LC-004

微小アプライアンス用カーネル UK におけるタスク実行方式
Study on a Task Execution of a Kernel "UK" for Tiny Sensor Devices

小熊 寿 † 田中 聡 † 山崎 憲一 †
Hisashi Oguma Satoshi Tanaka Kenichi Yamazaki

1 はじめに

将来のユビキタスコンピューティング環境では、小型化された計算機が実世界の環境に多数埋め込まれ、日常生活のあらゆる状況に応じて多種多様なサービスが提供される。環境中に埋め込まれた計算機はセンサ機能を持ち、センサから得られる実環境情報を、多数の計算機とそれらを接続するネットワークにより構成されるサイバー空間へ送信する。サイバー空間では、環境中の計算機から送られてきた実世界情報を集積、加工することにより、実世界の状況を理解し、その状況に応じたサービスが実世界のエンティティに提供される。

このような環境中に埋め込まれた計算機の実現に向けて、我々は近年微細化の進歩が著しい小型計算機に着目した。将来、センサ、計算機、無線通信機能、電源はチップ大の大きさの中に実現されると考えている。本稿では、このような機器を微小アプライアンスと呼ぶ。微小アプライアンスは、小型化により、任意の場所に導入することが可能である。しかし、チップに積載できるリソースは制限が大きい。特に、電力が不安定であり、安定しない電源条件においても、センサとしての機能を断続的に稼動させることが必要である。そのため、小さなオーバヘッドを実現したカーネルにより、微小アプライアンスが適切に制御されることが必須であると考えている。我々は、微小アプライアンスに適用するカーネルである Ubiquitous Kernel (以下、UK と略す) の設計と実現を行っている。本稿では、微小アプライアンスの要求条件を明らかにするとともに、不安定な電力の環境において有効な UK のタスク実行方式について述べる。

2 微小アプライアンス用カーネル UK

本章では、微小アプライアンスの目的と要求条件を明らかにし、UK に求められる機能について述べる。

2.1 目的

本研究は、センサやタグを持つ微小アプライアンスを利用し、実環境に対して多種多様なサービスを提供することを目的としている。実環境の状況を理解し、その場に適したサービスを提供するために、微小アプライアンスを利用して実環境の情報を取得する。

2.2 要求条件

実環境情報を取得する微小アプライアンスへの要求条件として、以下の項目を挙げる。

モビリティ 精度を高めるためには、センサが測定対象の近くに存在することが望ましい。そのため、環境に依存せず、任意の場所に導入できること、あるいは対象と一緒に移動できることが必要となり、高い自由度を実現しなければならない。

即時応答性 実環境情報にとって、時間は重要な要素である。情報の特性によっては再取得不可能なものあり、それらは最優先に処理されなければならない。またサービスによっては、対象情報の発生から獲得、提供までの遅延を最小限にすることが求められる。

長期間運用 微小アプライアンスはバッテリを持つことになる。しかしながらそのバッテリは、ある程度の容量しか期待できない。そのような微小アプライアンスの動作時間をできるだけ長くするために、アイドル時間の省電力化や、処理すべきタスクの特徴を利用したシステム制御機能などが求められる。

電源切断への対処 微小アプライアンスの電力は不安定であり、断続的な電力供給になると予想される。そのため、微小アプライアンスによる処理が停止した場合においても、再開時の処理継続を可能とし、かつ素早く復帰しなければならない。

自律的動作 微小アプライアンスは無線通信を利用するため、通信が途切れる可能性がある。通信不能のため処理が滞るのではなく、状態を把握し、適切な切断時動作 (Disconnected Operation) をすべきである。

スケーラビリティ 複数の微小アプライアンスを用意し分散配置することは、情報取得精度をより高めること、および微小アプライアンス間の連携により高度なサービスを提供できることにつながる。しかしながら微小アプライアンスの増大により、システムのオーバヘッドが増大してはいけない。

微小アプライアンスを利用した研究として、無線センサネットワークに関する研究が従来より数多く存在する。これらは、取得したセンサ情報の伝達を目的とし、その上で微小アプライアンスの消費電力を抑える研究や、微小アプライアンスの停止時にネットワークを再構成する研究がされている。しかしながら微小アプライアンス自身の制御に関する研究は少なく、特に断続的な電源供給ながらも、センサを利用したプログラムが実行できるよう制御する研究は見当たらない。

2.3 タスクの特徴

UK は、センサにより実環境情報を取得し、その情報を利用したサービスを提供するプログラムを対象としている。

UK にとってタスクとは、管理すべきプログラムの実行イメージである。センサによる実環境情報取得を目的としたタスクには、以下のような特徴がある。

単純な処理 微小アプライアンスは、計算リソースが乏しい。また単独で取得した実環境情報のみで、高機能なサービスを提供することは難しい。よってタスクは、単純な処理の組み合わせ、および繰り返しから構成される。

†株式会社 NTT ドコモ ネットワーク研究所
Network Laboratories, NTT DoCoMo, Inc.

動作条件　センサのタスクは取得した情報が処理対象データそのものであるため、情報取得のタイミングをトリガとして、タスク処理が開始される。取得のタイミングには、インターバルタイマなどを利用することにより、ある程度予測できるものと、測定対象である物理現象の発生時というタイミングが予測できないものがある。

予測可能タイプは再測定も可能であるため、タイミングがずれたとしてもある程度の補正が可能である。予測不能タイプは測定だけでなく、タイミングを知るためにもセンサが必要となるため、カーネルのサポートを必要とする。

2.4　実現に向けた問題点

本研究では、微小アプライアンスに要求する機能として、計算機能、無線通信機能、センサ、さらに不揮発性メモリである MRAM (Magnetoresistive Random Access Memory)[1] と発電機能がある。

MRAM は従来の電荷型メモリと異なり、磁化により状態を記憶するメモリである。そのため電荷を失った状態であっても、その情報を失うことはない。将来的には生産コストも目処がつき、従来の DRAM などにかわるメモリと期待されている。

微小アプライアンスは、消費電力も小さいことが予想されるため、少しの発電量で動作できると考えられる。例えば振動による加速度を測定する場合、同時に振動による起電力も得ることも可能であり、短時間の処理は可能となる [2]。

このようにセンサにより取得する情報によっては、バッテリに依存せずとも微小アプライアンスによる処理が可能となる。

前節に挙げた要求条件および処理すべきタスクの特徴より、UK は以下に示す機能が必要となる。

- 微小アプライアンスへの電力供給が断続的であってもタスク処理を可能とし、かつ停止、再開の処理が短時間でできる。
- 物理現象およびイベント発生を知らせる割り込み処理や、処理中のタスクに対する CPU 横取りが、小さなオーバヘッドでできる。
- メモリなどの計算機リソースが限られているため、優れたハードウェア利用効率を実現する。
- 微小アプライアンスの都合により、タスク処理不能とならないよう、近隣に存在する同仕様の微小アプライアンス間で、タスクの代行機能があることが好ましい。

このような機能を UK が実現するためには、以下に示す問題がある。

- 任意のタイミングでチェックポイントが可能であれば、電源切断から再開できる [3]。また実行中タスクのコンテキストを保存し復元できれば、タスク間優先順、センサ情報取得待ちや通信待ちによる CPU 横取りからの再開が可能となる。

  しかしながら双方とも、中断したイメージの保存および復元のオーバヘッドを伴い、即時応答性や、電源再開時から素早く復帰する際の妨げとなる。さらに微小アプライアンスはストレージや通信機能も乏しいため、先述のイメージを保存すること、またはネットワークで伝送することは難しい。
- 近年実用化が期待される不揮発性メモリを利用すれば、電源切断時においても情報が消えないため、メモリ状態を保存する必要がなくなる。しかしながらレジスタ情報は消えてしまうため、上記と同様に CPU 状態のチェックポイントが必要となる。さらに、ロールバック時に、CPU 状態とメモリ状態との一貫性を保つために、チェックポイントからのメモリ変更操作のログが別途必要となる。

次章では、これらの問題を解決するための方法として、現在我々が考えている方法について述べる。

3　実現に向けたアイデア

前章に述べたようにセンサのタスクは、取得した情報が処理対象データである。よってその情報を保存し、時間や電力残量などの決められた条件の範囲であれば、どのタイミングで実際のデータ処理を行っても問題ない。つまりその条件の範囲内であれば、そこまでに得た計算結果を破棄し、センサが情報を取得した直後から処理を再開することができる。計算結果を破棄してもよいのであれば、電源切断からの再開を実現するために、途中の計算結果を保存および回復する必要がない。

そこで UK では、タスク処理方法として連鎖型トランザクションを応用する。

UK では、タスクを複数のまとまった処理単位 (これをブロックと呼ぶ) に分割し、各ブロックの先頭をセーブポイントとする。これにより、あるブロックの処理中に中断した場合、中断したところから再開するのではなく、ブロックの先頭から処理を再開する。

ロールバック機能を持つ連鎖型トランザクションは、セーブポイントを保存するためにディスクなどのストレージを必要とする。MRAM はその性質から、ストレージとして代用できる。しかしながら前章で述べたように、ロールバックを実現するため、チェックポイントからのメモリ変更操作のログが必要となる。

UK のタスク処理において、ブロックは処理に必要なデータを前のブロックから受取り、次のブロックに渡すべきデータを書き出して終わる。電源切断時から再開するときには、受取ったデータをそのまま利用し、渡すべきデータは全て破棄する。これにより、メモリ変更内容のログが不要となり、電源復帰時にロールバックすべきブロックの即時実行が実現できる。

以降本文では、これまでに述べたハードウェア仕様と処理すべきプログラムに基づいた UK の設計について述べる。

4　UK の設計

4.1　システム構成

UK が扱うタスクの構造、および処理方法について述べる。

UK は、複数のタスクを管理、制御する。タスクの構成を図 1 に示す。タスクは 1 つ以上のブロック、ブロッ

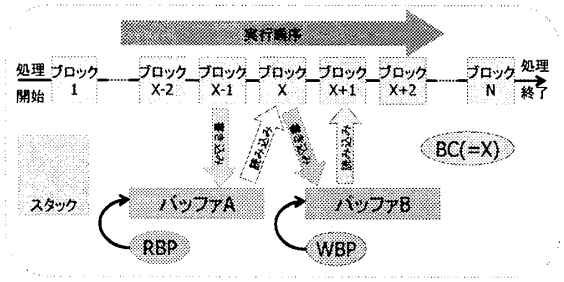

図 1: UK におけるタスク構造

ク間のデータの引継ぎを行うためのバッファ (A、B の 2 つ) とバッファを指示する *RBP* (Read Buffer Pointer)、*WBP* (Write Buffer Pointer)、および実行中のブロックを示す *BC* (Block Counter)、ブロックの実行に必要なスタックから構成される。各要素および要素間の動作を以下に述べる。

- 1 つのタスク内に存在する全てのブロックは、そのタスクが処理を担うプログラムに対応している。ブロックは処理順に従って実行される。
- ブロックは (原則として) 大域変数を利用せずに、局所変数のみで処理を行う。タスク内ブロック間通信は、直前および直後のブロックとのみ行い、その際には片方のバッファを介して行う。
- バッファ *A* はあるブロック *X* の処理に必要な入力データを持つ。*RBP* はバッファ *A* の先頭アドレスを持つ。ブロック *X* は *RBP* を介し、読み込み用としてバッファ *A* へアクセスする。
- バッファ *B* はあるブロック *X* の全ての処理結果を保管する。*WBP* はバッファ *B* の先頭アドレスを持つ。ブロック *X* は *WBP* を介し、書き込み用としてバッファ *B* へアクセスする。バッファ内の構造は、ブロック *X* とブロック *X+1* 間での取決めで、自由に使ってよい。
- ブロック *X* が処理中の時、*BC* は、ブロック *X* の識別子を持つ。
- ブロック *X* の処理が終了すると、次にブロック *X+1* を処理する。このとき *BC* は、ブロック *X+1* の識別子を持ち、*RBP* はバッファ *B* の先頭アドレスに、*WBP* はバッファ *A* に先頭アドレスがそれぞれ更新される。よってブロック *X* に対して書き込み用であったバッファ *B* は、ブロック *X+1* に対する読み込み用バッファとなる。
- ブロック内には、ネストした関数が存在しない。スタックは、ただ 1 つブロックの実行制御のために利用する。そのため 1 つのブロック処理が終了するごとに、スタックはその使用を完了する。

UK では、タスクを構成する要素を MRAM 上に実現する。そのため、ブロック *X* 処理中に計算機が停止後、電源

図 2: 実行ブロック切替えに利用するデータ構造

が復帰し、計算機の動作が再開可能となった時、MRAM に存在する情報からブロック *X* の処理を最初から開始することができる。このとき、ブロック *X* 処理の最初の状態から変化した要素は、*WBP* が示す書き込み用バッファとスタックのみである。よって、この 2 つの要素のみをリセットすればよい。

UK に複数のタスクが存在した場合、必要であればタスク切替えは実行ブロック切替え時に行われる。これによりタスク切替えの際に、スタックを保存する必要がなくなる。

4.2 実行ブロックの切替え処理方法

前節で述べたシステム構成により、ブロック実行中に電源切断が起きたとしても、処理再開が可能となる。しかしながらこのままでは、*RBP*、*WBP* および *BC* の更新中に電源切断が起きた場合に安全にブロック処理が再開されない可能性がある。

UK では安全に実行ブロックの切替えを実現するために、*RBP*、*WBP* および *BC* の他に *RBP'*、*WBP'*、*BC'* と大きさ 1 ビットの *commit* を持つデータ構造体を用意する (図 2)。同時にこの区間は、ハードウェア割り込み禁止とする。

RBP、*WBP*、*RBP'*、*WBP'* にはバッファのアドレスが、*BC* と *BC'* にはブロックの識別子が保存され、*commit* は 0 もしくは 1 を取りうるものとする。バッファ *A* のアドレスを P_A、バッファ *B* のアドレスを P_B とした時、ブロック *X* からブロック *X+1* に更新は、以下の手順により行う (表 1)。

1. *RBP'*、*WBP'*、*BC'* にはそれぞれ *RBP*、*WBP*、*BC* と同じ情報が存在する。
2. *RBP*、*WBP*、*BC* の更新時には、*commit* のビットを 0 とする。
3. *RBP*、*WBP*、*BC* の更新が終了後、*commit* のビットを 1 とする。
4. *RBP'*、*WBP'*、*BC'* の値を、それぞれ *RBP*、*WBP*、*BC* と同値にする。

commit は、*RBP*、*WBP*、*BC* が更新中であるか否かを縮退させた情報である。電源切断から再開した場合には *commit* の結果から、*RBP*、*WBP*、*BC* の更新中に切断したか、*RBP'*、*WBP'*、*BC'* の更新中に切断したかのどちらであるか判断できるため、以下の手順で更新処理を再開する。

- *commit* が 0 であった場合、*RBP*、*WBP*、*BC* が正しく更新されていない可能性がある。そのため表 1 の手順 1 にロールバックし、*RBP*、*WBP*、*BC* 更新をやり直す。

表 1: *RBP*、*WBP*、*BC* の更新手順

手順	*RBP*	*WBP*	*BC*	*RBP'*	*WBP'*	*BC'*	*commit*	電源復帰時の対応
(更新前)	P_A	P_B	*X*	P_A	P_B	*X*	1	—
1	P_A	P_B	*X*	P_A	P_B	*X*	0	—
2	P_B	P_A	*X+1*	P_A	P_B	*X*	0	手順 1 からやり直し
3	P_B	P_A	*X+1*	P_A	P_B	*X*	1	—
4	P_B	P_A	*X+1*	P_B	P_A	*X+1*	1	手順 3 からやり直し
(更新後)	P_B	P_A	*X+1*	P_B	P_A	*X+1*	1	—

- *commit* が 1 であった場合、*RBP'*、*WBP'*、*BC'* が正しく更新されていない可能性がある。そのため、表 1 の手順 3 にロールバックし、*RBP'*、*WBP'*、*BC'* 更新をやり直す。また *RBP'*、*WBP'*、*BC'* がそれぞれ *RBP*、*WBP*、*BC* と比較し同値であった場合、更新が正しく行われているため、ロールバックしない。

このような手順で UK を実現することにより、微小アプライアンスのようなリソースに乏しい計算機においても、突然の電源切断への対応という機能を実現することができる。また UK がカーネルの停止、再開を短時間で実現できるということは、電力供給が断続的である微小アプライアンスにとって、有効であると言える。

5 関連研究

SmartDust[4] などの微小アプライアンスを制御するカーネルとして、TinyOS[5] が存在する。TinyOS は多目的利用用途であるため、プログラム作成時において有用な機能を数多く提供している。UK は利用目的を実環境取得に絞り、微小アプライアンスの限られたリソースを有効利用し、より高精度な実環境情報取得を目指している。特に UK が意識した電源切断時における微小アプライアンス自身の対応は、TinyOS において考慮されていない。

文献 [6] では、不揮発性メモリである FeRAM を利用した Persistent System の実現を行っている。このシステムは FeRAM を利用して、電源切断時におけるシステムの実行状態を復元可能としている。電源切断からの復帰時は、2.4 節で述べたように、任意のタイミングで保存したレジスタ状態とメモリ状態との間で時制がずれてしまう。そのため、レジスタ状態保存時からのメモリ変更操作のログをとり、電源復帰時にそのログをメモリに反映して、レジスタ状態保存時に戻っている。UK でも同様のアプローチをとることはできるが、3 章で述べたように電源復帰時の実行開始遅延を伴ってしまう。

6 おわりに

本稿では、センサから得られた実世界情報を利用して、多種多様なサービスを提供するために必要な微小アプライアンスを制御するためのカーネル UK に置けるタスク制御方式について述べた。

微小アプライアンスは、電源が不安定な環境において、センサを利用したプログラムを断続的に実行し、実環境情報を取得できることが求められる。そこで、UK ではセンサのプログラムの特徴に合わせて、カーネルが管理すべきタスクを複数のブロックに分割する。さらにメモリ情報の更新を縮退させることにより、電源切断からの復帰などによるタスクの中断および再開が、最低限の情報管理で実現できる。

今後は UK の設計をより詳細にした上で、ブロックを生成する言語処理系、プログラム作成支援系、およびハードウェア上による実現とサービス提供を目指し、UK の評価を行う。

参考文献

[1] “MRAM, 256M への確信,” 日経エレクトロニクス, 2003 1-20 (no. 839), pp. 83–105, 2003.

[2] J. M. Rabaey, M. J. Ammer, J. L. da Sliva, Jr., D. Patel, and S. Roundy, “PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking,” IEEE Computer, vol. 33, no. 7, pp. 42–48, 2000.

[3] J. S. Plank, M. Beck, G. Kingsley, and K. Li, “Libckpt: Transparent Checkpointing under UNIX,” Usenix Technical Conference, pp. 213–224, 1995.

[4] B. Warneke, M. Last, B. Liebowitz, and K. S. J. Pister, “Smart Dust: Communicating with a Cubic-Millimeter Computer,” IEEE Computer, vol. 34, no. 1, pp. 44–51, 2001.

[5] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, “System Architecture Directions for Network Sensors,” Proc. 9th Int’l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX), pp. 93–104, 2000.

[6] 大村 廉, 山崎 信行, 安西 祐一郎, “主記憶に不揮発メモリを用いたシステムの実行状態復元手法,” 情処学論, vol. 45, no. SIG1 (ACS4), pp. 88–99, 2004.