Architecturing and Securing IoT Platforms With Microservices

Lessons Learned From Building Mainflux
It’s robot time!
## Mainflux - Internet For Robots And Machines

**Industrial IoT Messaging and Device Management Server**

- [http://mainflux.com](http://mainflux.com)

### GitHub Repository

- 333 commits
- 1 branch
- 0 releases
- 6 contributors

<table>
<thead>
<tr>
<th>Branch: master</th>
<th>New pull request</th>
<th><img src="https://github.com/mainflux/mainflux/blob/master/.git?raw=true" alt="Clone or download button" /></th>
</tr>
</thead>
</table>

### Latest commit summary

- **22 days ago**: Fix contributing guide
- **14 days ago**: Remove unused service references
- **7 days ago**: Remove printin
- **7 days ago**: Remove comments, clean code
- **7 days ago**: Ensure codestyle adherence
- **7 days ago**: Ensure codestyle adherence
- **13 days ago**: Add coap support
- **7 days ago**: Ensure codestyle adherence
- **7 days ago**: Integrate manager service
- **22 days ago**: Fix contributing guide
- **13 days ago**: Add coap support
- **22 days ago**: Integrate http adapter service
- **22 days ago**: Integrate manager service
- **22 days ago**: Integrate message writer service
- **2 years ago**: Switch LICENSE to Apache v2.0
- **22 days ago**: Remove lora-related artifacts
- **22 days ago**: Remove unused service references

#OREillySACon

**Software Architecture**
Drasko Draskovic
MSc. Electronics

Over 10 years of industry experience in semiconductor, telecom, system software, embedded and electronics. MSc in Electrical Engineering, Electronics, Telecommunication and Control
OMAP processors for Texas Instruments, LTE SoC with Sequans Communications, 5G connected world with NOKIA
Connected everything (IoE) with Mainflux

Janko Isidorovic
MSc. Telecommunication

Janko is the co-founder of Mainflux IoT open source project. He is also chair of the Application Work Group of Linux Foundation EdgeX project. Janko has a 10+ years background in Project Management, IT and Software integrations.
He holds MSc. In Telecommunications from Belgrade University.
Outline

Internet of Things (IoT)
▪ IoT Business Value
▪ IoT and Enterprise

Challenges
▪ IoT and Enterprise Integration
▪ Common IoT Project challenges
  - Networking
  - Power Consumption
  - Sensor Computing Power
  - Scalability
  - Security

IoT and Cloud
▪ Centralized vs Decentralized
▪ Typical Cloud Architecture
▪ Protocol Choices

Mainflux IoT Platform
▪ Architecture, Security and Technology Choices
▪ How Does it Work (Examples)
▪ Deployment

Q&A
Takeways

▪ How does IoT fits in enterprise and business

▪ Common IoT system problems and challenges

▪ Understanding of IoT system requirements and implications of constrained device capabilities and limitations.

▪ Mainflux as example of IoT Platform - microservices and OSS building blocks for integration of an IoT system.
What is IoT
What is Internet of Things?

The Internet of Things is the network of dedicated physical objects (things) that contain embedded technology to sense or interact with their internal state or external environment.

The IoT comprises an ecosystem that includes things, communications, applications and data analysis.
Internet of Things - IoT > M2M

Connectivity and embedded technology

The cost of connectivity and embedded technology is becoming less of a barrier to adoption. Broadband communication, Wi-Fi, Near Field Communication (NFC), Bluetooth and mobile networks are becoming ubiquitous and able to support large volumes of IoT connectivity at little incremental cost.

- Low Cost Sensors
- Long Range - Low Power Radio Networks
- Bitcoin like Micro transactions
IoT and Enterprise
IoT and Enterprise

Integration
Integration with existing systems within enterprise is a challenge.

Complexity
Small companies do not have resources to develop and maintain Smart Things.

Compete with Big Companies
Small and midsize companies want to be able to develop competitive products.

Manufacturing
Customers expect Smart devices, physical device is not enough.
IoT and Enterprise

Challenges for organizations and existing enterprise systems

- **Data** - Large and diverse volume of data. Velocity and demands of web scale.

- **Security** - Professionals expect IoT attacks to rise.

- **Integration** - IoT protocols, data formats, networking are different from existing enterprise IT solutions.

- **Diverse IoT Landscape** - Each vertical solution needs to use products from different vendors.

- **Lacking Hardware Expertise** - Most enterprises do not have experience with connected or intelligent devices.
IoT and Enterprise

Enterprise Architecture and IoT

▪ IoT will be crucial for digital transformation strategy of enterprises.

▪ But, IoT will disrupt existing IT architectures, because of IoT data volume velocity and demands.

▪ Implications of the business possibilities must be assessed for responding enterprise architecture design and plans.

Changing role for CIOs and ITs

▪ **History: Outside-in approach**
Existing IT infrastructure and enabling technologies align with the business strategy.

▪ **Present: Inside-out approach**
Enterprise architects have initiative and crucial role in cross-functional teams responsible for development of business vision and strategy.
IoT Business Value Chain

Note, the above is not an exhaustive list of companies and any company may have play in more than one component of value chain.
Copyright: Telecomcircle.com
IoT Business Value Chain
“We completely restructured Intel to be a cloud and Internet of Things company,”

Said Raejeanne Skillern, vice president of the Data Center Group and general manager of the Cloud Service Provider Business at Intel Corp.

*March 2017.*
Common IoT Challenges
Common IoT Challenges - Diversity

**Smart Agriculture**
- Livestock monitoring and reducing usage of chemicals

**Smart Industry**
- Predictive maintenance and reduced operational cost

**Smart City**
- Reduce consumption and increase efficiency

**Smart Energy**
- Predict energy production from renewable sources
Common IoT Challenges - Networking

No one size fits all solution

**Network Frequency**
- Sub-GHz Solutions
- LoRaWAN
- Sigfox
- 2.4GHz Solutions
- NB-IoT, LTE-M

**Network Range**
- Long range
- Short range

**Network Latency**
- Cloud Systems ~ 200ms
- On Premise Systems ~ 50ms
- Fog Systems ~ sub 5ms
# Common IoT Challenges - Networking

<table>
<thead>
<tr>
<th>Key Factor</th>
<th>Strength</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>Sub-GHz</td>
<td>Higher regulatory output power, reduced absorption, less spectral pollution, narrowband operation</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>Sub-GHz</td>
<td>Better circuit efficiency, improved propagation as Sub-GHz</td>
</tr>
<tr>
<td>Antenna Size</td>
<td>2.4GHz</td>
<td>2.4GHz is using smaller antennas; small antennas can be achieved for Sub-GHz</td>
</tr>
<tr>
<td>Data Rate</td>
<td>2.4GHz</td>
<td>Much higher throughput</td>
</tr>
<tr>
<td>Software cost</td>
<td>Proprietary</td>
<td>Small stack sizes, targeted applications</td>
</tr>
<tr>
<td>Multi-vendor</td>
<td>Sub-GHz, 2.4GHz</td>
<td>Most standards are 2.4GHz, many Sub-GHz also available</td>
</tr>
<tr>
<td>World Wide Deployment</td>
<td>2.4GHz, 433MHz</td>
<td>2.4GHz has an advantage, 433 can not be used in Japan, 868MHz/915MHz can be used in most of the world</td>
</tr>
</tbody>
</table>
Common IoT Challenges - Power Consumption

- Battery Powered Devices
- Battery life same as device life
- Tx/Rx Consumes lot of power on the device
- Encryption adds to power consumption
- Messages transmitted need to be as shot as possible
Common IoT Challenges - Computing Power

- Devices are CPU Constrained
- Devices are Memory Constrained
- IETF - RFC 7228 “Terminology for Constrained Node Networks”

<table>
<thead>
<tr>
<th>Name</th>
<th>data size (e.g., RAM)</th>
<th>code size (e.g., Flash)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 0, C0</td>
<td>&lt;&lt; 10 KiB</td>
<td>&lt;&lt; 100 KiB</td>
</tr>
<tr>
<td>Class 1, C1</td>
<td>~ 10 KiB</td>
<td>~ 100 KiB</td>
</tr>
<tr>
<td>Class 2, C2</td>
<td>~ 50 KiB</td>
<td>~ 250 KiB</td>
</tr>
</tbody>
</table>
Common IoT Challenges - Security

- September 16th 2016 - Largest DDoS Attack Ever.
- Some 150,000 smart/connected devices where hijacked.
- DNS Infrastructure was attacked
- Brought down many sites including: Twitter, the Guardian, Netflix, Reddit, CNN and others in Europe and the US

- Security
  - Device Security
  - System Security
  - Application Security
  - Data Security

- PKI Infrastructure
  - Blockchain like systems might prove to be valuable
Common IoT Challenges - Scalability

Scalability

▪ Large number of devices
▪ Fault tolerance
▪ By design concurrent programming languages like Go and Erlang

Micro services

▪ Modularity
▪ Independent scaling of services
▪ Division of responsibility within the server system
Common IoT Challenges - Integration

- IoT is not magic it will not resolve problems by itself
- Integration with existing enterprise systems is required
- Integration in near realtime decision making within the enterprise
Mainflux Solution

Typical Smart Device Technology Stack

User App
- Application Management
- User Management
- Access Control
- Device Message Broker
- Data Storage
- Device Provisioning
- Network Security
- Multi Protocol Connectivity
- Smart Device

Mainflux – Smart Device Technology Stack

User App
- OAuth 2.0 Identity Provider
- User Management
- Policy Based Access Control
- Device Message Broker
- Data Storage
- Device Provisioning
- Network Security
- Multi Protocol Connectivity (HTTP, WS, MQTT, CoAP)
- Smart Device

Benefits with Mainflux IoT Platform

User App
- [Command]
  - git clone https://github.com/Mainflux/mainflux
  - && cd mainflux
  - docker-compose up

Smart Device
IoT and Cloud
… which means we need to talk about electronics, firmware, RF, network protocols, server architecture, devops, security...
IoT and Cloud

We need to understand how sensor HW affects the connectivity
… which implicates protocol choices
… which implicates cloud architecture

IoT system architecture should be analyzed from the sensor towards the cloud and application - bottom up
Centralized VS Decentralized
IoT and Cloud - Decentralized vs Centralized

Decentralized:
- AllJoyn (Qualcomm)
- IoTivity (Intel)
- Telehash
- Blockchain, Blockchain, Blockchain

Centralized
- Everybody else (practically)
- Mainflux included
IoT and Cloud - Typical IoT Architecture
IoT Device Nodes
We are talking about something like this:
Or even this:
IoT Device Nodes - Characteristics

- MCU + RF front-end
- Bare-metal or small RTOS (for HAL and drivers)
- Constrained CPU clocking (low on kHz)
- Constrained flash and RAM
- Battery powered devices

<table>
<thead>
<tr>
<th>Name</th>
<th>data size (e.g., RAM)</th>
<th>code size (e.g., Flash)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 0, C0</td>
<td>&lt;&lt; 10 KiB</td>
<td>&lt;&lt; 100 KiB</td>
</tr>
<tr>
<td>Class 1, C1</td>
<td>~ 10 KiB</td>
<td>~ 100 KiB</td>
</tr>
<tr>
<td>Class 2, C2</td>
<td>~ 50 KiB</td>
<td>~ 250 KiB</td>
</tr>
</tbody>
</table>

IETF RFC 7228: Terminology for Constrained-Node Networks
Heavy networking stack just won't fit!
IoT Device Nodes - IoT Protocols

- UDP is more lightweight than TCP
- TLS becomes a problem (DTLS for UDP, but implementation missing in many languages)
- Elliptic Curve Cryptography - Diffie-Hellman
- HW encryption engine helps
And every radio TX/RX is power hungry!
IoT Device Nodes - Mode of Operation

- Sleep most of the time - wake up just to send telemetry, then go to sleep again
- Keep the messages short to save power (less protocol overhead, less metadata in the header, etc...)
- Accumulate (and filter) on the edge
IoT Protocols
Should I use WS to connect devices?
HTTP then?
CoAP?

YES. YES.
Mainflux - Example IoT Platform
“In the past, software products were largely developed in-house. Now we are repeatedly downloading code from the Internet to evaluate, prototype, and integrate. We only code the parts that are truly unique to our application.” - Linux Foundation
Mainflux is Written in Go

Go is an open source programming language that makes it easy to build simple, reliable, and efficient software.

- Compiled, statically typed
- Fast compiler
- Small footprint
- Performant, robust and stable
- Easy to read and maintain
- Testing framework
- Portability (compiled to a single static bin)
Mainflux - Example IoT Platform
Mainflux architecture is based on microservices
Mainflux IoT Platform - Microservice Architecture
IoT Message Subsystem
Messaging Subsystem

Messaging subsystem is composed of following microservices:

- HTTP Server
- MQTT (and WS) Broker
- CoAP Server
- NATS Broker
- Cassandra Adapter and Storage

It’s role is to distribute messages between various clients that can connect via various protocols - i.e. it makes a messaging bridge between them.
IoT Protocols - Protocol Choices

Devices
- MQTT
- CoAP

Apps
- HTTP
- WS
HTTP-to-WS-to-MQTT-to-CoAP?
Yes. Mainflux is bridging protocols.
Messaging Subsystem - NATS

- Brokers the IoT messages as an events in Mainflux NATS Message format

```
// RawMessage represents a message emitted by the mainflux adapters layer.
type RawMessage struct {
    Channel    string `json:"channel"`
    Publisher   string `json:"publisher"`
    Protocol    string `json:"protocol"`
    ContentType string `json:"content_type"`
    Payload     []byte `json:"payload"`
}
```
Messaging Subsystem - HTTP

- HTTP Message Sender is an HTTP Server that exposes RESTful API for sending (and sending only, not receiving) IoT messages received from HTTP clients (devices and applications)

- Message is sent in the form of SenML

- Endpoint: /channels/<channel_id>/messages
Messaging Subsystem - MQTT

- MQTT broker accepts and publishes SenML and binary messages to MQTT clients and at the same time on NATS broker for database persistence and analytics

- SenML JSON messages are published on channels/<channel_id>/messages/senml-json

- Binary messages are published on channels/<channel_id>/messages/octet-stream
Messaging Subsystem - CoAP

- CoAP server accepts CoAP (UDP) connections.
- RESTful-like API for sending IoT messages received from CoAP clients (devices and applications)
- RESTful-like API for CoAP-observing (similar to MQTT subscribing) of message channels
- SenML message format
Messaging Subsystem

Messaging between Microservices

- HTTP RESTful API (sync)
- NATS Events (async)
- ZMQ
- gRPC
SenML - Model for IoT Messages

- Provides simple model for retrieving data from sensors and controlling actuators
- Provides minimal semantics for the data inline and allows for more metadata with in-line extensions and links
SenML - Message Format

[
  {
    "bn": "urn:dev:ow:10e2073a0108006;",
    "bt": 1.276020076001e+09, "bu": "A",
    "bver": 5, "n": "voltage", "u": "V", "v": 120.1},
  {
    "n": "current", "t": -5, "v": 1.2},
  {
    "n": "current", "t": -4, "v": 1.3},
  {
    "n": "current", "t": -3, "v": 1.4},
  {
    "n": "current", "t": -2, "v": 1.5},
  {
    "n": "current", "t": -1, "v": 1.6},
  {
    "n": "current", "v": 1.7}
]
Binary Message Format

- Message is sent in the form of binary blob
- Mainflux doesn’t parse the binary message
- Binary message is stored in the database
- 3rd Party App would have to parse the message and use it
Provision Device Model

curl -s -S -i -X POST -H "Content-Type: application/senml+json" -H "Authorization: <user_token>" localhost:8180/clients -d '{"type":"device", "name":"weio"}'

HTTP/1.1 201 Created
Content-Type: application/json; charset=utf-8
Location: /clients/8293b8fa-9039-11e7-b6e2-080027b77be6
Date: Sat, 02 Sep 2017 23:50:33 GMT
Content-Length: 0
Provision Channel

curl -s -S -i --noproxy localhost -X POST -H "Content-Type: application/senml+json" -H "Authorization: <user_token>"
localhost:8180/channels -d '{"name":"mychan"}"

HTTP/1.1 201 Created
Content-Type: application/json; charset=utf-8
Location: /channels/7209d9b8-90af-11e7-9cf0-080027b77be6
Date: Sun, 03 Sep 2017 13:54:46 GMT
Content-Length: 0
Reporting Values (aka. Send Message)

curl -s -S -i -X POST -H "Content-Type: application/senml+json" -H "Authorization: <client_token>" localhost:7070/channels/7209db8-90af-11e7-9cf0-080027b77be6/messages -d '[{"bn":"some-base-name:","bt":1.276020076001e+09,"bu":"A","bver":5,"n":"voltage","u":"V","v":120.1},{"n":"current","t":-5,"v":1.2},{"n":"current","t":-4,"v":1.3}]'

cqlsh> SELECT * FROM message_writer.messages_by_channel;

<table>
<thead>
<tr>
<th>channel</th>
<th>id</th>
<th>bn</th>
<th>bs</th>
<th>bt</th>
<th>bu</th>
<th>bv</th>
<th>bvver</th>
<th>protocol</th>
<th>publisher</th>
<th>s</th>
<th>t</th>
<th>u</th>
<th>ut</th>
<th>v</th>
<th>vb</th>
<th>vd</th>
<th>vs</th>
</tr>
</thead>
<tbody>
<tr>
<td>7209db8-90af-11e7-9cf0-080027b77be6</td>
<td>fa22b310-90c3-11e7-2a58-7584557e38c9</td>
<td>some-base-name:</td>
<td>0</td>
<td>1.276e+09</td>
<td>A</td>
<td>0</td>
<td>5</td>
<td>voltage</td>
<td>HTTP eyJhbGciOiJIUzI1NiIsInR5cCI6...</td>
<td>0</td>
<td>0</td>
<td>V</td>
<td>0</td>
<td>120.1</td>
<td>False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7209db8-90af-11e7-9cf0-080027b77be6</td>
<td>fa252410-90c3-11e7-2a58-7584557e38c9</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>current</td>
<td>HTTP eyJhbGciOiJIUzI1NiIsInR5cCI6...</td>
<td>0</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>1.2</td>
<td>False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7209db8-90af-11e7-9cf0-080027b77be6</td>
<td>fa260e70-90c3-11e7-2a58-7584557e38c9</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>current</td>
<td>HTTP eyJhbGciOiJIUzI1NiIsInR5cCI6...</td>
<td>0</td>
<td>-4</td>
<td>0</td>
<td>0</td>
<td>1.3</td>
<td>False</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(3 rows)
MQTT PUB/SUB

mosquitto_pub -u <client_id> -P <client_token> -t mainflux/channels/7209d9b8-90af-11e7-9cf0-080027b77be6/messages/seml-json -m '[{"bn":"8293b8fa-9039-11e7-b6e2-080027b77be6","bt":1.276020076001e+09, "bu":"A","bver":5, "n":"voltage","u":"V","v":120.1}, {"n":"current","t":-5,"v":1.2}, {"n":"current","t":-4,"v":1.3}]

mosquitto_sub -u <client_id> -P <client_token> -t mainflux/channels/7209d9b8-90af-11e7-9cf0-080027b77be6/messages/seml-json

[{"bn":"8293b8fa-9039-11e7-b6e2-080027b77be6","bt":1.276020076001e+09, "bu":"A","bver":5, "n":"voltage","u":"V","v":120.1}, {"n":"current","t":-5,"v":1.2}, {"n":"current","t":-4,"v":1.3}]
Securing IoT System
Mainflux - Reverse Proxy

- Load Balance - scaling, HA, Fault Tolerance
- Reverse Proxy
- SSL termination - keep these certs at one place. MQTT is pure TCP, and CoAP is UDP
Intercept the requests and send them for Auth
Mainflux – Auth Subsystem

- Policy Based Auth
  - Grouping devices by connection to channels
  - Channel corresponds to MQTT topic
- JWT and Certs for Devices
  - JWT (short TTL)
  - Bearer Token
- Evaluate support for OAuth2.0 for Apps
Auth Server

**AuthX - Authenticate devices and users**
- CA must be burned into device flash for server auth for TLS
- Client-side certificates for higher security - use small-size encryption (constrained devices), for example TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
- JWT (current Mainflux implementation)
- OAuth2.0 Client Secret - how do we issue temporary token to devices?

**AuthZ - Authorize devices and users**
- Scoped API keys - JWT
- Drawback - no revoke, and device JWT has infinite TTL
- Revoke destroys stateless approach - so use just for session tokens with low TTL and no revoke
- Mainflux uses Access Control Policies - inspired by AWS IAM Policies
Securing Microservices

▪ Continuous Monitoring Is Critical. Discover and monitor inter-service communications

▪ Diversify Security Tactics. Secure data in motion and data at rest

▪ How do we know that noone has changed the micro service?
  - Securing the containers before they land in production
  - Sign Docker Images

▪ Each microservice has to be able to stand on its own security-wise
  - Secure the microservice API with authentication or access tokens from centralized server
  - Certificate-based security. Use TLS for communication between microservices
  - Use anomaly detection system to monitor communication between microservices
Securing the Fog

- HW TEE (Trusted Execution Environment)
  - Required in platform to protect and isolate security sensitive values
- Key Store/Key Management
  - Required in platform to protect stored keys
  - Containerize Key Vault
  - Later someone would need to connect this to other OS services and hardware RoT
- RNG (Random Number Generator)
  - TRNG (True Random Number Generator)
  - DRNG (Deterministic RNG)

- Secure Boot
  - Signature validation at each boot level
  - Integrity checks at each boot level
  - Connection into chain of trust in EdgeX
  - System will only boot if integrity checks pass
- Digital Signature Algorithm
  - ECDSA
- Access Management
  - OAuth2.0 Roles
  - Resource Owner
  - Client
  - Resource Server
  - Authorization Server
Deployment - IoT Server/Cloud Deployment

▪ Go produces static binary
▪ Can be deployed on RPi
▪ Docker images prepared on DockerHub (https://hub.docker.com/u/mainflux/)
▪ Kubernetes to Orchestrate the containers
`docker-compose up` and you got your Mainflux system running!

I'm the king of the world!
Mainflux Benefits

Typical Smart Device Technology Stack

- User App
- Application Management
- User Management
- Access Control
- Device Message Broker
- Data Storage
- Device Provisioning
- Network Security
- Multi Protocol Connectivity
- Smart Device

Mainflux – Smart Device Technology Stack

- User App
- Mainflux – Open Source IoT Platform
  - OAuth 2.0 Identity Provider
  - User Management
  - Policy Based Access Control
  - Device Message Broker
  - Data Storage
  - Device Provisioning
  - Network Security
  - Multi Protocol Connectivity (HTTP, WS, MQTT, CoAP)
- Smart Device

Benefits with Mainflux IoT Platform

- User App
- Mainflux
- Multi Protocol Connectivity
- Smart Device

>> git clone https://github.com/Mainflux/mainflux.git
>>& cd mainflux
>> docker-compose up
Scalable Architecture for the Internet of Things

O'Reilly Report on Coming Soon

Data Driven Architecture series

Authors:
  Drasko Draskovic, MSc. EE
  Dejan Mijic, PhD. Candidate
  Ervin Varga PhD.

#OReillySACon
Thank you!

Drasko Draskovic
- drasko@mainflux.com
- @drasko
- @draskodraskovic

Janko Isidorovic
- janko@mainflux.com
- @janko-isidorovic
- @jankoisidorovic