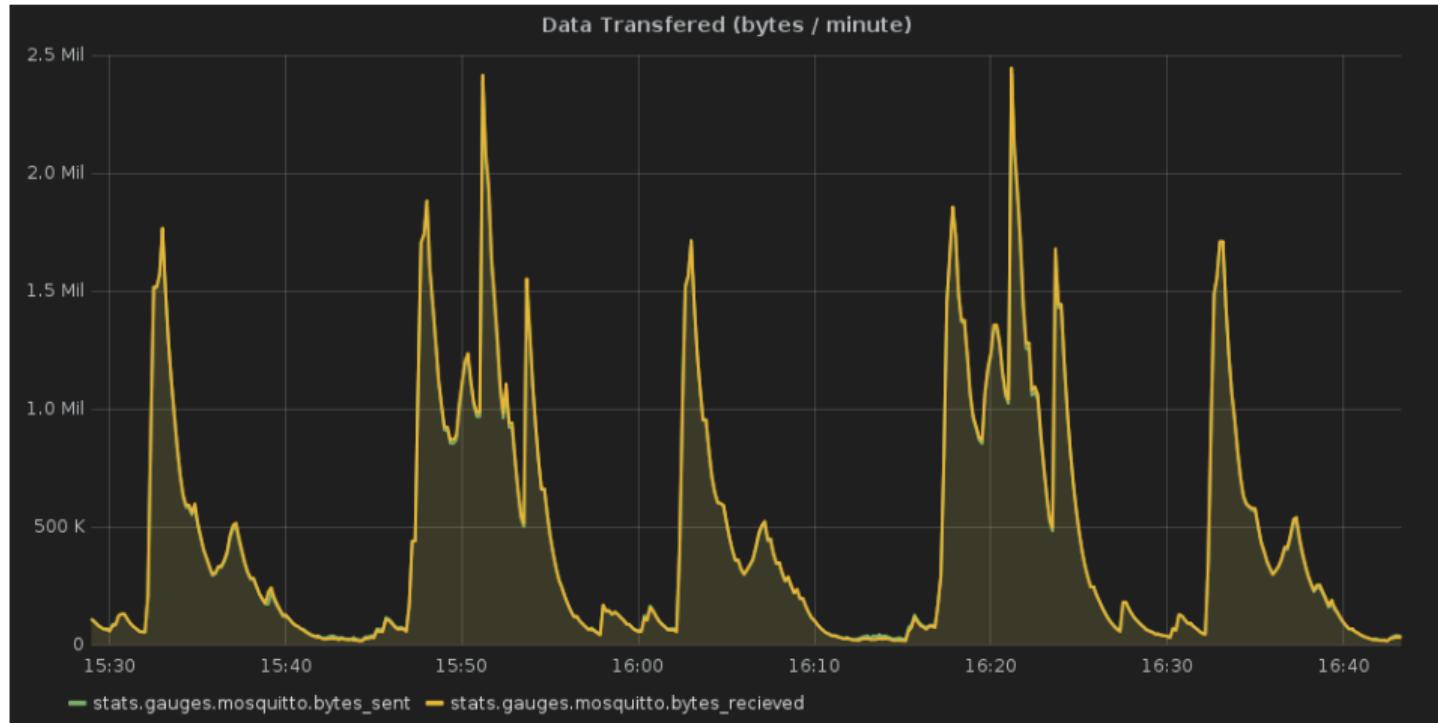


Services Using the Firehose

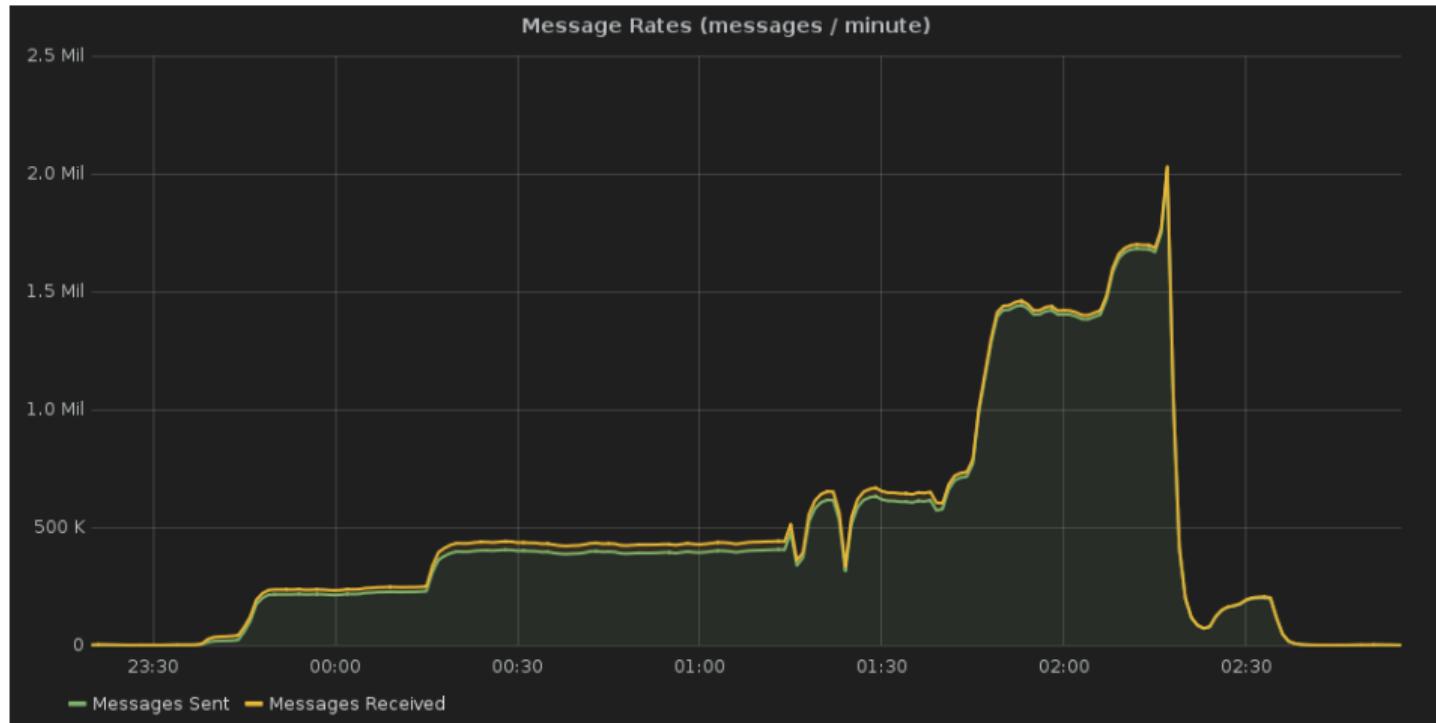
Service	Base Topic	Source of Messages
Ansible	ansible	Ansible MQTT Callback Plugin
Gerrit	gerrit	germqtt
Launchpad	launchpad	lpmqtt
Subunit Gearman Worker	gearman-subunit	subunit-gearman-worker

Typical Firehose Load

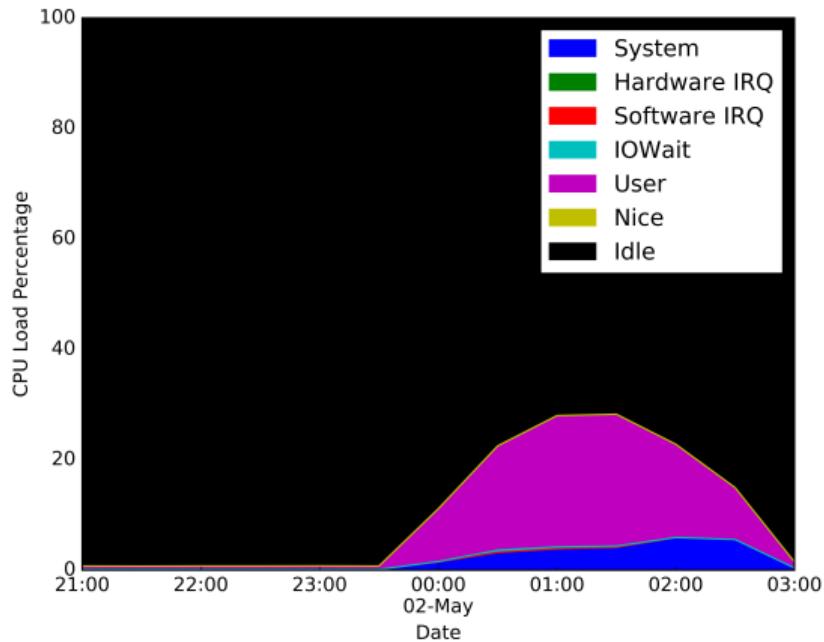




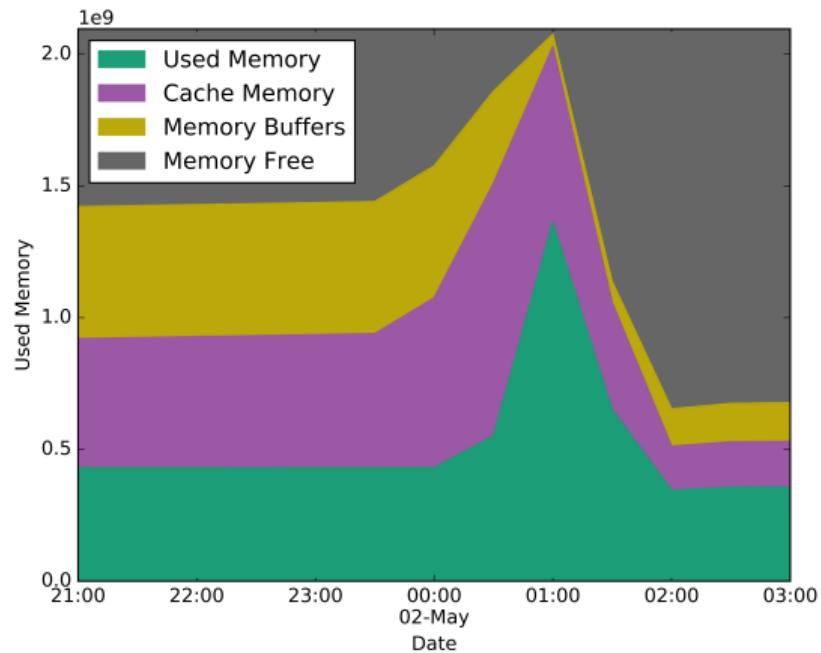
The limits of scaling



CPU Usage:

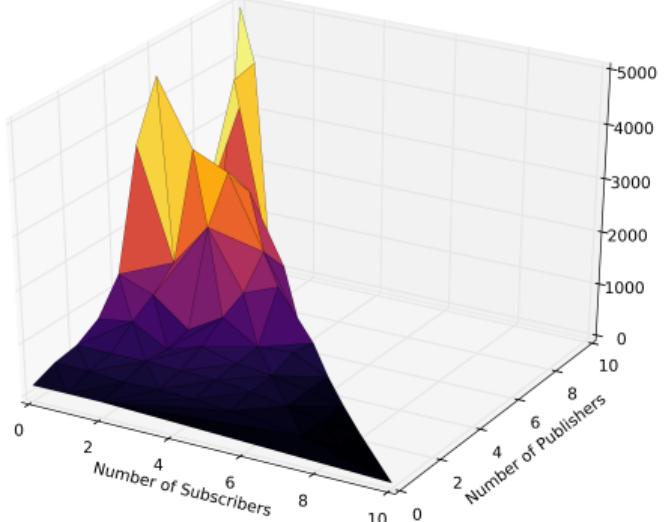


Memory Usage:

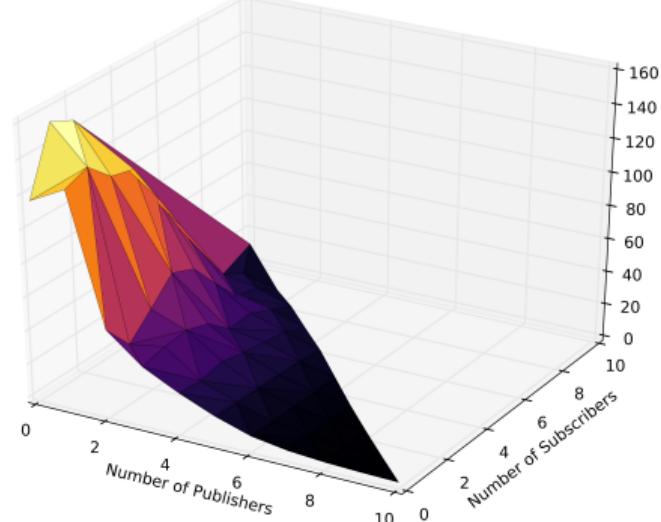


Benchmarking Performance

Subscriber Average Throughput (msg/sec)



Publisher Average Throughput (msg/sec)



<https://github.com/mtreinisch/pymqtbench>

Using Firehose

Example mosquitto client subscriber command line (provided in a mosquitto-clients package on many distros):

```
mosquitto_sub -h firehose.openstack.org --topic '#'
```

Listen for all Nova Comments in python

- ▶ **CLI:**

```
mosquitto_sub -h firehose.openstack.org --topic  
gerrit/openstack/nova/comment-added
```

- ▶ **Python:**

```
import paho.mqtt.client as mqtt  
  
def on_connect(client, userdata, flags, rc):  
    print("Connected with result code " + str(rc))  
    client.subscribe('gerrit/openstack/nova/comment-added')  
  
def on_message(client, userdata, msg):  
    print(msg.topic+" "+str(msg.payload))  
  
# Create a websockets client  
client = mqtt.Client()  
client.on_connect = on_connect  
client.on_message = on_message  
  
# Connect to the firehose  
client.connect('firehose.openstack.org', port=1883)  
# Listen forever  
client.loop_forever()
```

Listen for all ansible tasks on `health.openstack.org`

► **ruby:**

```
require 'rubygems'
require 'mqtt'

client = MQTT::Client.new
client.host = 'firehose.openstack.org'
client.port = 1883
client.connect()
client.subscribe('ansible/playbook/+/task/health.openstack.
org/#')

client.get do |topic,message|
  puts message
end
```

► go:

```
package main
import (
    "fmt"
    MQTT "github.com/eclipse/paho.mqtt.golang"
    "os"
    "strconv"
    "time"
)
func onMessageReceived(client MQTT.Client, msg MQTT.Message) {
    fmt.Printf("TOPIC: %s\n", msg.Topic())
    fmt.Printf("MSG: %s\n", msg.Payload())
}
func main() {
    hostname, _ := os.Hostname()
    opts := &MQTT.ClientOptions{
        ClientID: hostname + strconv.Itoa(time.Now().Second()),
    }
    opts.AddBroker("tcp://firehose.openstack.org:1883")
    opts.OnConnect = func(c MQTT.Client) {
        if token := c.Subscribe("ansible/playbook/+task/health.openstack.org/#", 0,
            onMessageReceived); token.Wait() && token.Error() != nil {
            fmt.Println(token.Error())
            os.Exit(1)
        }
    }
    client := MQTT.NewClient(opts)
    if token := client.Connect(); token.Wait() && token.Error() != nil {
        panic(token.Error())
    }
    for {
        time.Sleep(1 * time.Second)
    }
}
```

Potential Applications for Firehose

- ▶ 3rd Party CI Operators
- ▶ Desktop Notifications
- ▶ Inter Service communication

Future Plans

- ▶ Add germqtt subscription support to:
 - ▶ Zuul v3+
- ▶ Add more publishers:
 - ▶ Nodepool daemons
 - ▶ Zuul v3+
 - ▶ (your favorite service here)

Where to get more information

- ▶ openstack-infra ML openstack-infra@lists.openstack.org
- ▶ #openstack-infra on Freenode
- ▶ <http://docs.openstack.org/infra/system-config/firehose.html>
- ▶ https://docs.openstack.org/infra/system-config/firehose_schema.html
- ▶ <http://specs.openstack.org/openstack-infra/infra-specs/specs/firehose.html>
- ▶ <http://mqtt.org/>
- ▶ <https://mosquitto.org/>
- ▶ <https://www.eclipse.org/paho/>