2A-04

小型 IoT 機器を対象とした動的システムマイグレーションの研究状況

大谷　真†　　中尾　司ピエール†

湘南工科大学大学院電気情報工学専攻†

要旨

多数の小型 IoT 機器が協調動作している環境では、処理集中や一時的故障などで、不特定の IoT 機器に突然トラブルが発生することがある。このときに、当該 IoT 機器内で実行中のアプリケーションと OS を"生きた状態"でバックアップ PC に動的に移動（マイグレート）する方式の研究を行っている。現在までに、実現方式の考案と実装開発と実機検証が完了している。

１．はじめに

IoT 技術の進展に伴い、ネットワークで接続された多数の小型 IoT 機器が協調動作する形態が、従来からの機械制御などはもとより、近い将来、家庭、店舗、工場、倉庫、農地、公共スペースなどあらゆる領域に適用されていくと考えられる。これらの小型 IoT 機器は一般に、小さいマイクロプロセッサを搭載し、その上で小型のリアルタイム OS(RTOS)とアプリケーション(AP)が一体化して動作する形をとっている。なお、本論でいう"小さいマイクロプロセッサ"とは、MMU などの上位機能を持たず限定されたアーキテクチャのプロセッサ、例えば ARM Coretex-M3 などを意味する。安価ではあるが性能が低く信頼性も必ずしも高くない。したがって、処理集中や一時的故障などが発生し、不特定の IoT 機器に突然トラブルが発生することがある。このようなときに図１に示すように、トラブルが発生した IoT 機器内で実行中のシステム（AP と RTOS）を PC のような比較的高性能なバックアップコンピュータ（BKC）にその時点で動的に移動する（すなわち IoT 機器から移出(emigrate)し BKC に移入(immigrate)して実行を継続させる）ことが、我々の研究の目的である。ここで留意すべきは：(1)AP と RTOS は単一コアイメージ内で結合動作しているため AP プロセスだけの移動はできず、システム全体を"生きた状態"で移動しなければならない。および、(2) AP と RTOS は一体になってビルドされるので両者のバージョン/リビジョンの組み合わせで極めて多種のバイナリが環境内に存在する。しかしどのバイナリに対し移動が発生するかは事前予測できない。したがって BKC 内で交代システムを予め立ち上げ待機しておくことができない。

Research Status on the Dynamic System Migration for Small IoT Devices, † Makoto Oya and Tsukasa-Pierre Nakao, Shonan Institute of Technology

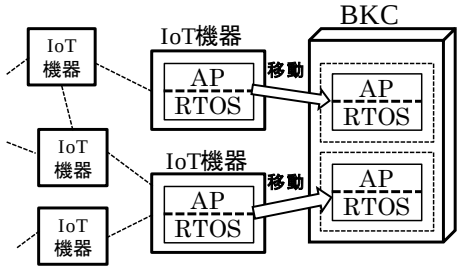

図１：動的システムマイグレーション

２．解決の方針と課題

研究目的達成のアプローチと課題を記す。

（１）考えられ解決方法

移出側（IoT 機器側）で実行状態データを取り出し移入側（BKC 側）でそれを元に実行を継続することが必要だが、その実行主体を何にするかで、図２に示す３つの方法が考えられる。我々はこれらのうち(c)を採用することにした。

(a) VM を介して移動：移出側と移入側ともに VM（移入側はエミュレータ機能を含む）を起動しておき、VM 間で実行状態データを受け渡す。OS や AP に手を加えず外側から移出と移入が可能。いわゆる VM 間ライブマイグレーションとしてすでに確立している方法である。しかし本研究が前提とする小さいマイクロプロセッサ内で VM を作動させることは現実的に不可能である。

(b) OS 間で直接移動：RTOS が自力で移出し移入する方法である。OS の大幅な改造が必要であり現実的でない。

(c) OS で移出し VM で移入：移出は RTOS が自力で行い、移入は VM(エミュレータを含む) が行う方法である。OS を（できれば AP も）を改造しないで済ませる方法を考案する必要がある。

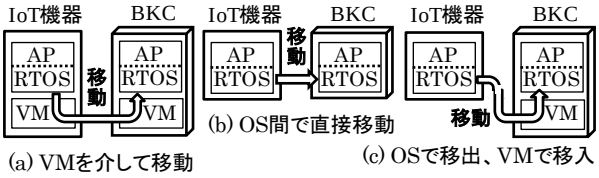

図２：考えられる移動方法

３．現在までの研究状況

現時点で、具体的な移動実現方式の考案と方式の妥当性検証のための実装開発と実験が完了

している[1]。以下にその概略を記す。

３．１　移動実現方式の概略

IoT 機器側での動的システムマイグレーションの開始は特定の割込み（DSM 割込みと呼ぶ）を起こすことで指示することにした。DSM 割込みの契機としては、AP 内での検知、OS 内での検知、タイマ、特別なハードウェア動作などが考えられる。DSM 割込みに対する割込み処理プログラムは DSMC と呼ばれ、本方式の主要部分である。

図３にシステム構成を示す。IoT 機器内のメモリ（ROM と SRAM によって構成される）には AP と RTOS の他に DSMC が入っている。また IOT 機器内には、プログラム実行のためのレジスタ（汎用レジスタ、スタックレジスタ、リンクレジスタ、プログラムカウンタ、その他のプログラム実行制御レジスタ）と、IO 制御のためのレジスタ（タイマ制御レジスタ、割込み制御用のレジスタ群、ペリフェラル制御レジスタなど）が存在する。BKC 側には、全体を制御するための DSM サーバ（図２(c)の VM に対応する）が存在する。DSM サーバは、IoT 機器側の DSMC からのデータを受信処理するとともに、IoT 機器のハードウェアアーキテクチャをエミュレートする機能を内包している。DSM サーバの配下にはエミュレーション対象プログラムを格納するためのエミュレーションメモリと IoT 機器側の実際のレジスタに対応する仮想レジスタが存在する。

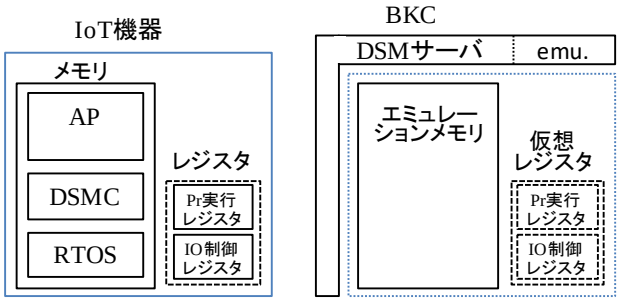

図３：システム構成

図 4 に IoT 機器側での移送処理の概略を示す。①DSM 割込みが発生すると、AP の実行が停止する(②)とともに DSMC の処理が開始する。DSMC の中では、③IoT 機器内のメモリの内容を DSM サーバに伝送し、DSM サーバは受信した内容をエミュレーションメモリに書き込む。同様にプログラム実行レジスタの内容を BKC 内の仮想レジスタに書き込む。ただしプログラムカウンタは DSMC 内の再開始アドレスとしておく。なお、IO 制御レジスタの内容はメモリ伝送前に DSMC 内メモリに退避するだけにしておく。

図５に BKC 側での移入処理の様子を示す。図４の処理が完了した段階で、DSM サーバは伝送さ

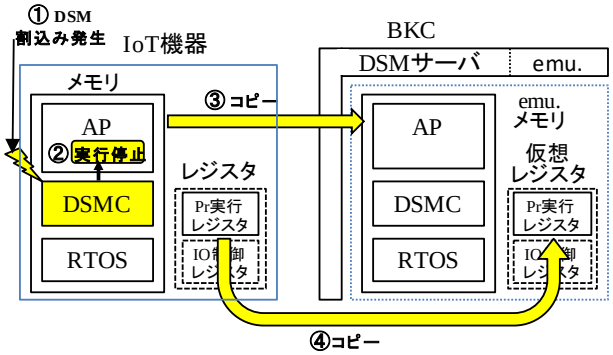

図４：IoT 機器側での移送処理

れてきた再開始アドレスにプログラムカウンタをセットしてエミュレーション実行を開始する。これにより BKC 内で DSMC の後半が再開始する DSMC の後半では、IO 制御レジスタを退避領域の内容を元にセットしたのち割込み処理から return する。その結果 AP が IoT 機器側で割込みが起こった時点のアドレスから実行を再開する。IO 制御レジスタをエミュレーション開始後に DSMC の中でセットするのは IO 制御レジスタのセットに伴いハードウェアや OS が副次的に動作する可能性があるためである。

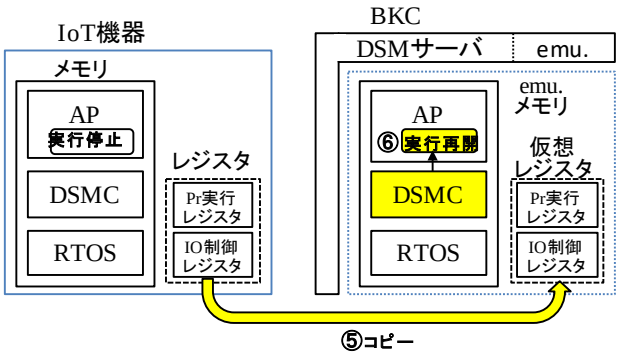

図５：BKC 側での移入処理

３．２　実装と実験

次の環境で提案方式検証のための実装開発を行った：IoT 機器側［FM3(ARM Coretex-M3)、μT-Kernel2.0］、BKC 側［Intel PC、Linux］、DSM サーバ［QEMU を改造］。検証実験では、DSM 割込みはタイマを用いた。IoT 機器側システム(AP とμT-Kernel)が動的に DSM サーバ下に移動し実行を再開することが確認できた。

４．まとめ

小型 IoT 機器を対象とした動的システムマイグレーションの方式考案と実装検証を完了した。なお、性能面についても一部評価済みである[2]。

参考文献

[1]中尾,大谷：小型 IoT 機器を対象とした動的システムマイグレーション（準備中）
[2]Nako and Oya, Provisional Process Migration From IoT Devices: A Performance Study, IEEE/GCCE2015, pp.493-494, 2015