情報処理学会論文誌　コンシューマ・デバイス & システム　Vol.11　No.1　1–13 (Mar. 2021)

招待論文

ネットワーク接続された組込みシステムの拡張

寺岡 秀敏[1,a)]　尾崎 友哉[2]

受付日 2020年6月30日, 採録日 2020年11月4日

概要：通信技術の発展にともない，我々を取り囲む多くのモノは，ネットワークを介して接続されてさまざまな機能を実現するようになった．このような環境において，これら偏在するモノを再構成し，技術の進展や環境の変化に合わせて進化させることが重要になっている．本稿では，このようなモノを実現する形態の 1 つである組込みシステムに着目し，その再構成に関する技術を体系的に整理した．本稿ではネットワーク接続された組込みシステムを再構成することをシステム拡張と定義し，システム拡張に必要な技術を，システムの構成方法に関する技術とシステム拡張を実行するプロセスにおける技術に分類し，それぞれの課題と既存手法を示す．

キーワード：組込みシステム，再構成，システム拡張，ソフトウェア更新

A Survey on Extensibility of the Networked Embedded Systems

Hidetoshi Teraoka[1,a)]　Tomochika Ozaki[2]

Received: June 30, 2020, Accepted: November 4, 2020

Abstract: With the evolution of communication technology, many things that surround us have come to archive various functions by being connected via networks. In such an environment, it is important to reconstruct these unevenly distributed objects and evolve them in accordance with technological progress and changes in the environment. In this paper, we focused on embedded systems, which are one of the forms that realize such things, and systematically organized the technologies related to their reconstruction. In this paper, reconfiguring an embedded system connected to a network is defined as system expansion, and the technologies required for system expansion are classified into technologies related to system configuration methods and technologies in the process of executing system expansion. And the existing method is shown.

Keywords: networked embedded systems, reconfiguration, system extensibility, software update

1. はじめに

近年，あらゆるモノがインターネットに接続される Internet of Things（IoT）が急速に拡大している．パソコンや携帯電話などのインターネット接続端末に加えて，家電や自動車，ビルや工場などさまざまなものがインターネットへつながることで，IoT デバイスは 2020 年には 300 億に達すると予測されている [1]．

IoT システムは Cyber Physical System（CPS）としての側面も有し，その付加価値をソフトウェアとして具現する物理システムである [2]．IoT システムまたは CPS は，社会課題の解決や新たな価値の創造への貢献が期待されている．

文献 [1] において IoT デバイスは，民間調査機関 IHS Technology の定義が引用され，固有の IP アドレスを持ちインターネットに接続が可能な機器およびセンサネットワークの末端などとして使われる端末などとされている．また，ITU-T における定義では IoT デバイスは，必須の通信機能とオプションのセンシング・アクチュエーション・データ取得・データ蓄積・データ処理機能を有するデバイスとされている [3]．このような機器のうち，パソコンな

[1] 株式会社日立製作所研究開発グループ
Hitachi, Ltd. Research & Development Group, Yokohama, Kanagawa 244–0817, Japan
[2] 長崎大学情報データ科学部
School of Information and Data Sciences, Nagasaki University, Nagasaki 852–8521, Japan
[a)] hidetoshi.teraoka.rf@hitachi.com

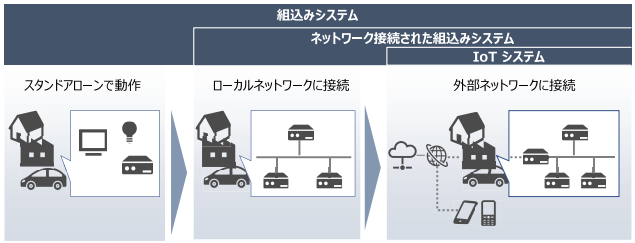

図 1　組込みシステムのネットワーク化
Fig. 1　Evolution of networked embedded system.

どのコンピュータ以外は組込みシステムとして構成される．組込みシステムは，特定用途の機能を実現するマイクロプロセッサベースのシステム [4] である．従来の家電機器など単独で機能を実現してきた組込みシステムは，ネットワークに接続されて他の組込みシステムと連携し，ホームネットワークシステムや自動車，ビルシステム，工場システムを構成するようになった．さらに，通信技術の進展にともない，これらがインターネットに接続することで，さらなる機能の向上が進展している．たとえば，スマートスピーカなどはインターネット越しにサーバと連携して機能を実現している．また，ホームネットワークシステムや自動車，ビルシステム，工場システムはシステムごとインターネットに接続されることで，スマートハウスやホーム/ビルエネルギーマネジメントシステム，コネクテッドカー，スマートファクトリーなどの IoT システムを構成する．

図 1 に組込みシステムのネットワーク化の概要を示す．すなわち，単独で動作していた組込みシステムがネットワークに接続され，ネットワーク接続された組込みシステムとして機器どうしが連携して必要な機能を実現する．さらに，これらのネットワーク接続された組込みシステムが，インターネットに接続し，汎用コンピュータや，携帯電話などの単独の組込みシステムとして構成される機器とともに，IoT システムを構成する．すなわち，IoT システムは，ネットワーク接続された組込みシステムの一形態である．本稿では，ネットワーク接続されたシステムを構成する要素としての組込みシステムをノードと称する．

ネットワーク接続された組込みシステムがノード間で連携して動作する際の基本構成として，集中型と分散型，サーバ連携の 3 通りの構成がある．**図 2** に基本構成の概要を示す．

(a) 集中型システム：コントローラを中心としてノードを制御することで機能を実現する．例として，ホームゲートウェイが家電と連携するホームネットワークシステムがある．

(b) 分散型システム：単一のコントローラを必要とせず，ノードどうしが連携して機能を実現する．例として，Electronic Control Unit（ECU）どうしが連携する自動車システムがある．

(c) サーバ連携システム：ノードとサーバ側のアプリケーションが連携して機能を実現する．例として，スマートスピーカと連携するホームネットワークシステムなどがある．

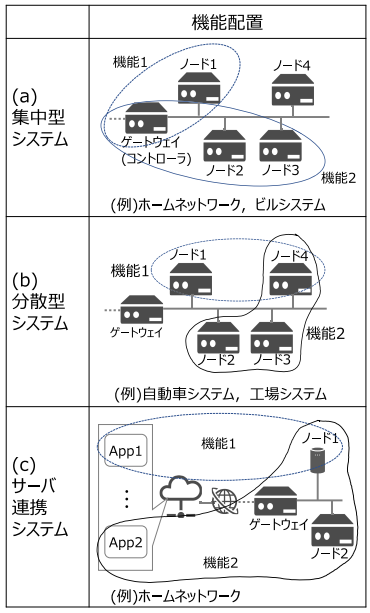

図 2　ネットワーク接続された組込みシステムの基本構成
Fig. 2　Basic configurations of a networked embedded system.

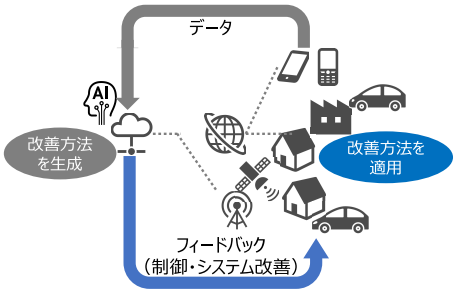

図 3　IoT/CPS におけるサイバー空間と実システムの連携
Fig. 3　Collaboration between cyber world and real world.

IoT 化したシステムの特徴の 1 つに，実システムから収集したデータに基づきサイバー空間で導出した改善策を実システムにフィードバックする枠組みがある（**図 3**）．

このような実システムの改善は，これらのフィードバックを反映した新たな製品を設計・開発する [5]，あるいは運用されているシステムそのものに適用する [6] ことで実現される．本稿では，改善手段や拡張機能を運用中のシステムの再構成により実世界へ反映することを「システム拡張」と称する．このため，本稿におけるシステムの拡張には，機能の追加だけではなく不具合の修正やセキュリティパッチなどを含む．

運用中のシステムの再構成には，システムに新たなノードを追加する方法や，ノードを構成するハードウェアまたはソフトウェアを再構成する方法がある．ノードの追加の事例としては，ホームネットワークシステムにおいて，新規購入した機器を追加した際に，当該機器もエネルギー管

理の対象とするような拡張が行われる場合がある．また，ノードの再構成の事例としては，携帯電話やスマートフォンでは，インターネットを経由したソフトウェアのアップデートにより，機能の追加などを行う方法が広く普及している [7], [8]．他にも，自動車システムにおいて，ソフトウェアを更新することで，航続距離延長や，自動運転機能の追加などのシステム拡張が行われる場合がある．また，近年普及しているスマートスピーカでは，端末側だけでなくクラウド側に処理ロジックを追加することで機能が追加される．

このように現在，あらゆる機器がネットワークに接続されることによりシステム化が進んでいる．また，ハードウェアまたはソフトウェアのアップデートや新たな機能を有するノードの追加により，システムの機能が拡張されるようになってきている．特に，2013 年にガートナーより発表された Software Defined Anything（SDx）[9] という言葉に示されるように，近年の大きなトレンドとしてさまざまな分野でソフトウェア化が進んでいる．今後は，さらにこの傾向が進展してプログラマブルな世界が広まり，特にエッジ側の更新が重要になると予測されている [10]．エッジ側は組込みシステムとして構成されることが多く，この観点からも組込みシステムの拡張は重要である．

システムを拡張することは，価値を向上し続けるという観点と合わせて，サイバーセキュリティリスクへの対策の観点からも重要になる．たとえば，スマートメータ調達におけるソフトウェア更新への要求 [11] や自動車におけるサイバーセキュリティ [12] およびソフトウェア更新 [13] に関する法制度の整備など，システムを危殆化させないための手段としても活用が進んでいることが分かる．

ソフトウェアの更新によるシステム拡張が進展している一方で，前述のとおりシステムの拡張手段としてはソフトウェアの更新を含めて以下の 4 通りの方法が考えられる．

(A) 連携するノードの追加：ノードを追加することで，システムの機能の追加などを行う
(B) ノードのハードウェアの再構成：ノードのハードウェアを再構成することで，機能の拡張などを行う．
(C) ノードのソフトウェアの再構成：ノードのソフトウェアをアップデートすることで，機能の拡張などを行う．
(D) サーバ連携（主にサーバ側の機能拡張）：端末側がユーザインタフェースのみを提供し，サーバ側にロジックを変更・追加することにより，機能の変更・追加などを行う

CD/CI や DevOps の普及により運用と改善適用のサイクル化が進んでいるサーバ側のアプリケーション（(D)）に加えて，ネットワークでつながるエッジ側のノードの拡張（(A)，(B)，(C)）が重要になっていく．

このようなシステムを拡張する技術のうち，ハードウェアの再構成 (B) に関する技術は，再構成可能コンピューティング（Reconfigurable Computing）と称されてさまざまな取り組みがなされている．再構成可能コンピューティング技術は，Field Programable Gate Array（FPGA）に代表される細粒度アーキテクチャと，Dynamic Reconfigurable Processor（DRP）のような粗粒度アーキテクチャに大別できる．それぞれについてアーキテクチャやその分類，性能，ツールなどの観点からまとめられている [14], [15], [16], [17], [18]．

ネットワーク接続された組込みシステムのソフトウェアの再構成 (C) に関する技術は，2000 年代に Wireless Sensor Network（WSN）向けや携帯電話向けにさまざまな取り組みがなされた．また，放送のデジタル化にともない，デジタルテレビやレコーダなどの放送受信端末のファームウェア更新も実用化された．これらの技術は IoT デバイス向けにも適用可能であり，文献 [19] にて Over the Air（OTA）によるファームウェア更新のための差分生成や更新データ配布プロトコルといった運用面，セキュリティ，OTA をサポートするプラットフォームについてまとめられている．他にも，ソフトウェアの再構成に関しては，Self-Adaptive System [20], [21]，動的 SPL [22] などの取り組みがなされている．

さらに文献 [23] では，ハードウェア・ソフトウェアの両方を含む動的再構成可能システムについて，その分類，プラットフォームと方法論，適用範囲についてまとめられている．

本稿では，これらの切り口にノードの追加によるシステム拡張 (A) を含め，ネットワーク接続された組込みのシステム拡張に関する技術を概観する．具体的にはシステム拡張を実現するための「構成」と「プロセス」に着目し，それぞれを要素として体系化して論じる．これにより，システム拡張を可能または容易にするノードの構成における課題を解決する技術と，システム拡張を実行するプロセスにおける課題を解決する技術を示す．

本稿の構成は以下のとおりである．2 章では，システム拡張のための構成とプロセスに関する技術の体系を述べ，3 章では，システム拡張のための構成に関する技術について述べる．4 章では，システム拡張を実行するプロセスに関する技術について述べる．最後に，5 章でまとめと今後の展望について述べる．

2. システム拡張を支える技術

システム拡張を実現するためには，どのようにシステムを構成し，どのような手順すなわちプロセスで再構成を行うかが課題になる．そのため，本稿では，システム拡張技術をシステム拡張のための構成技術とシステム拡張プロセスに関する技術に大別する（図 4）．

(1) システム拡張のための構成技術

システムを拡張するためには，そもそも運用時に再構成が可能な構成であることが前提となる．このような前提と

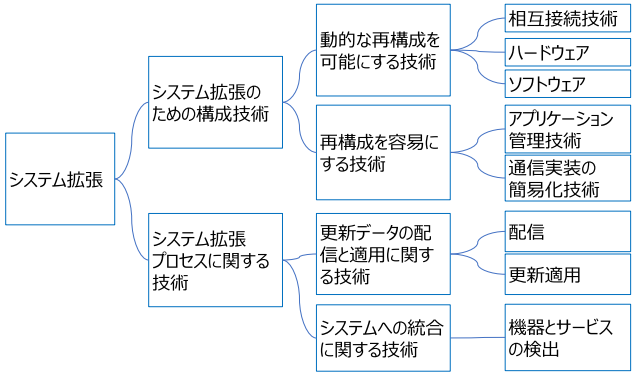

図 4　システム拡張を支える技術

Fig. 4　Technologies for extension of embedded systems.

して，ノードを追加する際には追加するノードとシステムの相互接続性が，ハードウェアを再構成する際には，再構成可能なハードウェアが必要である．また，ソフトウェアを再構成する際には，目的とする粒度でコンポーネントを更新可能とするアーキテクチャでノードを構成する必要がある．さらに，システム拡張を行う際には，それを容易にする技術も重要である．再構成する機能，たとえばアプリケーションの管理や，再構成する機能の開発を容易化することで，システム拡張そのものも容易化することができる．

(2)　システム拡張プロセスに関する技術

システムを拡張するためには，ノードに再構成のためのデータを作成・配信し，そのデータをノードに適用する必要がある．このとき，再構成の仕組みを攻撃・悪用されないようにセキュリティリスクから保護する仕組みも重要になる．さらに，再構成により新規に追加されたノードや機能をシステムに統合するためには，追加・再構成したノードや機能を検出して自律的にシステムに組み込む必要がある．

本稿では，これらの技術を概観することで，ネットワーク接続された組込みシステムの拡張において，どのようにシステムを構成し，どのような手順すなわちプロセスで再構成を行うかを明確化し，今後ますます進展が想定されるプログラマブルな世界の加速に貢献する．

3.　システム拡張のための構成技術

1 章で述べたとおり，本稿ではノードの追加およびノードの再構成によるシステム拡張を取り扱う．本章では，システム拡張の前提となる構成技術として，ノードの動的な再構成を可能にする構成技術として，相互接続技術およびハードウェアとソフトウェアそれぞれの再構成技術を示す．また，システム拡張を容易にする技術として，アプリケーション管理技術およびアプリケーション開発の容易化技術を示す．

3.1　動的な再構成を可能にする技術

3.1.1　ノードの相互接続技術

ネットワーク接続された機器どうしを連携させるためには，通信の標準化による相互接続性の担保が重要となる．機器追加などによりシステム拡張を行う場合においても，互換性のない通信方式の機器・機能を追加・連携させることは一般的に困難なためである．

相互接続を実現するためには，通信媒体や変復調方式・データリンク処理などの規定と，そのうえで，各機器のアドレッシングやルーティング，End to end の通信処理方式や，各種情報の表現形式やコマンドを規定する [24]．OSI 参照モデル [25] やインターネットプロトコルスイート [26] では，それぞれ 7 層または 4 層でこれらの機能が階層化して整理されている．本稿では，通信媒体や変復調方式・データリンク処理などを規定するレイヤを下位層，各機器のアドレッシングやルーティング，End to end の通信処理を行うとともに，各種情報の表現形式やコマンドを規定するレイヤを上位層とし，さらにその上位で各種サービスを実現するレイヤをアプリケーション層と呼称する．ネットワーク接続された組込みシステムを構成する通信技術は実現する機能による通信への要求に応じて複数の組合せ構成される場合が多い．以下，代表的なシステムとしてホームネットワークシステム，ビル・工場システム，自動車システムを取り上げ，システムごとに相互接続技術を説明する．

(1)　ホームネットワークシステム

ホームネットワークはデジタル AV 機器を接続する AV 系ネットワークと空調機器などを接続する設備系ネットワークとその組合せで構成される．AV 系ネットワークでは，容量の大きい映像音声データをリアルタイムで送受信するため，下位層には広帯域な伝送が可能である Ethernet や WiFi などが利用され，上位層には TCP/UDP/IP や HTTP などが利用される．また，アプリケーション層では，機器やサービスの検出を行うための UPnP [27] やコンテンツ情報の共有と伝送要求などを行う DLNA [28] などが利用されている．設備系ネットワークでは，空調・照明機器や各種センサなど多様な機器に接続するために，下位層には配線が容易な電力線通信や省電力で無線接続が可能な 6LowPAN や Bluetooth など利用され，上位層には TCP/UDP/IP などに加え，狭帯域でもデータの伝送を可能にする ECHONET [29] や Zigbee [30] が利用される．また，アプリケーション層では，ECHONET Lite [31] や Zigbee が各機器への制御指示や各種センシングデータを規定している．本分野では，ネットワークに関する知識などのない一般ユーザが機器を管理・接続する．そのため，接続や問題の切り分けが簡易に行える必要がある．そのような課題を解決する技術として，ホームネットワークマップを特定する HTIP が提案されている [32]．

(2)　工場・ビルシステム

工場システムやビルシステムのネットワークは，センサおよびアクチュエータ/空調機などを接続するフィールドネットワークと，Programmable Logic Controller（PLC）や Distributed Control System（DCS）などのコントローラを接続して協調制御を実現するためのコントローラネットワーク，上位のサーバなどに制御状況や機器の状態を提供する管理用の情報ネットワークの組合せで構成される．フィールドネットワークではリアルタイムにセンシングデータや制御指示伝送するために下位層には RS-485 や Ethernet，CAN などが利用され，上位層からアプリケーション層には CC-Link，PROFIBUS [33]，DeviceNet [34]，Modbus [35] など多数が標準化されて利用されている．コントローラネットワークにおいても，制御装置間でリアルタイムに情報を共有するためにフィールドネットワークと同様に下位層には Ethernet などが利用され，上位層からアプリケーション層には EtherNET/IP [36]，EtherCAT，PROFINET，Modbus/TCP などが利用されている．これらの通信技術では，リアルタイム性を確保するために一部の処理を簡素化してスタックが構成される．また，近年では，リアルタイム性を確保するための同期機能を有する Ethernet TSN [37] の導入が検討されている．さらに，情報モデルを定義して情報ネットワークやクラウドも考慮した情報透過性を持つ OPC UA [38] の適用も進みつつある．

前述のとおり，近年では上記のネットワーク接続された組込みシステムがインターネット経由で外部のシステムに接続される．このようなインターネット経由でのデータ配送を HTTP/XML を用いて IEEE1888 を軽量に実装する方式が提案されている [47]．

(3)　自動車システム

自動車システムは，制御系 ECU を接続する制御系ネットワークと，情報系 ECU を接続する情報系ネットワークで構成される．制御系ネットワークではリアルタイムに制御情報を共有するため，下位層では CAN [39] を中心に LIN [40] や FlexRay [41] が必要な伝送データ量に応じて利用され，上位層では CAN-TP [42] などが利用される．また，アプリケーション層は診断通信に標準化された UDS [43] などが利用される一方，制御メッセージは各自動車メーカ独自の規定であることが多い．近年は制御データの大容量化にともない，自動車分野でも Ethernet [44] および TCP/UDP/IP の導入が進んでいる．Ethernet の導入にあたっては配線を軽量化するため，1 対 2 線の方式が開発され [45]，100BASE-T1 [46] として標準化された．

3.1.2　ハードウェアの再構成技術

一般的に，組込みシステムが出荷された後で，たとえば，基板に部品を追加するなど出荷されたハードウェアを変更することは困難である．

一方，デジタル回路については，内部構成回路の書き換えによりハードウェアを再構成する技術が研究，実用化されているこのような再構成可能なハードウェアは Reconfigurable Device や Programmable Logic Device（PLD）と呼ばれ，さまざまな技術が提案されている [14], [15], [16], [17], [18]．

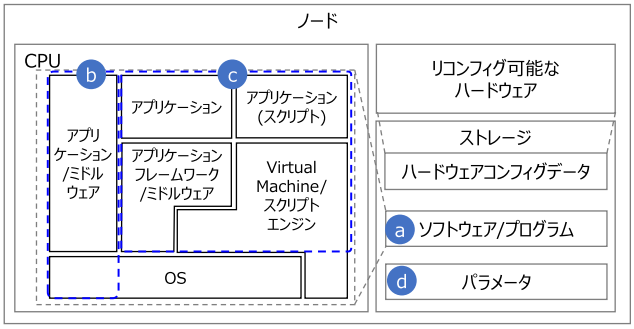

図 5　再構成に関するノードの構成

Fig. 5　A configuration related to reconfiguration of a node.

(1)　FPGA

このような PLD の代表例としては SRAM 型の Field Programable Gate Array（FPGA）がある．FPGA は，Look up table（LUT）を基本論理ブロックとして演算回路などさまざまな回路を構成することができる．近年，Xilinx 社 [48] などから CPU と FPGA を組み合わせた Reconfigurable System が発売され，利用されている．文献 [14] では FPGA のハードウェア・ソフトウェア・動的な再構成についてのサーベイがまとめられている．また，文献 [18], [49] では，FPGA の動的な部分更新に関してのアーキテクチャ，ツール，オーバヘッド削減技術，ランタイム管理，アプリケーションがまとめられている．

(2)　DRP

FPGA は細粒度で回路の構成が可能である一方で，コンフィグレーションデータが大きくなり，再構成時間がかかるという課題があった．これに対し，Dynamic Reconfigurable Processor（DRP）[50] は演算器とレジスタなどからなる粗粒度な Processing Element（PE）を要素として構成することで FPGA の再構成の課題を解決するとともに，面積効率や性能向上を果たしている．DRP はデジタルカメラに搭載される [51] など，近年では，Deep neural network をアクセラレートする技術も発表されている [52]．

3.1.3　ソフトウェアの再構成技術

ソフトウェアの再構成は一般的にはソフトウェア更新と呼ばれ，さまざまな分野で広く利用されている．組込みシステムのソフトウェアは，Flash ROM などのストレージに格納され，そのまま，または RAM に展開されて CPU で実行される．ソフトウェアは一般的に，OS・ミドルウェア・アプリケーションから構成されることが多いが，組込みシステムでは OS を必要としない構成もある．**図 5** に示すようにソフトウェア更新はストレージに格納されたバイナリイメージ全体を更新するケース (a)，OS に更新の仕組

みを設け，OS 上で動作する任意のミドルウェアおよびアプリケーションモジュールを更新するケース (b)，ミドルウェアなどにソフトウェア更新の仕組みを設け，当該ミドルウェア上で動作する任意のアプリケーションを更新するケース (c)，ソフトウェアの動作に係るパラメータのみを更新するケース (d) と，さまざまな範囲で再構成が可能である．一方で，更新範囲によって柔軟性や必要なコストが異なる [53]．

バイナリイメージ全体を更新する場合，動的な拡張を可能とする構造への考慮は不要であるが，本章の冒頭で述べたような設計手法やアーキテクチャ選定が拡張のしやすさに影響する．一方でバイナリモジュールやスクリプトなど，ソフトウェアの一部を更新する場合は，このような部分的な更新や，部分的な更新を動的に実行することを可能とするアーキテクチャでノードを構成することが課題となる．このような部分的なソフトウェア更新を可能にするソフトウェアの構造に関しては，OS，ミドルウェア/アプリケーションフレームワーク，バーチャルマシン/スクリプトエンジンなどさまざまなレイヤでの取り組みがなされてきた．以下では，それぞれのレイヤにおけるソフトウェア更新を容易にする構成する技術について説明する．

(1)　OS

OS レイヤでは，リソースの少ないセンサネットワークシステムのセンサノード向けにプログラムモジュールの位置独立化やメモリ再配置により，ソフトウェアモジュールの動的な追加を可能にする SOS [54] や Contiki [55] などが提案されている．また，iTronOS 向けにソフトウェアモジュールの動的な追加を可能にする仕組みとして，サーバでリンクしてアドレスを決定しターゲットで当該モジュールを決定したアドレスに配置して実行する RLL（Remote Link Loader）や DLM（Dynamic Loading Manager）が開発されている [56]．近年では Android 向けなどにカーネルの live patch 技術が提案されている [57]．

(2)　ミドルウェア

Java [58] に代表されるようなバーチャルマシン（VM）技術も，ソフトウェアの動的な再構成を可能にする．組込みシステム向けには，Android で利用される DalvikVM や Android Runtime（ART）がある [59]．また，低リソースなセンサネットワークノード向けに，Squawk [60] や Mate [61]，VM* [62] などが提案されている．さらに，ノード上でスクリプトを解釈して実行するようなスクリプト技術も VM 技術の一部でありシステム拡張を容易にする．このようなスクリプト技術のうち，組込みシステムでも利用が可能な軽量なものとしては Lua [63]，mruby [64]，konoha [65] などがある．また，よりリソースの制限されたセンサノード向けのスクリプトとして SCript [66] が提案されている．バーチャルマシン技術を用いることにより，アプリケーションやスクリプトなどのモジュール単位でソフトウェアを更新することが可能になる．センサネットワーク向けの動的再構成技術については文献 [67] にまとめられている．本分野では近年，アプリケーションだけでなくその実行環境も合わせて管理するコンテナ技術の組込みシステムへの適用が検討されている．初期検討として，Docker や Linux container（LXC）などのコンテナ技術を組込みシステムに適用する際の課題であるリソースと性能・リアルタイム性への影響が評価されている [68], [69]．また，自動車分野での Plug and play を実現するために，OS を含むシステム全体を管理するシステム仮想化技術の組込みシステムへの適用も検討されている [70]．

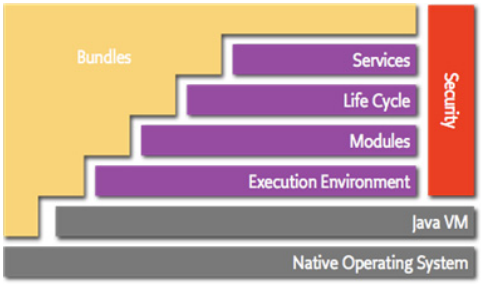

図 6　OSGi 階層モデル [71]
Fig. 6　The layer model of OSGi.

3.2　再構成を容易にする技術

動的な再構成が可能な環境においては，アプリケーションの管理やアプリケーション間の連携が課題となる．このために，前述の OS や VM をベースとして，アプリケーションのライフサイクル管理などを行うフレームワークが提案されており，Android 向けの Java API フレームワーク [59] や，サービスゲートウェイ向けに開発された前述の OSGi フレームワーク [71] が該当する．また，通信プロトコルをサポートするミドルウェアが提案されており，ECHONET で定義された通信ミドルウェアなどがこれに該当する．

3.2.1　アプリケーション管理技術

(1)　OSGi フレームワーク

OSGi フレームワークは，開発当初，ホームネットワークのサービスゲートウェイ向けの実行基盤として規定されたが，車載機 [72], [73], [74] をはじめとするさまざまな分野への適用が行われている．**図 6** に OSGi の階層モデルを示す．

OSGi フレームワークは，1 つの JavaVM 上でバンドルと呼ばれるソフトウェアモジュールの管理を行う実行基盤であり，1 つの JavaVM 上でバンドル＝サービスの動的な追加や実行が可能である．すなわち，複数のバンドルが稼働しているときに，特定のバンドルだけを停止・再起動したり，更新したりすることが可能であり，環境や提供するサービスが異なっていても，必要なバンドルの組合せで適切なシステムを構築することが可能となる．

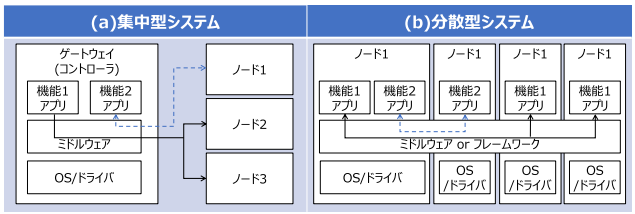

図 7　ネットワーク接続された組込みシステムの構成とアーキテクチャ例

Fig. 7　Examples of configuration and architecture of networked embedded system.

3.2.2　アプリケーション開発の容易化技術

さらに，ネットワーク接続されたシステム向けにもアプリケーションの開発を効率化して拡張性を高めることは課題となる．たとえば，ノード間の通信プロトコルをアプリケーションごとに実装する必要がある場合などは開発が非効率になり拡張も容易ではなくなる．このため，ネットワーク接続されたシステム向けにも拡張性を考慮した構造が研究され，**図 7** に示す集中型・分散型のシステムにそれぞれ適用されている．

(1)　集中型システム向け技術

集中型システムの場合，各種サービスの実現はゲートウェイに搭載されたアプリケーションが担い，通信機能は共通のミドルウェアとする構成をとることが多い（図 7 (a)）．このような構成においては，ゲートウェイのアプリケーションを更新する，またはゲートウェイに新たなアプリケーションを追加することで，システムが拡張される．集中型システムにおいて拡張性を考慮した構成方法としては，前述の OSGi が広く普及している．OSGi では，モジュールの管理に必要となるバージョン管理，依存関係管理，アクセス制御などの基本機能に加え，UPnP や HTTP など標準的なプロトコル向けのインタフェースもサービスとして規定されている．OSGi を用いたサービスゲートウェイでは，ネットワークへのノードの追加などにともない，当該ノードの制御を行うバンドルをアプリケーションとして追加する．また前述のとおり，ECHONET では，通信ミドルウェアを規定し，通信ミドルウェアとアプリケーション間のアプリケーションインタフェース（基本 API）を定義して，アプリケーションから共通に利用されるミドルウェア仕様を定義することで，アプリケーション開発の容易化を図っているが，アプリケーションは ECHONET のプロトコルを意識して実装する必要がある．これに対し，OSGi フレームワークをベースに，プロトコルを意識しない，機器を抽象化したインタフェースを定義して多様な機器を統一的に管理・制御するためのホームネットワーク向けのフレームワーク [75] や，ホームネットワーク全体を OS と見立てて，各種機器の制御アプリケーションを PC のアプリケーションと同様に扱う HomeOS [76] が提案されている．これらの構成方法を適用し，共通のミドルウェアを利用する構成をとることで，アプリケーション開発の効率向上が見込まれ，ノードの拡張が容易になる．

(2)　分散型システム向け技術

分散型システムの場合，各ノードが共通のミドルウェアまたはフレームワーク上にアプリケーションを構築し通信プロトコルなどが隠蔽されたインタフェースで連携し，位置透過性のあるアプリケーション実行環境を構成する（図 7 (b)）．このような構成においては，アプリケーションの更新や追加に加え，アプリケーションの再配置などによりシステムが拡張される．分散型システムの構成方法として，たとえば前述の AUTOSAR [77] では，Virtual Function Bus（VFB）[78] と呼ばれるアプリケーション間の抽象化インタフェースの概念が導入され，Run Time Environment（RTE）として実装されて同一ノード上またはノード間の通信を隠蔽する．これによりアプリケーションのノードへの配置の自由度を高めている．しかしながら，VFB は，設計時の配置自由度を高めるフレームワークであり，動的なシステム拡張には対応していない．このため，Belaggoun らは AUTOSAR を拡張して動的なソフトウェアコンポーネントの配置を可能にする技術を提案している [79]．Belaggoun らの提案では，提案する動的な再配置機構により，ECU 失陥時の機能再配置を行う例が示されている．他にも企業システム向けに標準化された CORBA（Common Object Request Broker Architecture）を組込み向けに軽量化しつつサービス拡張のためのオブジェクトの動的追加機構を追加した Embedded CORBA [80] や，Intelligent electronic device や PLC などの産業分野の機器向けに IEC61499 ベースの動的再構成の手法が示されている [81]．

以上述べたように，ソフトウェアの更新によるシステム拡張にあたっては，更新を可能とする OS やバーチャルマシンなどのアーキテクチャと，拡張を容易にするミドルウェアやフレームワークといった技術の導入が重要となる．さらにこれら技術の導入にあたっては，システム構成やシステムで利用される通信プロトコルの特徴を考慮した設計が課題となる．

4.　システム拡張プロセスに関する技術

前述のとおり，ネットワーク接続された組込みシステムは，ノードの追加やノードの再構成により拡張される．**図 8** に，システムの拡張プロセスを示す．ノードの追加の場合は，最初にノードをネットワークに接続する．一方，ノードの再構成の場合は，再構成用のデータ（更新データ）を生成し，ノードに配信し，ノード上で受信した更新データを適用するというステップが必要となる．ノードの追加またはノードの再構成が完了すると，これらのノードまたは再構成された機能を検出することでシステムへの統合が行われた後，他のノードとのデータ送受信や制御が開始さ

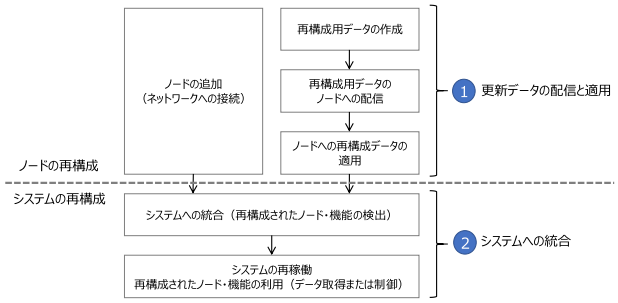

図 8　ネットワーク接続された組込みシステムの再構成プロセス

Fig. 8　Reconfiguration process of networked embedded systems

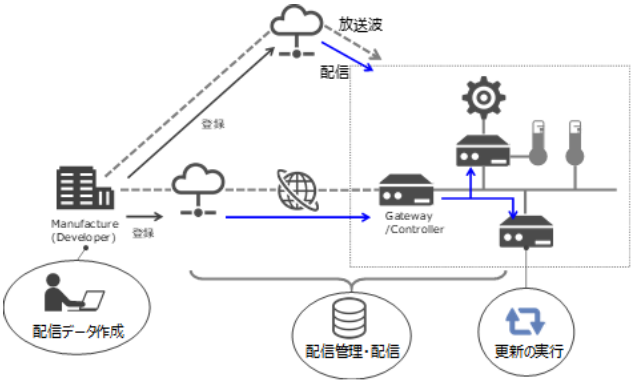

図 9　ソフトウェアの遠隔更新システム概要

Fig. 9　Overview of the software remote update systems.

れ，拡張されたシステムとしての稼働が始まる．これらの過程は，大きく，ノードとしての再構成過程と，システムとしての再構成過程に分類できる．

これらの課程において，ノードとしての再構成過程に関する技術として，更新データの配信と適用に関する技術があり，システムとしての再構成過程に関する技術として，システムへの統合に関する技術がある．

4.1　更新データの配信と適用に関する技術

ノードの再構成は，図 5 のノード構成図に示すストレージに格納されたハードウェアのコンフィグデータやソフトウェア/プログラム，あるいはパラメータを更新することで実現される．このため，更新するためのデータをどのように生成・配送し，ストレージを書き換えるかが課題となる．特に，ネットワーク接続された機器のソフトウェアを遠隔で更新する技術は，ソフトウェアの大規模化が先行した PC や携帯電話，デジタル放送受信機，ゲーム機などで実用化され，頻繁にアップデートが行われている．この場合の更新データの配信経路としては，インターネットのほかに，携帯電話網や放送波が利用されている．このように携帯電話網などの無線接続を利用してソフトウェア/ファームウェアを端末に配信し，遠隔で更新する技術は OTA（Over the Air software/firmware update：無線によるソフトウェアの更新）と呼称される．また，複数のノードがネットワーク接続されてシステムを構成するようなシステムでは，センサネットワークのセンサノードのソフトウェア更新方式が提案されている [82]．このような，ネットワーク経由でのノードの再構成にあたっては，更新データの作成，ノードへの配信管理・配信，ノードにおける更新の実行が必要である（**図 9**）．このような更新データの配信と適用にあたっては，通信コストの削減・消費電力の低減・配信時間の短縮・メモリなどのリソースの少ない環境での更新・システムあるいはノードのダウンタイムの短縮，ロバスト性が共通の課題となる．

また，前述の共通課題に加えてそれぞれのフェーズで個別の課題もある．

4.1.1　配信

(1)　配信データ作成

配信データの作成では，ノードの更新のために何をどのような形式で配信データとして構成するかが課題である．配信管理においては配信対象のシステムやノード・更新内容の特定，ノードへの配信においては End to end でのセキュリティが課題となる．

配信データの形式としては Linux 向けの RPM [83] や Android アプリケーション配布用に APK [59] などのパッケージフォーマットが広く利用されているが，本稿で取り扱うシステム向けに統一的に利用されているフォーマットはいまだ存在しない．

(2)　配信管理

配信管理技術としては OMA DM [84]，Rsync [85]，TR069 [86] などが利用されている．OMA DM は Opne Mobile Alliance（OMA）において定められたデバイスの管理を行う標準規格であるが，そのなかでファームウェアやソフトウェアのアップデート機構として対象特定のための構成同期手順や，ソフトウェアの配信・更新の実行手順を規定している．また Rsync では，サーバとノード間でバイナリレベルでの比較を行い，差分を配信する方式を定義している．OMA では，メモリなどのハードウェアリソース向けのデバイス管理プロトコルとして LwM2M を定義し，そのなかでも軽量なソフトウェアの更新を定義している [87]．

(3)　通信データの削減技術

配信時の課題である通信コストの削減・消費電力の削減・可用性向上を目的とした配信時間の短縮に対応するための共通の手段として，通信データの削減がある．通信データを削減するための差分更新技術も多数提案されている [7], [19], [88], [89], [90], [91], [92], [93]．差分更新では，開発元などで新旧プログラムから差分ファイルを生成して更新対象に転送し，更新対象上の旧プログラムと差分ファイルから新プログラムを復元する（**図 10**）．このようにプログラム全体ではなく差分のみを送信することでネット

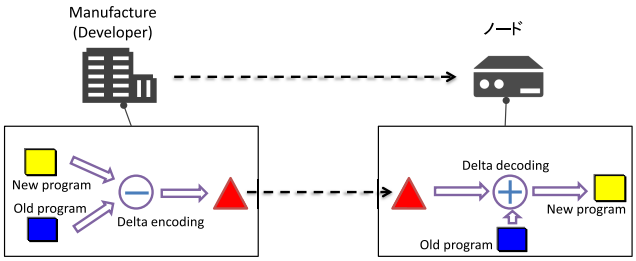

図 10　差分更新の概要
Fig. 10　Overview of the incremental update.

ワーク上を伝送するデータ量を削減できる．

ワイヤレスセンサネットワークの分野では，配信時に加えノードへの更新適用時の課題にも対応したネットワーク経由でのソフトウェア更新方式が提案されている [94]．Tsiftes らは，複数の圧縮アルゴリズムを比較し，ネットワーク伝送量とノード上での処理時間のトレードオフを行い，GZIP が最も消費電力を削減できることを示している [95]．

また，Stolikj らは，複数の圧縮アルゴリズムと差分圧縮アルゴリズムについて同様のトレードオフを行い，bsdiff と LZ77 の組合せが最も好適であることを示している [96], [97]．他にも，中西らは自動車向けに更新途中で給電が停止した場合も復旧後に更新を継続できる，ロバスト性を考慮した差分更新方式を提案している [98]．

4.1.2　更新の適用

(1)　セキュリティ

システムの再構成，特にソフトウェアの更新は，ソフトウェアの危殆化によるセキュリティリスクへの対策として適用が推奨または義務化されている．一方で，再構成の仕組み自体がセキュリティホールとなりリスクを高める懸念がある．情報セキュリティの 7 つの要素（機密性・完全性・可用性・真正性・責任追跡性・否認防止・信頼性）のうち，ソフトウェア更新においては特に完全性と真正性が重要である．たとえば，再構成の仕組みが悪用されて不正なソフトウェアの実行される（完全性と真正性の侵害）ことは重大な問題となりうる．他にも，更新データが盗聴されて知的財産権が脅かされるため機密性も重要である．

コンシューマデバイス向けには，ID ベースの相互認証と鍵計算，およびハッシュチェーンを用いたファームウェアの分割送信による信頼性保証が提案されている [99]．また，WSN 向けに，チャレンジ–レスポンスによりセンサノードのコード改竄を検出してコードを更新することによりそれらを修復または除外する SCUBA プロトコル [100] や，分割した更新データにデジタル署名とハッシュチェーンを付与することで低リソースなデバイスにおいても更新データ全体を検証して完全性を保証する Sluice [101] などが提案されている．Sluice に類似のデジタル署名とハッシュチェーンを用いて更新のセキュリティを確保する仕組みは自動車の ECU 更新向けにも提案されている [102]．自動車向けにはほかにも，IT システム向けに開発されたセキュアな更新フレームワーク TUF [103] を自動車向けに拡張した Uptane [104] が提案されている．近年では，Blockchain を用いてバージョンおよびファームウェアのバイナリを検証する方式が IoT デバイス向け [105], [106] や DoS 攻撃への耐性を持ち可用性も考慮された自動車向け [107] に提案されている．また，上記に加えて，大量に存在する IoT 機器向けに，OEM とデバイスの直接接続がなくても更新データの検証を行うフレームワークが提案されている [108]．

以上，ソフトウェア更新の先行研究について述べたが，ハードウェアのコンフィグデータをソフトウェアの更新データと同様に扱えば，ソフトウェアの遠隔更新の枠組みは，再構成可能なハードウェアを遠隔更新する場合にも適用できる．たとえば，永田ら [109] は FPGA の遠隔再構成システムを提案している．

前述のとおり，従来，遠隔でのソフトウェア更新が適用されてきた分野は携帯電話やスマートフォンなど，直接インターネットに接続する組込みシステムか，同種のセンサノードから構成されるセンサネットワークが想定されていた．このため，システムが多様なノードからなる構成のソフトウェア更新方法については十分に検討されておらず，その更新制御が課題となる．

また，低消費電力化やネットワーク帯域の有効活用のために差分更新を適用するにあたっては，ROM，RAM などのノードのハードウェアリソースを考慮した方式の適用が課題となる．

4.2　システムへの統合に関する技術

あらたに接続されたノードや，あらたに構成された機能をシステムに組み込むためには，システムが接続されたノードや機能を認識し，連携できるようにすることが必要である．このような管理を，ノード追加を行った人間が都度，他のノードの設定変更などによりその対応を行うのは困難である．このため，追加されたノードをシステムに統合するための仕組みとして，ノードや機能の検出の仕組みを構築することが課題となる [110]．

4.2.1　検出と利用

このような検出の仕組みとして前述の UPnP では，Simple Service Discovery Protocol（SSDP）[111] が利用されており，Apple 社 Bonjour [112] では multicast DNS（mDNS）と DNS service discovery（DNS-DS）が組み合わせて用いられる [113]．また，ECHONET Lite などでも独自のノード検出方法が定義されている [31]．自動車分野では現在，AUTOSAR において，サービス検出・利用の仕組みとして SOME/IP [114] が標準化されている．また，産業分野では，前述の OPC UA において機器の登録や検出の方法が定義されている [115], [116], [117]．他にも，アドホッ

クネットワーク向けの Konark [118] やリソース制約のあるセンサネットワーク向けに Bonjour Contiki [119] が提案されている．これらの技術では，検出対象をある機能を提供する「サービス」ととらえ，サービスを検出するための枠組みとして定義されている．このようなサービス検出の仕組みを用いることで，利用者はマニュアルで機器の登録作業などを行う必要なく，ネットワークに組み込まれた機器を利用できるようになる．このようなサービスの特定と識別にあたっては，ノードやサービスをトレースするための識別方法も重要となる．

5. まとめと今後の展望

本稿では，ネットワーク接続された組込みシステムの拡張に関する関連研究を体系的に整理した．最初に，システム拡張に必要な技術を，システム拡張のための構成に関する技術とシステム拡張を実行するプロセスに関する技術に分類し，それぞれの先行技術を示した．システム拡張を可能または容易にする構成に関する技術においては，ハードウェアの再構成技術および再構成を考慮したソフトウェアの構成に関する技術を示し，ソフトウェアの構成に関する技術としては，再構成の前提となる OS・バーチャルマシンレイヤの技術，再構成を容易にするミドルウェアやフレームワーク技術を整理した．システム拡張を実行する過程に関する技術においては，ノードの再構成のための更新データの生成と配送に関する技術と，追加または更新されたノードのシステムへの統合に関する技術について述べた．前者については配信データの作成技術，配信管理およびノードへの配信技術，ノードにおける更新適用技術について述べ，後者については，サービスの検出技術について述べた．

今後は，安全やセキュリティを確保するためにシステム拡張に追従してどこでどのようなシステムが動作しているかのトレーサビリティの管理が要求されるようになっていくと考えられる．具体的には，多数のハードウェアとソフトウェアが稼働している環境下でバージョン管理が複雑化するとともに，それらの組合せにおいて動作を担保できるようにするなどの課題に対応するため，製品のライフサイクル全般にわたる構成管理が重要になっていく．

また，近年スマートフォンのアプリケーションやスマートスピーカなど，サーバと連携して機能を実現する構成が増加している．また，現在分散型のアーキテクチャである自動車においても，Central Computer への集約が進み，最終的には機能の一部がサーバサイドに配置されるというビジョンが描かれている [120] ように，5G など通信技術のさらなる進展にともない，今後はさまざまな分野で組込みシステムとクラウドシステムの境界がなくなり，プログラマブルな世界がさらに進展するものと予想する．

参考文献

[1] 総務省：情報通信白書平成 29 年版 (2017).

[2] 日本学術会議：日本の展望—学術からの提言 2010 情報学分野の展望 (2010).

[3] Sector, I.T.S.: Recommendation ITU-T Y. 2060: Overview of the Internet of things, Series Y: Global information infrastructure, internet protocol aspects and next-generation networks-Frameworks and functional architecture models (2012), available from ⟨https://www.itu.int/rec/T-REC-Y, 2060-201206⟩.

[4] Heath, S.: Embedded systems design, EDN series for design engineers (2nd ed.) (2003).

[5] Tao, F., Cheng, J., Qi, Q., Zhang, M., Zhang, H. and Sui, F.: Digital twin-driven product design, manufacturing and service with big data, *The International Journal of Advanced Manufacturing Technology*, Vol.94, No.9-12, pp.3563–3576 (2018).

[6] Glaessgen, E. and Stargel, D.: The digital twin paradigm for future NASA and US Air Force vehicles, *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA*, p.1818 (2012).

[7] 清原良三：携帯電話のソフトウェアアドインに関する研究 (2008)，入手先 ⟨https://ir.library.osaka-u.ac.jp/repo/ouka/all/2273/22506_%E8%AB%96%E6%96%87.pdf⟩.

[8] Boie, A.: Android Software Updates, *Embedded Linux Conference* (2015).

[9] Zhu, X., Song, B., Ni, Y., Ren, Y. and Li, R.: Software Defined Anything — From Software-Defined Hardware to Software Defined Anything, *Business Trends in the Digital Era*, pp.83–103, Springer, Singapore (2016).

[10] Taivalsaari, A. and Mikkonen, T.: A roadmap to the programmable world: software challenges in the IoT era, *IEEE Software*, Vol.34, No.1, pp.72–80 (2017).

[11] Caskey, J.: Nema sg-ami 1 requirements for smart meter upgradeability, National Electrical Manufacturers Association, Technical Requirements Document (2009).

[12] World Forum for Harmonization of Vehicle Regulations: Proposal for a new UN Regulation on uniform provisions concerning the approval of vehicles with regards to cyber security and cyber security management system (2018), available from ⟨https://unece.org/fileadmin/DAM/trans/doc/2020/wp29grva/ECE-TRANS-WP29-2020-079-Revised.pdf⟩ (accessed 2020-12-28).

[13] World Forum for Harmonization of Vehicle Regulations: Proposal for a new UN Regulation on uniform provisions concerning the approval of vehicles with regards to software update and software updates management system (2018), available from ⟨https://unece.org/fileadmin/DAM/trans/doc/2020/wp29/ECE-TRANS-WP29-2020-080e.pdf⟩ (accessed 2020-12-28).

[14] Bobda, C.: *Introduction to reconfigurable computing: Architectures, algorithms, and applications*, Springer Science & Business Media (2007).

[15] Compton, K. and Hauck, S.: Reconfigurable computing: A survey of systems and software, *ACM Computing Surveys* (csuR), Vol.34, No.2, pp.171–210 (2002).

[16] Liu, L., Zhu, J., Li, Z., Lu, Y., Deng, Y., Han, J., Yin, S. and Wei, S.: A Survey of Coarse-Grained Reconfigurable Architecture and Design: Taxonomy, Challenges, and Applications, *ACM Computing Surveys* (*CSUR*), Vol.52, No.6, pp.1–39 (2019).

[17] Podobas, A., Sano, K. and Matsuoka, S.: A Survey on Coarse-Grained Reconfigurable Architectures from

a Performance Perspective, arXiv preprint arXiv:2004.04509 (2020).
[18] Shoa, A. and Shirani, S.: Run-time reconfigurable systems for digital signal processing applications: A survey, *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, Vol.39, No.3, pp.213–235 (2005).
[19] Arakadakis, K., Charalampidis, P. and Fragkiadakis, A.: Firmware over-the-air programming techniques for IoT networks — A survey, arXiv preprint arXiv:2009.02260 (2020).
[20] Salehie, M. and Tahvildari, L.: Self-adaptive software: Landscape and research challenges, *ACM Trans. Autonomous and Adaptive Systems* (*TAAS*), Vol.4, No.2, p.14 (2009).
[21] 中川博之，鄭　顕志，田原康之：特集 ソフトウェア工学の最前線〜ソフトウェアが社会のすべてを定義する時代［未来に向かって］IoT 時代の環境適応型ソフトウェア，情報処理，Vol.58, No.8, pp.702–704 (2017).
[22] Hallsteinsen, S., Hinchey, M., Park, S. and Schmid, K.: Dynamic software product lines, *Computer*, Vol.41, No.4, pp.93–95 (2008).
[23] Fornari, G. and de Santiago Júnior, V.A.: Dynamically Reconfigurable Systems: A Systematic Literature Review, *Journal of Intelligent & Robotic Systems*, pp.1–21 (2018).
[24] 水野忠則（監修），中条直也，井上雅裕，山田圀裕：未来へつなぐデジタルシリーズ 20：組込みシステム，共立出版 (2013).
[25] ISO/IEC 7498-1:1994 Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model (1994).
[26] RFC1122: Requirements for Internet Hosts — Communication Layers (1989).
[27] available from ⟨https://openconnectivity.org/developer/specifications/upnp-resources/upnp⟩ (accessed 2018-12-23).
[28] Digital Living Network Alliance, available from ⟨https://www.dlna.org/⟩ (accessed 2019-01-20).
[29] ECHONET コンソーシアム「ECHONET 規格書」，入手先 ⟨https://echonet.jp/spec_v321/⟩.
[30] available from ⟨https://zigbeealliance.org/⟩ (accessed 2019-01-20).
[31] ECHONET コンソーシアム「ECHONET Lite 規格書」，available from ⟨https://echonet.jp/spec_v113_lite/⟩
[32] 美原義行，山崎毅文，岡本　学，佐藤　敦：ホームネットワークマップ特定プロトコル HTIP の設計と診断ツールへの適用，情報処理学会論文誌コンシューマ・デバイス & システム（CDS），Vol.2, No.3, pp.34–45 (2012).
[33] available from ⟨https://www.profibus.com⟩ (accessed 2019-01-20).
[34] available from ⟨https://www.odva.org/technology-standards/key-technologies/devicenet/⟩.
[35] available from ⟨http://www.modbus.org⟩ (accessed 2019-01-20).
[36] available from ⟨https://www.odva.org/technology-standards/key-technologies/ethernet-ip/⟩ (accessed 2019-01-20).
[37] available from ⟨https://1.ieee802.org/tsn/⟩.
[38] available from ⟨https://opcfoundation.org/⟩ (accessed 2019-05-18).
[39] Bosch, R.: CAN specification version 2.0, Rober Bousch GmbH, Postfach, 300240 (1991).
[40] ISO 17987 Part 1-7 (2016).
[41] ISO 17458-1:2013 Road vehicles — FlexRay communications system — Part 1: General information and use case definition (2013).
[42] ISO 15765-2:2011 Road vehicles — Diagnostic communication over Controller Area Network (DoCAN) — Part 2: Transport protocol and network layer services (2011).
[43] ISO 14229-1:2013 Road vehicles — Unified diagnostic services (UDS) — Part 1: Specification and requirements (2013).
[44] 木谷光博，片岡幹雄，寺岡秀敏：自動運転向け車内ネットワークシステムにおけるデータ伝送方式の開発，情報処理学会論文誌コンシューマ・デバイス & システム（CDS），Vol.6, No.2, pp.43–51 (2016).
[45] available from ⟨http://www.opensig.org/⟩ (accessed 2019-01-20).
[46] IEEE 802.3bw-2015: IEEE Standard for Ethernet Amendment 1: Physical Layer Specifications and Management Parameters for 100 Mb/s Operation over a Single Balanced Twisted Pair Cable (100BASE-T1) (2015).
[47] 落合秀也，井上博之，寺西裕一，江崎　浩：IEEE1888 通信スタックの組込み向け軽量実装，情報処理学会論文誌，Vol.54, No.7, pp.1849–1860 (2013).
[48] Zynq UltraScale+ MPSoC, available from ⟨https://japan.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html⟩.
[49] Vipin, K. and Fahmy, S.A.: FPGA dynamic and partial reconfiguration: A survey of architectures, methods, and applications, *ACM Computing Surveys* (*CSUR*), Vol.51, No.4, p.72 (2018).
[50] 本村真人，藤井太郎，古田浩一朗，安生健一朗，矢部義一，戸川勝巳，佐々木僚子：新世代マイクロプロセッサアーキテクチャ（後編）：3. 実例　4. 動的再構成プロセッサ（DRP），情報処理，Vol.46, No.11, pp.1259–1265 (2005).
[51] 日経 BP 社：やわらかくなる LSI—広がり始めた動的再構成，日経エレクトロニクス，pp.59–66 (2011).
[52] Fujii, T. et al.: New Generation Dynamically Reconfigurable Processor Technology for Accelerating Embedded AI Applications, *2018 IEEE Symposium on VLSI Circuits*, pp.41–42, DOI: 10.1109/VLSIC.2018.8502438 (2018).
[53] Balani, R., Han, C.C., Rengaswamy, R.K., Tsigkogiannis, I. and Srivastava, M.: Multi-level software reconfiguration for sensor networks, *Proc. 6th ACM & IEEE International Conference on Embedded Software*, pp.112–121, ACM (2006).
[54] Han, C.C., Rengaswamy, R.K., Shea, R., Kohler, E. and Srivastava, M.: SOS: A dynamic operating system for sensor networks, *Proc. 3rd International Conference on Mobile Systems, Applications, And Services* (*Mobisys*), pp.1–2 (2005).
[55] Dunkels, A., Gronvall, B. and Voigt, T.: Contiki-a lightweight and flexible operating system for tiny networked sensors, 29th Annual IEEE International Conference on Local Computer Networks, pp.455–462, IEEE (2004).
[56] available from ⟨https://www.toppers.jp/⟩ (accessed 2019-01-20).
[57] Chen, Y., Zhang, Y., Wang, Z., Xia, L., Bao, C. and Wei, T.: Adaptive android kernel live patching, *26th USENIX Security Symposium* (*USENIX Security 17*), pp.1253–1270 (2017).
[58] available from ⟨https://java.com/ja⟩ (accessed 2019-01-20).
[59] available from ⟨https://developer.android.com/guide/

platform/⟩.
[60] Simon, D., Cifuentes, C., Cleal, D., Daniels, J. and White, D.: Java^{TM} on the bare metal of wireless sensor devices: the squawk Java virtual machine, *Proc. 2nd International Conference on Virtual Execution Environments*, pp.78–88, ACM (2006).
[61] Levis, P. and Culler, D.: Maté: A tiny virtual machine for sensor networks, *ACM Sigplan Notices*, Vol.37, No.10, pp.85–95, ACM (2002).
[62] Koshy, J. and Pandey, R.: VMSTAR: Synthesizing scalable runtime environments for sensor networks, *Proc. 3rd International Conference on Embedded Networked Sensor Systems*, pp.243–254, ACM (2005).
[63] available from ⟨https://www.lua.org/⟩ (accessed 2019-02-21).
[64] Tanaka, K., Nagumanthri, A.D. and Matsumoto, Y.: mruby — Rapid Software Development for Embedded Systems, *2015 15th International Conference on Computational Science and Its Applications* (*ICCSA*), pp.27–32, IEEE (2015).
[65] 倉光君郎：拡張性のある組み込みアプリケーションを実現するスクリプティング言語の開発，情報処理学会論文誌，Vol.51, No.12, pp.2185–2194 (2010).
[66] Dunkels, A.: A low-overhead script language for tiny networked embedded systems, SICS Research Report (2006).
[67] Khan, I., Belqasmi, F., Glitho, R., Crespi, N., Morrow, M. and Polakos, P.: Wireless sensor network virtualization: A survey, *IEEE Communications Surveys & Tutorials*, Vol.18, No.1, pp.553–576 (2015).
[68] Sollfrank, M., Loch, F. and Vogel-Heuser, B.: Exploring Docker Containers for Time-sensitive Applications in Networked Control Systems, *2019 IEEE 17th International Conference on Industrial Informatics* (*INDIN*), Vol.1, pp.1760–1765, IEEE (2019).
[69] Noronha, V., Lang, E., Riegel, M. and Bauschert, T.: Performance evaluation of container based virtualization on embedded microprocessors, *2018 30th International Teletraffic Congress* (*ITC 30*), Vol.1, pp.79–84, IEEE (2018).
[70] Lin, C.W., Kim, B. and Shiraishi, S.: Hardware virtualization and task allocation for plug-and-play automotive systems, *IEEE Design & Test* (2019).
[71] OSGi Alliance「OSGi Service Platform Core Specification」, available from ⟨http://www.osgi.org/⟩ (accessed 2012-12-13).
[72] Li, Y., Wang, F.Y., He, F. and Li, Z.: OSGi-based service gateway architecture for intelligent automobiles, *Proc. IEEE Intelligent Vehicles Symposium*, pp.861–865, IEEE (2005).
[73] Sun, Y., Huang, W.L., Tang, S.M., Qiao, X. and Wang, F.Y.: Design of an OSEK/VDX and OSGi-based embedded software platform for vehicular applications, *IEEE International Conference on Vehicular Electronics and Safety* (*ICVES*), pp.1–6, IEEE (2007).
[74] Park, P., Yim, H., Moon, H. and Jung, J.: An OSGi based in-vehicle gateway platform architecture for improved sensor extensibility and interoperability, *33rd Annual IEEE International Computer Software and Applications Conference* (*COMPSAC'09*), Vol.2, pp.140–147, IEEE (2009).
[75] 宮田克也，寺岡秀敏，関原拓也，芳野明彦：ホームゲートウェイ向け機器管理制御フレームワークの開発，情報処理学会論文誌コンシューマ・デバイス & システム（CDS），Vol.3, No.3, pp.39–48 (2013).
[76] Dixon, C., Mahajan, R., Agarwal, S., Brush, A.J., Lee, B., Saroiu, S. and Bahl, P.: An operating system for the home, *Proc. 9th USENIX Conference on Networked Systems Design and Implementation*, pp.25–25, USENIX Association (2012).
[77] available from ⟨https://www.autosar.org/⟩ (accessed 2019-01-20).
[78] Dafang, W., Shiqiang, L., Bo, H., Guifan, Z. and Jiuyang, Z.: Communication mechanisms on the virtual functional bus of AUTOSAR, *2010 International Conference on Intelligent Computation Technology and Automation* (*ICICTA*), Vol.1, pp.982–985, IEEE (2010).
[79] Belaggoun, A. and Issarny, V.: Towards adaptive autosar: A system-level approach, FISITA 2016 World Automotive Congress (2016).
[80] 伊賀徳寿，中本幸一，奥山嘉昭，佐藤直樹，檜原弘樹：組込みシステム向け CORBA の開発と評価，情報処理学会論文誌コンピューティングシステム（ACS），Vol.44, No.SIG10(ACS2), pp.164–176 (2003).
[81] Strasser, T., Rooker, M., Ebenhofer, G. and Zoitl, A.: Standardized dynamic reconfiguration of control applications in industrial systems, *Rapid Automation: Concepts, Methodologies, Tools, and Applications*, pp.776–793, IGI Global (2019).
[82] Han, C.C., Kumar, R., Shea, R. and Srivastava, M.: Sensor network software update management: A survey, *International Journal of Network Management*, Vol.15, No.4, pp.283–294 (2005).
[83] Ewing, M. and Troan, E.: The rpm packaging system, Proc. 1st Conference on Freely Redistributable Software, Cambridge, MA, USA (1996).
[84] Open Mobile Alliance, available from ⟨http://www.openmobilealliance.org/wp/Overviews/dm_overview.html⟩ (accessed 2016-04-16).
[85] Tridgell, A. and Mackerras, P.: The rsync algorithm (1996).
[86] TR-069 CPE WAN Management Protocol, Issue: 1 Amendment 6, Approval Date: March 2018, CWMP Version: 1.4 (2018).
[87] Lightweight M2M – Software management Object (LwM2M Object – SwMgmt) Approved Version 1.0 (2018).
[88] available from ⟨http://www.pocketsoft.com/⟩ (accessed 2019-02-21).
[89] available from ⟨http://www.redbend.com/en⟩ (accessed 2016-04-16).
[90] available from ⟨http://www.daemonology.net/bsdiff/⟩ (accessed 2019-02-21).
[91] available from ⟨http://xdelta.org/⟩ (accessed 2019-02-21).
[92] available from ⟨http://cis.poly.edu/zdelta/⟩ (accessed 2015-03-16).
[93] available from ⟨http://code.google.com/p/open-vcdiff/⟩ (accessed 2019-02-21).
[94] Wang, Q., Zhu, Y. and Cheng, L.: Reprogramming wireless sensor networks: Challenges and approaches, *IEEE Network*, Vol.20, No.3, pp.48–55 (2006).
[95] Tsiftes, N., Dunkels, A. and Voigt, T.: Efficient sensor network reprogramming through compression of executable modules, *5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks* (*SECON'08*), pp.359–367, IEEE (2008).
[96] Stolikj, M., Cuijpers, P.J. and Lukkien, J.J.: Energy-

aware reprogramming of sensor networks using incremental update and compression, *Procedia Computer Science*, Vol.10, pp.179–187 (2012).
[97] Stolikj, M., Cuijpers, P.J. and Lukkien, J.J.: Efficient reprogramming of wireless sensor networks using incremental updates, *2013 IEEE International Conference on Pervasive Computing and Communications Workshops* (*PERCOM Workshops*), pp.584–589, IEEE (2013).
[98] Nakanishi, T., Shih, H.H., Hisazumi, K. and Fukuda, A.: A software update scheme by airwaves for automotive equipment, *2013 International Conference on Informatics, Electronics & Vision* (*ICIEV*), pp.1–6, IEEE (2013).
[99] Choi, B.C., Lee, S.H., Na, J.C. and Lee, J.H.: Secure firmware validation and update for consumer devices in home networking, *IEEE Trans. Consumer Electronics*, Vol.62, No.1, pp.39–44 (2016).
[100] Seshadri, A., Luk, M., Perrig, A., van Doorn, L. and Khosla, P.: SCUBA: Secure code update by attestation in sensor networks, *Proc. 5th ACM Workshop on Wireless Security*, pp.85–94 (2006).
[101] Lanigan, P.E., Gandhi, R. and Narasimhan, P.: Sluice: Secure dissemination of code updates in sensor networks, *26th IEEE International Conference on Distributed Computing Systems* (*ICDCS 2006*), pp.53–53, IEEE (2006).
[102] Nilsson, D.K. and Larson, U.E.: Secure firmware updates over the air in intelligent vehicles, *ICC Workshops-2008 IEEE International Conference on Communications Workshops*, pp.380–384, IEEE (2008).
[103] Samuel, J., Mathewson, N., Cappos, J. and Dingledine, R.: Survivable key compromise in software update systems, *Proc. 17th ACM Conference on Computer and Communications Security*, pp.61–72 (2010).
[104] Karthik, T., Brown, A., Awwad, S., McCoy, D., Bielawski, R., Mott, C. and Cappos, J.: Uptane: Securing software updates for automobiles, *International Conference on Embedded Security in Car*, pp.1–11 (2016).
[105] Lee, B. and Lee, J.H.: Blockchain-based secure firmware update for embedded devices in an Internet of Things environment, *The Journal of Supercomputing*, Vol.73, No.3, pp.1152–1167 (2017).
[106] Yohan, A. and Lo, N.W.: An over-the-blockchain firmware update framework for IoT devices, *2018 IEEE Conference on Dependable and Secure Computing* (*DSC*), pp.1–8, IEEE (2018).
[107] Baza, M., Nabil, M., Lasla, N., Fidan, K., Mahmoud, M. and Abdallah, M.: Blockchain-based firmware update scheme tailored for autonomous vehicles, *2019 IEEE Wireless Communications and Networking Conference* (*WCNC*), pp.1–7, IEEE (2019).
[108] Asokan, N., Nyman, T., Rattanavipanon, N., Sadeghi, A.R. and Tsudik, G.: ASSURED: Architecture for secure software update of realistic embedded devices, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.37, No.11, pp.2290–2300 (2018).
[109] 永田和生，原田英雄，牛嶋和行，久我守弘，末吉敏則：FPGA 遠隔再構成システムの設計と実装，電子情報通信学会論文誌 D，Vol.90, No.6, pp.1357–1366 (2007).
[110] Guinard, D., Trifa, V., Karnouskos, S., Spiess, P. and Savio, D.: Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services, *IEEE Trans. Services Computing*, Vol.3, No.3, pp.223–235 (2010).
[111] Simple Service Discovery Protocol/1.0, available from ⟨https://tools.ietf.org/html/draft-cai-ssdp-v1-03⟩ (accessed 2019-01-20).
[112] Bonjour, available from ⟨https://developer.apple.com/bonjour/⟩ (accessed 2019-01-20).
[113] Edwards, W.K.: Discovery systems in ubiquitous computing, *IEEE Pervasive Computing*, Vol.5, No.2, pp.70–77 (2006).
[114] Scalable service-Oriented MiddlewarE over IP (SOME/IP), available from ⟨http://some-ip.com/⟩
[115] OPC Foundation: OPC UA Specification Part 12: Discovery, OPC Foundation, Technical Report (2015).
[116] Dürkop, L., Imtiaz, J., Trsek, H., Wisniewski, L. and Jasperneite, J.: Using OPC-UA for the autoconfiguration of real-time Ethernet systems, *2013 11th IEEE International Conference on Industrial Informatics* (*INDIN*), pp.248–253, IEEE (2013).
[117] Profanter, S., Dorofeev, K., Zoitl, A. and Knoll, A.: OPC UA for plug & produce: Automatic device discovery using LDS-ME, *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation* (*ETFA*), pp.1–8, IEEE (2017).
[118] Helal, S., Desai, N., Verma, V. and Lee, C.: Konark-a service discovery and delivery protocol for ad-hoc networks, *2003 IEEE Wireless Communications and Networking* (*WCNC 2003*), Vol.3, pp.2107–2113, IEEE (2003).
[119] Klauck, R. and Kirsche, M.: Bonjour contiki: A case study of a DNS-based discovery service for the internet of things, *International Conference on Ad-Hoc Networks and Wireless*, pp.316–329, Springer, Berlin, Heidelberg (2012).
[120] Baic, D., Langjahr, P., Haas, W. and Fessard, A.: Safe computing with central ECUs, *Internationales Stuttgarter Symposium*, pp.155–163 (2018).

寺岡 秀敏　（正会員）

2002 年京都大学大学院工学研究科修士課程修了．同年（株）日立製作所．組み込みシステム，ネットワーク，車載システムに従事．博士（情報学）．

尾崎 友哉　（正会員）

1990 年名古屋大学大学院工学研究科修士課程修了．同年（株）日立製作所入社．2020 年 4 月より長崎大学情報データ科学部教授．組み込みシステム，ユーザインタフェース，EMS（Energy Management System）に関する研究開発に従事．博士（情報学）．