LOD の物理世界拡張と空間 OS

Extensions to LOD for Cyber physical System and Space-OS

中川雅三 [1,2]

[2]日本総合システム株式会社産業ソリューション部（Nippon Sogo Systems, Inc.）

Masami Nakagawa[1]

[1] 先端 IT 活用推進コンソーシアム　ビジネス AR 研究部会
[1] Advanced IT Consortium to Evaluate, Apply and Drive
Committee on Business AR

Abstract: 身の回りのあらゆるコンピュータ ―モバイルフォン、情報家電、HEMS、ロボット、など ― を連携させて数十年にわたって生活の場を支えるインフラストラクチャとすることで、高齢化などの社会問題の緩和に役立たせることが急務であると考える。そのためには、様々なコンピュータがもつサービス API とデータを相互運用可能にする必要がある。私たちは LOD を拡張し、サービス共有のための機能とセキュリティ機能を追加した CPLOD(Cyber Physical LOD)とよぶ実装を提案する。CPLOD を使えば、サービスが提供するデータ構造を RDF で表現し、API の実行を SPARQL クエリで記述することができるようになる。すべての規格の構築基盤を共通化することで、異なる規格に基づくサービスの相互運用を容易にする。CPLOD の応用として、空間 OS と呼ぶアーキテクチャを構想している。空間 OS は、人、家、車、オフィス、ビル、街、地域などの場所に対応する CPLOD とその場所にあるコンピュータを連携させたメタ OS である。そこで活動する機械と人々との協働を支えるインフラストラクチャとする。インターネット上の LOD とも連携し、プライバシーなどのセキュリティを守りながら、サイバー空間と物理空間にあるすべてのサービスとデータを相互に利用できるようにする。

1.はじめに

2025 年には日本の 65 歳以上の人口が 30%を超え、その 1/5 が認知症に罹患すると予想されている[1][2]。

こうした状況に対処するためには、高齢者の自立を支え、介護作業を軽減し、生産年齢の人々の生産性を高め、地域のサービス・モノ・エネルギーの流通を最適化し、人々の安心感や幸福感を高めることを支援するコンピュータシステムを作ることが必要であると考える。

身の回りのあらゆるコンピュータ　―モバイルフォン、情報家電、HEMS、ロボットなど―　を連携できるようにし、コンピュータを水道や電気同様のインフラストラクチャとして整備する、すなわち家庭・オフィス・街などすべての場所で、共通の規格に準拠したサービスを提供・利用できるようにすることが、このようなシステムを作り出すために必要である。これは市場原理を否定するものではなく、あらゆるシステムのサービス連携基盤を統一し、開発競争を機能や性能の差別化やそれらを通じた感動の創造という本質に集中させることを意図している。これによって人々を支えるコンピュータシステムの効率的な実現を促すのである。

本稿では、コンピュータのインフラストラクチャ化を妨げる課題を示し、LOD にサービス共有のための機能を付加した CPLOD と呼ぶ技術と、CPLOD を応用した空間 OS と呼ぶアーキテクチャを使って、これらの課題を解決することを提案する。また、開発中の実装について簡単に紹介する。

2. コンピュータの総連携

2.1 インフラストラクチャとしてのコンピュータ

一般のビジネスや政策の設計では、水道、電力、通信、交通などのインフラストラクチャの存在を前提とすることができる。同様に、生活圏に多数存在するようになったコンピュータ群をインフラストラクチャとして利用できるようになれば、これまでに実現できなかった事業や政策が実現できるようになるだろう。

たとえば、「人の一生の生活を快適にしながら、病気の兆候や老いてゆく状況を把握し、適切に支援するシステム」「地域の災害状況や人の所在をリアルタイムに把握し、災害による被害を最小限にくい止め、迅速な復旧を支援するシステム」などを実現することができるようになる。

プライバシーの問題や責任の所在の問題といった社会的な課題はあるが、技術的には、現在使われている技術でこうしたシステムを実現可能だと考える。

生活の現場にある様々な機器 ―TV、エアコン、電話、インターホン、ロボット、HEMS など― がサービスを API として公開し、それらを制御するコンピュータが現場やクラウドに存在するといった実装形態は、すでにスマートハウスと呼ばれる形で一部

実現している。

しかし現在のスマートハウスはインフラストラクチャではない。プロプライエタリな技術で作られていて、すべての事業者が使えるようなものではないからである。また、どの家にでもあるわけではなく、数十年にわたってシステムを使い続けられる見通しが不透明であるといった点でもインフラストラクチャと言えるものではない。

2.2 総連携によるインフラストラクチャ化

コンピュータをインフラストラクチャとしてゆくロードマップを考える。

人の一生にわたって動作し続けることが可能なハードウエアは存在しない。だから、インフラストラクチャとなるべきシステムは、構成要素が入れ替わりながらも全体として存在を続けるものとなるはずである。

またこのようなシステムは、綿密な設計のあと一気に作るというものではなく、部分的な実装が発展して、最終的にインフラストラクチャとしての体裁を整えるようになり、さらに発展を続けるというものになるだろう。

構成要素が入れ替わりながらもシステムは存続を続けるという実装の本質は、構成要素間のサービスインタフェースと、代々の構成要素間で受け継がれるデータとに、それぞれ互換性を持たせて、構成要素を入れ替えても連携可能とすることである。

本稿では、このような互換性の実現を「総連携」と呼ぶことにする。総連携は、つぎの 2 通りの連携を含む。

- 現在ある機器を連携させる、空間軸での連携。
- 古いデータの利用、未来へのメッセージ送信、古い機器と新しい機器の相互運用といった、時間軸での連携。

2.3 総連携の実現のための課題

総連携は、サービスインタフェースとデータの規格化で実現できる。すべての機器が同じ規格のインタフェースとデータを持てば、空間軸、時間軸ともに連携できるからである。

しかし、現在使われている規格にはつぎのような問題点があって総連携を実現することが難しい。

1. 様々なコンピュータ上のサービスが準拠する規格（API など）は、メーカーや業界ごとに分断されており、異なる規格に準拠した機器を相互に連携させることが困難である。
2. 新興分野や小規模な分野では、連携のためのサービス規格を作ること自体が困難である。
   規格の設計はトップダウンの重い作業となっており、規格作成のための経済基盤や実績を持っている企業や団体にしか実現できないと思われている。
3. 規格の寿命が、製品やビジネス形態の寿命によって抑えられており、人の一生や代々にわたってデータやサービスを相互利用することが困難である。

3. CPLOD による総連携の実現

3.1 LOD の応用

LOD を使えば、前項で述べたような課題の一部を解決することができる。

1 アクセス方法が HTTP、データ構造は RDF、クエリ記述は SPARQL で統一されている。あらゆる OS、あらゆるプログラミング言語からアクセスできるような抽象度をもったインタフェースであり、既存や未来の様々な実装からの利用が可能である。

2 IRI を使って独立な名前空間を作ることで、名前の衝突を避けながら、様々な事業者や個人が勝手に規格を作ることができる。
また、その規格を誰もが利用できるように公開することができる。

3 RDF ではあらゆる構造のデータを、データの意味を記述するメタデータとともに格納できる。時代の変化によるデータセットの追加や仕様の変更を加えることを事前に想定しており、規格の設計時に予期していないような様々な用途に適合させて規格を成長させてゆくという運用に適している。

3.2 CPLOD

LOD はオープンなデータ共有のための技術であり、サービスを共有する機能や、身の回りのデータを、相手を選んで公開するための機能がない。

そこで、本稿では、LOD につぎの 3 つの機能を追加することで、総連携を実現することを提案する。これらの拡張機能を追加した LOD を CPLOD(Cyber Physical Linked Open Data)と呼ぶことにする。LOD に物理空間と Cyber 空間とを連携する機能を持たせ

ることになるからである。

3.2.1 物理空間とのリンクと変化通知

RDF のノードの一部を、物理空間に実在するセンサーやサービスが持つ値へマッピングする。メモリマップド I/O に似ている。

- 物理空間とリンクする IRI を定義し、SPARQL による読出し操作のときにこの IRI を持つノード（物理ノードと呼ぶ）の値を、RDF ストアが現在値と置き換えて応答する。
  たとえば、SPARQL でセンサーの現在値を読み出すことができる。

- SPARQL による更新操作でこの物理ノードの値を書き換えると、その値を対応する物理空間の機器へ伝達する。
  たとえば、SPARQL で他のコンピュータが公開している API を操作できる。

- 物理ノードへ WebSocket で直接アクセスできるようにする。最初に SPARQL クエリで物理ノードを特定すれば、以後 SPARQL クエリの実行は不要となり、オーバーヘッドを減らせる。
  センサーは、WebScoket を使って物理ノードの値をリアルタイムに報告する。
  物理ノードの値の利用者は、WebSocket を使って値の変化をリアルタイムに取得する。

3.2.2 RDF データの履歴記録

CPLOD は、RDF データの履歴を記録する。

コンピュータ群が連携して多様なサービスを実現するときには、パラメータを渡すだけでなく、環境の状態やこれまでの履歴といった「コンテキスト」を共有する必要がある。CPLOD はコンテキストとして、環境の構成、現在値データ、データの履歴を提供する。

CPLOD を介した API では、単純なパラメータの受け渡しだけでなく、共有するコンテキストを使った連携動作が可能となるのである。

3.2.3 RDF ノードごとのアクセス権限の制御

多様な用途でサービスやデータを共用するために、RDF の個々のノードの公開範囲を細かく制御できるようにする。アクセス制御情報を RDF で記述し、RDF ストアの検索エンジンがこのデータを参照して認可された情報のみを処理する。

3.3 CPLOD による総連携

CPLOD を使うことで、つぎのようにして総連携の課題を解決できる。

1. メーカーや業界ごとに分断されている既存の API やデータの規格を CPLOD へ変換することで、メーカーや業界を超えた相互運用を可能にできる。また、新たに作る規格を CPLOD 上に構築すれば、最初から総連携が可能となる。

2. CPLOD 上での規格制定は、語彙とオントロジーの定義という本質的な作業だけになる。
   語彙とオントロジーの定義は簡単な作業ではないが、標準的な作業手順を確立してゆくことで、規格制定作業がこれまでより遥かに単純なものになると期待できる。
   また、RDF による表現の柔軟性や、名前空間の分離によるバージョン管理を使えば、技術が枯れるのをまたずに規格を作り、試行錯誤しながら発展させることができる。

3. CPLOD 上の規格に、サービスやデータの仕様を記述するメタデータを含めれば、つぎのことが可能となる。
   - 新旧の機器や、異なる規格に属する機器が、互いにサービスを利用する。
   - 古くから蓄積され、時代によって構造が変化してきたデータを利用する。

4. 空間 OS

4.1 CPLOD による空間 OS の実装方針

CPLOD でコンピュータのインフラストラクチャ化を実現するためには、その利用方法、すなわちシステムのアーキテクチャを規定する必要がある。

本稿では、CPLOD の応用形態の一つとして、空間 OS と呼ぶアーキテクチャを提案する。

CPLOD が厳密な規格（HTTP,RDF,SPARQL）に基づいて互換性を維持することに対し、空間 OS はシステムの漸進的な発展を可能とするためのゆるやかな指針を提案するものである。

以下に空間 OS に対する要件と実装方針を列挙する。

4.1.1 要件：コンピュータ群の総連携

サービス範囲となる「空間」を定義し、空間 OS を、「その空間に所属あるいは滞在するコンピュータの総連携を司る OS」として実装する。

空間 OS という名前はここに由来する。空間 OS は、一般の OS の上に実装された各種のコンピュータの集合体であるという点で、いわゆる「OS」ではない。　　しかし、様々なデバイス、OS、言語などの実装を隠蔽し、抽象的なインタフェースでサービスを提供するという点で、空間「OS」と命名した。

サービス範囲となる空間には、つぎのようなものがある。

- 固定された空間：部屋、家、オフィス、ビル、街、地域など
- 移動する空間：人（のまわり）、自転車、自動車、飛行機など
- 概念的な空間：家族、サークル、会社、部署、仮想世界（セカンドライフのような）など

4.1.2 要件：既存のコンピュータとの連携

現存する一般のコンピュータ群を総連携可能とすることが最初の実装となる。

初期の空間 OS では、CPLOD 仕様の RDF ストアを持ったサーバー（CPLOD サーバーと呼ぶ）の設置でこれを実現する。家やオフィスなどの空間ごとに CPLOD サーバーを設ける。

既存デバイス側には、CPLOD サーバーとの通信で、既存の API とデータを CPLOD へ提供するプログラムを追加する。

最初から空間 OS に対応した実装では、スマートフォンなどの処理能力の高いコンピュータそれぞれが CPLOD サーバーを内蔵することになるだろう。

4.1.3 要件：「社会」をシステム構成要素とする

これまでのコンピュータの利用目的の大部分は自動化であり、人間や他のコンピュータの指示でコンピュータが動くというものであった。

しかし、空間 OS が実現するアプリケーションには、自動化できなかったり、自動化すべきでなかったりするユースケースが存在する。

たとえばつぎのようなものである。

- 空調管理アプリケーションが、家の住人に窓の開閉を依頼する、
- 倒れた人を検出した警備アプリケーションが、通りがかりの人や管理責任者に救援を求める。
- エアコンの故障を検出した空調管理アプリケーションが、メンテナンス会社へ修理サービスを依頼する。
- 買い物の指示を受けた家事アプリケーションが、運動不足解消のために買い物に出かけることを提案する。

このように、空間 OS のアプリケーションでは、様々な人や組織（＝社会）を単なるユーザではなく、ワークフロー上のシステム構成要素として設計できるようにする必要がある。

空間 OS に、人や組織の認証機能を持たせる。また、人や組織、それらが持つ責任やミッション、人やプログラムによる意思決定の経緯などを扱う語彙を持たせることでこれを実現する。

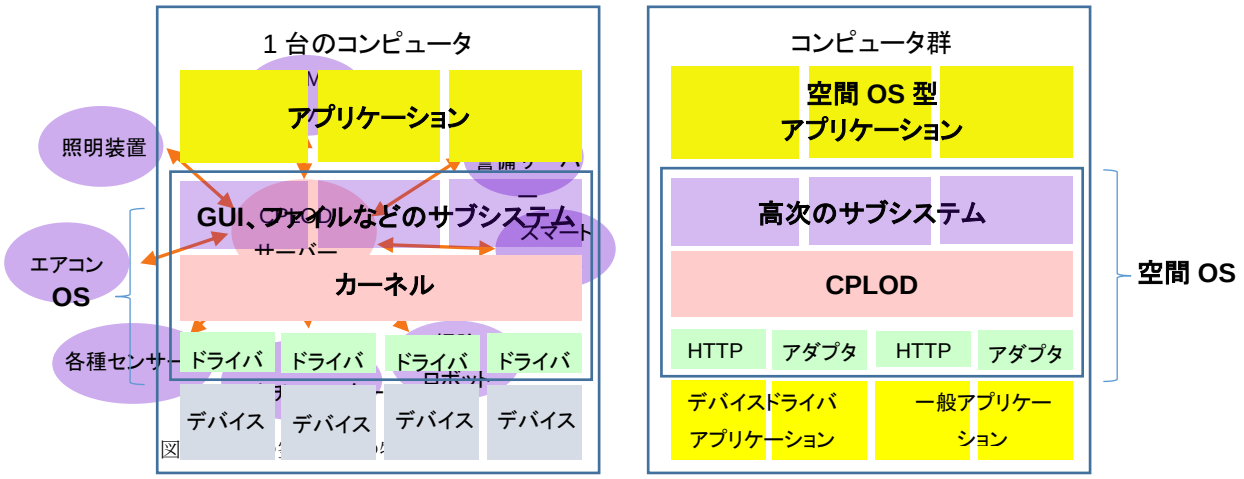
PKI や RFID などの既存の認証技術に加え、「いまそこにいる」という空間認識に基づく認証も必要である。

日常生活では、「匿名だがそこにいる人」を扱うユースケースが存在するからである。

- 店舗を訪れた客にサービスを提供する。
- 助けを求める人を援助する。

このようなユースケースで、「そこにいる」ことを認証して臨時に認可するサービスを実現できるようにしておく必要がある。

4.2 空間 OS の構成



既存 OS とアプリケーション　　空間 OS とアプリケーション

図 2. 空間 OS の論理構成

たとえば「個人の家」という空間を対象とする空間 OS の物理的なシステム構成は図 1 のようなものになる。

CPLOD へ、センサーやアクチュエーターがデータとサービスを登録し、様々な意図をもったサーバーがそれらを利用して高次のサービスを実現する。

空間 OS は既存のシステムを一気に刷新するように実現されるとは限らない。たとえば最初に CPLOD を設置するときには、既存のシステムが直接制御しているセンサーやアクチュエーターを CPLOD には接続せず、それらを制御する HEMS などが代理でサービスを提供することになるだろう。

空間 OS は複数のコンピュータで構成するシステム全体を抽象化する。図 2 に既存の OS との対比で、空間 OS の構成を示した。

既存の OS は、1 台のコンピュータの中で、デバイスを抽象化し、多くのアプリケーションが共用する機能をサブシステムとして提供する。

これに対して空間 OS は、図 1 のようなコンピュータ群が空間内へ提供するサービスとデータ全体を抽象化する。空間 OS では CPLOD がコンピュータ群を連携させるハブとなる。各コンピュータ上のサービスはそれぞれが属するレイヤや分野の語彙を使ってサービスやデータを空間内に公開する。

図 2 で「デバイスドライバアプリケーション」は、センサーやアクチュエーターを制御するプログラムである。一般 OS のデバイスドライバに当たる。デバイスドライバアプリケーションは、センサーの読み取り値を CPLOD へ記録し、CPLOD から得たリクエストによってアクチュエーターを制御する。これらのアプリケーションは標準的には CPLOD へ HTTP で接続する。しかし、デバイスの中にはきわめて低い消費電力での運用や、既存規格との互換性のために TCP/IP や HTTP を実装できないものもある。こうした TCP/IP+HTTP 以外のプロトコルによる通信に対しては、アダプタを使って CPLOD への接続を実現する。

図 2 で「高次のサブシステム」は、様々なアプリケーションが共通に使う機能である。たとえば、人の追跡、デバイスの管理などの機能を提供する。

「人の追跡」は、監視カメラなどが間欠的に報告する顔認識情報や、人感センサーが報告する「誰かはわからないが人がいる」といった情報を総合して、家のどこに誰がいるかを追跡する。このとき高次のサブシステムは、CPLOD から読み取ったデータから導いた高次の認識結果を CPLOD へ書き戻す。

「デバイスの管理」は、掃除ロボットなどの状況を把握し、アプリケーションからの利用リクエストの競合を捌いたり、故障を検出したりする。

図 2 で「空間 OS 型アプリケーション」とは、空間 OS の上で、与えられたミッションを実行するように作られたアプリケーションである。家の空間 OS であれば、住人の健康管理、ハウスキーピングなどのミッションを実行する主体である。

個人のスケジュール表、会議室予約などの既存 OS 上の一般アプリケーションは、多くの場合、空間 OS ではデバイスドライバ相当の役割となる。空間 OS

型アプリケーションが、スケジュールを読み書きしたり会議室を予約したりする動作は、CPLOD を介してセンサーやアクチュエーターを操作するのと同じ形態になるからである。

空間 OS 型アプリケーションや、高次のサブシステムの設計や実装の方法は自明でなく、今後開発してゆくべき課題である。様々なアプリケーションが空間 OS 上に作られたあと、その中から共通の機能が抽出されて、空間 OS の高次のサブシステムとなってゆくだろう。

空間 OS は、これまでの OS と違って OS のパッケージが供給されるようなものではない。空間 OS の本体は、つぎのようなソフトウエア設計のための規格の集合体である。これらの規格を満たすコンピュータを接続したときに空間 OS の実装が出現する。

- CPLOD 上の語彙とオントロジーの設計規約。
- デバイスドライバ、高次のサブシステム、空間 OS 型アプリケーションなどの階層ごとに作られる語彙、オントロジーと、サービス利用の上位プロトコル（手順）。

4.3 空間 OS の連携

4.3.1 空間 OS 同士の連携

空間 OS は、CPLOD を空間外へ公開することで、外部空間にある空間 OS と連携する。すべての CPLOD は互換なので、すべての空間 OS は連携することが可能である。

こうした連携には 2 通りの形態がある。

- サービス実現のための連携
  住宅の空間 OS が、電気設備メンテナンス業者のオフィスの空間 OS と連携するといった形態である。
- 空間の包含関係に基づく連携
  集合住宅の各戸の空間 OS が、ビルの空間 OS に接続し、ビルの空間 OS が街の空間 OS へ接続するといった形態や、 支社の空間 OS が本社の空間 OS へ接続するといった形態がある。このような連携では、連携した空間 OS が上位の空間 OS を構成するという再帰構造をもつことになる。

4.3.2 インターネットとの連携

空間 OS は、サービス実現のためにインターネット上のサイトと直接連携することがある。

- インターネット上の Web サービスを利用する。
  一般の Web サービスを CPLOD へ変換する CPLOD サーバーを置けば、空間 OS から直接アクセスできるようになる。
- LOD として公開されている様々な情報を利用する。
  メーカーが公開するデバイスやサービスインタフェースのメタデータや、DBpedia などに蓄積された情報である。

インターネットを、包含関係に基づく連携の最上位である「世界」に属する空間 OS とみなすこともできる。

既存の LOD サイトを CPLOD サーバーへバージョンアップして下位の空間 OS の情報を集約することができる。たとえば DBpedia を CPLOD サーバー化して自治体の空間 OS と接続すれば、河川の水位といった各種の公開情報をリアルタイムに更新することが可能となる。

このとき、DBpedia のような公開 LOD と空間 OS とは私的なリンクで接続する。公開 LOD が空間 OS の情報をキャッシュすることで、空間 OS への攻撃や負荷集中を防ぐことができるようになる。また、包含する家庭やオフィスの空間 OS と私的に接続している自治体 OS は、家庭やオフィスで発生したデータを匿名化して公開 LOD へプッシュすることができる。

アクセス制限機能をもつ CPLOD は一見オープンデータの精神に反するように見える。しかしこのような構成によって、これまで公開しにくかったデータをリアルタイムにオープンにすることができるようになるのである。CPLOD や空間 OS は、すべての公開可能なデータに公開する手段を与え、オープンデータの拡充と応用の拡大にも寄与するだろう。

4.4 空間 OS サーバーの現場への設置

空間 OS サーバーは、クラウドだけでなく現場（司る対象の、家、街などの空間）にも設置することができる。現場に空間 OS サーバーを設置すると、クラウドサービスでは実現できない機能を実装できるようになる。

4.4.1 プライバシーやデータの所有権の制御

家の CPLOD サーバーを家庭内に設置すれば、家の中で発生するすべてのデータを家庭内で処理・保存することができる。

これにより、きわめて機微なデータを外部に出さずに利用することができるようになる。

データの所有者は、データの所有権と所在を明確にしたうえで、必要に応じて、公開先や公開方法を選んで公開できる。

4.4.2 クラウド障害時のサービス続行

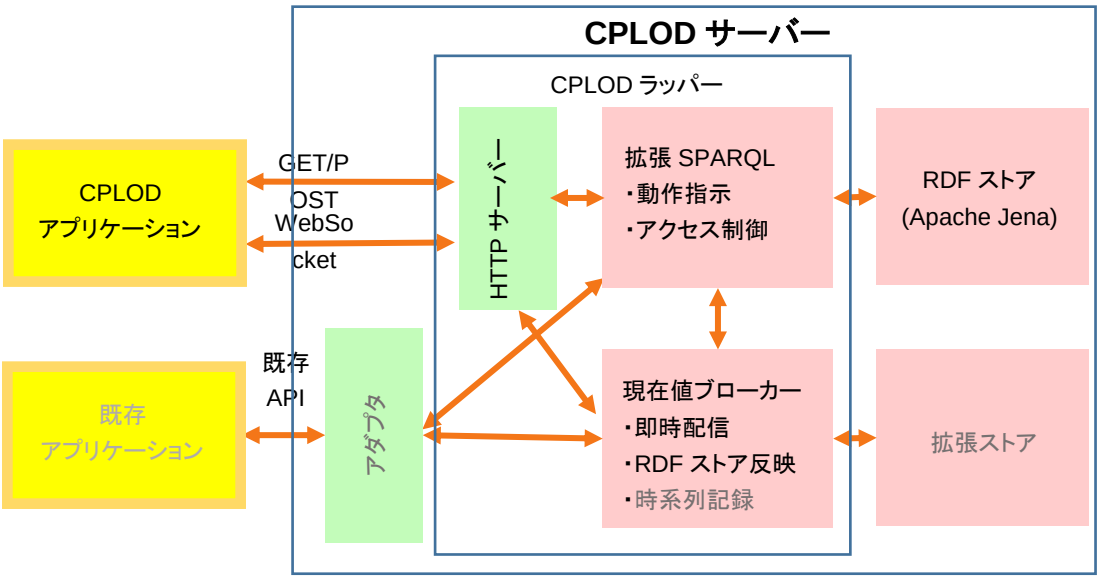


図 3. CPLOD サーバーの試作

自然災害、事故、故障、テロなどによってネットワークやクラウドに障害が発生したとき、クラウドに依存した実装では、ローカルのサービスまで停止してしまう。

空間 OS を現場に設置していれば、こうした場合でも空間 OS を介してサービスを続行できる。

広域ネットワーク障害時にも生き残れるネットワークに、ローカル接続を使ったメッシュ型ネットワークがある。空間 OS はメッシュ型ネットワークとも相性が良い。

現在はローカル側のコンピュータ上のサービスのほとんどがクライアント型である。メッシュ型ネットワークで通信接続できたとしても、両方がクライアントでは多くのサービスが成立しない。これに対して空間 OS は CPLOD サーバーを持つため、メッシュ型ネットワークの連携のハブとなることができる。たとえば地域の災害情報やボランティア情報を、その地域内で広域ネットワークを使わずに共有することができる。

4.4.3 リアルタイムの連携

空間 OS を現場に設置することで、現場の機器同士の総連携における遅延を小さくすることができる。空間内に配備した様々な装置を連携させて、いわゆる VR や AR タイプのユーザインタフェースを実現するには、空間 OS が適している。

クラウドとの通信では遅延が発生する。通信機器を極限まで高速化したとしても光の速さを超えることはできないし、実際には中継装置の存在によって数 10～数 100ms の遅延が発生する。

UI の実装では、概ね 100ms 以上の遅延を避けなければならないケースが多く、ローカル接続による高速化には大きなメリットがある。

5. CPLOD サーバーの実装実験

CPLOD サーバーの実装実験を行った。図 3 に構成を示す。RDF ストアである Apache Jena を CPLOD ラッパーと呼ぶモジュールがラップする形で実装した。図中に薄色で示した、「時系列記録」「アダプタ」「拡張ストア」はこれまでの説明との整合のために示したもので、これまでの実験では実装していない。

CPLOD ラッパーは外部からの HTTP アクセスを解釈し Apache Jena に蓄積した RDF 情報にアクセスしながらクエリを実行する。

拡張 SPARQL は、パーザジェネレータの ANTLR と、インターネット上に公開されている SPARQL の文法定義ファイルを使って、Python 言語で実装した。HTTP サーバーが受信した SPARQL リクエストを構文解析して実行する、CPLOD サーバーの本体である。

現在値ブローカーは、物理空間と RDF 内部データの写像を行う。たとえばセンサーの値を読み出すリクエストを実行すると、拡張 SPARQL が Apache Jena へクエリを発行し、IRI として記述してあるセンサーの ID を読み出す。現在値ブローカーは、その ID に対応するセンサー値を拡張 SPARQL へ与えて現在値を応答する。

拡張 SPARQL は、受信した SPARQL の構文を加工してアクセス権限制御を実現する。元のクエリを加工し、与えられた SPARQL クエリに含まれる変数それぞれについてその変数に関するアクセス権限情報を取り出し、その利用権限の公開レベルに達しない値を SPARQL の FILTER で削除するようなクエリを合成する。これを Apache Jena へ送信し、アクセス権限のフィルタリングを Apache Jena 側で実行するようにした。

ある権限者に対して個々のデータが持つ公開レベルは、公開、内部利用可、非公開の 3 段階とした。内部利用可とは、SPARQL によるパターンマッチングと CPLOD サーバーの内部では使えるが、検索結果の中には含めることがないレベルである。たとえばメールアドレスを内部利用可にしておくと、「この部屋にいる人全員のメールアドレスを取得する」というクエリの応答は空とし、「この部屋にいる人全員のメールアドレスへメールを出す」を実行可能とすることができる。

実験結果の概要はつぎのとおりである。

- CPLOD は実装できる。
  RDF ストアの履歴機能は実装しなかったが、センサーなどの測定値に限れば現在値ブローカーで記録できる見通しである。RDF ストア内の比較的低速で変化するデータは、テキスト化してバージョン管理システムで管理するといった方法を検討している。
- 家の中での制御に使えそうである。
  パフォーマンスを確保するために RDF ストアをオンメモリで実行する必要がある。家の中にある数百～数千のデバイスという範囲ではメモリ容量内でデータを表現できるだろう。WebSocket での変化通知の遅延は ms オーダーとなった。
- ラッパー方式では実現できないことがある。
  たとえば、プロパティパスの経路の途中での権限チェックができない、アクセス権限チェックが正しく働いていることを証明しにくい、物理ノードへの書き込みを検知するために SPARQL の構文拡張が必要だった、などである。

この結果から、現在は Apache Jena の SPARQL エンジン自体を改造する方法で、CPLOD 実装を進めている。

6. おわりに

本稿では、CPLOD および空間 OS を実装することが現行の技術で可能であり、空間 OS によってコンピュータを社会のインフラストラクチャにすることが技術的には可能であろうことを示した。

本稿で示した空間 OS アーキテクチャは、これまで懸案とされているつぎの問題の解決策の一つとしても使えるだろう。

- IoT が生成するプライベートなデータのセキュリティ確保と権利問題を、生成データのローカルでの蓄積と公開制御によって解決する。
- IoT が生成するデータを、家、町、県、国といった空間の階層で集約することによって、処理を適切に分散し、IoT ビッグデータの処理能力の問題を解決する。各階層で匿名化などのセキュリティを確保するための処理を加えることで、IoT ビッグデータの権利やセキュリティ問題を解決し、多様な目的での利用を可能とする。
- クラウドや広域ネットワークのダウン時にも、空間 OS がローカルでの処理を続行できるようにすることによって、災害やテロなどによる広域障害に対するレジリエンスを実現する。
- 機械学習技術を様々な現場の処理に適用するための共通基盤を、空間 OS が提供する。
- 空間 OS を、それが司る空間に関連する人や事業者が利用する共通基盤とし、スモールビジネスや地域経済といった、大きな初期投資が難しい分野でのインフラストラクチャとする。

空間 OS の本格的な実装開発の実現が今後の課題である。

試作中の空間 OS 実装は部分的なものであり、実用化には多くの作業が必要である。語彙やオントロジーの設計、機器や人の参加や移動に伴う RDF データ構造の自動生成、様々な認証機能の実装など、実現方法が自明ではない課題も多数存在する。

本稿によって、空間 OS の課題解決や開発への参加を検討していただければ幸いである。

謝辞

筆者に活動の場を与えている日本総合システム株式会社、空間 OS の構想と実装をともにしてきた先端 IT 活用推進コンソーシアムのビジネス AR 研究部会メンバーと、「空気を読む家」プロジェクトで空間 OS の実験実装を使っていただいている他の部会の方々に感謝いたします。

講習会で LOD の基礎知識をご講義いただいた LOD イニシアチブの皆様と、SIG SWO で本稿の発表の機会が得られることを示していただいたオントロノミー合同会社の小出誠二氏に感謝いたします。

参考文献

[1] 内閣府：平成 28 年版　高齢社会白書,2016
[2] 厚生労働省：認知症施策推進総合戦略～認知症高齢者等にやさしい地域づくりに向けて～（新オレンジ

プラン)」（概要）,2015