

個人クラウドを利用したIoTプラットフォームの提案

樋口 裕次郎[†]

東京工科大学 大学院 バイオ・情報メディア研究科[†]

1. はじめに

近年，IoT 洗濯機やIoT 冷蔵庫，スマートスピーカーなどの個人に密着したIoT 機器が増えてきている．

それらのような生活に密着したIoT 機器は生活に密着しているが故に多くの個人情報を保持している．また，会話型のユーザーインターフェースなどは従来の画面を通じたユーザーインターフェースよりも多くの個人情報がサーバー側に蓄積される．そのため，従来のような大規模サーバーを使って多数の利用者に対して一括にサービスを提供する形式では，個人情報保護の観点から好ましくない．そこで，個人ごとにサーバーをクラウド上に用意し，そのサーバー上で各IoT 機器が必要としているアルゴリズムを実行することで個人情報が大規模サーバーに蓄積されるような状況を防ぐ方式を提案する．

また，IoT 機器はそれ単体で動作するだけでなく，スマートスピーカーを介して他のIoT 機器を操作するなど，それぞれが連携することが求められている．しかしながら，相互に制限なく操作・情報を取得できる状況は予期せぬ動作や個人情報保護の問題などがある．そこでこのプラットフォームでは，IoT 機器を管理するエッジルーターを利用し，各IoT 機器間の通信制限や操作の制限などを行う．

2. 提案するIoTプラットフォームの基本概念

提案するIoTプラットフォームでは個人ごとに，クラウドサーバーを用意し，そこでIoT 機器群必要とするアルゴリズムを展開し，IoT 機器間で連携することのできるIoTプラットフォームの構築を目標とする．

ここでは，IoT 機器・IoT 機器が必要とするアルゴリズムそれぞれをResourceと呼称する．

図1はこのIoTプラットフォームの概念図である．IoTプラットフォーム側ではそれらのResourceをプラグイン式に接続することができる型を作成し，その型にあった任意のResourceをユーザーが選択し，接続することで一つの機能を

構成することができる形にする．

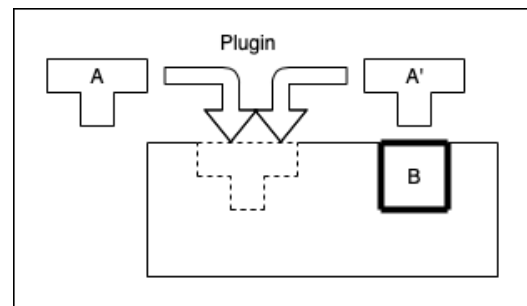


図 1: 概念図

3. 提案するIoTプラットフォームの構成

提案するIoTプラットフォームの構成として以下のようなものがある．

- ・ Resource
IoT 機器・ アルゴリズム
- ・ Resource Manager
Resource の統合管理
- ・ iSpace
各 Resource 間でのデータ共有

3.1 Resource

ResourceとはIoT 機器・IoT 機器が必要とするアルゴリズムそれぞれのことである．Resource上ではアプリケーションが動作する．

IoT 機器が必要とするアルゴリズムは Applicationが Docker コンテナとなっていることを想定している．Docker[1]コンテナとすることによって，ある特定の環境でなく，Dockerの実行環境が存在すればクラウド上でも容易に動作させることが可能になるためである．また，このResourceは個人クラウド上にResource Managerにより必要とされた際に展開される．

3.2 Resource Manager

Resource ManagerはEdge Router上で動作し，Resourceの管理，IoT 端末の管理，ストレージの管理を行う．各種リソースはそれぞれDriverと

呼ばれる ファイルでリソースの操作方法を記述しロードすることで様々な Resource を簡単に管理対象にすることができる。

3.3 iSpace

iSpace は、Python オブジェクトの共有とストリーム・シグナル等のイベントによる 関数の実行機能を提供する。これにより、Resource 間でのデータのやり取りを行う。

4. 提案する IoT プラットフォームの実装

4.1 個人クラウド環境

今回、個人クラウド環境として Kubernetes[2]を構築した。Kubernetes は Docker を利用することができるデプロイ環境であるため、Resource を展開するのに適している。

4.2 Edge Router

Edge Router とクラウド間のネットワークに関しては Kubernetes のノード間で利用している flannel を利用した。

Edge Router 上で flannel を動作させブリッジ処理を行うことで、Edge Router に IoT 機器が接続するだけでクラウド上の Kubernetes 環境と仮想的に同一ネットワークに存在するようにした。

これにより、IoT 端末が Kubernetes 上で動作している Docker に対してもアクセスできる。

4.3 Resource Manager

Resource Manager は、HTTP インターフェイスとコマンドラインのインターフェイスを通してリソースの管理機能を提供している。HTTP インターフェイスでは HTTP の URI を用いて階層上にリソースの管理機構を提供する。また、階層上に管理した Python オブジェクトのメソッドを POST メソッドを用いて実行することもできるため、柔軟な操作を行うことができる。

4.4 iSpace

iSpace は分散処理ライブラリである rpyc を用いて構築した。iSpace では、Python のデータ形式であるディクショナリ形式でデータの共有機能を提供する。

iSpace では、標準的な Python のデータ形式に加えて iSpace で提供するデータ形式が存在する。

以下のデータ形式はいずれも同期、非同期的に実行が可能のため状況に応じて使い分けることが可能である。これらは、通常の Python オブジェクトと同様に取得・実行することが出来るが内部的には他のノードで登録されたメソッド等呼び出すことが可能となっている。

- ・ ストリームブローカ
ソケットオブジェクトのように使用することが出来る
ストリーム機能を提供する
- ・ キュー
順序化されたバッファを提供し、dequeue 時の非同期実行をサポートする
- ・ シグナル
何らかのイベントの発生を伝えることができ、複数の通知先にイベントと値の送信をすることができる。

5. 終わりに

本研究では Kubernetes を用いて、個人クラウド環境を作成し、従来のような大規模サーバーを使って多数の利用者に対して一括にサービスを提供する形式ではなく、個人ごとにサーバーをクラウド上に用意し、そのサーバー上で各 IoT 機器が必要としているアルゴリズムを実行することで個人情報が大規模サーバーに蓄積されるような状況を防ぐ方式を実現した。

また、Edge Router・Resource Manager・iSpace を作成することで、それぞれの IoT 機器間で相互に操作・情報を取得できるようにプラットフォームを構築した。

参考文献

- [1] Docker Inc. : “Docker” <https://www.docker.com>
- [2] The Linux Foundation : “Kubernetes Document” <https://kubernetes.io>
- [3] Tomer Fillba : “Rpyc” <https://rpyc.readthedocs.io/en/latest/>