情報処理学会論文誌 Vol. 50 No. 11 2643–2653 (Nov. 2009)

10 MbpsCAN プロトコルの設計と評価

倉地 亮[†1] 高田広章[†1]
手嶋茂晴[†1] 宮下之宏[†2]

今日までに提案されている次世代車載プロトコルの多くは既存システムの再構成を強く要求しており，高い安全性が求められる自動車の制御ネットワークへ適用するには高いコストとリスクがともなう．そこで本論文では，自動車内の制御系ネットワークに広く使われる CAN をベースとした新しい次世代車載プロトコル 10 MbpsCAN を提案する．10 MbpsCAN プロトコルはスタートポロジへの変更と新しい衝突解決アルゴリズムにより高速化を実現した．さらにシミュレーション結果によれば，10 MbpsCAN は従来 CAN に比べ十分なメッセージ量を送信できる能力を示した．最後に，10 MbpsCAN コントローラの実装について説明する．

Design and Performance Analysis of 10 MbpsCAN

Ryo Kurachi,[†1] Hiroaki Takada,[†1]
Shigeharu Teshima[†1] and Yukihiro Miyashita[†2]

Recently, automotive industry has atracted next genaration automotive protocols such as FlexRay. However, these solutions which shift to new applications will not meet the cost and risk requirements. In this paper, we propose a new automotive network protocol which can transmit data at a speed of 10 Mbps. The new automotive protocol “10 MbpsCAN” is based on several changes to CAN; a star topology, in which nodes are physically connected using a multiport gateway, and a new collision resolution algorithm which guarantees delivery of a message within a finite period of time. According to our analysis using a simulator, the “10 MbpsCAN” protocol demonstrated several advantages over throughput performance of a conventional CAN, including a maximum possible data transmission speed, high scalability, and reduction of priority inversion. The implementation considerations of “10 MbpsCAN” are also reviewed briefly.

1. はじめに

近年，自動車に求められる機能数の増加にともない，車両 1 台あたりに搭載される ECU（Electronic Control Unit）の数は年々増加している．現在，エンジン制御やブレーキ補助などの走行に関わる様々な機能を実現するために，車両 1 台あたりに数十個の ECU が搭載され，各 ECU が CAN（Control Area Network）[1] を中心とするネットワークを介して，多くの情報を共有し制御を実現している．特に高いリアルタイム性が要求される制御系システムにおいて，今後は X-by-Wire [2] と呼ばれる電子制御を中心としたシステムが標準となることが期待されており，次世代車載プロトコルにはより高いデータ転送能力やリアルタイム性が要求されている．

現在，CAN に代わる次世代車載プロトコルとして FlexRay [3] が注目されている．FlexRay に代表される次世代車載 LAN プロトコルの多くは新しいノードの追加を行うためにシステムの再構成が必要となるタイムトリガをベースとしており，CAN で構築された既存システムの再構成を強く要求している．一方で，制御系システム全体を一新するにはコストやリスクがかかりすぎる現実があり，従来 CAN を高速化するための研究が行われている [4]．

そこで，本研究ではイベントトリガアプローチにより，従来 CAN の 10 倍の回線容量をターゲットとする 10 MbpsCAN プロトコルを提案する．10 MbpsCAN は FlexRay と同様に物理的なネットワークトポロジを変更する必要はあるが，従来 CAN で開発されたソフトウェアと高い互換性を保つことで，論理的なネットワークを壊すことなく高速な回線を利用することを目的としている．次にプロトコルの評価として，従来 CAN で用いられるメッセージの最大遅れ時間解析の手法を適用し，10 MbpsCAN の解析手法について提案する．さらにシミュレーションにより CAN と 10 MbpsCAN の最大遅れ時間メッセージを比較することで，プロトコルの伝送能力を評価する．

本論文では，2 章で CAN 高速化の課題について整理し，3 章では 10 MbpsCAN プロトコルの提案とそのシステムの構成について述べる．続く 4 章ではプロトコルの評価として，10 MbpsCAN のシステム構成を考慮したゲートウェイにおけるメッセージの最大遅れ時間解析について提案し，プロトコルシミュレータの結果と比較することで提案する解析手法

†1 名古屋大学大学院情報科学研究科附属組込みシステム研究センター
Center for Embedded Computing Systems, Nagoya University
†2 株式会社オートネットワーク技術研究所
AutoNetworks Technologies, Ltd.

の有効性を確認した．さらに 5 章では実際の車両で使われるメッセージセットを用いてシミュレーションを行い，CAN と比べ 10 MbpsCAN が十分なメッセージ量を送信できることを評価した．6 章では実環境下でのコントローラの実装について述べ，最後に 7 章で本論文のまとめと今後の展望について言及する．

2. CAN 高速化の課題

CAN では同時に 2 つ以上のノードが送信を開始しメッセージが衝突する場合，衝突はビットワイズアービトレーション（アービトレーション）に従って解決される．CAN における伝送路上の値は論理的に優勢の信号（ドミナント）と劣勢の信号（レセシブ）のいずれかで表され，劣勢の信号に対し優勢の信号が上書きされることにより，その優劣を判断することができる．これに従いビットワイズアービトレーションでは，送信を行うノードが伝送路上で互いにメッセージに付与されるユニークな ID を 1 ビットずつぶつけあうことで，その優劣を判断し，劣性の信号を送信していたノードは自発的に送信を中止し，最後まで送信を続けた 1 つのノードのみが送信を行うことができる．このアービトレーションが成立する前提条件として，ノード間が伝送路上の 1 ビット時間内の距離に配置される必要がある．つまり，ビットレートはケーブルによる遅延率とノード間の距離に依存しており，伝送路遅延が 1 ビット時間よりも十分に小さくなければ，アービトレーションが成立しないことを意味する．また CAN の ACK シーケンスについても，アービトレーションと同様に 1 ビット時間内の距離に配置することが前提として求められる．

加えて，転送速度が増加するにつれ，波形の反射やノイズを要因とする伝送路上の波形乱れから，1 つのバスにつながるノード数を減らす必要がある．

本研究ではこれら伝送路遅延の影響を最小にするよう新しいアービトレーションと ACK シーケンスを用いた高速なプロトコルを提案する．

3. 提案するプロトコル：10 MbpsCAN

この章では，10 MbpsCAN プロトコルの概要と CAN からの変更点を説明し，そのうえでシステムの特徴を説明する．

まず前提として，10 MbpsCAN における伝送路上の 1 ビットは，CAN と同様にドミナントとレセシブのいずれかで表現される．そのうえで，10 MbpsCAN のメッセージ送信はチャネルの状態によって次のような 2 つの場合に分けられる．メッセージの送信要求が発生したときにバスがアイドルと判断できる場合，ノードはすぐに送信を開始する．一方，バスがビジーである，すなわち送信あるいは受信がすでに行われている場合には，互いのノードが交互に送信するようなスケジュールに従い，メッセージの送信が行われる．しかしながら，互いのノードがアイドルであると判断し同時に送信を開始した場合には，バス上で送信メッセージが衝突する可能性がある．それゆえ，この衝突を解決するためのアルゴリズムが新たに必要となる．

これらプロトコルの概要をふまえ，10 MbpsCAN プロトコルでは CAN からの次の 3 つの変更により，高速化を実現した．

（1）トポロジ変更

伝送路遅延の影響を克服するために，それぞれの ECU は物理的にマルチポートゲートウェイに接続する．この接続方法は一般にポイント・ツー・ポイント接続と呼ばれ，ゲートウェイを中心とするスタートポロジのネットワークとして運用され，リンギングを減らすための方法として広く用いられている．

（2）ACK シーケンス

信頼性の高い通信とするために，メッセージを送信後，次のメッセージを送信する前に ACK を待つようなストップ・アンド・ウェイトを採用した．ストップ・アンド・ウェイトを行うために，受信結果はフレーム単位で返却するよう変更し，受信結果を ACK 情報として付与するようフレームフォーマットを変更した．従来 CAN から変更されるデータフレームの詳細については，付録 A.1 を参照されたい．また具体的な動作として，**図 1** はゲートウェイが ECU に対しメッセージ A を送信した場合について示しており，それぞれ受信ノードと送信ノードとして以下のように動作する．まず受信ノードは，メッセージを受信すると所定のインターバル時間（EOF+IFS [⋆1]）を待った後，受信したメッセージの応答として ACK 情報を付与したフレームを送信する．一方，送信ノードは，メッセージの送信が完了すると受信ノードからの応答を一定時間待つ．もし応答が所定の期間内に到着しないようであれば，送信ノードはタイムアウトを判断しメッセージを再送する．なお，送信ノードが応答を待つ時間はノードの優先度に従い静的に決定される．この優先度は ECU よりもゲートウェイの再送が優先されるよう設計されており，この衝突解決アルゴリズムによって再送時には衝突が発生することなく送信が成功することを保証している．

⋆1 EOF（End of Frame）とはネットワーク上の 7 ビット時間，IFS（Interframe Space）は 3 ビット時間を表し，CAN では EOF+IFS 期間を標準的なフレーム間インターバルと定義している．

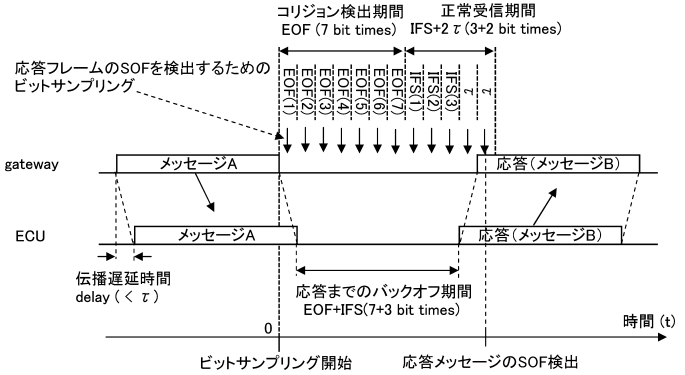


図 1　伝播遅延の影響を克服するための方法
Fig. 1　Method to overcome the effects of propagation delays.

（3）衝突解決アルゴリズム

メッセージ衝突時のデッドタイムを短くするために，提案する本アルゴリズムではユーザが伝送路遅延の情報をパラメータとして与える．これは ECU とゲートウェイの間に発生する実際の伝送路遅延をその伝送路に用いられるケーブル特性と伝送距離から見積もることによって，ユーザが設定する．この伝送路遅延の情報を使った，我々が提案するアルゴリズムの詳細を**図 2** に示す．送信ノードはバスアイドル中にメッセージの送信要求が発生すると，すぐにメッセージの送信を開始する．そして CRC デリミタの送信が完了した時点を $t = 0$ とし，応答メッセージの SOF を検出するためにビットサンプリングを開始する．このビットサンプリングの結果は次の 3 つの状態に分かれる．

- $t < 7$：CRC デリミタの送信完了後，7 ビットサンプリングするまでにドミナントビットを検出した場合，コリジョンを意味する．
- $7 \leq t < rtout$：CRC デリミタの送信後 7 ビット以上サンプリング後，正常受信期間 $rtout$ 内にドミナントを受信した場合，応答の SOF を受信したものと判断し受信処理を行う．これは送信に対し応答が正常な期間に返却されたことを意味する．
- $t \geq rtout$：受信ノードからの応答がない状態であり送信失敗を意味する．このとき，送信ノードでは再送が行われる．

このアルゴリズムで用いられる待ち時間 $rtout$ は極力短いデッドタイムとするために EOF + IFS + 2τ で設定され，ユーザがゲートウェイか ECU かを設定することで，ゲートウェイが必ず先に再送するような非対称な再送開始時間を持つことになる．ここで使われる τ はユーザが設定する伝送路遅延の情報であり，実際の遅延時間よりも大きな値で設定され，少なくとも 1 ビット時間以上にする必要がある．なお，図 2 中に示される α はノード間のクロックドリフトを許容するためのマージンを想定しており，これらは実装依存によって決定されるパラメータである．

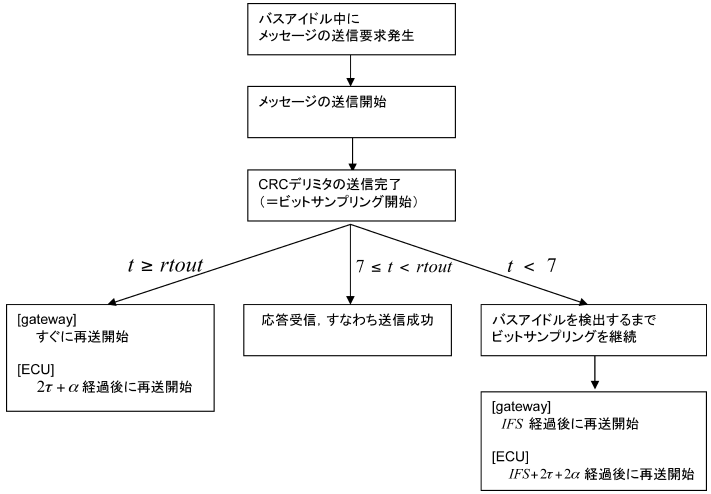


図 2　提案する衝突解決アルゴリズム
Fig. 2　Proposed collision resolution algorithm.

3.1　提案するシステム

ここでは 10 MbspCAN を用いたシステムの具体例をあげ，各機器の構造とメッセージの流れを示したうえで，システムの特徴について説明する．

まず，システムの例として，**図 3** では ECU1 と ECU2 がゲートウェイの支線 1 と支線 2 の先に接続される簡易なシステムを取り上げる．システムを構成する機器はゲートウェイと ECU の 2 種類が存在し，互いにポイント・ツー・ポイントで物理的に接続されている．ECU はホストプロセッサと 1 つの 10 MbpsCAN コントローラで構成され，ホストプロセッサのメモリ上には複数メッセージ分の送信キューと受信キュー，コントローラには送信メールボックスと受信メールボックスが少なくとも 1 つずつ用意される．ECU から送信されるメッセージは，ECU 内のアプリケーションからメッセージの送信要求が発生すると，送信

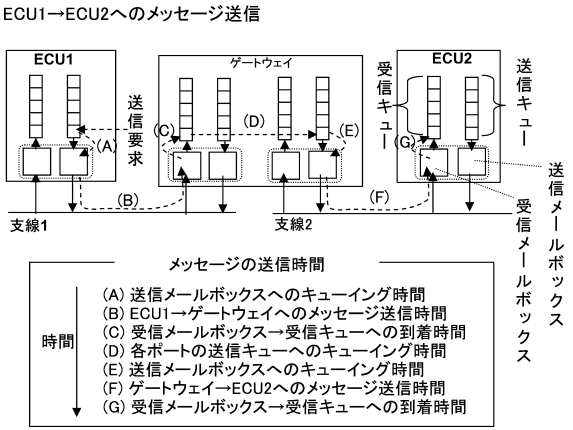


図 3　ゲートウェイを介した ECU 間通信
Fig. 3　Transmission between ECUs via a gateway.

キューへとメッセージが展開され，その後，送信キュー内に存在するメッセージは優先度の高いメッセージから順に送信メールボックスへと挿入され，伝送路へと送出される．一方，ゲートウェイは 1 つのホストプロセッサと各ポートにコントローラが用意され，各ポートから受信したメッセージを転送先となる支線へ中継する．ゲートウェイの中継処理は，コントローラの受信メールボックスにメッセージが到着すると，割込みなどを通じて受信タスクへと通知され，受信タスクによりホストプロセッサのメモリ上にある受信キューへと展開されるとすぐにゲートウェイ内に用意されるルーティングテーブルに従い，転送先のポートに用意される送信キューへと展開される．その後，送信キューでは ECU と同様にメッセージの優先度に従い，順次送信メールボックスへと挿入され，伝送路へと送信が行われる．

次に，ECU1 の送信メッセージが ECU2 へ到達するまでのメッセージの流れと送信に要する時間について，各ステップに分解し説明する．

まずメッセージの送信ステップとして，次の 2 段階に分けて解析できる．1 つは ECU1 からゲートウェイに到着するまでのメッセージ送信時間（図 3 の (A) + (B) + (C)）であり，もう 1 つはゲートウェイに到着したメッセージが転送先である ECU2 へ送信されるまでの時間（図 3 の (D) + (E) + (F) + (G)）である．前者については各 ECU 内で発生する送信要求の起動タイミングを制御することで ECU 内のメッセージを最適に分散させることが可能[5] であり，ゲートウェイから送信されるメッセージとの衝突は発生するかもしれないが，遅延時間はほとんど発生させることなく制御することができる．一方，後者については各 ECU から到着するメッセージは時刻同期などされていない限りは制御できないため，各 ECU から送信されるメッセージ量が増えれば増えるほど，ゲートウェイ内部に滞留するメッセージ数が増加し，ゲートウェイ内での滞留時間が異常に長くなることが懸念される．

従来 CAN におけるメッセージの送信遅延は，CAN バス上におけるアービトレーションにより，他ノードから送信されるメッセージの優先度やその送信頻度に影響されるため，バス負荷率の規定を与えることでメッセージの最悪送信遅延を保証していた．一方，提案する 10 MbpsCAN プロトコルを使ったシステムにおいては，各支線の送信時間よりもゲートウェイでの遅延がボトルネックになると予想でき，従来 CAN におけるバス負荷率同様の設計指南を与えるためには，そのシステム構成を考慮したゲートウェイの遅延を見積もる方法が必要となる．そこで本論文では，ボトルネックになるであろうゲートウェイにおける最大遅れ時間を求めるための解析手法について提案し，その評価を行う．なお，ここでのゲートウェイにおけるメッセージの遅れ時間とは，ゲートウェイにメッセージが到着してから転送先の ECU へ到着するまでの時間（図 3 の (D) + (E) + (F) + (G)）を指す．

4. ゲートウェイにおける最大遅れ時間の解析と評価

本章の目的は，各メッセージの最大遅れ時間を求めるための解析手法を与えることにある．そのため，ここで示す解析手法を用いて計算された値が正しいことを検証するために，同環境下におけるシミュレーション結果と比較し，その妥当性を確認した．

まず，ここで用いる従来 CAN におけるメッセージの最大遅れ時間の解析手法について説明する．飯山らは，Tindell らにより示された CAN メッセージの最大遅れ時間を求める手法を発展させ，より現実的なシステムに解析手法を適用している[6),7)]．これらの従来手法は単一プロセッサにおけるタスクの静的優先度ベーススケジューリングである Rate Monotonic Analysis をメッセージのスケジューリングに適用したものであり，CAN における critical instant 定理が示されている．CAN における critical instant とは，そのメッセージよりも優先度の高いメッセージの送信要求がすべて同時に発生したときであり，あるメッセージの送信要求が発生可能になってから，実際に送信されるまでの時間が最も長くなる状況である．

次に，ゲートウェイの構造と送信キューに着目し，従来手法をゲートウェイに適用するためのアプローチを説明する．ここで解析の対象となるゲートウェイは，図 3 で示される一般的なアーキテクチャであることを想定し，メッセージの送信は優先度ベースで行われるものとする．

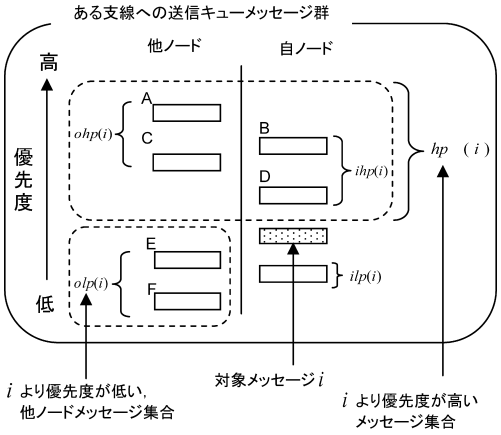


図 4　ゲートウェイの送信キューにおけるメッセージ集合
Fig. 4　Grouping of messages in a transmit queue of gateway.

図 4 はゲートウェイ内の各ポートの送信キューに存在するメッセージ群，つまりはある ECU を転送先とするメッセージの集合を表している．これらのメッセージは解析の対象となるメッセージ i を基準に送信元で 2 つに分類され，“i” プレフィックスは対象となるメッセージ i と同じ ECU から送信されたメッセージを示し，“o” プレフィックスは対象となるメッセージ i とは別の ECU から送信されたメッセージを指す．さらにメッセージの優先度を用いて分類すると，すべてのメッセージは対象となるメッセージ i と以下の 4 グループに分類される．

- $ohp(i)$：メッセージ i とは別の ECU から到着したメッセージ i より優先度の高いメッセージ集合
- $olp(i)$：メッセージ i とは別の ECU から到着したメッセージ i より優先度の低いメッセージ集合
- $ihp(i)$：メッセージ i と同じ ECU から到着したメッセージ i より優先度の高いメッセージ集合
- $ilp(i)$：メッセージ i と同じ ECU から到着したメッセージ i より優先度の低いメッセージ集合

ここで対象となるメッセージ i の最大遅れ時間を解析するために，次のシナリオをゲートウェイの最悪シナリオであると仮定した．ここではメールボックスが 1 個である場合についてのみ議論する．

Critical instant　メールボックス数が 1 個の場合，ゲートウェイの到着と同時に，次の条件をすべて満たす状況において，メッセージのゲートウェイにおける Critical instant を定義できる．(1) ある ECU へ向けたメッセージ i よりも優先度の高いすべてのメッセージが同時にゲートウェイに到着し，(2) 送信メールボックスに，メッセージ i よりも優先度の低い他ノードから到着したメッセージのうち，最大の滞留時間を持つメッセージが格納され（ただし，メッセージが存在しない場合には，この条件はなくなる），(3) その最初の送信において，コリジョンが発生し最大の送信時間を要する．

4.1　解 析 手 法

これまでに提案されている従来 CAN の解析手法では ECU のアプリケーションがメッセージの送信要求を発行する場合について議論されているが，ゲートウェイの場合においても到着したメッセージに関して議論することで同様の解析手法が適用できる．まず ECU が送信要求を発行する場合については，ゲートウェイでは受信したメッセージがつねに送信要求可能な状態であると仮定すれば，ゲートウェイの中継処理は ECU のアプリケーションの送信要求と同等であると見なすことができる．なお，本論文で扱う解析手法は優先度ベースに処理されるゲートウェイの送信キューに着目し，従来 CAN における最大遅れ時間解析手法を適用したものであり，それ以外の方法で処理されるゲートウェイには適用できないので，この点に注意されたい．

ここで扱うすべてのメッセージは周期的に送信が要求されるものとし，n 個の周期メッセージからなる集合がゲートウェイの送信キューに到着すると考える．メッセージ i $(1 \leq i \leq n)$ は (P_i, T_i, C_i) のパラメータを持つ．ここで，P_i は優先度を示すメッセージ ID，T_i はメッセージの送信周期で表される．またメッセージの送信に要する最大の送信時間 C_i は，従来 CAN の解析で使われた方法を用いて，式 (1) のように表される．ここで表されるメッセージ i の最大の送信時間は，スタッフビットの対象となるヘッダ 35 ビットとペイロードであるデータ部 s_i バイトに対し 5 ビットに 1 ビットのスタッフビットが挿入され，これに 46 ビットのフレームのヘッダ領域とデータ $8\,s_i$ ビットを加え，τ_{bit} を掛け合わせたものが送信時間として表すことができる．

$$C_i = \left(\left\lfloor \frac{35 + 8s_i - 1}{4} \right\rfloor + 46 + 8s_i \right) \tau_{bit} \tag{1}$$

メッセージ i の最大遅れ時間 R_i，つまり，メッセージ i がゲートウェイに到着してからその送信が完了するまでの最大時間は以下のように表される．

$$R_i = J_i + Q_i + C_i \tag{2}$$

J_i はメッセージ i の最大キューイングジッタで，ゲートウェイへメッセージ到着後，送信キューへの転送要求を出すことが可能となる最も遅い時刻と最も早い時刻との差を表している．Q_i はメッセージ i の最大キューイング時間で，メッセージがゲートウェイへ到着以降，実際に送信が開始されるまでの最大時間を示している．この時間は他のメッセージに邪魔される時間を含んでおり，次式を満たす最小の Q_i として示される．

$$Q_i = B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{Q_i + J_j + \tau_{bit}}{T_j} \right\rceil (C_j + E_{max}) \tag{3}$$

ここで示される E_{max} は転送先となる ECU から送信される応答メッセージのうち，最大長のメッセージの送信にかかる時間を表している．また，B_i は低優先度メッセージが存在する場合と存在しない場合とで与える式が異なり，低優先度メッセージが存在する場合にはその低優先度メッセージを送信するのに必要な最大時間を表し，一方で，低優先度メッセージが存在しない場合には対象メッセージの衝突に要する最大時間を表す．

（1）低優先度メッセージが存在しない場合

低優先度メッセージが存在しない場合では，その衝突に要する時間 B_i は対象となるメッセージ i と転送先の ECU から送信されるメッセージとの衝突に要する時間であり，伝送路の遅延時間 $delay$ を用いて，次式で表すことができる．

$$B_i = delay + \tau_{bit} + \max(C_i, E_{max}) \tag{4}$$

（2）低優先度メッセージが存在する場合

一方，低優先度メッセージが存在する場合には，低優先度メッセージの送信に必要な最大の時間であり，次式で表すことができる．

$$B_i = D_i + \max_{\forall j \in olp(i)} C_j + E_{max} \tag{5}$$

D_i は低優先度メッセージと ECU から送信されるメッセージとの衝突にかかる最大時間を表し，次のように表すことができる．

$$D_i = delay + \tau_{bit} + \max(C_j, E_{max}) \tag{6}$$

提案するプロトコルでは衝突検知時にはゲートウェイからの再送が優先されるよう非対称に制御されているため，ここで表される式はゲートウェイの場合についてのみ適用できる．なお，ここでは両ノードが正常に応答可能な状況であることを前提としており，受信ノードが応答しないなどの異常時については議論しない．ここで定義するブロッキング時間に関する詳細な状況については付録 A.2 を参照されたい．

4.2　10 MbpsCAN シミュレータ

ゲートウェイにおけるメッセージの最大遅れ時間の解析の有効性を確認するために，プロトコルシミュレータの OPNET Modeler [8] を使い，10 MbpsCAN のシミュレーションモデルを開発した．この 10 MbpsCAN のシミュレーションモデルは，伝送路とコントローラとアプリケーションの 3 つのモデルで構成される．まず最初に，伝送路のモデルは正確なネットワークモデルを実現するためにネットワーク上の 1 ビット単位でモデル化した．次にコントローラのモデルは，10 MbpsCAN プロトコルで定義されるフレームのコーディング，エンコーディングとビットスタッフィングを行い，実際のコントローラと同等の機能を持つ．最後にアプリケーションのモデルは ECU とゲートウェイで異なり，ECU のアプリケーションは与えられたメッセージセットから周期的にメッセージの送信要求を発生させるモデルであり，ゲートウェイのアプリケーションはメッセージを受信するとすぐに中継処理を開始し，各ポートの送信キューへとメッセージ転送を行うモデルである．なお，ここでは中継にかかる時間はネットワーク上の 1 ビット時間に比べて十分に小さいものとして，受信キューから送信キューへの転送時間（図 3 の (C) + (D) + (E)）を 0 として扱っている．これらのモデルに，シミュレーション結果として各メッセージの最大遅れ時間を得るためのログ出力機能を追加したシミュレーションモデルを評価に用いた．

ここで解析手法を評価するために，ゲートウェイに最もメッセージが集中するよう 1 つのゲートウェイに 20 個の ECU をポイント・ツー・ポイントで接続するネットワークを用いた．このシミュレーションモデルの外観は**図 5**，シミュレーションに用いるパラメータは**表 1** に示すとおりであり，実際の車両で使われるメッセージセットを使い評価を行った．

4.3　シミュレーションシナリオ

システムにおける最大遅れ時間を評価する観点から，ゲートウェイの各ポート，すなわち各送信キューが最もメッセージで混み合うようなシナリオを実現するために，ポートごとに送信するメッセージを抽出してシミュレーションを行った．そして各ポートのシミュレーションから得られた全メッセージの遅延率を比較することで，システム上最も遅延率の大きいメッセージを 1 つ選び出す方法をとった．またこのシミュレーションでは，シミュレーションが開始されると同時に，各 ECU は送信するすべてのメッセージの送信要求を発行し送信が開始されるため，シミュレーション開始と同時に最もゲートウェイが混み合う状況にある．そのため，このシミュレーションにおいて，最悪シナリオはシミュレーション開始とほぼ同時に発生し，その後，与えられたシミュレーション時間以内に定常状態へ回復するよ

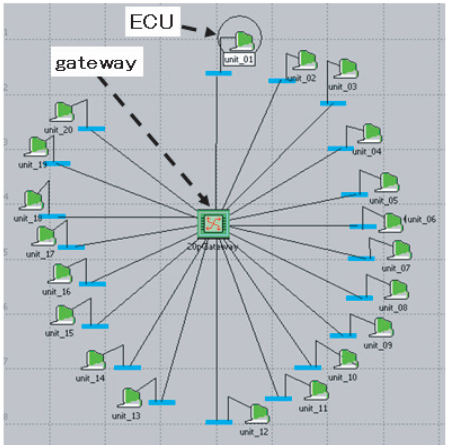


図 5　10 MbpsCAN のシミュレータ概観
Fig. 5　Simulator overview (10 MbpsCAN).

表 1　10 MbpsCAN におけるシミュレーションパラメータ
Table 1　Simulation parameters of 10 MbpsCAN.

Parameters	Status
Number of ECUs	20
Number of gateways	1
Size of message queue	200 messages
Data rate	10 Mbps
Simulation time	10 seconds

うなシミュレーションシナリオとなる．

4.4　解析手法とシミュレーション結果の比較

システム上，最も大きな遅延率を持つポート 20 のシミュレーションにおいて，シミュレーション開始後すぐに発生する最悪シナリオ時には，最大で 181 メッセージが送信キューにキューイングされ，送信キューはほとんど満杯になった．

図 6 は各ポートごとのシミュレーションから得られた最大遅れ時間メッセージと，解析による計算結果から得られた最大遅れ時間メッセージの比較を示しており，どちらの結果もポート 20 で最大の遅延率を持つメッセージが存在することを示した．またすべてのポートにおいてシミュレーション結果よりも計算結果の方が大きいことから，本解析は容易な計算とするために悲観的ではあるが，ゲートウェイにおける遅延率を容易に見積もるための方

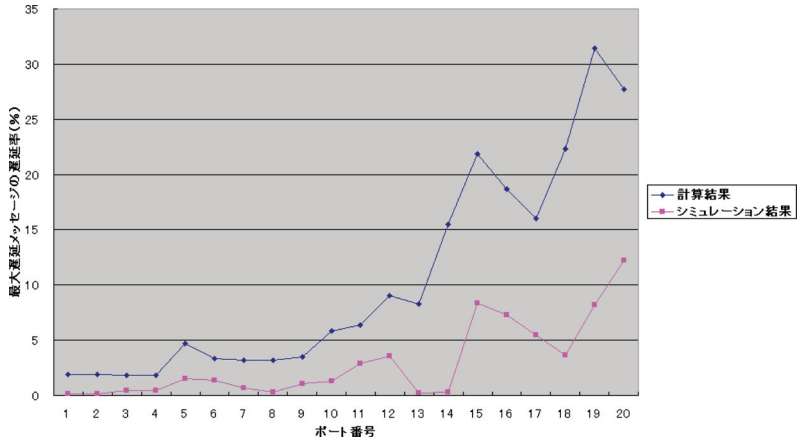


図 6　シミュレーションと計算結果によるゲートウェイ遅延率の比較
Fig. 6　Comparison of worst-case response time of gateway ports for the simulation and calculated results.

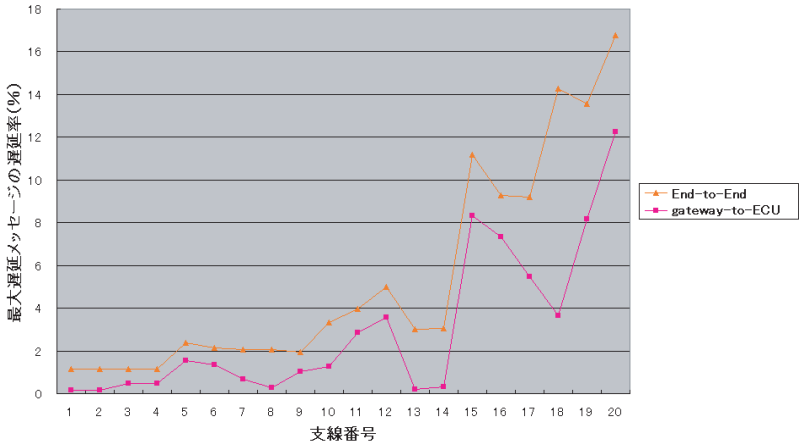


図 7　End-to-End の遅延率とゲートウェイ遅延率の比較
Fig. 7　Comparison of worst-case response time of gateway ports for the gateway-to-ECU and the End-to-End results.

法として有効であるといえる．この計算結果とシミュレーション結果の違いは，目的となるECU から到着するメッセージの送信時間 E_{max} が容易な計算になるよう過大に見積もっているためと考えられる．

次に，このシステムにおけるメッセージの最大遅れ時間とゲートウェイにおける遅れ時間の関係を考察するため，開発した 10 MbpsCAN シミュレーションモデル用いて，各ポートにおける End-to-End（図 3 の (A) + (B) + (C) + (D) + (E) + (F) + (G)）までのメッセージの最大遅れ時間を測定した．**図 7** に示されたシミュレーション結果のとおり，複数の ECU から同時にメッセージが到着するようなトラフィックの高いポートでは，メッセージの全送信時間に対するゲートウェイの遅れ時間の占める割合が非常に高くなっていることが明らかとなった．

5. 10 MbpsCAN と従来 CAN のシミュレーション比較

10 MbpsCAN のメッセージ伝送能力を定量的に示すために，従来 CAN と 10 MbpsCAN のシミュレーション比較を行った．ここで用いる 10 MbpsCAN のシミュレーションモデルは前章で開発したものを使用し，従来 CAN のシミュレーションモデルは次のとおり用意した．なお，従来 CAN のシミュレーションも 10 MbpsCAN と同様にネットワーク上の 1 ビット単位でモデル化しているため，ネットワークレベルでは正確なネットワークモデルである．

5.1 従来 CAN シミュレータ

比較に用いる従来 CAN のシミュレータは 20 個の ECU を 1 つのバスに接続させるバス型ネットワークとし，その通信速度は従来 CAN の最大通信速度である 1 Mbps とする．実際の車両ではリンギングの問題から最大通信速度を 500 kbps で使用しており，1 つのバスに接続できる ECU を 10 数個までと制限しているが，ここでは 10 MbpsCAN と比較するために理想的なネットワークを用いている．またシミュレーションで使用されるメッセージセットはおおよそ 100 種のメッセージを有する実際の車両で使われているメッセージセットであり，このメッセージ量を基準の 1 倍量とする．なお，**図 8** は従来 CAN のシミュレータの外観を表している．

5.2 比 較 方 法

プロトコルとトポロジが違う従来 CAN と 10 MbpsCAN を比較するために，評価指標として転送元 ECU から目的の ECU へ到着するまでの End-to-End におけるメッセージの最大遅れ時間を比較した．さらに，10 MbpsCAN は従来 CAN の 10 倍の回線容量を持つことから，公平な比較を行うために，10 MbpsCAN では 2 つのメッセージセットでシミュレーションした．1 つは比較対象である従来 CAN のメッセージセットと同じ 1 倍量で，もう 1 つは回線速度に応じたメッセージ量，すなわち CAN で扱うメッセージセットの 10 倍量でシミュレーションを行い評価した．

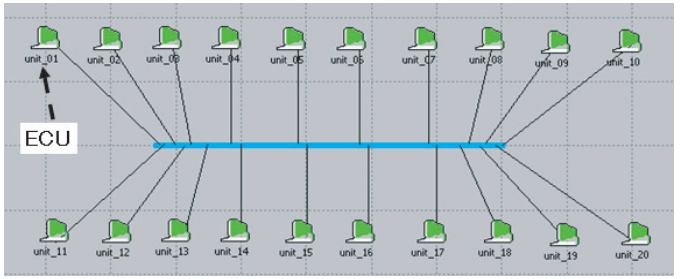


図 8　従来 CAN のシミュレータ外観
Fig. 8　Simulator overview (CAN).

表 2　End-to-End におけるシミュレーション結果の比較
Table 2　Simulation results in end-to-end simulation.

	Protocol (speed)	Quantity of message sets	Worst case response message	
			Transmission cycle (ms)	Delay rate (%)
1	CAN (1 Mbps)	x1	12	15.867
2	10 MbpsCAN (10 Mbps)	x1	24	1.745
3	10 MbpsCAN (10 Mbps)	x10	24	16.763

5.3 10 MbpsCAN と従来 CAN の比較結果

表 2 は従来 CAN と 10 MbpsCAN の各シミュレーションから得られた最大遅れ時間を持つメッセージを示している．これらのメッセージは送信周期に対する送信時間の割合である遅延率で評価され，システム上最も大きい遅延率を持つメッセージのみを比較している．シミュレーション 1 と 2 の遅延率の比較によれば，10 MbpsCAN がほとんど従来 CAN の 10 倍に近い回線速度を実現している．さらに，シミュレーション 1 と 3 の遅延率の比較によれば，10 MbpsCAN では伝送路遅延の影響を克服するために連続する交互送信を行い従来 CAN よりも多少のオーバヘッドが存在するにもかかわらず，10 MbpsCAN における遅延

率がほとんど CAN の遅延率と同じであることから，従来 CAN と比較しても回線速度に見合うメッセージ量の送信が可能であることを示している．またシミュレーション 3 が 2 と比べ 10 倍の遅延率よりも下回っているのは，回線速度の向上と送信方法の変更により低優先度メッセージの影響が小さくなっていたためである．これらシミュレーションの比較結果から，我々のプロトコルは従来 CAN を高速化した場合とほぼ同等の高い能力を示していることが分かる．

6. FPGA への実装

実際のハードウェア上での効果を検証するために FPGA への 10 MbpsCAN コントローラの実装を行った．実装時における 10 MbpsCAN と従来 CAN の大きな違いは，新しい ACK シーケンスや提案する衝突解決アルゴリズムを 10 MbpsCAN プロトコルのステートマシーンとして実装している点である．

今回，試作環境として Nios II Development Kit, Cyclone II Edition を使った．このテストボードに実装した FPGA のコンポーネントは NiosII プロセッサ，タイマ，オンチップ RAM，DDR-SDRAM コントローラを各 1 個に加えて，ECU の場合は 10 MbpsCAN コントローラが 1 個，ゲートウェイの場合は 10 MbpsCAN コントローラを 2 個実装した．このデザインを合成した結果のリソース使用量を**表 3** に示す．この結果より，1 つの 10 MbspCAN コントローラで使われるハードウェア資源は 949 ロジックエレメントであり，ボード上のピンを 4 個使用している．

このデザインの主要なコンポーネントである NiosII プロセッサと 10 MbpsCAN コントローラはそれぞれ，10 MbpsCAN コントローラのホストプロセッサとして，また本論文で提案する 10 MbpsCAN プロトコルを実現するネットワークコントローラとして実装されている．10 MbpsCAN コントローラの機能は主にフレームのデコードやエンコード，ビットスタッフなどを行うことであり，このコンポーネントは**図 9** に示すとおり，3 つの大きな部品で構成される．まず 10 MbpsCAN のステートマシーンを要するプロトコル制御ユニットが存在し，その他 2 つは従来 CAN でも同様に用いられるビットサンプリングユニットとメッセージレジスタで構成される．

本実装の評価として，まず 1 つ目に従来 CAN コントローラとの互換性の高さを評価するために従来 CAN とのレジスタ構成を比較した．その結果，バックオフ時間の優先度を設定するためのゲートウェイ/ECU の属性設定ビットと『伝送路遅延の情報』を追加している以外は従来 CAN コントローラと同等であり，従来 CAN とはコントローラの初期化処理の

表 3　テストボードにおける合成結果
Table 3　Synthesis results on the test board.

	ECU	ゲートウェイ
Total logic elements	4,251/33,216 (13%)	5,200/33,216 (16%)
Total pins	81/475 (17%)	85/475 (18%)
Total memory bits	48,384/483,840 (10%)	48,512/483,840 (10%)

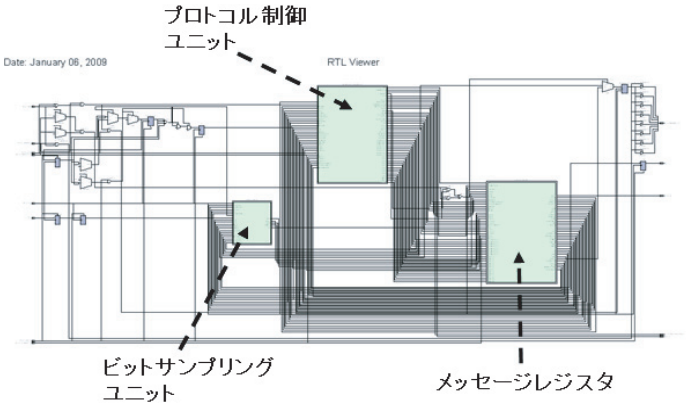


図 9　10 MbpsCAN コントローラ IP の外観
Fig. 9　Overview of 10 MbpsCAN controller IP.

み変更する必要はあるが，それ以外は変更する必要がないレジスタ構成とすることで，高いソフトウェア互換性を保つことができた．2 つ目にコントローラの動作を確認するため，AUTOSAR[9] COM スタックおよびその上で動作するデモアプリを作成し動作を確認した．ここで作成した AUTOSAR の COM スタックは CAN の仕様で定義される CAN Driver と CAN Interface，その上位層にあたる COM，PDU Router とデモアプリで構成され，PDU Router はゼロコストオペレーションで実装した．また，本デモのトポロジとしてゲートウェイ 1 個と ECU2 個のネットワークを用いたが，ゲートウェイのデモアプリで中継処理を行うと中継にかかる時間が大きすぎるため，今後ゲートウェイのポート数を増やすことを検討すると同時に，中継処理のハードウェア化の検討を行う．

7. ま と め

本論文では，まず従来 CAN を高速化するための課題について議論し，そのうえで，CAN

をベースとする次世代車載プロトコル 10 MbpsCAN を提案した．次に，このプロトコルを用いたシステムの詳細について示し，ゲートウェイがシステムのボトルネックになりうるため，ゲートウェイにおける最大遅れ時間の解析手法の提案を行い，シミュレーション結果との比較から解析手法の妥当性を確認した．さらに，プロトコルを定量的に評価するためにシミュレータを使い，従来 CAN と 10 MbpsCAN の伝送能力の比較を行い，10 MbpsCAN が CAN と比べても十分に高い伝送能力があることを示した．そして最後に FPGA への実装を行い，実環境下で構築されたコントローラを通じて，プロトコルが有効であることを確認した．

最後に今後の課題として，本論文ではゲートウェイの中継に関わる時間をネットワーク時間に対し，十分に小さい時間として扱ってきたが，ソフトウェアで中継処理を行う場合にはこの予想よりも大きい時間が必要になるため，中継処理のハードウェア化を検討する必要がある．さらに，ゲートウェイのハードウェア化により実際の中継処理にかかる時間を反映させたより正確なシミュレーションモデルと解析モデルを提案したりする必要がある．

謝辞　本研究はオートネットワーク技術研究所と名古屋大学との共同研究である．本研究を行うにあたり，貴重なご意見をいただいたオートネットワーク技術研究所の夏目晃宏氏，山本秀樹氏，堀端啓史氏には，ここに厚く御礼申し上げます．

参　考　文　献

1) International Organization for Standardization, Road vehicles: Controller area network (CAN), Part1: Data link layer and physical signaling, ISO IS11898-1 (2003).
2) Nolte, T., Hansson, H. and Bello, L.L.: Automotive communications – past, current and future, *Proc. 10th IEEE International Conference on Emerging Technologies and Factory Automation* (*ETFA'05*), Vol.1, pp.985–992 (2005).
3) FlexRay Consortium. http://www.flexray.com/
4) Compton, B.T.: Goldilocks Serial Communication Protocol, SAE World Congress (2008).
5) 飯山真一，冨山宏之，高田広章，城戸正利，細谷伊知郎：オフセット付き CAN メッセージの最大遅れ時間解析，情報処理学会論文誌：コンピューティングシステム，Vol.45, No.SIG11(ACS 7), pp.455–464 (2004).
6) Tindell, K. and Burns, A.: Guaranteed Message Latencies for Distributed Safety-Critical Hard Real-Time Networks, Technical Report YCS 229, Department of Computer Science, University of York (1994).
7) 飯山真一，高田広章：システム構成を考慮した CAN の最大遅れ時間解析手法，情報処理学会論文誌：コンピューティングシステム，Vol.45, No.SIG1(ACS 4), pp.66–76 (2004).
8) OPNET Modeler. http://www.opnet.com/
9) AUTOSAR. http://www.autosar.org/

付　　録

A.1　10 MbpsCAN のデータフレーム

10 MbpsCAN のデータフレームは従来 CAN で使用されるフレーム構造から 2 点変更されている．まず 1 つ目に，ACK を情報ビットとしてフレームに含めるため，ACK フィールドをデータフィールドの前に移動させている．次に，CRC デリミタの値をレセシブからドミナントへ変更しており，これは提案する衝突解決アルゴリズムでコリジョンを検出するためにフレームの最初（SOF）と最後（CRC デリミタ）を認識する必要があるためである．フレームの概要は，**図 10** に示すとおりである．

A.2　10 MbpsCAN のブロッキング時間

前述のとおり，メッセージ i の最悪ブロッキング時間 B_i は，低優先度メッセージが存在する場合としない場合の 2 つのに分けて考える必要がある．

まず，メッセージ i よりも低優先度のメッセージが存在しない場合，低優先度メッセージに邪魔されることはないため，ブロッキング時間 B_i は，メッセージ i と ECU から送信される最大長のメッセージの衝突にかかる最悪時間を想定しなければならない．**図 11** は，対

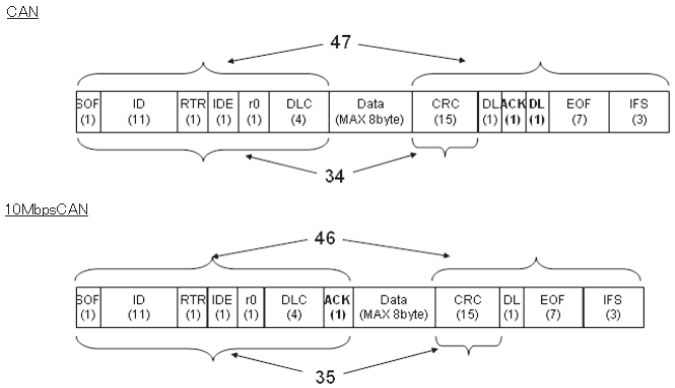


図 10　従来 CAN からのデータフレームの変更
Fig. 10　Data frame format changed from CAN.

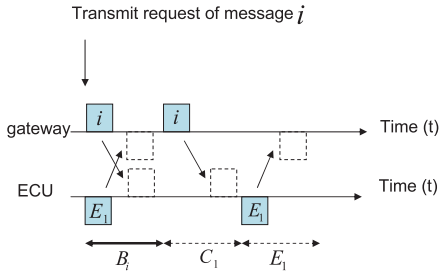


図 11　低優先度メッセージが存在しない場合
Fig. 11　When non-blocking by a lower priority message.

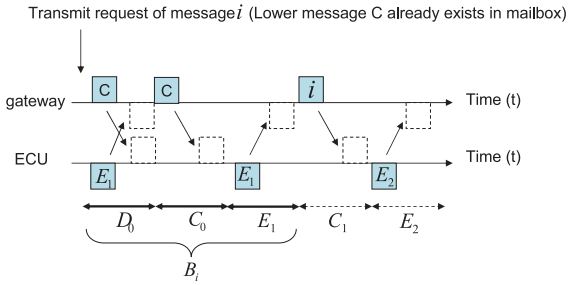
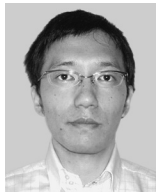


図 12　低優先度メッセージが存在する場合
Fig. 12　When blocking by a lower priority message.

象となるメッセージ i と ECU からのメッセージ E が衝突する場合のブロッキング時間 B_i を示している．

次に，低優先度メッセージが存在する場合，メッセージ i が送信要求をすると同時に送信メールボックスに低優先度メッセージが入ると，メッセージ i は低優先度メッセージの送信に必要な最悪時間だけ邪魔される．**図 12** は，対象となるメッセージ i が低優先度メッセージ C に邪魔される場合のブロッキング時間 B_i を示している．

(平成 21 年 2 月 2 日受付)
(平成 21 年 9 月 11 日採録)



倉地　亮（正会員）

名古屋大学大学院情報科学研究科附属組込みシステム研究センター研究員．2000 年東京理科大学基礎工学部電子応用工学科卒業後，アイシン AW 株式会社にてカーナビゲーションシステムの開発に従事．2007 年東京理科大学専門職大学院総合科学技術経営研究科技術経営専攻修了後，同年より現職．車載 LAN に関する研究に従事．

高田　広章（正会員）

名古屋大学大学院情報科学研究科情報システム学専攻教授．1988 年東京大学大学院理学系研究科情報科学専攻修士課程修了．同専攻助手，豊橋技術科学大学情報工学系助教授等を経て，2003 年より現職．2006 年より名古屋大学大学院情報科学研究科附属組込みシステム研究センター長を兼務．リアルタイム OS，リアルタイムスケジューリング理論，組み込みシステム開発技術等の研究に従事．オープンソースの ITRON 仕様 OS 等を開発する TOPPERS プロジェクトを主宰．博士（理学）．IEEE，ACM，電子情報通信学会，日本ソフトウェア科学会各会員．

手嶋　茂晴（正会員）

京都大学大学院工学研究科情報工学専攻修士課程，名古屋大学大学院工学研究科情報工学専攻博士後期課程修了．博士（工学）．1986 年株式会社豊田中央研究所入社，現在に至る．2006〜2009 年名古屋大学大学院情報科学研究科附属組込みシステム研究センター特任教授/ディレクター等歴任．

宮下　之宏

1988 年大阪大学工学部機械工学科卒業．現在，株式会社オートネットワーク技術研究所勤務．車載 LAN に関する研究に従事．