Cloud Scale IoT Messaging

EclipseCon France 2018

Dejan Bosanac, Red Hat
Jens Reimann, Red Hat
IoT: communication patterns

Cloud

Telemetry

Inquiries

Commands

Notifications
Things optimized for throughput scale-out with messages

Telemetry

optimized for reliability scale-out with devices

Command & Control

Cloud arbitrary providers & deployment options

many existing protocols HTTP, MQTT, CoAP etc
What makes an IoT platform?

Business Services
- Monitoring
- Real time streaming
- Machine Learning
- ...

IoT Core Services
- Authentication & Authorization
- Device Registration
- Device Provisioning

Messaging infrastructure
Eclipse Hono provides a uniform API for interacting with millions of devices connected to the cloud via arbitrary protocols.
Eclipse Hono

- An Eclipse Foundation IoT project ...
  - Bosch and Red Hat as main contributors
- [https://www.eclipse.org/hono/](https://www.eclipse.org/hono/)
Eclipse Hono


Goals

- Tailored general messaging for IoT solutions
- Provide standard APIs for interacting with devices
- Support for arbitrary protocols (MQTT, AMQP 1.0, HTTP, ...)

Features

- Scalability
- Multi-tenancy
- Device-based security
- Multi-protocol support
Eclipse Hono


- Open source IoT connectivity platform running on ...
  - Kubernetes
  - OpenShift
  - Docker Swarm
- On-premise & in the cloud
- Provided by a set of Docker containers
Building Blocks

- Devices
  - HTTP
  - MQTT
  - etc.
- Protocol Adapters
- API Endpoints & Security
- AMQP 1.0
- AMQP messaging network
- Business Applications
  - AMQP 1.0

end to end flow control

- Hono
Eclipse Hono

AMQP 1.0

- International Standard (ISO/IEC ISO 19464)
- Binary Protocol
- Rich feature set:
  - conversation multiplexing
  - advanced flow control
  - Type system
  - QoS Guarantees
- Symmetrical message exchange
  - No Broker required
Routing vs Brokering

Producer → Broker → Consumer

- Send message
- Accepted

Acceptance process:

1. Producer sends a message to the broker.
2. The broker accepts the message.
3. The broker sends the message to the consumer.
4. The consumer accepts the message.
Routing vs Brokering

Direct

Producer

Send message

Consumer

Accepted
Routing vs Brokering

Producer

Router

Consumer

Send message

Accepted

Send message

Accepted
Basic idea
Addressing semantics

- Store and Forward
  - Queue
  - Topic

- Direct
  - Anycast
  - Multicast (Broadcast)
Micro-Service Design

- MQTT Adapter
- Hono Messaging
- Qpid Dispatch Router
- ActiveMQ Artemis Broker
- Device Registration
- Device Registry
- Auth Server
- Authentication
- Telemetry
- Event
- Credentials
- Provided by 3rd Party/Demo Implementation
Eclipse Hono
Telemetry & Event

- used by devices to send data/event downstream
- leverages on "direct messaging" ...
  - Telemetry
  - Devices can send data only if consumers are online
  - No broker involved
- ... “store and forward”
  - Event
  - Broker for storing event with a “ttl” eventually
- consumers receive data published by devices belonging to a particular tenant
Eclipse Hono

Credentials

- handle authentication for devices on protocol adapters
- used by **protocol adapters** to retrieve credentials used to authenticate devices connecting to the adapter (MQTT, HTTP, ...)
- different types of credentials
  - psk, hashed password, public key, ...
- operations
  - add, get, update, remove
- where an **identity management system** is already in place (i.e. Keycloak) ...
  - needs for having a “facade” from this API to such a system
Eclipse Hono

Device Registration

- used to make Hono **aware of devices** that will connect to the service
- solutions/consumers may use the API to get information about devices
- operations
  - register, deregister, get information, assertion
- for every message sent by a device ...
  - a **registration assertion** (JWT) is attached by the protocol adapter
  - it’s verified by messaging before sending the message downstream
  - a disabled device will have such check fails
Eclipse Hono

Tenants

- provides **multitenancy** in the system
- business applications can consume applications only for a certain tenant
- device registry and credentials are scoped per tenant
- tenants can be configured to use only certain protocol adapters
- protocol adapters check tenant of the device
Eclipse Hono
Command & Control

- used by applications to **send commands to devices**
- direct commands follow a request-response pattern and expect an immediate confirmation of their result.
- connection oriented protocols like MQTT or AMQP are easy
- request-reply protocols like HTTP need a little help
Eclipse Hono

Device notifications

- used by devices to update business applications about specific states
- time until disconnect notification (ttd)
- can be sent using any downstream message or sent in an “empty” notification
- can be defined for a device in device registry
- can be set by default for tenant or protocol adapter (in the future)
Simple deployment

Devices → Protocol Adapters

HTTP, MQTT

AMQP 1.0

ЯAPI Endpoints & Security

AMQP 1.0

Qpid Dispatch Router

AMQP 1.0

ActiveMQ Artemis Broker

AMQP 1.0

Business Applications
Scalable deployment

- EnMasse ...
  - a messaging-as-a-service platform
  - elastic scaling
  - multiple communication patterns
  - more info: enmasse.io
  - ... and more and more ...
Basic idea
EnMasse

- Open source cloud messaging running on Kubernetes and OpenShift
- enmasse.io
Features

EnMasse

- Multiple communication patterns: request/response, publish/subscribe and competing consumers
- Support for “store and forward” and direct messaging mechanisms
- Scale and elasticity of message brokers
- AMQP 1.0 and MQTT support
- Simple setup, management and monitoring
- Multitenancy: manage multiple independent instances
- Deploy “on premise” or in the cloud
Standard space

MQTT clients → MQTT gateway → Qpid Dispatch Router network → ActiveMQ Artemis brokers

AMQP clients

Admin

kubernetes

openshift by Red Hat
Brokered space

AMQP & JMS clients

ActiveMQ Artemis brokers

Admin

kubernetes

Openshift
Monitoring - Telemetry

Messaging Instances: 1
Total Processed Telemetry Per Sec (Ø/Min): 1.0

Processed Telemetry Per Second (Ø/Min)

- Meter: hono.messaging.messages.processed.m1_rate.mean (host: hono-service-messaging-1-frwqj)

Processed Telemetry: 218
Discarded Telemetry: 0

Monitoring - Events

Total Processed Events Per Sec (Ø/Min): N/A

Processed Events Per Second (Ø/Min)

No data points

Processed Events: 0
Undeliverable Events: 0
Resources

- Eclipse Hono - https://www.eclipse.org/hono
- EnMasse - http://enmasse.io
- ActiveMQ Artemis - https://activemq.apache.org/artemis/
- Qpid Dispatch Router - http://qpid.apache.org/components/dispatch-router/
Thank you! Questions?