Vol.2017-DPS-172 No.25
Vol.2017-SPT-26 No.25
Vol.2017-EIP-78 No.25
2017/11/29

# 車載ECU向けソフトウェア更新システム

小沼 寛[1]　寺島 美昭[3]　清原 良三[2]

**概要**：自動車に搭載される Electronic Control Unit(ECU) の数は，自動運転機能や Advanced Driver Assistance System(ADAS) などの高度システムの搭載により増加している．更に，自動車内のソフトウェア規模は大規模化している．結果，ソフトウェアに起因するリコール件数が増加している．また，コネクテッドカーの普及により，自動車がサイバー攻撃の対象となる可能性がある．一方で，自動車はスマートフォンのアプリケーションダウンロードサービスのように，必要動作環境を満たしていれば，販売後でも新しい機能の追加を行うことが可能となり，ユーザの快適性や安全性の向上が期待される．自動車は，バグや脆弱性の修正，車載システムのアップグレードのために，ECU ソフトウェアを更新する必要がある．本論文では，車載 ECU ソフトウェアアップデートプロトコルを提案し，更新時間について評価する．

## 1. はじめに

Advanced Driver Assistance System(ADAS) や自動運転機能などの搭載により，自動車に搭載される Electronic Control Unit(ECU) の数が増加している．自動車内のソフトウェアは大規模化かつ複雑化している．結果，ソフトウェアに起因するリコール件数が増加している [1]．特に，Advanced Safety Vehicle(ASV) 技術に関するリコール件数は増加傾向にある．ASV などの安全に関わる部分の不具合は，事故に繋がる恐れがあるため，早急な修正が重要となる．

自動運転車両の実用化を目指し，自動車メーカや IT 企業が中心となり開発が進められている．自動運転技術は，車両に取り付けられた各種センサー，車車間や路車間の通信を用いることで実現される．

車両間で通信を行う際に，共通のプロトコルを使用できなければ，正常な通信を確立できない．自動運転車両が普及した場合，様々なメーカの車種間で通信が行われる．また，自動運転技術の開発が進むにつれ，より高精度の自動運転を実現するために，車車間の通信プロトコルが変更されることが考えられる．また，通信プロトコルが変更された場合，古いプロトコルを使用している車両と新しいプロトコルを使用している車両間の通信では，望んだ挙動を得られないことが考えられる．そのため，古いプロトコルを使用している車両に対して，ソフトウェアアップデートを行うことで，新しいプロトコルに対応させることが可能となる．

また，インターネットへの接続機能を備えたコネクテッドカーが登場している．自動車がインターネットに接続することにより，センサにより取得した車両状態や道路状況などのデータをクラウド上で集積し分析することで，ユーザは様々なサービスを受けることが可能となる．欧州では，2018 年 4 月から，新型車への緊急通報システムの搭載を義務付けられている [2]．また，自動車はスマートフォンのアプリケーションダウンロードサービスのように，必要動作環境を満たしていれば，販売後でも新しい機能の追加を行うことが可能となり，ユーザの快適性や安全性の向上が期待される．これらのことから，今後，コネクテッドカーの普及が見込まれる．一方で，インターネットに接続することにより，自動車がサイバー攻撃の対象となる恐れがある．実際に，無線インターフェースを持つ自動車の脆弱性を突くことで，自動車の遠隔操作が可能となり，大規模なリコールに発展した例がある [3]．

不具合修正や新機能の追加により，ECU ソフトウェアアップデートの重要性が高まる．更に，セキュリティリスクへ対応するため，従来に比べアップデートの頻度が高まる．

ECU ソフトウェアのアップデート方法は二種類挙げられる．一つは従来行われている，ユーザがディーラに自動車を持ち込み，専用の診断機器を用いる方法である．従来の方法では，ソフトウェアアップデートのためにユーザはディーラまで自動車を持ち込まなければならない．アップ

[1] 神奈川工科大学大学院
Graduate School of Kanagawa Institute of Technology
[2] 神奈川工科大学
Kanagawa Institute of Technology
[3] 創価大学
Soka University

Vol.2017-DPS-172 No.25
Vol.2017-SPT-26 No.25
Vol.2017-EIP-78 No.25
2017/11/29

デートの頻度が高まった場合，従来のアップデート方法は，ユーザにとって大きな負担となりかねない．また，アップデートには専用の診断機器や駐車スペースが必要となるため，一度にアップデート可能な車両数に制限がある．このような制限は，アップデートがリリースされた直後や製造ラインでアップデートを行う際に支障となり，アップデート時間を大きく増加させる可能性がある．

一方で，Over The Air(OTA) による ECU ソフトウェアアップデートが研究されている [4][5][6]．今後，コネクテッドカーが普及することで OTA によるアップデートが可能となり，インターネットに接続できる環境化であればアップデートが可能となる．OTA によるアップデートでは，ユーザはソフトウェアアップデートのために自動車をディーラまで持ち込む必要がなくなる．そのため，ユーザは即座にアップデートを受け取ることが可能となり，ユーザのソフトウェアアップデートに対する負担を軽減することができる．また，無線通信により同時に複数の車両に対してアップデートが可能であり，専用の診断機器や駐車スペースを必要としないため，制限を受けずにアップデートを実行できる．

ECU ソフトウェアアップデートでは以下の要件が求められる．

(1) ソフトウェアアップデート時間の短縮
(2) ソフトウェアアップデートの完全性保証
(3) ソフトウェアアップデートのセキュリティ

ECU ソフトウェアのアップテート中に，ユーザが自動車を使用することは危険である．よって，ソフトウェアアップデート中は自動車を駐車していることが前提となる．そのため，ユーザはソフトウェアアップデートの間に自動車を使用することがでいないため，ソフトウェアアップデート時間は可能な限り短いことが望まれる．

自動車という環境上，ソフトウェアの不具合や誤りが事故を誘発する可能性がある．よって，更新データの欠落や改ざん，誤った更新データの適用は重大な事故に繋がる恐れがある．そのため，更新データの欠落や改ざんを防ぎ，ECU ソフトウェアのアップデートが正しく完了したことを保証することが重要となる．

コネクテッドカーの普及により利便性が高まる一方，セキュリティリスクが増大することが考えられる．無線ネットワークを介したソフトウェアアップデートでは，転送する更新データの盗聴や改ざん，不正データの転送といった攻撃を受ける可能性がある．これらの攻撃は，ユーザのプライバシーの侵害や，車両と周囲を危険な状態に陥れる．そのため，外部からの攻撃に対して，ユーザと自動車を保護するためのセキュリティが必要となる．

本論文では，ソフトウェアアップデート時間に焦点を当て，要求を満たした ECU ソフトウェアアップデートプロトコルを提案し，提案プロトコルを用いた場合のソフトウェアアップデート時間について評価を行う．

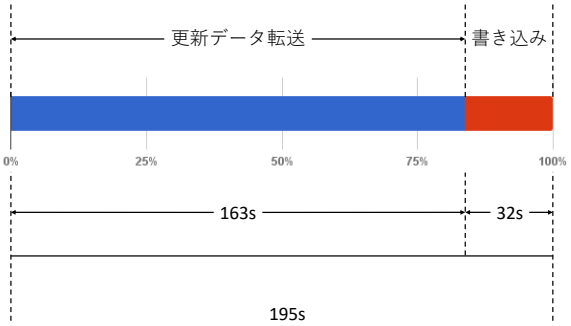


**図 1** ソフトウェアアップデート時間内訳例

**表 1** 主な車載ネットワーク規格

| | Bit rate | Payload | Use |
|---|---|---|---|
| CAN | 10 Kbps - 1 Mbps | 0 - 8 byte | Body<br>Powertrain |
| LIN | 20 Kbps | 8 byte | Body |
| FlexRay | 10 Mbps | 254 byte | Powertrain<br>Safety |
| Ethernet | 100 Mbps | 46 - 1500 byte | Backbone<br>Infotainment |

## 2. 関連研究

ECU ソフトウェアアップデートの要件を満たすため，様々な研究が行われている．ソフトウェアアップデート時間を短縮するアプローチとして，転送データを削減する手法と，更新データの転送を分散させる手法が研究されている．

ECU ソフトウェアアップデート時間の内訳例を図 1 に示す．ECU ソフトウェアアップデート時間は，更新データの転送時間と，フラッシュメモリの書き換え時間に大きく分けられる．

サーバから受信，または診断機器により車両まで転送された更新データは，車載ネットワークを介して対象となる各 ECU へと転送される．車載ネットワークで使用されている主な規格を，表 1 に示す．現在，車載ネットワークの標準規格として Controller Area Network(CAN)[7] が使用されている．CAN の最大ビットレートは 1Mbps，最大ペイロードは 8byte であり，昨今のモバイルネットワークの通信速度に比べ低速となっている．そのため，OTA によるソフトウェアアップデートを行う場合，CAN 上の更新データの転送時間がボトルネックとなり，ソフトウェアアップデート時間の大部分を占める．

更新データの転送時間は，更新データのサイズに依存する．そのため，更新データの転送時間を短縮するためには，CAN 上を通る更新データサイズを小さくすることが重要

Vol.2017-DPS-172 No.25
Vol.2017-SPT-26 No.25
Vol.2017-EIP-78 No.25
2017/11/29

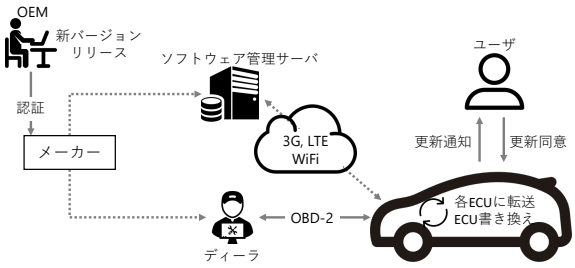


図 2　ECU ソフトウェアアップデートシステム概観

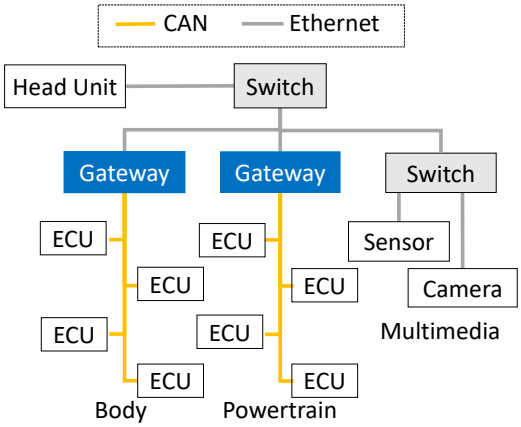


図 3　車載ネットワーク構成図

となる．

転送すべき更新データサイズ削減のために，差分更新による転送データサイズの削減が研究されている [8][9][10]．これらの研究では，汎用的な差分更新ツールである bsdiff[11] を車載 ECU ソフトウェア向けに改良を行っている．また，bsdiff をインプレース型差分更新に改良することで，差分更新時に必要なメモリ使用量の削減と，ECU の不安定な電源環境化においても差分更新を可能とする研究がされている [12]．[13] では，bsdiff を ECU の小さい RAM 上でも実行できるように改良し，差分更新を車載 ECU ソフトウェア更新に適用することで，更新時間を 90%削減できることを示している．

[14] では，複数の ECU の更新を実行する際に，異なるバス上に存在する更新対象 ECU に対して並列に更新を行うことで，更新時間全体を短縮する手法を提案している．

また，自動車セキュリティに関する分析は様々行われている [15][16]．[17] では，既存の手法では OTA によるソフトウェアアップデートに必要なセキュリティ要件をすべて満たせないことを指摘し，自動車に OTA ソフトウェアアップデートを適用するために包括的なセキュリティコンセプトを提案している．

## 3. ECU ソフトウェア更新プロトコル

### 3.1 想定環境

#### 3.1.1 ネットワーク

図 2 に提案する ECU ソフトウェアアップデートシステムの概観を示す．ECU ソフトウェアの更新は，更新データを管理するサーバ，Telematics Communication Unit(TCU) と各更新対象 ECU 間での通信となる．

サーバから TCU までの更新データの転送は，3G や LTE などのモバイルネットワークを介して転送される．また，TCU から各更新対象 ECU へは車載ネットワークを介して転送される．

車載システムの高度化に伴う通信量増加により，車載ネットワークの標準とされている CAN では帯域不足になりつつある．そこで，車載ネットワークに Ethernet が導入されている．Ethernet を導入することで，CAN に比べ高速で柔軟な車載ネットワークを実現することができる．しかし，従来の CAN を用いたシステムの全てを同時に Ethernet へ置き換えることは難しい．そのため，車載ネットワークの一部には従来通り CAN が使用されることが想定される．

想定される車載ネットワークを図 3 に示す．想定する車載ネットワークでは，バックボーンや通信量の多い部分に Ethernet を用い，それ以外のブンブンは従来通り CAN を使用する．このようなネットワークで ECU ソフトウェアアップデートを行う場合，CAN のビットレートが低いため，CAN 上の ECU に更新データを転送する時間がボトルネックとなる．

#### 3.1.2 更新データ

想定する車載ネットワークの一部に低速な CAN を使用するため，サイズの大きい更新データを転送する場合，多くの時間を要する．転送すべき更新データサイズを削減するために，差分更新を適用する．

差分更新は，図 4 に示すように，ソフトウェア全体ではなくソフトウェアの新旧のバージョン間の差分のみを用いて更新を行うことで，転送すべきデータ量を削減している．そのため，更新を適用するためには，転送先の ECU 内の旧版のソフトウェアに受信した差分を適用する必要がある．

従来の ECU の多くは，NOR 型のフラッシュメモリを採用している．NOR 型フラッシュメモリではメモリの書き換えを行う際，イレーズブロック単位で書き換えを行う．そのため，差分更新を行う場合，少なくともイレーズブロック分のデータと差分データを RAM 上に展開する必要がある．しかし，一部の ECU では RAM のサイズが小さく，イレーズブロックのデータと差分データを RAM 上に展開することができず，差分更新を適用できない場合がある．よって，対象 ECU が差分更新を適用可能であるかを判断し，適当な更新データを用いることが重要となる．

Vol.2017-DPS-172 No.25
Vol.2017-SPT-26 No.25
Vol.2017-EIP-78 No.25
2017/11/29

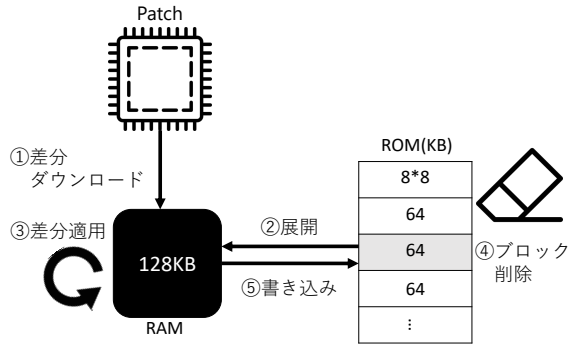


図 4　差分更新概要

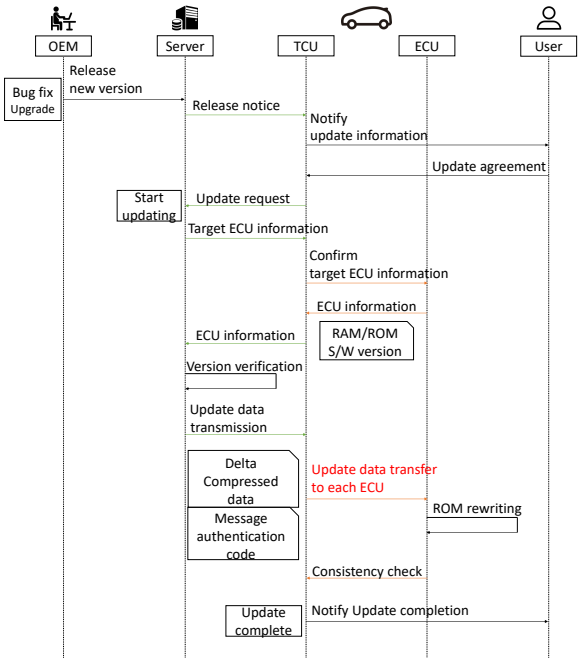


図 5　ECU ソフトウェアアップデートプロトコル

### 3.2　更新手順

図 5 に提案する ECU ソフトウェアアップデートプロトコルの流れを示す．

不具合や脆弱性の発見，新機能の追加などが行われる場合，自動車メーカあるいは ECU サプライヤが新しいバージョンのソフトウェアをリリースする．新しいソフトウェアバージョンの情報は，対象となる ECU を持つ各車両に通知され，ユーザの同意のもと更新が開始される．

更新が開始され，各車両に更新対象となる ECU のリストが送信される．受け取ったリストを基に，車両内に存在する各 ECU の RAM，ROM，インストールされているソフトウェアのバージョン情報を返す．

サーバ側は，返された車両内に存在する ECU の情報から，差分更新可能であるかを判断し，対象 ECU に適当な更新データを無線ネットワークを介して TCU へ転送する．TCU で受け取った更新データは，車載ネットワークを介して各 TCU へ転送される．各 ECU は受信した更新データを適用し，ROM へと書き込む．

正しく更新が完了されたことを確認した後，更新が完了したことをユーザへ通知し，ECU ソフトウェアアップデートを完了とする．

### 3.3　更新時間

#### 3.3.1　計算方法

本論文では，ECU ソフトウェアアップデートに要する時間に関して評価する．ソフトウェアアップデート時間を評価するに当たり，ソフトウェアアップデートプロトコルを 3 つのセクションに分ける．

第一段階を，新しいバージョンのリリースからユーザへの更新通知までとする．第一段階については，ユーザが待機しなければならない時間に影響を与えないため，本評価では考慮しない．

第二段階を，更新の開始から更新データを TCU に転送するまでとする．第二段階については，TCU などで更新データをバッファリングすることが可能であれば，バックグラウンドで更新データをダウンロードしておくことでソフトウェアアップデート時間から無視することができる．

第二段階では，自動車内の更新対象 ECU の情報の確認と，その情報から更新データを生成し，TCU まで更新データを転送する．ECU からの各情報の取得を CAN の 1 フレームで行えると仮定する．CAN のフレームサイズを $F_{size}$，更新対象 ECU 数を $N$，CAN ビットレートを $CAN_{bitrate}$ としたとき，ECU 情報取得のための CAN 上の通信時間 $T_1$ は以下の式で表される．

$$T_1 = \frac{F_{size} * N}{CAN_{bitrate}} \tag{1}$$

次に，取得した ECU の情報から更新データを生成し，モバイルネットワークを用いて TCU へ更新データを転送する．更新データサイズを $UPDATE_{size}$，モバイルネットワークの通信速度を $WN_{speed}$ としたとき，サーバから TCU までの更新データ転送時間 $T_2$ は以下の式で表される．

$$T_2 = \frac{UPDATE_{size}}{WN_{speed}} \tag{2}$$

第三段階を，第二段階以降の処理とし，これらの処理はソフトウェアアップデート時間に必ず含まれるものとなる．

第三段階では，車載ネットワークを介して TCU から各 ECU へ更新データを転送する．CAN を介しデータを転送する際に，メッセージ認証のために Message Authentication Code(MAC) をメッセージに付加する方式が AUTOSAR[18] で検討されている．MAC を用いた認証は，メッセージの改ざんやなりすましを防ぐことが可能となる．しかし，MAC を付加する場合，MAC 生成処理や認証処理が必要となる．また，CAN メッセージのペイロードは最

Vol.2017-DPS-172 No.25
Vol.2017-SPT-26 No.25
Vol.2017-EIP-78 No.25
2017/11/29

大 8byte であり，MAC のために数バイトを使用すると転送すべきデータ量が増加する．ここで，MAC によるオーバーヘッドを $MAC_{size}$ とし，CAN メッセージのペイロードを $CAN_{payload}$，$N$ 個の ECU を更新するとき，CAN 上の更新データ転送時間 $T_3$ は以下の式で表される．

$$T_3 = \frac{UPDATE_{size} + \frac{UPDATE_{size}}{CAN_{payload} - MAC_{size}}}{CAN_{bitrate}} * N \quad (3)$$

### 3.3.2 目標時間

ECU ソフトウェアアップデート中は，ユーザは自動車の運転を行えないため，アップデート時間は可能な限り短いことが望まれる．

ここで，ソフトウェアアップデート時間の目標値として Electric Vehicle(EV) の充電時間を設定する．現在，欧州を始めとし，将来的にディーゼル車やガソリン車の販売を終了する方針の発表や，EV の普及を促す取り組みを行っている．そのため，今後 EV が普及することが予想される．EV では，自動車の走行のためには充電が必要となる．EV の充電中は，ガソリン車の給油と同様に自動車を走行させることはできない．よって，EV の充電中にソフトウェアアップデートを行うことで，ユーザに不要なソフトウェアアップデート中の待機時間を減らすことができる．

EV の充電時間は充電方法によって大きく異なる．充電方法は，普通充電と急速充電の二種類が存在する．普通充電は，主に一般家庭などに設置可能な普通充電設備を用いて充電を行う．急速充電は，高速道路のサービスエリアなど短時間での充電が求められる場所に設置されている急速充電設備を用いて充電を行う．各充電方法の充電目安時間は，普通充電では充電完了までに約 6 時間程度，急速充電では 80%までの充電で約 30 分程度とされている．しかし，ユーザは必ずしも目安値まで充電するわけではなく，利用状況に応じて充電を行うと考えられる．例として，長距離移動などで充電設備の少ない地域に向かう場合は，多くの充電が必要となる．一方で，充電設備の多い地域や短距離の移動であれば，少ない充電でも移動が可能である．そのため，実際に充電に所要する時間は，充電目安時間よりも短くなると考えられる．そこで，ソフトウェアアップデートの目標時間には，急速充電の平均目安時間の半分である 15 分と設定する．

## 4. 評価

### 4.1 アップデート例

ECU ソフトウェアアップデート時間について評価を行う．表 2 に想定する環境を示す．アップテート例として，リアルタイムカーネル TOPPERS/APS[19] のバージョン 1.7.0 から 1.9.0 へのアップデートを想定する．

一つの車載システムは複数の ECU から成り立っている．そのため，車載システムを更新する場合，同時に複数の ECU を更新する必要がある．図 6 に，更新対象 ECU 秒毎

表 2 評価環境

| Item | Value |
|---|---|
| Size of CAN frame | 111 bit |
| CAN bit rate | 500 kbps |
| Overhead | 1 byte |
| Mobile network bit rate | 2000 kbps |
| Number of target ECUs | 10 ～ |
| Update data size | 221,303 byte |
| Delta data size | 21,798 byte |

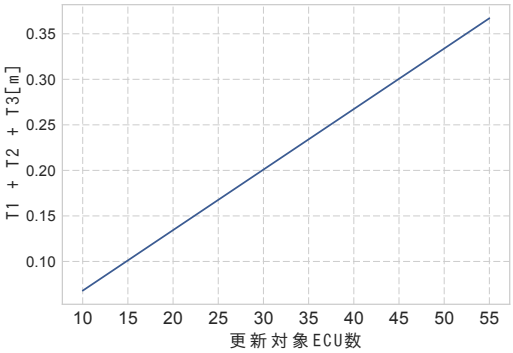


図 6 ECU 数毎の転送時間

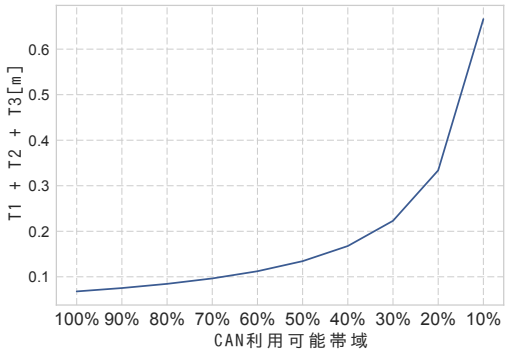


図 7 CAN 利用可能帯域毎の転送時間

の $T_{1_3}$ の合計値を示す．図 6 の結果から，設定した環境下においてソフトウェアアップデートのために更新時間を要さないことがわかる．

しかし，CAN 上に更新データを転送する際，更新データ以外のメッセージが流れていることが想定される．そのため，更新データ転送のために CAN の全ての帯域を利用することは難しい．よって，更新データ転送のために利用できる CAN の帯域は制限されることが想定される．図 7 に CAN の利用可能帯域毎の $T_{1_3}$ の合計値を示す．図 7 の結果より，CAN の帯域が制限される場合，大幅に転送時間が増加することがわかる．

また，ECU の中には RAM とフラッシュメモリのイレーズブロックの関係から，差分更新を適用できないものが存在する．差分更新では，差分情報のみの転送でソフトウェアアップデートを実行できるが，差分更新を適用できない

Vol.2017-DPS-172 No.25
Vol.2017-SPT-26 No.25
Vol.2017-EIP-78 No.25
2017/11/29

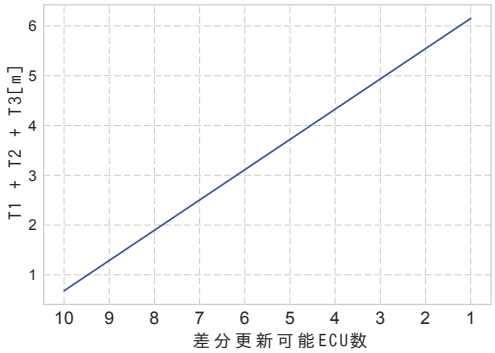


**図 8** 差分更新可能 ECU 数毎の転送時間

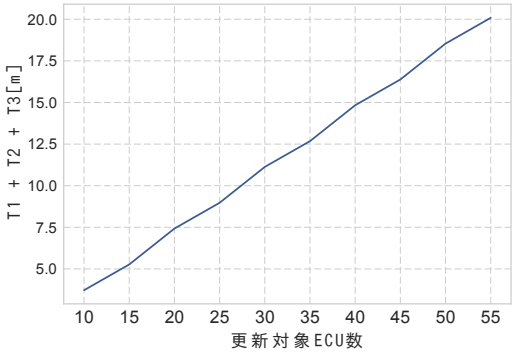


**図 9** ECU 数増加の転送時間への影響

**表 3** T の各値

| Item | Time[s] |
|---|---|
| T1 | 0.022 |
| T2 | 0.885 |
| T3 | 222.264 |

場合，ソフトウェア全体を転送しなければならないため転送すべきデータ量が大幅に増加する．図 8 に，差分更新が適用できない ECU が存在する場合の影響を示す．また，CAN の利用可能帯域を 10%とする．図 8 より，差分更新可能 ECU 数が減るごとに転送時間が大幅に増加していることがわかる．

図 9 に，図 8 から更新対象 ECU 数を増加させた場合の更新時間を示す．図 9 では，更新対象 ECU の半数を差分更新可能としている．この場合，更新対象 ECU 数が 45 個の時点で目標時間とする 15 分を超えている．

図 9 の差分更新可能 ECU 数が 5 個のときの $T_{1-3}$ の各値を表 3 に示す．表 3 から，更新時間の多くが車両内での処理の $T_3$ で要していることがわかる．

### 4.2 考察

本評価では，ECU ソフトウェアアップデート時間の大部分を占める更新データの転送時間のみに焦点を置いている．しかし，実際のソフトウェアアップデート時間には，差分データの適用や ROM への書き込み，認証処理などの処理時間が含まれる．また，本評価では CAN メッセージ認証のためのオーバーヘッドを 1byte と仮定したが，メッセージの重要性によってはより大きいオーバーヘッドが付加される．そのため，実際のソフトウェアアップデート時間は，評価結果よりも増加すると考えられる．

また，本評価では，差分更新を適用できない場合に，ソフトウェア全体を送信している．しかし，差分更新よりも RAM の使用量が少ない圧縮を用いることで，転送すべき更新データサイズを削減することができる．そのため，ECU の限られた RAM 上で展開可能な圧縮方式が必要となる．

## 5. まとめ

本論文では，車載 ECU ソフトウェアアップデートプロトコルを提案し，提案プロトコルで要する更新時間について評価を行った．評価の結果，最適な環境下であればソフトウェアアップデートがユーザに影響を与えることがないことを示した．一方で，環境によっては目標時間を上回り，ユーザをソフトウェアアップデートのために待たせる可能性があることを示した．

今後の課題として，ECU ソフトウェアアップデートに向けた，限られた RAM 上でも実行可能な圧縮方式について検討する．また，将来，車載ネットワークが Ethernet などの高速なネットワークに移行することにより，更新データ転送時間の占める割合が減少し，書き込み時間が占める割合が増加する．更に，車載システムの規模が大きくなるに伴い，更新データサイズや ROM への書き込みに要する時間が増加する．そのため，更新データ転送時間以外の ECU ソフトウェアアップデート時間全体について評価を行う．

### 参考文献

[1] 国土交通省　自動車局，平成 27 年度　リコール届出内容の分析結果について，http://www.mlit.go.jp/jidosha/carinf/rcl/common/data/h27recallbunseki.pdf （参照 2017-10-31）．
[2] European Commission, eCall in all new cars from April 2018, https://ec.europa.eu/digital-single-market/news/ecall-all-new-cars-april-2018 （参照 2017-10-31）．
[3] FCA US Media website, http://media.fcanorthamerica.com/newsrelease.do?id=16849 （参照 2017-05-09）．
[4] D. K. Nilsson, L. Sun and T. Nakajima, *A Framework for Self-Verification of Firmware Updates over the Air in Vehicle ECUs*, 2008 IEEE Globecom Workshops, New Orleans, LO, 2008, pp. 1-5.
[5] G. de Boer, P. Engel and W. Praefcke, *Generic Remote Software Update for Vehicle ECUs Using a Telematics Device as a Gateway*, Advanced Microsystems for Automotive Applications 2005, pp 371-380.
[6] R. Miucic and S. M. Mahmud, *Wireless Reprogramming of Vehicle Electronic Control Units*, 2008 5th IEEE Consumer Communications and Networking Conference, Las Vegas, NV, 2008, pp. 754-755.
[7] Bosch CAN 2.0 Specification, http://esd.cs.ucr.edu/

webres/can20.pdf （参照 2017-10-31）.
[8] Y. Onuma, M. Nozawa, Y. Terashima and R. Kiyohara, *Improved Software Updating for Automotive ECUs - Code Compression* -, The 4th IEEE International Workshop on Consumer Devices and Systems conjunction with COMPSAC 2016.
[9] H. Teraoka, F. Nakahara and K. Kurosawa, *Incremental update method for resource-constrained in-vehicle ECUs*, 2016 IEEE 5th Global Conference on Consumer Electronics, Kyoto, 2016, pp. 1-2.
[10] D. Bogdan, R. Bogdan and M. Popa, *Delta flashing of an ECU in the automotive industry*, 2016 IEEE 11th International Symposium on Applied Computational Intelligence and Informatics (SACI), Timisoara, 2016, pp. 503-508.
[11] C. Percival, *Matching with Mismatches and Assorted Applications*, doctoral thesis, Wadham College University of Oxford, http://www.daemonology.net/papers/thesis.pdf
[12] Hsin-Han Shin, T. Nakanishi, K. Hisazumi and A. Fukuda, *Differential Update of Automotive Device Firmwares with Broadcasting*, The Special Interest Group Technical Reports of IPSJ, Vol. 2011-SLDM-149, No. 23, pp. 1-6 (2011).
[13] H. Teraoka, F. Nakahara, K. Kurosawa, *Incremental Update Method for In-vehicle ECUs*, The Special Interest Group Technical Reports of IPSJ, Vol. 2016-CDS-16, No. 5, pp. 1-8 (2016).
[14] Y. S. Lee, J. H. Kim, S. J. Jang and J. W. Jeon, *Automotive ECU Software Reprogramming Method Based on Ethernet Backbone Network to Save Time*, Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication, IMCOM 2016, Danang, Vietnam, January 4-6, 2016.
[15] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham and S. Savage, *Experimental Security Analysis of a Modern Automobile*, 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 2010, pp. 447-462.
[16] C. Miller and C. Valasek, *Adventures in Automotive Networks and Control Units*, DEF CON 21, 2013.
[17] M. Steger, C. Boano, M. Karner, J. Hillebrand, W. Rom and K. Römer, *SecUp: Secure and Efficient Wireless Software Updates for Vehicles*, 2016 Euromicro Conference on Digital System Design (DSD), Limassol, 2016, pp. 628-636.
[18] AUTOSAR, https://www.autosar.org/ （参照 2017-10-31）.
[19] TOPPERS Project/ASP Kernel, https://www.toppers.jp/asp-kernel.html （参照 2017-10-31）.