ConnectedCooking powered by MQTT

Gerrit Brehmer
Köln, 5.6.2018
Project Overview

› Customer: Rational AG
› Target: iot Platform with
  Web & App Clients for Rational combi oven
› Remote Configuration, Monitoring,
  Remote Control
› Statistics, Reports,
  FW-Management
› ...

RATIONAL
main decision #1: Protocol

› Requirements:
  › Proxy support/Firewall friendly
  › embedded client libraries
  › fast, scalable & lightweight

› Candidates:
  › WebSocket / Stomp
  › AMQP / JMS
  › Cloud Push Provider (Parse)
  › MQTT
MQTT in a nutshell

› initially developed by IBM
› OASIS Standard
› lightweight (binary, tcp or websocket tunnel)
› fault tolerant (multiple QoS Level support, ...)
› PubSub
main decision #2: MQTT Broker

IOT PLATFORM?

NO PROBLEM, WE SELL THOSE!

https://goo.gl/JTUaS2
main decision #2: MQTT Broker

Disclaimer: evaluation 2,5 years ago

› HiveMQ
› Apache Apollo
› ApacheMQ, RabbitMQ
› VerneMQ
› EMQ (emqtt)
challenge #1: MQTT Java library

› fuse source
  › auto reconnect, fluent API
  › but: concurrency issues (especially after some runtime)
  › in the meantime: repo is dead

› eclipse paho
  › supports many languages
  › API could be more lightweight
  › but: no concurrency issues
  › main eclipse project (part of iot eclipse ecosystem)
challenge #2: Integration of devices

› pros:
  › HW & platform dev team: common API (MQTT, HTTP)

› cons:
  › device firmware delay: many dry runs for Sprint reviews of iot platform
  › no automated integration tests
  › some problems only occur on production with real live traffic
improvement #1: dummy devices

› manual integration tests
› getting debug infos easier/faster
› check specification of export/import formats
› fun with colleagues
challenge #3: integration of bigger customers

https://imgflip.com/i/2bm169
challenge #3: integration of bigger customers

› current/previous solution:
  › complex desktop sw / single host
  › can manage intranet devices only
› first wish: on-promise systems
  › very expensive & high operating costs
  › complex version/upgrade management
› migration from current system to the new iot cloud
challenge #4: BlackBox

- few new technologies and blackbox devices
  - mqtt as protocol, mqtt broker
  - handling thousands of concurrent connections
  - device firmware developed by another team which is not integrated in the same development workflow
IT WORKED FINE IN TEST

OPS PROBLEM NOW
improvement #2: DevOps

› CI-CD Pipeline
› Enhanced monitoring probes
  › Health checks, API statistics, Log statistics
› Monitoring Dashboards
› Alarming via Slack Messenger
› Log Analysis
  › currently a lot of manually work
  › Elastic Stack planned (Kibana, ...
improvement #3: WhiteBox

› monitor & analyse API usage
› sometimes APIs are not used as intended in all use cases
› get more understanding for client requirements and special use cases
   › If APIs does not cover each use case, the client developer will find a way
› discussed many improvements, but...
challenge #5: firmware development

https://imgflip.com/i/2bl74a
challenge #5: firmware development

› embedded development not agile and very conservative
› iot platform team & firmware dev team are separated: e.g. different roadmaps
› devices with different firmwares must be handled like supporting different mobile app versions
› workarounds on backend side
› Restriction: forcing fw updates not possible
challenge #6: MQTT broker updates

› Breaking changes (monitoring API, config files)
› Long unresolved issues (emq-auth-http)
› Problem due to permanent connections: broker restarts
  › Client reconnects to the first node back online
challenge #7: Optimized message handling

› (Pre-)Sharding device TOPICs
  › /<shard>/<device-uuid>/<device-topic>
  › message processing can be distributed
› Combine MQTT messages for same device in a short timeframe
  › only update database once, ignore duplicates
› Using (mqtt) sessions
challenge #8: Last will lie

LOOK. I DIDN'T LIE

I JUST DIDN'T TELL THE TRUTH.
challenge #8: Last will lie

› “Last will” often used for connection status updates
  › server send “offline” on client connection loss
› but what if server or node died
  › e.g. operating team have to clear device online state
  › bulk updates are dangerous
› self-healing if device reconnects successfully
further challenges...

› Remote display & slow connections: Great wall of china
  › AWS China
› Handling auth server problems
  › by default: bad username or password (Code 4)
  › better solution: server unavailable (Code 3)
whats next #1: MQTT 5

› Shared subscriptions
  › load balancing of message processing possible
  › scalable subscriber
› TTL for messages
› Alternate auth mechanisms (challenge/response)
› error handling improvements
› more flexibility (topic alias, custom header, user properties)
whats next #2: Shared subscriptions

› EMQ current state: only usable together with local subscriptions
  › subscription client must know all cluster nodes any time
› other brokers already support cluster wide subscriptions (HiveMQ, VerneMQ)
› but: concrete implementation very unclear
  › e.g. will online clients be preferred, will node local clients be preferred
whats next #2: MQTT 5 & EMQ

› no official EMQ roadmap available
› GitHub ticket comments from devs:
   › EMQ version 3.0 with MQTT 5 support is coming soon (probably beta phase)
   › monitoring: Prometheus support announced (already exists in paid version EMQ X/Plus)
Any further questions? ConnectedCooking
Vielen Dank

Gerrit Brehmer
Developer / Architect

inovex GmbH
Ludwig-Erhard-Allee 6
76131 Karlsruhe

gbrehmer@inovex.de
0173 3181 007