情報処理学会論文誌 Vol.54 No.2 659–671 (Feb. 2013)

インターネットサービスプラットフォームにおけるロボットサービス提供手法

加藤 由花[1,a)] 岡部 泉[1] 村川 賀彦[2] 岡林 桂樹[2] 植木 美和[2] 土屋 陽介[1] 成田 雅彦[1]

受付日 2012年5月14日, 採録日 2012年11月2日

概要：我々は，ロボットサービスのインターネット化を実現するための研究開発環境 RSi-Cloud（RSi Resaserch Cloud）の研究を進めている．これまで，Web ブラウザ経由でロボットカメラの画像を監視する見守りサービスを構築し，RSNP（Robot Service Network Protocol）を実装したロボットからの接続を実現してきた．今回，RSi-Cloud 上に，ロボットのためのソフトウェア部品（RT コンポーネントなど）を，RSNP を実装したロボットと同様に扱う仕組みを実現したのでその結果を報告する．この仕組みを利用すると，サービス機能をサーバ上にデプロイする必要がなく，またファイアウォール内に配備可能になるため，ロボットサービス開発の拡充が期待できる．本稿では，RT コンポーネントである VisionModule を組み込んだプロトタイプシステムを開発し，提案手法の有効性を検証する．

キーワード：サービス基盤，ロボットサービス，Web サービス，クラウド環境，RT コンポーネント

A Method Providing Robot Services on an Internet Service Platform

Yuka Kato[1,a)] Sen Okabe[1] Yoshihiko Murakawa[2] Keiju Okabayashi[2] Miwa Ueki[2]
Yosuke Tsuchiya[1] Masahiko Narita[1]

Received: May 14, 2012, Accepted: November 2, 2012

Abstract: We have proposed the RSi Research Cloud (RSi-Cloud), which enables integration of robot services with internet services. Until now, we have developed a surveillance service using robot cameras via Web browsers, and have realized internet connection from robots implemented with RSNP (Robot Service Network Protocol). This paper reports an implementation result of the mechanism which treats software components for robots (such as RT Components) by the same way as RSNP robot clients. The mechanism makes it possible to provide service components without deploying in a server side, and to deploy the components within firewall systems, and as a result, a robot service expansion can be expected. In this paper, we develop a prototype system using an RT Component, VisionModule, and verify the effectiveness and usefulness of the proposed method.

Keywords: service platform, robot services, web services, cloud environment, RT components

1. はじめに

近年，クラウドロボティクスへの関心が高まっている．クラウドコンピューティングはインターネット分野の技術であるが，コンピュータのユビキタス化にともない，Internet of Things [1]，Cyber-Physical Systems [2] など，実世界サービスとの融合が急速に進んでいる．ロボットやロボットサービスのネットワーク化については，これまでも RT（Robotics Technologies）ミドルウェア [3] や ROS [4] などの活動があったが，上記分野との融合を視野に，ロボットサービスのクラウド化が進んでいる．たとえば，RoboEarth [5]，CloudRobotics [6], [7] などの研究が行われている．ここで，ロボットサービスとは，ネットワー

[1] 産業技術大学院大学
Advanced Institute of Industrial Technology, Shinagawa, Tokyo 140–0011, Japan
[2] 株式会社富士通研究所
FUJITSU LABORATORIES LTD., Kawasaki, Kanagawa 211–8588, Japan
[a)] yuka@aiit.ac.jp

クを介してロボットが提供するサービスもしくは物理的なサービスと定義する．

このような背景の下，我々は，ロボットサービスのインターネット化を実現するための研究開発環境である RSi Research Cloud（RSi-Cloud）[*1]の研究を進めている [8], [9]．RSi-Cloud は，ロボットサービスイニシアティブ（Robot Service initiative: RSi）[10] によって策定されたオープンなロボットサービス向けプロトコル仕様である RSNP（Robot Service Network Protocol）[11], [12] を利用して構築されたサービスプラットフォームである．サービスロボット[*2]の産業化の試みは，ソニーのアイボの発売開始をはじめ，2000 年ごろから大手ベンダの参入により活発化してきたが，RSi はこのような中で 2004 年に設立され，ロボットサービスのインターネット化とそのための標準仕様の作成を推進してきた．RSNP の現在のバージョンは 2.3 であり，RSi ではこれまで，仕様を実装した Java ライブラリを SDK として提供してきた．しかし，様々な立場の人が開発に参加し，ロボットサービスが普及していくためには，Web 系の開発者も容易に開発を実施できるオープンな研究開発環境が不可欠であり，我々は RSi-Cloud を提案するに至った．プラットフォームへの要求条件は下記のとおりである．

- インターネットとロボットサービスのシームレスな統合が可能であること．つまり，RSi のロボットサービスモデル上で実世界データ利用サービスとの連携が可能であること．その結果，様々なロボットが共通して利用できるロボットサービスの提供を可能にする．
- ロボットの専門家ではないインターネットサービスの研究者が，容易にロボットサービスを開発できる仕組みを構築すること．ロボットには，認識，タスク処理，割込みなど，データ収集とアクションの依頼以外の複雑な処理が要求される．このようなロボットサービスを開発できる，または利用できる仕組みを実現する．
- 開発のしきいを低くするために，様々なサービス部品を用意しておくこと．また，部品の開発，プラットフォームを介した機能の提供が容易にできる仕組みを用意すること．サービス部品には，Google Map のようにインターネットサービスとして提供される API と，ロボットシステムを構成する機能要素ごとのプログラムであるロボット機能モジュールの 2 種類が存在する．

RSi-Cloud では，Web ベースの仕様を利用することにより 1 番目の課題を，ロボットに共通した仕様である RSNP を利用することにより 2 番目の課題を，さらにアカウント管理，GUI など共通の機能を実装することにより 3 番目の課題を解決してきた．現在，ロボットを RSi-Cloud に接続すると，Web ブラウザ経由で地図上にロボットが表示され，カメラ画像により遠隔見守りが可能になるサービスを提供している．これにより，ロボットシステム（ロボット側に実装されるロボットアプリケーション）を容易に開発可能な仕組みを提供している．しかし，サービス部品やサーバ側のサービス自身の開発は，RSi-Cloud 上にソフトウェアを配備する必要があり，プラットフォーム上への実装が不可能であるモジュール（特定のハードウェアへの実装を前提としている，地理的な制約がある，ライセンス上の問題があるなど），または実装が困難であるモジュール（環境依存のコードが存在する，ロボット側のプログラムに対してサーバ側のプログラムの開発の難易度が高い）が存在するという問題があった．

本稿では，この問題を解決するために，サービス部品，特にロボット機能モジュールをプラットフォーム上に実装せず，HTTP 通信上の擬似 Push 機能を利用してローカル環境に実装する方式を提案する．さらに，RSi-Cloud 上に提案手法を実装し，その有効性を検証する．ロボットシステムでは様々な粒度での部品化が行われ，たとえば，モータやセンサといった単機能のデバイス，移動ロボットやアームなど複合的なシステム，あるいは様々な処理を行うアルゴリズム（認識，判断，行動制御など）といった単位が考えられ，それら階層的な集積によりシステムが構築される．これら様々な粒度のロボット機能モジュールのインターネットサービス化が本稿での考察対象になる．

以下，2 章で関連研究を紹介した後，3 章でプラットフォーム（RSi-Cloud）の概要を述べる．次に，4 章において提案するロボット機能モジュール提供手法について詳述した後，5 章でその性能評価結果を示す．その後，6 章において提案手法の AR（Augumented Reality）サービスへの適用例を述べた後，7 章で本稿をまとめる．

2. 関連研究

本稿では，ロボットサービスをインターネット経由で利用するための仕組みを提案するが，近年，このようなサービスプラットフォーム技術に関する多くの研究開発が行われている．本章では，ロボットサービスのためのクラウド基盤に関する研究，センサネットワークのためのクラウド基盤に関する研究，ロボット機能モジュールのコンポーネント化に関する研究について説明する．

2.1 ロボットサービスのためのクラウド基盤

ネットワークロボット分野におけるクラウド基盤としては，RoboEarth [5] や CloudRobotics [6], [7] などが提案されている．RoboEarth は，クラウド環境をデータストアとして利用し，クラウド上で Web サービスとの連携を実

*1 将来構想としてクラウド環境への展開を予定しているが，現在はインターネットに接続された学内サーバ上で運用している．
*2 サービスロボットとは，工場内などで利用される産業用ロボットとは異なり，人のそばでサービスを提供するロボットのことを指す．

現するプラットフォームである．ただし，共通利用可能なロボットサービスをクラウド上に配備する仕組みは提供されておらず，ロボットサービス自身はロボット側に実装する仕組みになっている．一方，CloudRobotics は，これまでロボット側に実装されてきたロボット機能をクラウド側に配備し，ロボット側の処理を軽量化することを目的としている．近年，軽量で簡易なロボットが数多く製品化されており，クラウド上のサービスと組み合わせることにより様々なサービスの提供が期待できる．ただし，様々な種類のロボットで共通利用可能なロボットサービスを開発する仕組みは用意されておらず，サービス開発者にとっては，ロボット上で実装していた機能を，クラウド上で同様に実装する必要がある．また，他のロボットとの機能の連携などは考慮されていない．

その他，ロボット技術に関するソフトウェア基盤としては，次世代ロボット知能化技術開発プロジェクト [13]，ROS [4], Robotics Studio [14], OpenRoads [15], Ubiquitous Robotic Companion（URC）[16]，iRobot ConnectR [17]，ネットワークロボットフォーラム [18]，などが存在するが，これらはハードウェアとロボットアプリケーション間の通信を規定したものであり，クラウド環境は主にデータストアまたは膨大な計算資源として利用されている．

2.2 センサネットワークのためのクラウド基盤

ユビキタスコンピューティングの進展 [19], [20] にともない，インターネットサービス基盤は実世界サービスと融合する方向に進んでいる．センサネットワークのためのクラウド基盤はその 1 つの実現形態である．前節の研究がロボット分野から発展したのに対し，本節の研究は通信・コンピュータ分野から発展している点に特徴がある．具体的には，Web of Things [1]，Sensor-Cloud [21]，X-Sensor [22] などが提案されている．これらのプラットフォームは，センサから収集される大量のデータをクラウド上で処理するモデルとなっており，これらのデータを利用したアプリケーションは原則クラウド上に実装する必要がある．センサの仮想化が目的であるため，ロボットのように自律的な処理を実行することは想定されていない．つまり，これらの基盤を利用してロボットサービスを開発するには，すべてのサービスを，各自でクラウド上に実装する必要があり，ロボットサービスの開発者以外の参入は非常に困難である．

2.3 ロボット機能モジュールのコンポーネント化

本稿における研究のモチベーションは，様々なロボットで共通に利用可能なサービス部品を，インターネット経由で利用可能な仕組みを実現したいというものである．近年，サービスロボットを対象に，ロボット機能モジュールをコンポーネント化して提供する研究が活発に行われており，これらのコンポーネントの利用が本稿におけるターゲットの 1 つである．

代表的なロボット機能モジュールとしては，RT コンポーネント [23] と ROS パッケージ [24], [25] がある．RT コンポーネントは，RT ミドルウェア規格を用いたコンポーネントであり，次世代ロボット知能化技術開発プロジェクト [13] などがこれを利用した研究開発を推進してきた．プロジェクトの成果はソースコードやバイナリで公開することが奨励されており，数多くの RT コンポーネントが一般に公開されている．Kinect センサから全身の位置を取り出し出力するモジュール [26]，ハードウェアに組み込まれてファームウェアとして提供されるアームユニットモジュール（ソースコードは非公開）[27]，LRF 距離データ，オドメトリ情報を取得し，地図とのマッチングを行うことで自己位置を推定するモジュール [28]，入力画像から顔を検出し，顔の位置や検出した顔の数を出力するモジュール [29] など，様々な粒度のコンポーネントがある．これらはロボット内あるいは LAN 環境内での利用が前提であり，開発したコンポーネントやそれらを統合したシステムをインターネット越しに連携して利用する技術や方法は確立されていない．一方，ROS は米 Willow Garage 社で開発が進められているロボット向けのハードウェアおよびソフトウェアであるが，現状ネットワークを介したサービスは提供していない．

3. RSi-Cloud

次に，本稿で考察対象とするプラットフォームである RSi-Cloud について，その概要を説明する．まず，RSi-Cloud で利用している RSNP について説明した後，システムの構成，提供サービスの例を示す．

3.1 RSNP の概要

3.1.1 モデルの概要

RSi では，ロボットサービスを，ネットワークを介してロボットが提供する情報サービス，もしくは物理的サービスと定義している．**図 1** にモデルの概要を示す．ロボット

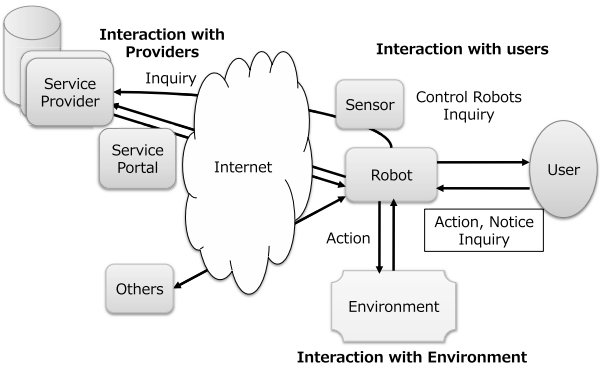

図 1 RSi のロボットサービスモデル
Fig. 1 The robot service model on RSi.

やサービスプロバイダ，ユーザなどから構成され，同期・非同期の通信による動作や動作パターンの指示や結果の取り出し，ロボットからプロバイダへの問合せ・通知，サービスの提供，ユーザを含む外界とのやりとりを行うことができる．このようにロボットをインターネットに接続する利点は，人とロボットが協調してサービスを行う「協調型」ロボットを実現できること，インターネット上のコンテンツを再利用できることなどにある．RSNP は，このモデルに従ってサービスのプロトコルを規定しており，異なるベンダで独立して開発したロボット/サービスの間での相互運用が可能になっている．

3.1.2 RSNP の特徴

RSNP の特徴は下記のとおりである．

- インターネットとの整合性が高く，標準化された豊富な機能の上にサービスを構築でき，多様なロボットと各種サービスに適用できる．さらに複数実装間で相互接続ができる．
- 計算機資源の限られているロボットへの情報提供や指示が迅速に行える．
- ロボットらしいサービスを提供できる．画像やセンサ情報をロボットサービスへ連続的に転送でき，種々プラットフォームとの連携が実現できる．

プラットフォームのベースは，インターネットやシステム構築向け通信基盤である Web サービス基盤を利用している．そのため，高信頼メッセージング機能，セキュリティ機能など，インターネットと整合性の高い標準化された機能を利用可能である．プラットフォーム自身は共通サービスとロボットサービスからなり，共通サービスでは，各種サービスをサポートするために，Pull 型・Push 型，同期・非同期型の通信モデルを提供している．一方，ロボットサービスは基本プロファイルと応用プロファイルからなり，カメラ・音声入出力などのマルチメディア機能や，前後回転動作など単純な動作・パターン動作などのロボットの動きを基本プロファイルとして提供し，天気サービス・見守りサービス・リモート制御などのサービスを応用プロファイルとして提供する．このように，ロボットの機能とロボットサービスの機能をモジュール化して提供することにより，機能の異なる多様なロボットや各種サービスへの適用を可能としている．

3.1.3 RSNP の実装

RSNP2.3 準拠のライブラリ実装としては，富士通研究所で開発された FJLIB がある．これは，ロボット用 API とサーバ用 API のそれぞれに対して呼び出し側のインタフェース実装と，呼び出された側の処理を記述するためのフレームワークを提供している．サービス開発実行環境を**図 2** に示す．FJLIB では，RSNP の下位層に 2 経路の HTTP 通信を使用し，それぞれ上り方向，下り方向の通信を担っている．上り方向通信は，HTTP リクエストのペイロードにデータを入れサーバへ送信し，下り方向の通信は，HTTP レスポンスのペイロードにデータを入れロボットへ送信する．下り方向の通信は，迅速にデータを転送するために，Comet を使用している．

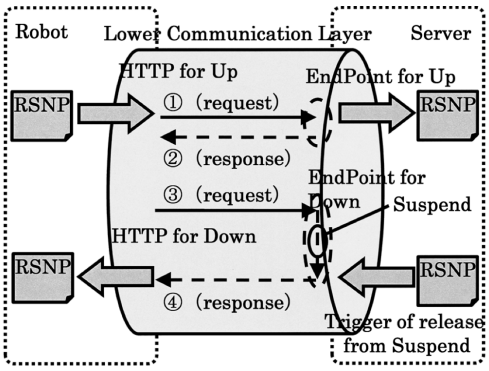

図 2 RSNP2.3 におけるロボットサービス開発環境
Fig. 2 Robot service development environment on RSNP 2.3.

3.2 システムの構成

インターネット上の RSi ロボットサービスの開発を促進し立ち上げを支援するためには，(1) ロボット側で RSNP に対応した機能を容易に開発でき，サービスをただちに利用して効果を確かめ，速やかにインターネットを通して実運用を実現すること，(2) ユーザやサービス提供者が容易にロボットサービスを開発でき，それが広く他のロボットから利用できること，(3) 開発者を増やし，ユーザの開発したロボットやロボットサービスが広く利用される環境を提供することが必要である．これらの設計指針に従い，RSi-Cloud を構築した．システムの構成を**図 3** に示す．

システムは，RSNP サーバ，RSNP クライアント，GUI，アカウント管理の 4 つのモジュールから構成される．以下，それぞれについて説明する．なお，本稿では図 3 に示すように，主にエンドユーザが Web サーバを介してロボットサービスを利用する形態を考えるが，コミュニケーションロボットなど，ロボットとエンドユーザが地理的に同じ場所に存在し，ロボットから直接サービスを受ける形態での利用も可能である．

3.2.1 RSNP サーバ

RSNP サーバは，サーバアプリケーション（ロボットサービス）を実装するモジュールである．サーバ開発者は，RSNP サーバ（RSNP に対応した機能のサーバ側実装）と開発したアプリケーションを同一クラスローダ上に実装する．このとき，ロボット側の機能は RSNP ライブラリ（ここでは FJLIB）に隠蔽されるので，サーバ開発者はこの実装を意識する必要がない．つまり，インターネットサービスの研究者がロボットと独立にサービスを開発することが可能になる．

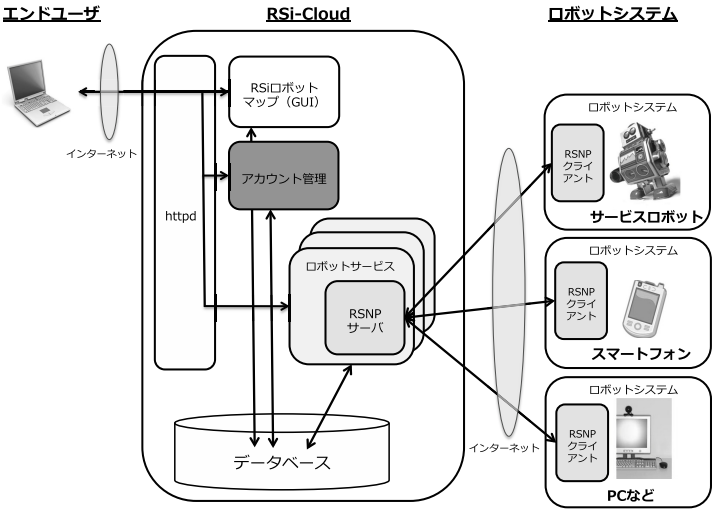

図 3 システムの構成
Fig. 3 System architecture.

3.2.2 RSNP クライアント

RSNP クライアントは，クライアントアプリケーション（ロボットシステム）を実装するモジュールである．RSNP クライアント（RSNP に対応した機能のクライアント側実装）を利用し，それぞれのロボットやデバイスに適したソフトウェアを開発する．RSNP を利用することで，RSi-Cloud 上で提供されている様々なサービスの利用が可能になる．

なお，RSNP では，ロボットとサーバ間の接続に擬似 Push 機能を利用しているため，NAT 越え通信が可能である．そのため，ロボットをファイアウォール内に設置したセキュアな運用が実現できる．

3.2.3 GUI

これは，図 3 における RSi ロボットマップの部分に相当し，RSi-Cloud のフロントエンドの役割を果たすモジュールである．Web サーバとして実装され，エンドユーザは Web ブラウザ経由でこの GUI にアクセスする．GUI は別途実装することも可能であるが，現状では，RSi-Cloud の基本サービスとして，RSi ロボットマップを提供している．地図アプリ上にロボットの現在位置を表示するインタフェースを提供しているが，ロボット表示だけではなく，ロボットをクリックすることで吹出しに各種情報を表示する機能を持っている．ここから各種サービスへのリンクをたどることも可能であり，ロボットサービスのポータルとしての役割を果たしている．GUI のイメージを図 **4** に示す．

3.2.4 アカウント管理

アカウント管理機能を提供するモジュールである．ロボットサービスへのアクセス制御などを実現するためには，ロボットやそのユーザを管理する機能が必要であり，RSi-Cloud ではプラットフォームの機能として本機能を提供している．ロボットの追加，登録情報の修正，変更，削除などはこの機能を介して実施され，管理情報はデータベースに保存される．サーバ開発者は必要に応じてこの機能を利用可能である．

図 4 RSi ロボットマップ
Fig. 4 RSi robot map.

3.3 提供サービスの例

現在，RSi-Cloud 上でロボットシステムに提供しているサービスは，基本サービスとして，前述した RSi ロボットマップ，アカウント管理機能の 2 つがある．これらに追加して，一実装例として，ロボットみまもりサービスを提供している．これは，ロボットカメラで撮影された映像を，Web ブラウザ上で監視できるサービスである．RSi-Cloud では，ロボットシステム用のサンプルコード（PC に Web カメラを接続した形態を想定）を提供しており，本プログラムを実行するとただちに RSi-Cloud 上でサービスを利用できるようになる．ロボットみまもりサービスのイメージを図 **5** に示す．

図 5 ロボットみまもりサービス
Fig. 5 An image of the surveillance camera service.

4. マイクロサービス

前章で述べたように，RSi-Cloud では，基本的なサーバ機能（ポータルに相当する GUI，アカウント管理機能など）をプラットフォーム上で提供し，RSi-Cloud に接続するロボットシステムを容易に開発できる環境を実現してきた．本章では，ロボット機能モジュールが提供する機能を，RSi-Cloud 経由で利用・提供を可能にする仕組みを提案する．これにより，サービス部品としてサーバ機能を容易に提供することが可能になる [30], [31]．以降，このロボット機能モジュールが提供するサービスのことをマイクロサービスと呼ぶ．

4.1 機能要件

マイクロサービスに対する機能要件を以下に示す．

- 機能の提供が容易であること．マイクロサービスをプラットフォーム上に組み込む，またはロボットにビルドする必要がないことが望ましい．
- マイクロサービス提供者にとって，複雑なサーバ管理やセキュリティ対策が不要であること．
- 多数の利用者が同時に利用可能であること．分散処理が必要になるが，必要に応じてマイクロサービス自身の増設が可能である仕組みが要求される．

2 章で述べたように，現在，数多くのロボット機能モジュールのコンポーネントが開発されている．これらのコンポーネントはロボット側での実行を前提に構築されているが，これらの機能を外出しし，インターネット経由で利用できるようになると様々なメリットが生まれる．たとえば，安価な低機能ロボットであっても，インターネット経由で様々なコンポーネントを組み合わせて利用することにより，多様なサービスの利用が可能になる．これは，スマートフォンをクラウドサービスと組み合わせて利用するモデルと同様の考え方である．また，ロボットは多様なハードウェア構成を持つことから，ソースコードやバイナリとして提供されたアプリケーションの導入は予想以上に困難である．そのため，ソフトウェアのデプロイ作業を必要とせず，ただちにサービスとして機能を利用できるメリットは大きい．一方，コンポーネント開発者にとっても，多くのユーザに機能を利用してもらえるというメリットがある．

4.2 機能の概要

機能要件に従い，マイクロサービス提供手法の設計を行う．RSNP は，擬似 Push 機能を利用することで，ファイアウォールの内側に配備されたロボットにインターネット経由でデータを送付する仕組みを持っている．この仕組みをマイクロサービスに拡張すると，プラットフォーム上に機能を配備することなく，インターネット経由で共通の機能を利用する仕組みが実現できる．RSi-Cloud ではこの仕組みを利用し，コンポーネントの外出しを実現する．さらに，サービスの同時利用を可能にするために，プラットフォームが受信したマイクロサービス本体へのサービス要求をキューイングし，サービスに割り当てていく．

マイクロサービスの実現形態としては，コンポーネントの機能を RSi-Cloud 上に実装する形態も考えられる．2 章で述べたように，ロボット機能モジュールには様々な粒度のものが存在し，行動制御や認識などの各種アルゴリズムであればプラットフォーム上への実装も可能である．CloudRobotics など，既存のロボットサービスのためのクラウド基盤では，通常この形態を想定している．RSi-Cloud においても，RSNP サーバとしてこれらの機能を実装することは可能である．しかし，ロボット機能モジュールには，ハードウェアに組み込まれたコンポーネント（アームの制御など），コンポーネントの地理的な位置が意味を持つコンポーネント（ある地点のロボットの遠隔操作など），ライセンスの問題からクラウド環境へのデプロイができないコンポーネントなどが存在し，既存のクラウド環境ではこれらへの対応は不可能である．そこで本稿では，コンポーネントの外出しによりマイクロサービスの機能を実現することとした．

マイクロサービスの仕組みを利用すると，コンポーネント自身がインターネットに公開されたインタフェース（API）として安全に利用できるようになり，1 番目の機能要件を満足する．またサーバの管理や運用はプラットフォーム側で実施されるため，2 番目の機能要件を満足する．さらに，キューイング機構により分散処理を実現するため，3 番目の機能要件も満足する．マイクロサービスのイメージを**図 6** に示す．図中の CR，MF，MB などの記号，および（4）などの番号については次節で説明する．この例では，カメラ機能を持たない簡易ロボット（サービスロボットの部分に相当）が，ある場所に設置された監視カメラ（各種デバイスなどの部分に相当）を操作（ズーム，

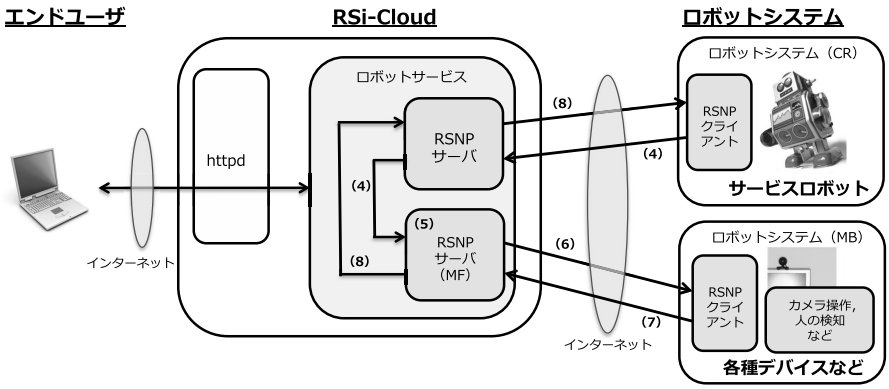

図 6 マイクロサービスのイメージ
Fig. 6 An image of micro-services.

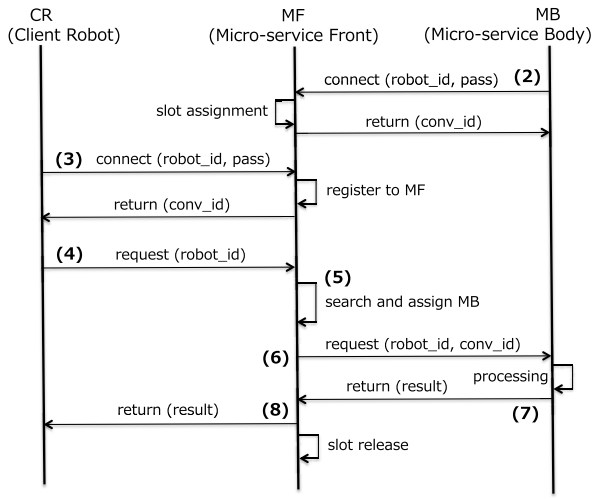

図 7 処理のシーケンス
Fig. 7 A sequence diagram.

パン，チルトなど）しながら RT コンポーネント化された人検知機能を利用し，エンドユーザからの遠隔操作を実現するサービスを想定している．カメラ操作，人の検知機能は RT コンポーネントとしてカメラとともに提供されるため，これを外出しし，RSNP の擬似 Push 機能を利用し，RSi-Cloud 経由でファイアウォール内部の機能にアクセスする．この例では，カメラの物理的な位置が意味を持つため，カメラ機能そのものをプラットフォーム上に実装することはできない．人検出機能のみを切り出し，プラットフォーム上に実装することは可能だが，Web サービス化した上でサーバへのデプロイが必要になり，コンポーネントを開発するロボット技術者にとってはしきいが高い．

4.3 機能の詳細

マイクロサービスの機構を実現するために，マイクロサービスを呼び出すロボット，プラットフォーム上で要求を受付けるモジュール，実際にサービスを提供するマイクロサービス本体との間の処理の流れを設計した．処理シーケンスを**図 7** に示す．ここでは，呼び出し側のロボットを CR（Client Robot），プラットフォーム上のモジュールを MF（Micro-service Front），マイクロサービス本体を MB（Micro-service Body）と呼ぶ．CR と MB は，RSi-Cloud におけるロボットシステムの部分に相当し，MF は RSNP サーバとして実現されている．処理の概要を以下に示す（下記の番号は図 6，図 7 中の番号に対応する）．

(1) サービス開始前に，MB と CR の情報（*robot_id*, *password*）を RSi-Cloud 上に登録しておく．これらの情報は認証に利用される．
(2) MB が MF に接続する．認証が成功すると会話 ID（*conv_id*）が発行され，該当 MB は MF に登録される．
(3) CR が MF に接続する．認証が成功すると *conv_id* が発行される．
(4) CR が *robot_id* を指定し，MF にロボットサービスを要求する．
(5) MF は該当 MB を検索し，処理を割り当てる．
(6) 擬似 Push 機能を利用し，MF は該当 MB に処理を依頼する．
(7) MB は処理を実施し，結果を MF に返す．
(8) MF は結果を CR に返す．

提案方式では，多くのロボットサービスがサービスの同時利用を可能とするために，キューイング機構を導入している（図 7 の (5)）．RSi-Cloud では，本機能は共有メモリとして実装され，実行可能なサービス（MB）に対し，1 つのスロットを割り当てている．以下に，マイクロサービスにおける処理の割当て方式について説明する．ここで，MF に登録される MB の *robot_id*（MB 名に相当）を $\{b_0, b_1, \ldots\}$，*conv_id* を $\{cb_0, cb_1, \ldots\}$，CR からの要求（*robot_id*, MB 名）を $\{r_0, r_1, \ldots\}$，*conv_id* を $\{cr_0, cr_1, \ldots\}$，i 番目のスロットの状態を $S_c(i)$，スロットに登録された MB の *conv_id* を $S_b(i)$，スロットに割り当てられた CR の *conv_id* を $S_r(i)$ $(i = 0, 1, \ldots, N-1,\ N$ はスロット数），キューイングされた CR からの要求を $Q(j)$ とする $(j = 0, 1, \ldots)$．MF は *robot_id* と *conv_id* の対応表を保有する．スロットの状態 $S_c(n)$ は $[ready, waiting, running, terminated, empty]$

の5つの値をとり，$S_b(n)$ には $cb_0, cb_1, \ldots$ のいずれかまたは $null$，$S_r(n)$ には $cr_0, cr_1, \ldots$ のいずれかまたは $null$，$Q(n)$ には $cr_0, cr_1, \ldots$ のうちの1つ以上または $null$ が入る．

Step 0 スロットの初期化：
すべての i に対し，$S_c(i)$ は $empty$，$S_b(i)$，$S_r(i)$，$Q(i)$ は $null$ を初期値とする．

Step 1 MBの登録（登録対象となるMBを b_j とする）：
(1) $S_c(i)$ が $empty$ であるスロット i を探す．
(2) i が見つからない場合，一定期間待機した後(1)に戻る．見つかった場合，$S_c(i)$ を $ready$ とし，b_j に対応する cb_k を生成して $S_b(i)$ に cb_k を登録する．

Step 2 CRからの要求（対象となるCRを cr_l とする）：
(1) b_m と cb_n の対応表より，$b_m = r_l$ となる cb_n をすべて探す．b_m が見つからない場合，CRにエラーを返し処理を終了する．
(2) $S_b(i) = cb_n$ となるスロット i をすべて探す．i が見つからない場合，CRにエラーを返し処理を終了する．
(3) $ready$ 状態のスロットがあれば，$S_c(i)$ を $waiting$ とし，$S_r(i)$ に cr_l を登録する．$ready$ 状態のスロットがない場合には，$Q(n)$ に cn_l を追加する．

Step 3 MFからの処理依頼：
(1) MFは $waiting$ 状態のスロット i を探し，$S_r(i)$ の要求をBFに依頼する．$S_c(i)$ を $running$ とし，処理の完了を待つ．
(2) 終了通知を受けると，$S_c(i)$ を $terminated$ とし，結果をCRに返す．

Step 4 後処理：
(1) MFは $terminated$ 状態にあるスロット i を探し，終了処理を実施したうえで $S_r(i)$ を $null$ に，$S_c(i)$ を $waiting$ にする．
(2) MBがサービスを終了した場合はスロットを初期化する．

Step 5 例外処理：
MFはタイマを保持し，MBからの応答がない場合にはタイムアウト処理を実施する．上記のいずれの状態であっても，処理の要求元にエラーを送信し，スロットの初期化を行う（MFでMBの状態管理は行わない）．

マイクロサービスはロボットシステムと同等に扱われるため，アカウント管理モジュールを利用し，事前にID（ロボットID）をRSi-Cloudに登録しておく必要がある．登録されたサービスは，起動時に認証処理が行われ，認証時に発行される会話IDを利用して，処理の受付が可能になる．現状，利用可能なマイクロサービスを動的に検索する機構は実装していないが，今後機能追加を予定している．

5. 性能評価

マイクロサービスは，サービス部品をロボットと同様に扱いプラットフォーム外に実装するアーキテクチャを採用しているため，プラットフォーム上に機能を実装した場合と比較し，性能が劣化する可能性がある．また，処理性能は，送信するデータ量，ネットワーク帯域にも大きく依存する．本稿では，標準的な利用形態においてどの程度の処理性能が期待できるのか，評価実験を行った．

5.1 実験の内容

本稿では，図6の形態を想定し，実運用の環境でマイクロサービスにおけるPush通信の擬似的な性能を測定することにした．MBにおける処理時間の影響を除くために，MBは受信したデータをそのままMFに返送するプロセスとしてPC上に実装し，またネットワーク遅延の影響を最小限にするために，実験はLAN環境で行った．RSNPでは，各種データ（カメラ映像，センサデータ，各種情報など）は添付ファイルとして送受信されるため，サイズの異なるファイルを添付することにより，この状況を模擬することにした．具体的には，擬似Push通信がマイクロサービス機能を実行するのに十分実用的な性能を提供できることを確認するために，MFにおける1秒あたりの処理数，CRがMFに処理を要求してから応答が返ってくるまでの応答時間，CR，MF，MBにおけるCPU使用率を測定した．

図6では，CR，MBともに1台だが，実環境においては，同一または異種の複数のCRが，同一または異種の複数のMBにアクセスする形態が想定される．負荷分散を行うために複数台のMBが存在する場合や，カメラ操作と人の検知のように複数のMBとして登録される場合など，様々な組合せが考えられる．本稿では，同一PC上に複数のプロセスを生成することにより，これらの状況を模擬することとした．

5.2 実験の環境

実験システムは，CR用の2台のPC（R1，R2），MF用の1台のサーバ（S1），MB用の2台のPC（M1，M2）からなり，互いに1 GbpsのLANで接続されている．システム諸元を以下に示す．

- R1, R2, M1, M2：Xeon @ 3.00 GHz (2 Core), 2.0 GBメモリ，CentOS 5.7
- S1：Xeon X3430 @ 2.40 GHz（4 Core），4.0 GBメモリ，CentOS 5.5
- マシン間通信：添付ファイル付きのデータ通信（0 KB〜62 KBのファイルを添付）

5.3 実験の結果

CRをR1とR2に35台ずつ分散させ，0 KBから62 KBのサイズのファイルを添付して測定を行った．CR数は，予備実験の結果から，各PCでの処理性能が頭打ちにならない数を決定した．各測定は，CRから1,000回の送信を

表 1 実験結果 1
Table 1 Experimental result 1.

File size		0 KB	10 KB	21 KB	45 KB	62 KB
Performance (times/s)	R1	101.6	100.0	95.5	89.2	74.9
	R2	105.4	104.2	101.7	97.2	86.8
Round trip time (ms.)	R1	195.3	207.0	234.5	256.4	254.7
	R2	190.5	226.1	239.4	257.3	260.6
CPU load of RC (%)	R1	53.5	57.9	53.5	47.5	44.6
	R2	61.0	58.8	56.7	52.5	52.3
CPU load of MF (%)	S1	24.0	23.6	22.5	21.6	24.3
CPU load of MB (%)	M1	80.4	82.8	87.8	87.8	85.6

表 2 実験結果 2
Table 2 Experimental result 2.

File size		0 KB	10 KB	21 KB	45 KB	62 KB
Performance (times/s)	R1	122.8	128.2	126.4	130.6	129.2
	R2	125.0	128.7	133.6	133.1	133.6
Round trip time (ms.)	R1	181.5	150.6	141.9	135.0	145.2
	R2	178.1	170.6	156.7	139.2	138.3
CPU load of RC (%)	R1	75.0	74.1	76.4	81.4	77.7
	R2	76.4	81.5	83.0	82.8	84.9
CPU load of MF (%)	S1	30.7	31.3	30.8	31.5	31.8
CPU load of MB (%)	M1	90.4	92.1	88.3	89.8	86.1
	M2	86.2	85.6	84.7	83.1	78.6

3 セット実行しその平均値を算出した．MB を 1 台のマシンに 70 プロセス生成した場合の結果を**表 1** に示す．

これらの結果から，本実験においては，ファイルを添付しない場合（ファイルサイズ 0 KB）で 207 件/秒，62 KB のファイルを添付した場合（RSi-Cloud 上で提供される見守りサービスで利用されている 640 × 480 の JPEG ファイルのサイズ）で 151 件/秒程度のトランザクションを処理できることが分かった．本稿におけるマイクロサービスの実装では，MB と CR の増加は容易に行えるので（物理マシンを追加するだけ），それぞれのマシンで CPU 処理が頭打ちになった場合には処理の分散を行えばよい．そのため MF の限界，またはネットワーク帯域の限界が処理性能の限界になる．以上より，今回の実装では 150 件/秒以上の処理が可能であることが分かった．

次に，処理分散の効果を調べるために，MB 側のマシンパワーを増やして，2 台の MB マシンに 35 プロセスずつを分散して実験を行った．結果を**表 2** に示す．処理性能が増加し，RTT も 150 ms 程度に短縮していることが分かる．具体的には，62 KB のファイルを添付した場合でも 260 件/秒程度のトランザクションを処理できており，通常のロボット操作であれば実運用を行う場合でも十分な性能であることが分かった．

5.4 考察

以上の実験結果から，擬似 Push 通信の性能が測定された．この測定値と，MB での処理時間，ネットワーク遅延の影響などを考慮すると，マイクロサービスの性能を推定することができる．ここでは，クラウド上にロボット機能モジュールを実装した場合と，マイクロサービス化した場合（外出しした場合）との比較を試みる．このとき，ロボット機能モジュールの種類によって，通信回数や通信量が異なるため，クラウド上に実装可能なモジュールとして，顔検出モジュールおよび人検知モジュールの 2 種類を考えることにした．

顔検出モジュールは，与えられた画像から顔の位置を検出し，その座標を返すモジュールである．まず，このモジュールをクラウド上に実装した場合の通信回数と通信内容は下記のようになる．

- CR → MF：画像ファイルを送信する．
- MF：顔検出を実施する．
- CR ← MF：得られた結果（座標データ）を送信する．

一方，このモジュールをマイクロサービス化すると，通信回数と通信内容は下記のようになる．このとき，MB は PC などの計算主体として実現可能である．

- CR → MF：画像ファイルを送信する．
- MF → MB：画像ファイルを送信する．
- MB：顔検出を実施する．
- MF ← MB：得られた結果を送信する．
- CR ← MF：得られた結果を送信する．

以上から，クラウド上に実装した場合は，通信回数は 2

回，画像転送は 1 回であり，マイクロサービス化した場合は，通信回数は 4 回，画像転送は 2 回であることが分かる．前節の結果から，後者の RTT は MB での処理時間を除き 150 ms 程度と推定され，前者の RTT はその半分程度と考えられるため 80 ms 程度と推定できる．これより，ロボット機能モジュールをマイクロサービス化した場合の処理時間の増分は 100 ms 以下であり（MB での処理時間は同等と仮定），実運用上は問題のないでレベルであると考えられる．

人検知モジュールは，カメラの設置場所でカメラ画像から人の存在を検知し，検知された人数を返すモジュールである．このモジュールをクラウド上に実装する場合，カメラの設置場所での画像が必要になるため，カメラの物理的な位置はそのままで，取得した画像をクラウド上のモジュールに送付することになる．この場合の通信回数と通信内容は下記のようになる（この場合のカメラデバイスは厳密には MB ではないが，便宜上 MB と記載する）．

- CR → MF：処理を依頼する．
- MF → MB：処理を依頼する．
- MF ← MB：MB が取得した画像ファイルを送信する．
- MF：人検知を実施する．
- CR ← MF：得られた結果（人数）を送信する．

一方，このモジュールをマイクロサービス化すると，通信回数と通信内容は下記のようになる．このとき，MB は RT コンポーネント化されたカメラなどが想定される．

- CR → MF：処理を依頼する．
- MF → MB：処理を依頼する．
- MB：人検知を実施する．
- MF ← MB：得られた結果を送信する．
- CR ← MF：得られた結果を送信する．

以上から，クラウド上に実装した場合は，通信回数は 4 回，画像転送は 1 回であり，マイクロサービス化した場合は，通信回数は 4 回，画像転送は 0 回であることが分かる．前節の結果から，前者，後者ともに RTT は MB での処理時間を除き 150 ms 程度と推定され，通信量を考慮すると，マイクロサービス化した場合の方が若干有利になることも分かる．

以上のように，マイクロサービス化の性能劣化への影響は，多くても 100 ms 程度であり，ロボット機能モジュールの種類によっては影響を受けない場合もあることが分かった．

6. 実装例

提案手法の有効性を検証するために，RT コンポーネントとして実装されたステレオビジョンモジュールを利用し，インターネットサービスとの融合を検証するために AR（Augumented Reality）サービスと連携したプロトタイプシステムを構築した．ここでは，マイクロサービスとして，RT コンポーネントと RSNP とのゲートウェイ（既存の RT コンポーネントをマイクロサービス化するためのコンポーネント）を開発し [32], [33]，RSi-Cloud 上でサービスを実現した．

今回の実装では，準備できるリソースの制約から，CR，MB ともに 1 台だが，プロトタイプシステムとしては，複数台の CR が 1 台の MB に処理を依頼する形態（MF のキューイング機構を利用），複数台の CR が複数台の MB に処理を依頼する形態（同一のロボット ID を持つ複数の MB の実装）などで実装することが可能である．

6.1 RSNPGatewayRTC

まず，RT コンポーネントと RSNP とのゲートウェイとして開発した RSNPGatewayRTC について説明する．RSNPGatewayRTC は，RT コンポーネントと接続しこれをマイクロサービス化することにより，RSi-Cloud 経由での利用を可能にするモジュールである．概要を**図 8** に示す．本稿で利用した RT コンポーネントは，富士通製ステレオビジョンモジュール用の RT コンポーネント「VisionModule」である．前述したように，RT コンポーネントは様々な粒度で部品化されており，この VisionModule も，特徴点クラスタのトラッキングや顔認識など，5 つの RT コンポーネントを実装している．セットアップのイメージを**図 9** に示す．

図 8 に示すように，RSNPGatewayRTC は 2 つの機能を持つ．1 つは人の位置情報を RT コンポーネント側から RSi-Cloud に送信する機能（送信機能）であり，もう 1 つは送信開始，終了などの制御を RSi-Cloud 側から RT コンポーネントに対して実施する機能（制御機能）である．送信機能では，特徴点クラスタのサイズから人の特徴点を判定し，その座標情報（xyz 座標系）を位置情報（緯度・経度．高度）に変換したうえで，RSNP を使って XML フォーマットでサーバ側に送信する．制御機能では，端末から受け取った制御要求を共有データ領域に書き込み，アクティビティが共有データ領域をポーリングすることにより VisionModul の設定情報（送信フラグなど）を更新する．

6.2 歩行者情報 AR サービス

前節で開発した RSNPGatewayRTC を利用し，インターネットサービスとの融合の一例として，歩行者情報 AR サービスを構築した．サービスの概要を**図 10** に示す．本サービスでは，ステレオカメラを接続したステレオビジョンモジュールが，追跡した歩行者の座標を位置情報に変換し，その情報を RSNPGatewayRTC を介してインターネット経由でサーバに蓄積する．蓄積された情報は携帯端末（スマートフォンなど）で参照され，AR 表示される．表示される情報は，一定期間の歩行者の軌跡と追跡した歩行者の移動速度である．また，携帯端末からは，情報蓄積の開

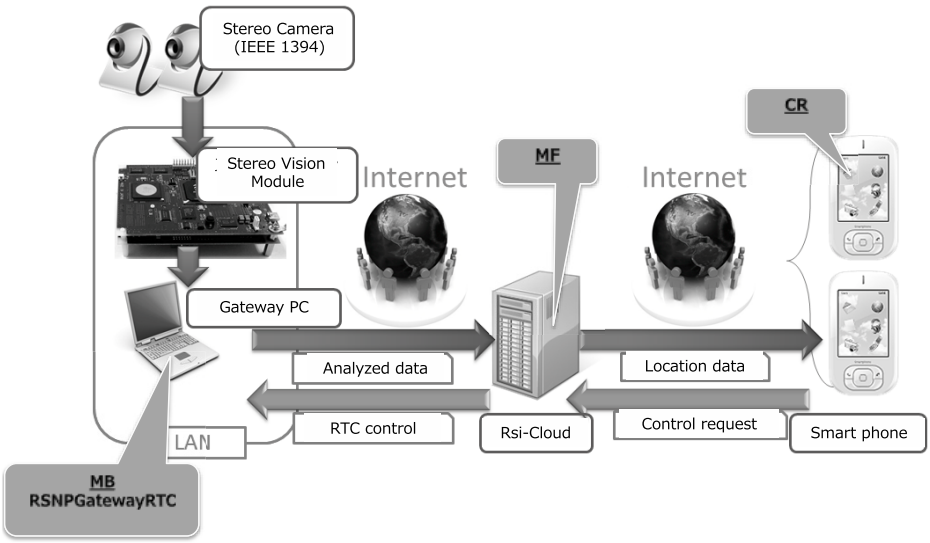

図 8 RSNPGatewayRTC の概要
Fig. 8 An overview of RSNPGatewayRTC.

図 9 セットアップのイメージ
Fig. 9 An image of camera setting.

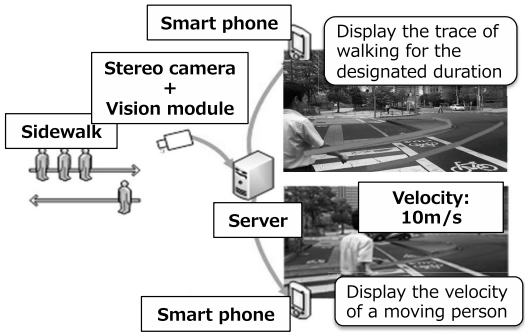

図 10 歩行者情報 AR サービスの概要
Fig. 10 An image of the AR service for pedestrian information.

始・停止，カメラの位置情報の変更などを行うことができる．ここでは，携帯端末も MB として実装することにより双方向の処理を実現している．

6.3 考察

今回利用した VisionModule は，ロボットに組み込むことを前提に開発された RT コンポーネントである．これをマイクロサービスとしてロボットの外に出すことにより，AR サービスとの組合せが実現した．サービス利用者にとっては，高機能なモジュール機能をサービスとして利用できるようになるという利点があり，コンポーネント開発者にとってはコンポーネント活用の選択肢が増えるという利点がある．同様の仕組みを実現するには，ゲートウェイを介さずにコンポーネントを直接インターネットに接続する，コンポーネントの機能をサーバ側に組み込むなどの方法が考えられる．しかし，前者ではリソースの限られた RT コンポーネントにインターネット接続機能（セキュリティ機能を含む）を組み込む必要があり実現は困難である．また後者では，物理的な制約（カメラの設置場所など）があり組み込みが不可能であったり，制約がない場合でもサーバ側の設定が必要になり，ロボット側のソフトウェア開発・設定に比較し難易度が大幅に上がる．RSi-Cloud におけるマイクロサービスの仕組みはこれらの問題に対する 1 つの解であり，今後，他の様々なコンポーネントのインターネットサービスとしての公開が期待される．

7. おわりに

本稿では，ロボット機能モジュールをインターネット経由で利用するための仕組みとして，マイクロサービスを提案した．ここでは，ソフトウェアコンポーネントとして実装されたロボット機能部品を，インターネット経由でプラットフォームに接続することにより，これらの機能をサービスとしてインターネット上に公開することを可能とした．さらに，プラットフォーム上に分散処理機構を構築し，マイクロサービス間での処理の分散，タスク管理，エラー処理などを実現した．本稿では，RSi-Cloud 上にマイクロサービス機構を実装し，性能評価を行った．また，プロトタイプシステムとして，VisionModule をマイクロサービスとして実装し，AR サービスと組み合わせた歩行者情

報 AR サービスを開発した．これらから，提案手法の有効性を確認した．

現在，複数コンポーネントを様々な粒度で組み合わせて利用する仕組みや，利用可能なコンポーネントを検索する仕組みなどは実現していない．今後これらの機能の検討を行っていく予定である．また，VisionModule 以外の様々な部品を接続し，検証を行っていく予定である．

参考文献

[1] Guinard, D., Muller, M. and Pasquier, J.: Giving RFID a REST: Building a Web-Enabled EPCIS, *Internet of Things 2010 International Conference* (*IoT 2010*) (2010).

[2] Cyber-Physical Systems (CPS), available from ⟨http://www.nsf.gov/pubs/2008/nsf08611/nsf08611.htm⟩.

[3] Ando, N., Suehiro, T., Kitagaki, K., Kotoku, T. and Yoon, W.: RT-Middleware: Distributed Component Middleware for RT (Robot Technology), *IEEE/RSJ International Conference on Intelligent Robots and Systems 2005* (*IROS 2005*), pp.3933–3938 (2005).

[4] Quigley, M., Conley, K. and Gerkey, B.: ROS: An open-source Robot Operating System, *IEEE International Conference on Robotics and Automation* (*ICRA*) *Workshop on Open Source Robotics 2009* (2009).

[5] RoboEarth, available from ⟨http://www.roboearth.org/⟩.

[6] Kuffner, J.: What's Next: Cloud Enabled Humanoids?, *10th IEEE-RAS International Conference on Humanoid Robots* (*Humanoids 2010*) *Workshop* (2010).

[7] Arumugam, R., Enti, R., Bingbing, L., Xiaojun, W., Baskaran, K., king, F., Kumar, A., Meng, K. and KitVikas, G.: DAvinCi: A Cloud Computing Framework for Service Robots, *IEEE International Conference on Robotics 2010* (2010).

[8] Kato, Y., Izui, T., Tsuchiya, Y., Narita, M., Ueki, M., Murakawa, Y. and Okabayashi, K.: RSi-Cloud for Integrating Robot Services with Internet Services, *The 37th Annual Conference of the IEEE Industrial Electronics Society* (*IECON 2011*), pp.2164–2169 (2011).

[9] Kato, Y., Izui, T., Murakawa, Y., Okabayashi, K., Ueki, M., Tsuchiya, Y. and Narita, M.: Research and Development Environment for Robot Services and its Implementation, *2011 IEEE/SICE International Symposium on System Integration* (*SII2011*), pp.306–311 (2011).

[10] RSi Robot Service initiative, available from ⟨http://robotservices.org/⟩.

[11] 成田雅彦，村川賀彦，植木美和，中本啓之，平野線治，蔵田英之，加藤由花：普及期のロボットサービス基盤を目指す RSNP（Robot Service Network Protocol）2.0 の開発，日本ロボット学会誌，Vol.27, No.8, pp.857–867 (2009).

[12] 成田雅彦，村川賀彦，植木美和，岡林桂樹，秋口忠三，日浦亮太，蔵田英之，加藤由花：インターネットを活用したロボットサービスの実現と開発を支援する RSi（Robot Service Initiative）の取り組み，日本ロボット学会誌，Vol.28, No.7, pp.829–840 (2010).

[13] 岡野克弥，安川祐介：次世代ロボット知能化技術開発プロジェクトの概要，第 27 回日本ロボット学会学術講演会，pp.1D1–01 (2009).

[14] 布留川英一：Microsoft Robotic Studio プログラミング，毎日コミュニケーションズ (2007).

[15] OpenRoads, available from ⟨http://www.speecys.com/openroads.html⟩.

[16] Ha, Y. et al.: Design and Imlementation of Ubiquitous Robitc Service Framework, *ETRI Journal*, Vol.27, No.6, pp.666–676 (2005).

[17] iRobot ConnectR, available from ⟨http://store.irobot.com/shop/index.jsp?categoryId=3311370⟩.

[18] 土井美和子，山本大介，荻田紀博：新たなサービスを興すネットワークロボット技術，東芝レビュー，Vol.64, No.1, pp.32–35 (2009).

[19] 大橋正良，徳田英幸，尾家祐二，森川博之，桐葉佳明，加藤正文，長谷川亨：ユビキタスネットワーク制御・管理技術（Ubila），電子情報通信学会誌，Vol.91, No.7, pp.569–582 (2008).

[20] 吉田 幹，奥田 剛，寺西裕一，春本 要，下條真司：マルチオーバレイと分散エージェントの機構を統合した P2P プラットフォーム PIAX，情報処理学会論文誌，Vol.49, No.1, pp.402–413 (2008).

[21] Yuriyama, M. and Kushida, T.: Sensor-Cloud Infrastructure, *13th International Conference on Network-Based Information Systems* (*NBiS2010*), pp.1–8 (2010).

[22] Kanzaki, A., Hara, T., Ishi, Y., Wakamiya, N. and Shimojo, S.: X-Sensor: A Sensor NEtwork Testbed Integrating Multi-Networks, *DMIEW2009*, pp.1082–1087 (2009).

[23] RT コンポーネント，入手先 ⟨http://openrtm.org/openrtm/ja/content/rt コンポーネント⟩.

[24] Cousins, S., Gerkey, B. and Conley, K.: Sharing Software with ROS, *IEEE Robotics & Automation Magazine*, Vol.17, No.2, pp.12–14 (2010).

[25] ROS packages, available from ⟨http://www.ros.org/browse/list.php⟩.

[26] RT_Kinect_UserTracking, available from ⟨http://www.openrtm.org/openrtm/ja/project/Kinect_UserTracking⟩.

[27] アームユニット RTC，入手先 ⟨http://www.openrtm.org/openrtm/ja/project/NEDO_Intelligent_PRJ_ID105⟩.

[28] モンテカルロ位置推定コンポーネント，入手先 ⟨http://www.openrtm.org/openrtm/ja/project/NEDO_Intelligent_PRJ_ID155⟩.

[29] 顔検出コンポーネント，入手先 ⟨http://www.openrtm.org/openrtm/ja/project/facedetect⟩.

[30] Narita, M., Kato, Y. and Akiguchi, C.: Enhanced RSNP for applying to the Network Service Platform – Implementation of a Face Detection Function, *4th International Conference on Human System Interaction* (*HSI2011*), pp.311–317 (2011).

[31] 成田雅彦：ロボットサービスイニシアティブのコンセプトとロボットネットワークサービスプロトコルの設計指針，信学技報，CNR2011-19，pp.49–54 (2011).

[32] 岡部 泉，奥平直仁，内藤裕幸，名倉真史，成田雅彦，加藤由花：RT コンポーネントと RSNP を利用した画像処理システムの構築，計測自動制御学会 SI 部門講演会（SI2011），1K4-1 (2011).

[33] Narita, M., Murakawa, Y., Akiguchi, C., Kato, Y. and Yamaguchi, T.: Push Communication for Network Robot Services and RSi/RTM Interoperability, *Proc. FUZZ-IEEE 2009*, pp.1480–1485 (2009).

加藤 由花 （正会員）

1989年東京大学理学部卒業．同年日本電信電話（株）入社．2002年電気通信大学大学院情報システム学研究科博士後期課程修了．博士（工学）．同年電気通信大学助手．2006年産業技術大学院大学准教授，現在，同教授．情報ネットワークサービス基盤等の研究に従事．電子情報通信学会，日本ロボット学会，計測自動制御学会，IEEE，ACM各会員．

岡部 泉

2011年産業技術大学院大学産業技術研究科修了（情報システム学修士）．同年より産業技術大学院大学客員研究員．

村川 賀彦 （正会員）

1981年東京工業大学理学部情報科学科卒業．同年富士通（株）入社．1997年北陸先端科学技術大学院大学博士後期課程修了．博士（情報科学）．2005年より（株）富士通研究所勤務．サービスロボットとビジネス化，ヒューマンロボットインタラクションの研究に従事．日本ロボット学会，人工知能学会，ソフトウェア科学会，認知科学会各会員．

岡林 桂樹

1988年佐賀大学大学院理工学研究科電子工学専攻修士課程修了．同年（株）富士通研究所入社．以来，ロボットビジョン，サービスロボット等の研究開発に従事．日本ロボット学会会員．

植木 美和

1993年東京大学工学部計数工学科卒業．同年（株）富士通研究所入社．以来，家庭用ロボット，サービスロボット等のソフトウェアの研究開発に従事．日本ロボット学会会員．

土屋 陽介

2006年拓殖大学大学院工学研究科博士後期課程修了．博士（工学）．同年産業技術大学院大学研究員．2007年同助教．現在に至る．音響信号処理，音響計測，ロボットサービス等の研究に従事．日本ロボット学会，計測自動制御学会各会員．

成田 雅彦

1980年早稲田大学大学院数学科修士課程修了．同年富士通（株）入社．以来，OS/ミドルウェアの企画・研究開発と関連の国際標準活動に従事．2008年より産業技術大学院大学教授．2010年首都大学東京大学院システムデザイン研究科博士後期課程修了．博士（工学）．電子情報通信学会，日本ロボット学会，精密機械工学会各会員．