mimik

Using mimik Hybrid Edge Cloud with AWS Greengrass

1 Abstract

In this whitepaper, we describe how AWS IoT technologies is complementary to mimik's distributed edge cloud platform. We describe how AWS solutions may be used together with the mimik hybrid edge cloud platform to address the industry challenge to empower the edge for internet of things (IoT) applications. We will show that AWS IoT technologies can be used as microservices to allow edge nodes to complement the mimik platform.

2 Introduction

The smart phone era has helped accelerate embedding of processors, memory, and sensors in all kinds of devices. At the same time, some machine-to-machine (m2m) applications mostly for telemetry have become popular and are poised to revolutionize various industry sectors. The edge nodes for typical applications such as m2m generate small amounts of data today. However, future applications for internet of things (IoT) encompass use cases where edge nodes generate large amounts of data. Surveillance systems today top data generation with 20-40 MBytes/sec but there are upcoming use cases where significant amount of data is produced at the edge. Self-driving cars for instance generate a GByte/sec. For these use cases, the central cloud architecture does not scale. It is not feasible for millions of cars to send GBytes/sec of information back to the central cloud. Hence, we need to analyze the data at the edge, communicate directly amongst local nodes, and send only filtered data or abstracted data back to the central cloud. In other words, distributed edge cloud computing is essential in the future.

To make edge computing practical and useful, the edge nodes should be able to:

1. discover other nodes with which they could share data, services, or resources
2. communicate at physical and microservice levels with other nodes
3. adapt to functions and roles based on their resources and capabilities
4. process and analyze data locally
5. be at least as secure and trustable as the central cloud

In this white paper, we discuss how this vision can be accomplished using mimik's hybrid edge cloud platform along with recently announced IoT solutions that address the 4th issue of processing and analyzing the data locally and then communicating and syncing the data with other nodes or the central cloud.

In Section 3 , we describe Amazon Greengrass microservices intended for edge processing of IoT devices, and in Section 4 we describe how AWS IoT solutions can be used together with the mimik platform to create an end-to-end solution for distributed edge cloud computing.

3 Amazon Greengrass

AWS Greengrass [1] provides an ecosystem for local computing, messaging via dedicated message broker and data caching for connected devices in a secure way (see Figure 1).

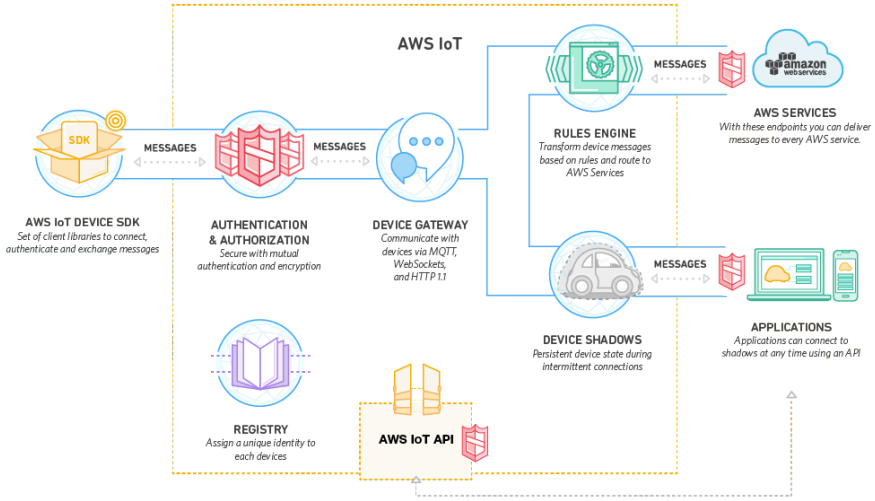

Figure 1: The Amazon Greengrass IoT ecosystem

The value proposition for Greengrass is centered on:

- versatile data-flow via publish / subscribe messaging
- filtering/processing using functional programming via amazon-lambda executed locally on edge nodes
- secure communications among edge nodes and with the cloud backend. Devices running AWS Greengrass core SDK act as a hub that can communicate with other devices (or things) [2] that have the AWS IoT Device SDK installed.

Devices connected to the same Greengrass group can process, communicate, and exchange data even when there is no Internet connection to the cloud backend (see Figure 2). To deal with intermittent links, "thing shadows" is supported, which maintains the "last known" status for devices that are off-line and takes care of synchronization when connection is restored.

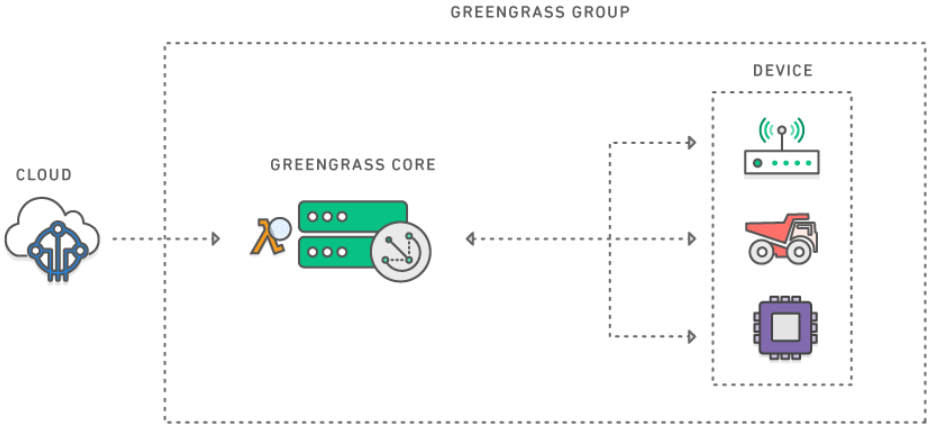

Figure 2: Local device group supported by Greengrass core

Registry, identity (via Amazon Cognito), policies and security are administered centrally and require manual setup [3]. Devices are obliged to keep their credentials safe in order to communicate securely with the message broker. Access to other AWS services in the cloud is via a "rules engine" that provides functional message processing and integration.
Using an SQL-based language, developers can select data from message payloads, process and send the data to other services, such as Amazon S3, Amazon DynamoDB, and AWS Lambda. The message broker can be used to republish messages to other subscribers. A full-stack development environment example is available through the AWS IoT button [4] (see Figure 3).

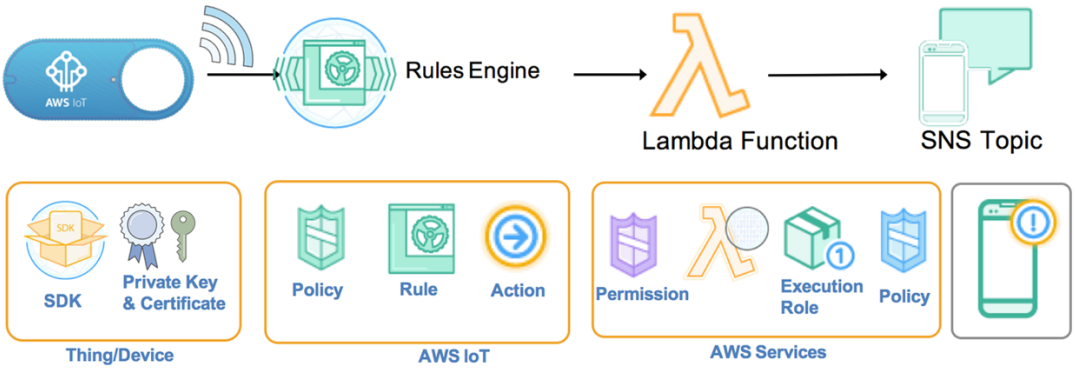

Figure 3: The AWS IoT button development environment example

Various commercial hardware platforms are supported including Raspberry-Pi 2 model B and any embedded Linux platform capable enough to run the AWS IoT device SDK. The open-source SDK is available for embedded C, Python, Java and Node.js, either on IoT platforms such as Arduino Yun, or mobile platforms such as iOS and Android [5]. Device addition/connection to the backend via the AWS IoT gateway requires preloading the software and manually configuring networking and credentials for access (see for example [6]).

4 mimik Hybrid Edge Cloud Platform + Greengrass

The mimik hybrid edge cloud platform enables the formation of edge clouds. It provides peer-to-peer communication (even without back-end Internet presence) amongst nodes within the edge cloud (cluster) and across edge clouds regardless of the network. The mimik edgeSDK enables any computing device (fixed or mobile) to acts as a server node and a computing platform to run microservices. Within an edge cluster, a node is dynamically elected as a *supernode*. A supernode selects nodes to take on specialized roles such as *network proxy* or *cache*. The selection depends on the physical characteristics and capabilities of the node, the edge cloud network topology and its communication capabilities with the cloud. The mimik edgeSDK is downloadable on Linux, iOS, Android, Windows, macOS, Raspbian, and QNX platforms and has been tested on a variety of mobile devices, smart TVs, WiFi routers, NAS appliances, connected car kits, etc.

The end-to-end technology has been tested for a variety of use cases such as rich media (TV, video, picture, music), content sharing amongst devices, STB and connected TVs within and outside homes, plugin for web-browsers, IOT for smart home and connected cars. edgeSDK

builds upon the standard TCP/IP stack, exposing functionality for discovery, connection and communication via REST API calls, both on the physical (network) and microservice levels.

Networking support is agnostic to the underlying infrastructure and independent of publish-subscribe protocol stacks employed by IoT gateways. As edgeSDK "lives" at the application layer in the stack (see Figure 4), it can be loaded on generic nodes running one of the supported platforms, as an "add-on".

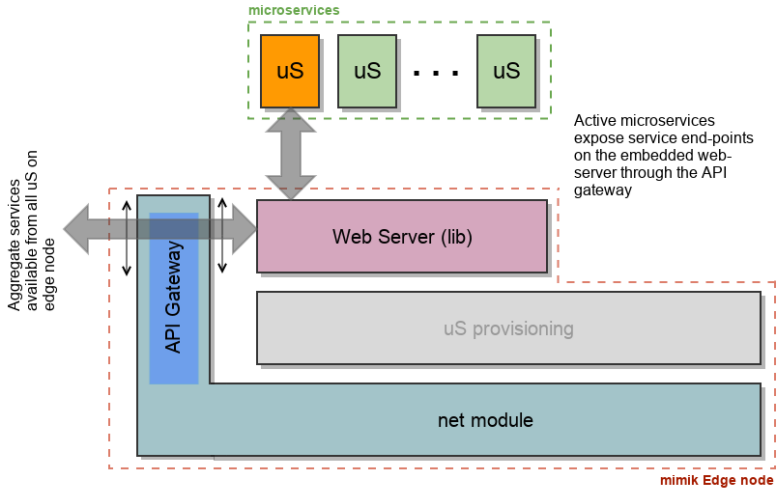

Figure 4: mimik edge node architecture

In this architecture, a node can function as an IoT gateway once the necessary microservices are deployed. In many cases the elected *supernode* can also be the *network proxy* for the link-local network. Nodes running in disjoint networks can establish two-way communication through the *network proxy* and on-demand signaling (SEP) and bearer (BEP) end-points (see Figure 5). In effect, each node can act as an ad-hoc IoT gateway which enables IoT "islands" to be joined into a mimik edge cloud (cluster) based on scopes (even supporting potentially other technologies such as Azure IoT). This enables the formation of application and use-case specific IoT network overlays on top of physically disjoint networks.

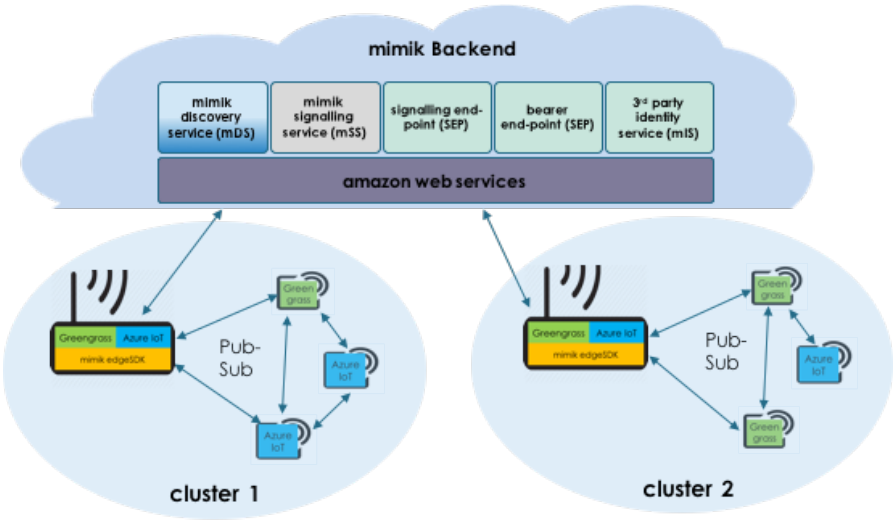

Figure 5: mimik combined with cloud-based IoT gateway for implementing IoT overlay networks

Advantages of combining the IoT gateway functionality with the mimik edge cloud include:

- extending local IoT groups across geographically remote sites, reducing fragmentation and enhancing analytics of preprocessed data before backend ingestion (e.g. joining two factories, aggregating sensors across both production floors in the same IoT group while using a single backend cloud ingestion gateway)
- enabling the use of general-purpose portable devices (e.g. smartphones or tablets) as IoT gateways for cases where IoT motes are deployed (e.g. in rural agricultural areas) but data collection is performed in sparse, ad-hoc intervals (e.g. when field workers visit the fields)
- supporting IoT frameworks from different vendors on the same physical gateway or hub and managing IoT motes from different manufacturers through the same application. The applications can be downloaded and installed on off-the-shelf devices.

The mimik edge node starts automatically right after a device is powered on, through a series of exchanges with the supernode (within the cluster). Interaction with the mimik back-end functions: the mimik Discovery (mDS), Signaling (mSS) and Profile (mPO) services, the on-demand Signaling End Point (SEP), or Bearer End Point (BEP), takes place only as needed and for communication across the edge cloud (cluster). The nodes within a cluster can discover, connect and communicate with each other without Internet. When connected to the internet, they can access the mimik backend functions if needed. The supernode acts as the mimik discovery service within the edge cloud (cluster) via a bootstrap mechanism.

The IoT gateway provided by AWS Greengrass can make services running on IoT nodes accessible to applications. Combining the mimik hybrid edge cloud platform with the local publish-subscribe messaging network implemented by an IoT gateway allows microservices running on IoT devices to become transparently accessible to applications anywhere, anytime. This is possible by hosting AWS Greengrass on mimik edgeSDK as an IoT messaging microservice (see Figure 6). mimik edgeSDK can be loaded on any device (e.g. PC, smart phone, router, tablet, etc.) effectively turning it into an IoT gateway or, alternatively, it can be loaded onto dedicated IoT gateway hardware.

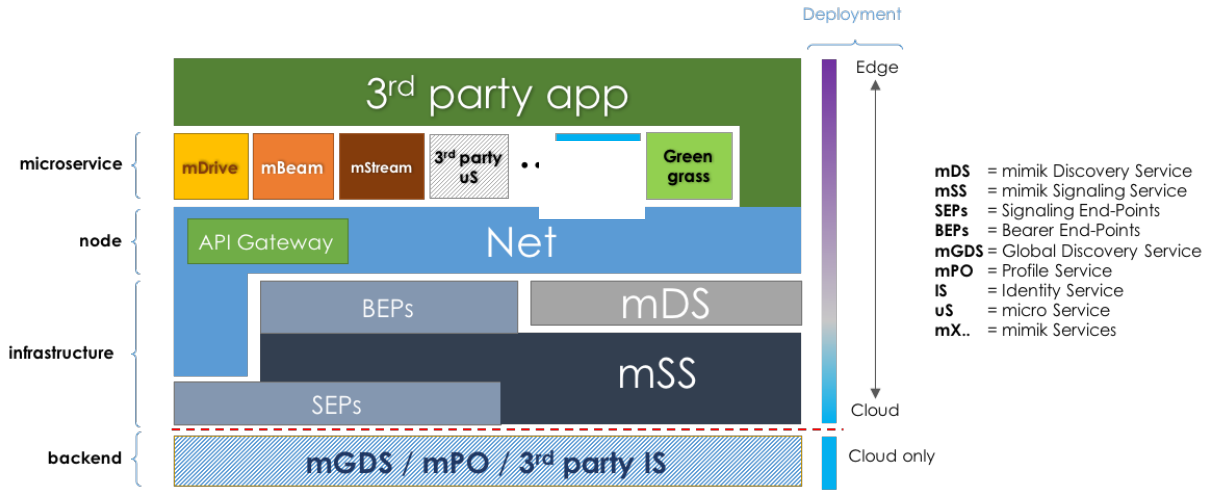

Figure 6: mimik hybrid edge cloud platform

By combining (see Figure 7) the different types of devices which may or may not support the usage of AWS FreeRTOS[7], AWS IoT Device SDK or edgeSDK, the AWS Greengrass groups can be combined with mimik ad-hoc clusters (by network, by proximity or by account) in order to address the dynamic nature of IoT solutions that go cross domain and cross technology including mobile solutions.

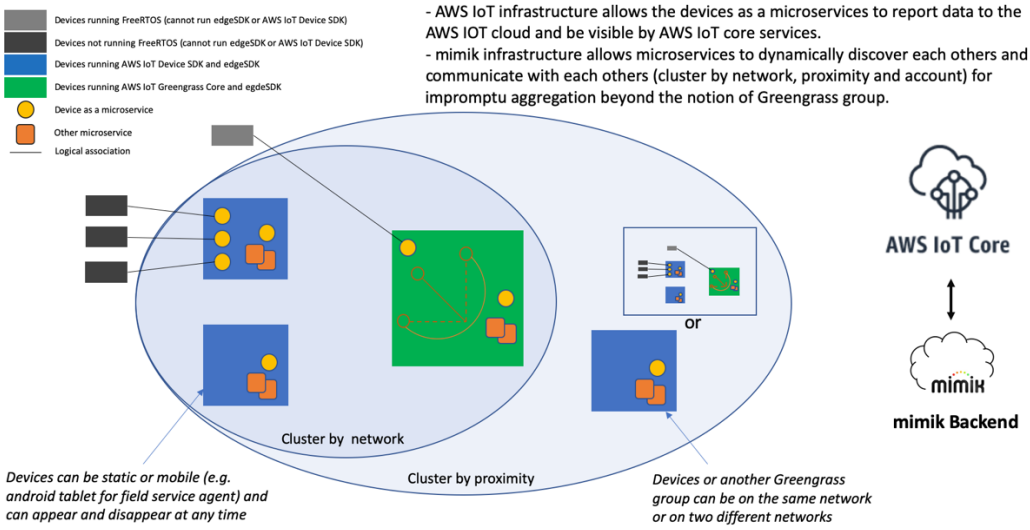

Figure 7: Combining AWS IoT infrastructure with mimik infrastructure

This combination of technologies leads to the creation of the notion of "device as a microservice"(see Figure 8) which provides the ability for devices of different nature and domain (mobile, smart cars, smart cities, home, …) to talk to each other and talk to other non-device related microservices at the edge or in the cloud and all this within a serverless microservice architecture environment.

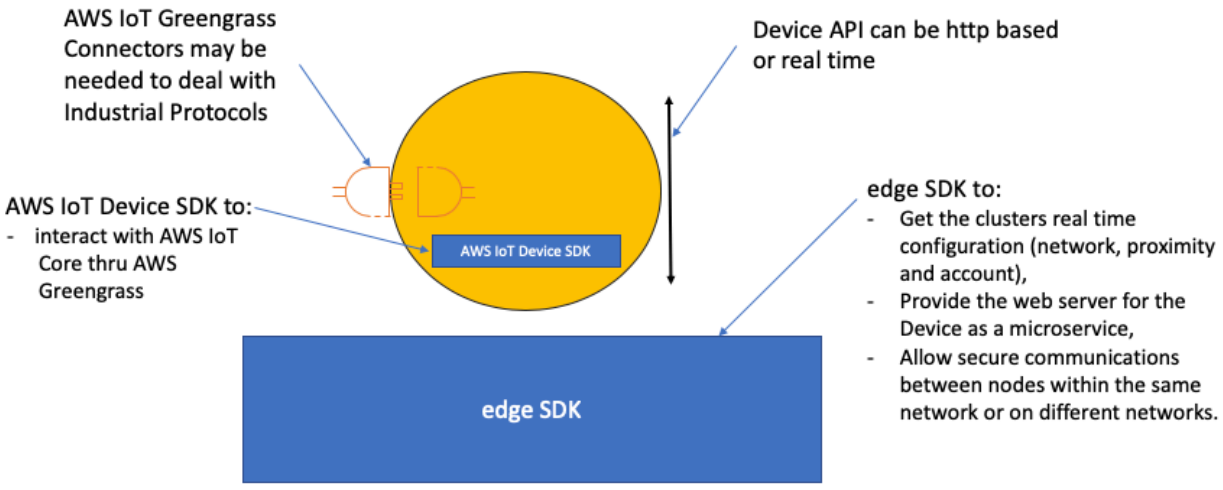

Figure 8: Device as a microservice

5 Conclusions

We have described the functions of Amazon Greengrass and have explained how these solutions can be run as microservices along with mimik edgeSDK. The mimik Hybrid Edge Cloud platform abstracts and exposes the computing capabilities of edge nodes and can turn any of these nodes into a gateway. In effect, any generic computing device such as a smart phone, tablet or android box can become a gateway. The mimik edgeSDK enables the edge nodes to dynamically form clusters, discover one another and connect within and across clusters. Furthermore, mimik provides a runtime environment for Amazon IoT enabled microservices to enable seamless messaging across cloud nodes. Building IoT gateways and applications with the mimik platform is simple and intuitive and works across hardware platforms and networks.

6 References

[1] AWS Greengrass https://aws.amazon.com/greengrass/
[2] AWS IoT components http://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html
[3] Greengrass device connection, registration, authorization example http://docs.aws.amazon.com/iot/latest/developerguide/iot-sdk-setup.html
[4] The AWS IoT button development environment http://docs.aws.amazon.com/iot/latest/developerguide/iot-gs.html
[5] Guide to the AWS IoT SDK https://aws.amazon.com/iot-platform/sdk/
[6] Using the AWS IoT Device SDK http://docs.aws.amazon.com/iot/latest/developerguide/sdk-tutorials.html
[7] AWS FreeRTOS https://aws.amazon.com/freertos/