IoT システムの制御アーキテクチャの一検討

松　本　真　佑[†1]

本ポジションペーパーでは，IoT システムの制御アーキテクチャについて，我々の研究グループで得られたプラクティスに基づき議論を行う．

Considering Management Architecture of IoT System

Shinsuke Matsumoto[†1]

1. はじめに

身の回りのあらゆるモノに IT 技術を組み込む，IoT の時代が現実味を増してきた．Kickstarter[⋆1] をはじめとするクラウドファンディングサイトだけでなく，一般市場にも様々なアイデアを盛り込んだ製品が登場してきている．例えば，飲み物のカロリーや糖分を計測するタンブラー [⋆2] や，植物の日照量や水やりを計測する複合センサー [⋆3] のような，現実世界の情報をデジタル情報として計測するデバイスが存在する．また，現実世界からの計測ではなく現実世界へのアクションの実行デバイスとしても，物理的なカスタマイズフルな汎用ショートカットボタン [⋆4] も登場している．

本ポジションペーパーでは，上記のような個々の IoT デバイスに閉じた観点ではなく，複数の IoT デバイス同士をどのようなアーキテクチャで管理するか，という点について議論する．IoT では，モノとモノがヒトを（極力）介さずに制御する M2M という性質を備え持っており，機械同士の自律制御が基本となる．この自律制御は特定のサーバを介する中央集権的な方法とは異なり，スケールアウトしやすいという性質を持つ．一方で，個々のモノが自律的に実行されるため，IoT デバイス全体の俯瞰や管理が困難になりやすい．

本稿では，この IoT デバイスの管理アーキテクチャについて，我々の研究グループ [⋆5] での研究・開発のプラクティスに基づいて議論を行う．

2. コンテキストアウェアサービス

議論の題材として，コンテキストアウェアサービス（以降，CAS と略す）を採り上げる CAS とは現実世界に設置された環境センシングデバイスの読み取った値に基づいて，現実世界の状況や空気，すなわちコンテキストを判別し，そのコンテキストに応じたアクションを実行するサービスのことである．具体的には，「寒い」というコンテキストを温度センサから検知し，自動的にエアコンをつけるサービスや人感センサ付きライトも一種の CAS であるといえる．さらに複雑な例としては，洗濯機の使用状況から「洗濯物がたまっている」コンテキストを，さらに天気予報から「久しぶりに晴れた」コンテキストを検知し，洗濯を勧めるサービスなども考えられる．

この CAS を IoT 環境で実現するためには，屋内において以下の 3 つの機能が実現される必要がある．

a) 現実世界の環境センシングを行う．
b) センサの値を監視してコンテキストの成立を調べる．
c) コンテキストの成立に伴って指定されたアクション実行リクエストを発行する．
d) 家電機器が実際にアクションを実行する

この内，a は屋内に設置された環境センサそのものが，d は家電機器そのものが実行する以外の選択肢はない．この a と d の機能の実現方法としてはマイクロサービス が考えられる．例えば，温度センサから気温を取得す

†1 大阪大学大学院情報科学研究科
Graduate School of System Infomatics, Osaka Univ.
⋆1 https://www.kickstarter.com/
⋆2 https://www.myvessyl.com/vessyl
⋆3 http://www.modernity.jp/brand/parrot/parrot-flower-power/
⋆4 https://www.indiegogo.com/projects/flic-the-wireless-smart-button#/
⋆5 http://www27.cs.kobe-u.ac.jp/wiki/home/

© 2016 Information Processing Society of Japan

る場合, `http://iot.cs27.ac.jp/temp_sensor/get` のような URI の呼び出しによって温度センサそのものから値を取得する．家電機能の実行も同様に，`http://iot.cs27.ac.jp/tv/on` という URI でテレビの電源を付けることができる．

現在では様々なハードウェアが省スペース化・省電力化・高性能化しており，センサそのものにリクエストの受け口となるサーバを設置することも用意である．このように個々の最小の機能を REST サービスとして提供することで，IoT システム全体の疎結合化を計っている．実際に我々の研究室は，Raspberry Pi のシングルボードコンピュータや，Phidgets 等のセンサデバイスを組み合わせて CAS を構築している．このアーキテクチャは我々の研究グループで培った一種のプラクティスであるともいえる．

一方で，b と c はほぼ一対一の制御フローで実現されるためこれらを分離する理由はないが，この機能 b と c を**誰が実現するか**は二転三転しているのが現状である．一つの案は環境センサそのものが行う自律分散型のアーキテクチャであり（3 説参照），もう一つの案はあるサーバが一括管理する中央集権型のアーキテクチャである（4 説参照）．

また，機能 a から d は CAS の実行に不可欠な要素であるが，ここに CAS の管理という視点を加えた場合，さらに以下の機能が必要となる．

e) コンテキスト成立条件のルールを管理する．

f) アクション実行条件のルールを管理する．

e の例としては「温度が 10 度以下なら寒い」というルールが，また f の例としては「寒いならばエアコンを付ける」というルールが挙げられる．この機能 e と f についても自律分散/中央集権の選択肢が存在しており，検討が必要となる．

3. 自律分散型アーキテクチャ

あらゆるデバイスが自主的に動作する, IoT や M2M で一般に想定されやすいアーキテクチャである．我々は CAS の実現に当たって，まずこのアーキテクチャに乗っ取ったシステムを実現した[1)]．コンテキストを判別するのは環境センサそのものであり，環境センサがそのコンテキスト成立イベントに応じて機器を実行するという考えに従たシステムである．実行のための機能 b と c，および管理のための機能 e と f の全てを個々のセンサに持たせるという選択肢となる．個々のデバイスが自律的に動作するため, 故障に強くスケールしやすいというメリットを持つ．一方で CAS のルールがばらばらに存在するため，CAS 全体の管理は極めて困難であった．何を切っ掛けとして誰がそのサービスを実行しているのか不明になるという状況が多々あった．

IoT デバイス同士の徹底した自律分散を実現するためには，機能 e や f 以外の管理 API の充足化や，CAS を拒否された際の学習といったより高度かつ複雑な機能の実現が必須であったといえる．

4. 中央集権型アーキテクチャ

特定の機能を実現するサービスを一つ設け，それが中央集権的にその機能を実現するアーキテクチャである．今回の題材の場合，CAS の実行と各ルールの一元管理を実現するサービスが一つ存在することになる．我々のグループでは，個々のセンサを機能 a 環境センシングのみを行うマイクロサービスとし，CAS 関連の機能 b，c，e，f を全て単一のサービスが実現するシステムを構築した[2)]．自律分散と比べ，CAS ルールの管理 e と f は極めて容易であった．

一方で CAS の場合，コンテキスト成立の確認には高いリアルタイム性が必要となるため，このアーキテクチャでは負荷が集中しやすいという課題も挙げられた．我々の構築した環境では，センサ約 50 個，家電機器 30 個程度であったが，より IoT の普及が進んだことを考えるとスケーラビリティの確保は一つの課題であるといえる．

5. おわりに

本ポジションペーパーでは，IoT の実現に向けて，IoT システムに対する制御アーキテクチャについて，その経験の紹介と議論を行った．ワークショップの場では，これらの経験に基づいたハイブリッド型のアーキテクチャについて議論させていただければ幸いである．

参考文献

1) M. Nakamura, S. Matsuo, S. Matsumoto, H. Sakamoto, and H. Igaki, “Application Framework for Efficient Development of Sensor as a Service for Home Network System,” Int’l Conf. Services Computing, pp.576–583, 2011.
2) H. Takatsuka, S. Saiki, S. Matsumoto, and M. Nakamura, “RuCAS: Rule-Based Framework for Managing Context-Aware Services with Distributed Web Services,” Int’l J. Softw. Innovation, vol.3, no.3, pp.57–68, 2015.

© 2016 Information Processing Society of Japan