Scaling out Eclipse Hono

EclipseCon Europe 2017
Dejan Bosanac, Red Hat
Kai Hudalla, Bosch Software Innovations GmbH
Who we are

Dejan Bosanac

Senior Software Engineer @ Red Hat
- Messaging and IoT
Open source committer
- Eclipse Hono
- Eclipse Kapua
- Apache ActiveMQ

Kai Hudalla

Chief Software Architect @ Bosch SI
- IoT Hub Team
Lead/Committer @ Eclipse
- Hono
- Californium
- Leshan
Eclipse Hono provides a uniform API for interacting with millions of devices connected to the cloud via arbitrary protocols.
Things

- many existing protocols
  - HTTP
  - MQTT
  - CoAP
  - etc

Cloud

- arbitrary providers & deployment options

Telemetry

- optimized for throughput
- scale-out with #messages

Command & Control

- optimized for reliability
- scale-out with #devices
Monolithic Design
≤ 0.5-M6
Micro-Service Design
≥ 0.5-M7

MQTT Devices

MQTT Adapter

Hono Messaging

Qpid Dispatch Router

ActiveMQ Artemis Broker

Device Registration

Auth Server

Device Registry

Credentials

Telemetry

Event

Authentication

Provided by 3rd Party/Demo Implementation
Registration Assertion

1. publish telemetry
2. assert (tenant-id, device-id)
3. check if device is registered and enabled
4. Json Web Token
5. create message (incl. token)
6. forward message
7. validate token
8. forward message

MQTT Adapter

Hono Messaging

Device Registry

<<public key>>
Features Hono 0.5

- Uniform APIs for consuming telemetry data and events
- MQTT, HTTP protocol adapters
- Device-level Authentication
- Tenant based Security Model
- Horizontal Scalability
Simple deployment
Routing vs Brokering

Producer → Broker → Consumer

- Send message
- Accepted

Consumer → Broker → Producer

- Send message
- Accepted
Routing vs Brokering

Producer → Router
Send message

Router → Consumer
Send message

Producer → Accepted
Consumer → Accepted
Addressing semantics

- Store and Forward
  - Queue
  - Topic

- Direct
  - Anycast
  - Multicast (Broadcast)
Scalable deployment

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

Diagram:

- Devices
- Protocol Adapters
- HTTP, MQTT
- AMQP 1.0
- API Endpoints & Security
- Business Applications
- AMQP 1.0
Basic idea
EnMasse

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

- **Queue**
  - store-and-forward = true
  - multicast = false

- **Topic**
  - store-and-forward = true
  - multicast = true

- **Anycast**
  - store-and-forward = false
  - multicast = false

- **Broadcast**
  - store-and-forward = false
  - multicast = true
Flavor examples

- Persistence
  - In memory
  - Persisted
- Scaling
  - Single broker
  - Pooled
- HA
Future
In progress/TODo

- Authentication and authorization
- Service broker API
- HTTP(S)
- Broker address space
  - Message grouping
  - Distributed transactions
  - Message ordering
- Multiple flavors
  - Apache Kafka?
- ...

...
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?
Evaluate the Sessions
Sign in and vote at eclipsecon.org