3G-05

bsdiffを応用したECUソフトウェア高速ダウンロード

小沼　寛†　　野澤　優尚†　　清原　良三†
神奈川工科大学情報学部情報工学科†

1 はじめに

ADAS の搭載やコネクテッドカーの登場など自動車の電子化，高度化は著しく進んでいる．自動車は以前よりも多くの電子制御を行うようになり，現在では，自動車一台に搭載される ECU の数は 70 個以上にものぼる場合がある[1]．また，システムの高機能，高精度化により ECU に組み込まれるソフトウェアは大規模，複雑化している．結果，ECU ソフトウェアの品質管理が困難となり，出荷後に不具合が発覚し修正が必要となる場合がある．問題が発覚した際に修正や更新を適用する方法として，ディーラなどに持ち込み対応してもらう方法と，更新プログラムをダウンロードし，ユーザ自身でインストールする方法がある．前者の方法による更新が主であるが，コネクテッドカーの普及により自動車がネットワークに接続されるようになった場合，後者の方法による更新が増加すると考えられる．

更新中は自動車のエンジンをかけたままにしなければならないが，運転をすることはできない．しかし，自動車から離れることは安全上好ましくない．ゆえに，ユーザは更新中に自動車の傍にいることを余儀なくされる．そのため，更新時間を短縮することが望まれる．

ECU の更新は車載ネットワークを介して行われる．車載ネットワーク規格の業界標準となっている CAN の最大通信速度は 1Mbps，一度に送信できるデータ量は最大 8 バイトとなっており[2]，ブロードバンド回線などに比べ通信速度が低く，帯域も狭いため大量のデータを送信するには時間がかかってしまう．よって，更新に必要なデータ量を削減することが重要と考えられる．

そこで，本論文では ECU ソフトウェア更新に必要なデータ量を削減することで，ダウンロード時間を短縮する手法を提案する．

2 関連研究

ソフトウェア更新システムを考えるにあたり重要事項となるのが，対象となる製品・サービスの更新中の停止時間である．停止時間となる要因として，更新に必要なデータをダウンロードする時間と，フラッシュメモリを書き換える時間がある．通信するデータ量を削減し，ダウンロード時間を短縮する有効な手法として差分圧縮が挙げられ，差分圧縮を効果的に行う研究がされている[3]．また，差分圧縮を車載ソフトウェアに適用する研究がされている[4]．この研究では，車載機器の，少ないメモリ制約下でも差分による更新が行えるようにメモリ消費量を抑える方式の提案がされている．フラッシュメモリの書き換え時間を短縮する研究がされている[5]．この研究では，ソフトウェアの構造に着目し，フラッシュメモリの書き換えが必要な部分を局所的にすることで，書き換えに必要な時間を短縮している．

本論文では既存の差分圧縮手法を改良し，更新に必要なデータ量を削減することでダウンロード時間の短縮を行う．既存の差分圧縮手法を改良し，ソフトウェアの更新時間を短縮する提案がされている[6]．この研究では汎用的な差分圧縮手法を固定長命令用に改良を行うことと，レジスタアサインのずれを考慮することで差分情報の削減を行っている．

3 差分圧縮

差分圧縮とはデータの転送や保存などを完全な状態ではなく，バージョン間での差分情報を用いることで，ソフトウェアの更新に必要なデータ量を削減する手法である．

ソースコード上で変更があった場合，バイナリコード上では，アドレスの参照先などが変わることにより，ソースコード上で変更が行われてない部分にも差分が生じる．そのため，ソースコード上の変更点から，バイナリコード上の差分を見つけ出すことは困難である．差分の抽出には様々なアルゴリズムが考案されている．

本論文では差分圧縮に bsdiff[7]を用いる．bsdiff による差分情報は，旧版のバイナリコードとの差と，新規に追加されたデータに分けられる．bsdiff は他の差分圧縮プログラムに比べ，高い圧縮率となっている．

Fast Downloading for ECU Software Based on Bsdiff
Yutaka Onuma, Nozawa Masanao, Kiyohara Ryouzou
Kanagawa Institute of Technology

4 提案手法

本論文ではサブルーチンの呼び出し規約に着目をした．プログラム中で，関数コールが行われる際に表 1，表 2 に示すような一定の処理が行われる．これらは関数毎にほぼ同一の処理が行われるが，局所変数の有無などにより即値が異なる．そこで，差分を抽出する前に，即値だけを保持し，一定の処理を架空の命令に置き換えることで命令量を減らすことができる．32 ビット ARM アーキテクチャを使用した例を図 1 に示す．命令量が減りバイナリコードが小さくなれば，同様に差分情報も小さくなると考えられる．

また，差分適用先で元の命令に戻すため，どのような置換を行ったかの情報が必要になる．そのため，置換前後のバイナリと即値の位置を記録するためのオーバーヘッドがかかる．

5 評価・考察

5.1 評価

標準の bsdiff と提案手法を用いた場合で比較し，評価を行った．提案手法では関数が呼び出されたときに実行される一定の処理を置換対象とした．評価には 32 ビット固定長 ARM のバイナリコードを用いた．それぞれで生成された差分情報のサイズを表 3 に示す．表 3 の結果から，提案手法により 1%弱のサイズ削減の効果が得られたことがわかる．

表 1 関数呼び出し

アセンブラ		
1:	push	{fp, lr}
2:	add	fp, sp, #imm
3:	sub	sp, sp, #imm

(imm:即値)

表 1 関数抜け出し

アセンブラ		
1:	sub	sp, fp, #imm
2:	pop	{fp, pc}

(imm:即値)

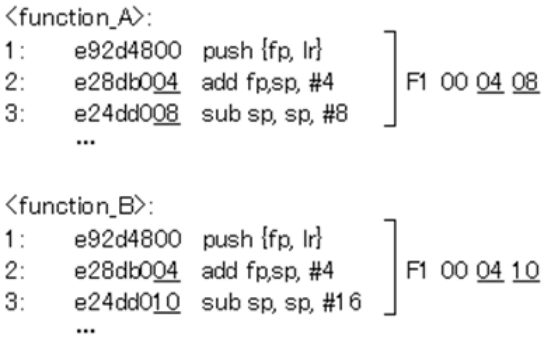

図 1 置換例

表 3 差分情報サイズ

	A1 -> A2	B1 -> B2
bsdiff	3, 464	5, 252
提案手法	3, 440	5, 204
サイズ比	99.3%	99.1%

5.2 考察

結果からバイナリコードのサイズを小さくすることで，差分情報を減らすことが可能であることがわかった．提案手法では関数コールに着目をしたが，サブルーチンの呼び出しに限らず，バイナリコード上から一定の処理が行われている部分を抽出し，より少ない命令で置き換えることでもサイズの削減を行うことができると考えられる．また，置換を行う一連の処理の命令量や，置換を行う回数を増やすことにより，更なるデータ量の削減が可能であると考えられる．

提案手法により差分情報のデータ量を削減することができた．しかし，提案手法を用いた場合，命令を復元するための処理時間が余計にかかってしまう．復元時間は，実装の仕方，復元を行う回数により左右されると考えられる．ゆえに，復元方法の最適化，復元を行う回数と処理時間とのトレードオフを考慮する必要がある．

6 まとめ

本論文では，複数の命令からなる一定の処理を一つの命令に置き換えることで，バイナリコードを小さくし，差分更新に必要なデータ量を削減する手法を提案し，評価した．結果，1%弱の効果を確認することができた．

今後の課題として，考察で述べたような処理の効率化が挙げられる．また，提案手法の有効性の確認のため，実環境に近い状態でのダウンロード時間への影響を調査する．

参考文献

[1] 堀川健一，目崎元太，渡辺章代，加櫓武，松本達治：自動車ソフトウェアの標準仕様”AUTOSA”の評価
[2] 佐藤道夫:車載ネットワーク・システム徹底解説
[3] 清原良三，三井聡，木野茂徳：組込みソフトウェア向けバイナリー差分抽出方式
[4] 施欣漢，中西恒夫，久住憲嗣，福田晃：放送による車載情報機器向けソフトウェア差分更新方式
[5] 清原良三，栗原まり子，古宮章裕，高橋清，橘高大造：携帯電話ソフトウェアの更新方式
[6] 野澤優尚，小沼寛，清原良三：車載 ECU 向けソフト更新のためのデータ圧縮方式
[7] http://www. daemonology. net/bsdiff/