動的更新が可能な組み込み OS に関する研究

樋上 真人†　横山 孝典†　志田 晃一郎†　兪 明連†
武蔵工業大学大学院　工学研究科‡

1. はじめに

組み込みシステムの高機能化に伴い，求められる処理も複雑になってきており，組み込みコンピュータに搭載されるソフトウェアの規模が増大している．そのため，製品出荷後にソフトウェアの不具合が見つかる場合がある．

ところが，家電や自動車など，ワンチップマイクロコントローラを用いた組み込みシステムの多くは内蔵された ROM 上でソフトウェアが動作し，従来は書き換え不可能な ROM にソフトウェアが保存されていた．ソフトウェアを修正するには機器を回収し ROM を交換しなければならず時間的，金銭的にコストがかかる作業であった．そのため，機器を回収せずに低コストで更新する方法が求められるようになった．

近年では組み込みコンピュータに，書き換え可能なフラッシュ ROM が用いられるようになり，機器を回収することなく，ネットワークを介してソフトウェアを修正できる可能性がでてきた．そこで，これを利用した新たな組み込みソフトウェアの更新方法の実現が期待されている．

稼働中の機器の組み込みソフトウェアをネットワークを介して更新する手法として RLL(Remote Link Loader) [1]がある．RLL は組み込み OS 上で動作するミドルウェアで，更新対象となるシステム(ターゲット)にアプリケーションプログラムやデバイスドライバといったモジュールを動的に追加，削除，更新することが可能である．しかし RLL では OS 上で動作するソフトウェアの追加や更新は可能であっても，RLL そのものは OS 上で動作する必要があり，OS そのものの更新には対応できず，OS の不具合は修正することができない．

OS 自身を更新可能な手法は提案されているが[2]，RAM 上で動作することを前提としており，ROM 上で動作する組み込みシステムに適用することができない．

そこで，本研究の目的は，内蔵フラッシュ ROM に搭載された組み込み OS の提供する機能(システムコール)をネットワークを介して動的に更新可能とすることである．

An Operating System with Dynamic Update Functions for Embedded Systems
† Masato Hikami, Takanori Yokoyama, Koichiro Shida and Myungryun Yoo
‡ Research Division in Engineering , Graduate School of Musashi Institute of Technology

2. システムの全体構成

RLL で OS の更新ができなかったのは更新システムが OS の機能を使っていたためである．そこで本研究では更新機能を提供するプログラムを OS とは独立して動作可能とすることで更新を実現する．また，組み込み OS の多くはシステムコールは通常の関数呼び出しで実装されるが，OS を更新後にアプリケーションの修正をすることなくシステムコールを呼び出せるようにするために，システムコールを間接呼び出し化する．提案するシステム構成を図 1 に示す．

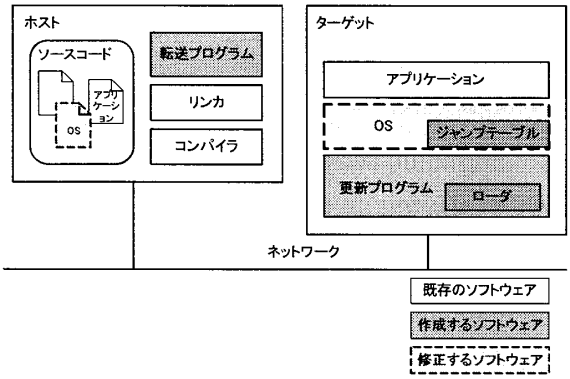

図 1 システム構成

ターゲット側には，間接呼び出し化するためのジャンプテーブル，更新に必要なデータを受け取りフラッシュ ROM に書き込みを行う更新プログラムを搭載する．またホストにはターゲットと通信するための転送プログラムを配置する．

3.ターゲット側の構成と機能

3.1 更新プログラム

更新データをネットワーク経由で受け取り，フラッシュ ROM に書き込みを行うプログラムである．ターゲット上のアプリケーションから更新プログラムの処理を呼び出すためのインターフェースであるメタシステムコールを提供する．このメタシステムコールの実体は更新プログラム内に実装し，OS の更新に関する手続きはメタシステムコールを用いてのみ行われるようにする．

更新プログラムを用いた更新の処理の流れを図 2 に示す．

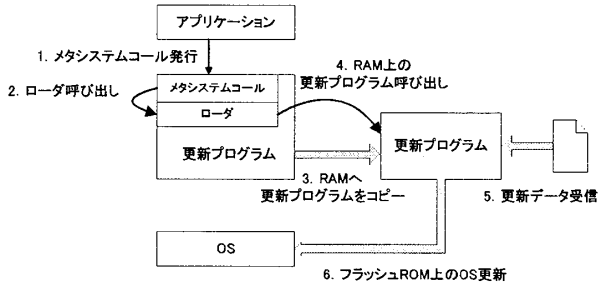

図2　ターゲット側の更新処理の流れ

アプリケーションからメタシステムコールが発行されると，更新プログラムのローダ機能により，フラッシュ ROM から RAM へ更新プログラム自身をコピーし，RAM 上で更新プログラムを実行させる．これは OS の更新のためにフラッシュ ROM の書き換えが生じるが，フラッシュ ROM に書き込みを行っている時にはフラッシュ ROM 上でプログラムを動作させることができないためである．RAM で動作する更新プログラムは，ネットワーク経由で更新に必要なデータを受信し，フラッシュ ROM に書き込みを行うことで更新を実現する．

3.2 システムコールの間接呼び出し化

OS の更新を行うとシステムコールのエントリポイントが変動し，アプリケーションからそのままでは更新後のシステムコールを呼び出せなくなる可能性がある．そこでジャンプテーブルを用いてシステムコールを間接呼び出し化する．システムコールの呼び出しにジャンプテーブルを用いた例を図 3 に示す．

システムコールを実行するときにはシステムコール実体ではなく，同名のラベルのジャンプテーブルへ飛び，テーブルに記述されているシステムコールのアドレスへジャンプする．システムコールの更新によりシステムコールのエントリポイントが変動しても，システムコール更新時にジャンプテーブルに記述されているシステムコールの実体のアドレスも書き換えることで，アプリケーションを更新することなく，更新後のシステムコールを呼び出すことができる．

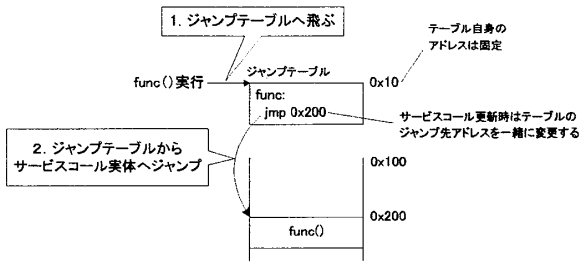

図3　ジャンプテーブルによる間接呼び出し

4. ホスト側の構成と機能

ホストには，ターゲットと通信し更新データを送信するための転送プログラムを用意する．

5. 更新手順

本提案手法を用いた実際の更新手順を示す．

1. ホスト側でソースコードのコンパイル
2. メタシステムコールを発行
3. 更新プログラムを RAM へコピーし実行
4. 更新プログラムがホストに更新要求
5. 転送プログラムがターゲットにデータ転送
6. 受け取ったデータをフラッシュ ROM に焼きこむ
7. ターゲット機器の再起動

6. 実装と動作実験

本研究では実験環境として，以下の環境を用いた．

OS：TOPPERS/JSP カーネル Release 1.4.3
ターゲット機器：
CPU：H8/3069F 20MHz
内蔵フラッシュ ROM：512KB
内蔵 RAM：16KB

OS コンパイル後のシステムコールのサイズは 31348byte，ジャンプテーブルは 648byte であった．通信インターフェースに RS-232C(38400bps)を用いて OS の更新を行ったところ，121 秒で更新が完了した(ソースコードのコンパイルにかかる時間は含まず)．

7. おわりに

本研究ではネットワークを介した動的な OS の更新ソフトウェアを作成し，OS の更新が行えたことを確認した．今後の課題として，更新後に再起動せずに更新前の動作に復旧する機能の実装があげられる．

参考文献

[1] Remote Link Loader
http://www.toppers.jp/rll-download.html

[2]Y.Yokote, “The Apertos Reflective Operating System: The Concept and Its Implementation”, Proc. The Conference on Object-Oriented Programming Systems, Languages, and Applications ’92, pp. 414-424, 1992.