システム構成を考慮したCANの最大遅れ時間解析手法

飯山真一[†] 高田広章[††]

現在，自動車の制御系ネットワークにおける事実上の標準としてCAN（Controller Area Network）が広く普及しており，CANメッセージの最大遅れ時間評価に関する研究が行われている．しかしながら，従来の評価手法は理想的なモデルの下で議論されており，実際のシステムにはあてはまらない．本論文では，従来手法の問題点を指摘し，この問題を解決するために，より現実的なモデルに対するメッセージの最大遅れ時間を求める手法を提案する．また，実際のシステムにおいて使用が検討されているメッセージセットに対して提案手法を適用し，提案手法の有効性を確認した．

Response Time Analysis with Architectural Considerations for CAN

Shinichi Iiyama[†] and Hiroaki Takada[††]

CAN (Controller Area Network) has become widespread as a de-facto standard of automotive networks for control, and methods to evaluate the worst-case response times of messages in CAN have been proposed. However, the conventional analysis for CAN message is discussed under an ideal model, and cannot be applied to an actual system. This paper points out the problems of the conventional method. As a solution to these problems, we propose a method to decide the worst-case response times of messages in more realistic system model. We also apply our method to message sets currently considered for a control network of an actual automotive, and confirm the method to be effective.

1. はじめに

近年，自動車制御は，低燃費化，走行性の向上，排気ガスや安全性に関する厳しい規制をクリアするために，複雑化・大規模化している．これにともない，車両に搭載されるECU（Electronic Control Unit）の数は増加する一方であり，その制御プログラムも複雑化，大規模化している．このような中で，自動車内ネットワークの導入が本格化してきている．自動車内のネットワークはボデー系，制御系，マルチメディア系の3つに分類することができる．これらの中で特に高いリアルタイム性が要求される制御系のネットワークにおいては，現在，Controller Area Network（CAN）[1)]が事実上の標準として広く受け入れられている．

また，CANに対する最大遅れ時間の評価に関する研究が，ハードリアルタイムスケジューリング理論の分野において進められている[2)~5)]．これらの従来手法は，メッセージを送信するノードに十分なリソースがあることを前提としたモデルの下で議論が行われている．しかしながら，このような前提はリソース制約が厳しい実際のシステムにはあてはまらず，従来手法をそのままの形で適用することはできない．実際のシステムにおいて，CANに対する最大遅れ時間を求めるためには，ノードのシステム構成を含めたモデルに対する議論が必要である．

そこで本研究では，自動車内ネットワークのための理論に基づく体系的な検証手法を提案し，実際のシステムへ適用することを目指している．本論文では，従来手法では実際のシステムを検証するためには不十分であることを指摘し，現実的なシステム構成を考慮した新たな評価手法を提案する．また，実際の自動車内ネットワークにおいて検討されているメッセージセットに対して提案手法を適用し，提案手法の有効性を確認する．さらに，提案手法を用いることにより，限られたリソースの中でもメッセージの最大遅れ時間を短く抑えるための方法について述べる．

本論文では，まず2章でCAN仕様[1)]の概要について述べる．3章では，本論文で想定するシステムの構

† 豊橋技術科学大学情報工学系
Department of Information and Computer Sciences, Toyohashi University of Technology
†† 名古屋大学大学院情報科学研究科情報システム学専攻
Department of Information Engineering, Graduate School of Information Science, Nagoya University

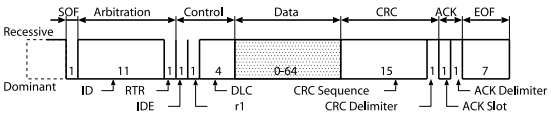

図 1 CAN のメッセージフォーマット
Fig. 1 CAN message format.

造について述べる．4 章では，CAN メッセージの最大遅れ時間を求める従来手法について述べた後，それを実際のシステムへ適用するためには，最悪性能評価の点において不十分であることを指摘する．5 章では，システム構成を考慮したメッセージの最大遅れ時間を求める新しい評価手法を提案する．6 章では，提案手法を実際のシステムに適用した事例について述べ，提案手法を用いたメッセージの最大遅れ時間を小さく抑える方法について 7 章で述べる．

2. Controller Area Network（CAN）

現在，自動車内の制御系ネットワークにおいて，Controller Area Network（CAN）[1]は事実上の標準となっている．CAN の仕様では，OSI の参照モデルの物理層とデータリンク層のみが規定されており，その実現はハードウェア（CAN コントローラ）で提供される．

CAN は最高で 1 Mbps の転送速度を実現できるブロードキャストバスである．CAN バス上では，最大長が与えられたメッセージ単位でデータがやりとりされる．CAN 仕様では，データフレーム，リモートフレーム，エラーフレーム，オーバロードフレームの 4 種類のメッセージフレームが定義されている．制御系ネットワークのノード間でデータを通信するために用いられるデータフレームには，**図 1** に示すように，47 ビット分のプロトコル制御用の情報（ID，CRC，ACK や同期ビットなど）に加えて，0 から 8 バイトまでのデータを含めることができる．メッセージには 11 ビットの ID が割り付けられる．ID はメッセージの優先度を表し，受信時には，必要なメッセージだけを受信するためのフィルタリングにも使用される．

ID がメッセージの優先度として用いられることが CAN での通信のリアルタイム性保証に関して最も重要となる．CAN のメッセージは ID の数値が小さいほど高い優先度を持つ．同時に 2 つ以上のノードが送信を開始しメッセージの衝突が起こった場合，衝突はワイヤード AND 機構による非破壊ビットワイズアービトレーション（Non-destructive bitwise arbitration）に従って解決される．具体的には，バス上の信号に優劣を設け，劣勢な信号（Recessive，論理 1）を優勢な信号（Dominant，論理 0）で上書きすることにより，

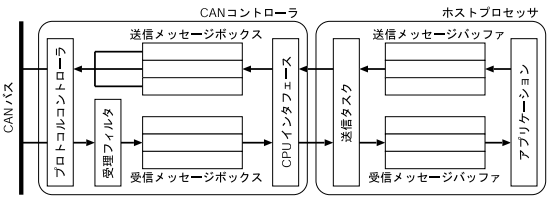

図 2 CAN ノードの通信モデル
Fig. 2 The communication model of a CAN node.

劣勢な信号を送信したノードはその状態を検知し自身の送信を中断する．これにより，優先度の高いメッセージがバス上に確実に送り出されることになる．このようなアービトレーションを正しく行うために，同じ ID を持つ 2 つ以上のメッセージが同時に送信されないように，メッセージにはユニークな ID が割り当てらなければならない．

CAN ではデータ転送に NRZ 方式を採用しており，複数のノード間での同期をとるために，同じ極性が 5 ビット連続するごとに逆の極性の 1 ビット（スタッフビット）が挿入される．挿入されるビット数はメッセージのビットパターンに依存し，それにともないメッセージサイズは変化する．たとえば，8 バイトのデータを持つ CAN メッセージは，47 の制御ビットを含めて，最大で 24 ビットのスタッフビットが付加されて転送される．

3. システム構成と動作

本章では，本論文で想定するシステム構成について述べる．一般的に使用されることが多い現実的なハードウェアやソフトウェアの構成とそれらの動作について言及する．

3.1 ハードウェア構成

本論文では，複数のノードが 1 つの CAN バスに接続されたシステムを考え，**図 2** に示すように，ノードはホストプロセッサと CAN コントローラから構成される．

コントローラは，プロトコルコントローラ，受理フィルタ，メッセージバッファ，ホストインタフェースから構成される．コントローラのメモリ上には，CAN メッセージを格納するためのメモリブロックであるメッセージボックスが用意されており，それらを送信用と受信用とに振り分けて使用される．

送信メッセージボックスが複数ある場合は，あらかじめノード内での調停が行われ，メッセージボックスに格納されたメッセージのうち最も高い優先度を持つ 1 つのメッセージのみが CAN バスのアービトレーションに参加できる．バスに送信を開始されたメッセージ

は，そのメッセージの送信が完了するまで，送信メッセージボックスに存在する．送信完了後，コントローラは，送信メッセージボックスが空になったことをホストプロセッサへ割込みを用いて通知する．受理フィルタは，バス上を流れるすべてのメッセージからそのノードが受け取るべきメッセージを選別し，受信メッセージボックスに格納し，受信したことをホストプロセッサへ割込みを用いて通知する．

3.2 ソフトウェア構成

ホストプロセッサでは，CAN コントローラを介して他のノードと通信を行う制御アプリケーションプログラムが動作している．制御アプリケーションプログラムは周期的にメッセージの送信要求を行う．

ホストプロセッサのメモリ上には，送信要求があったメッセージを格納する送信メッセージバッファと受信したメッセージを格納するための受信メッセージバッファがメッセージの種類ごとに固定的に確保されている．そのため，新しく到着した同じ種類のメッセージに上書きされない限り，メッセージがあふれてデータが消失することはない．

他のノードにメッセージを送信する場合は，送信メッセージバッファに送信したいデータを含んだメッセージを格納する．メッセージの送信要求ごとに起動される送信タスクは，コントローラの送信用メッセージボックスに空きがあれば，送信メッセージバッファ内で最も優先度の高いメッセージ送信用メッセージボックスに格納する．いったんメッセージボックスに格納されたメッセージは，その送信が完了するまで同じメッセージボックスに存在する．

複数のメッセージが送信メッセージボックスに格納されている場合，その中で最も優先度の高いメッセージが，前述したバスのアービトレーションに参加でき，メッセージの送信が開始される．送信メッセージボックスに空きが発生した場合，割込みを用いて送信タスクに通知し，新たなメッセージの格納を要求する．

受信され受理フィルタを通過したメッセージはコントローラ内の受信メッセージボックスに格納され，受信したことを割込みにより受信タスクに通知する．通知を受けた受信タスクは，受信メッセージボックスに格納されたメッセージをホストプロセッサ内の受信メッセージバッファに移動する．メッセージは，ホストプロセッサ内の受信メッセージバッファに格納された時点で，アプリケーションにより利用可能となる．

4. 従来手法

Tindell らは，CAN メッセージの最大遅れ時間を求める手法を示している[2)~4)]．この従来手法は，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[6)]である Rate Monotonic Analysis をメッセージのスケジューリングにおいて適用したものである．

本章では，従来手法の概要について述べた後，その問題点について指摘する．従来手法では，前節で示したようなシステム構成は考慮されておらず，メッセージボックスの数に制限のない理想的なモデルの下で議論されていることに注意されたい．

4.1 従来手法の概要

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メッセージ集合を考える．メッセージ $i(1 \leq i \leq n)$ は，3 つ組 (P_i, T_i, C_i) のパラメータを持つ．ここで，P_i，T_i はそれぞれメッセージ i の優先度（ID に相当し，小さいほど優先度が高い），送信周期を表している．C_i はメッセージ i を送信するのに要する最大時間である．先に述べたように，CAN はメッセージごとに 47 ビットの制御ビットを持ち，5 ビットごとにスタッフビットが新たに加えられる．47 ビット中 34 ビットとデータがビットスタッフィングの対象となるため，s_i をメッセージに含まれるデータのバイト数とすると，C_i は次のように定義される．

$$C_i = \left(\left\lfloor \frac{34 + 8s_i}{4} \right\rfloor + 47 + 8s_i \right) \tau_{bit} \quad (1)$$

メッセージ i の最大遅れ時間 R_i，つまり，メッセージの送信要求からメッセージの送信が完全に完了するまでの時間は，次の式で表すことができる．

$$R_i = J_i + Q_i + C_i \quad (2)$$

J_i はメッセージ i のキューイングジッタで，メッセージの送信を要求するタスクの中で，送信要求を出すことが可能な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 のキューイング遅れ時間で，メッセージが送信メッセージバッファに格納されてから実際に送信が開始されるまでの時間を示している．この時間は他のメッセージにより邪魔される時間を含み，次のように表される．

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

ここで，B_i は最大ブロッキング時間で，メッセージ i よりも優先度の低いメッセージに邪魔される時間を表している．メッセージ i はその送信が要求されると同時に送信を開始した低い優先度を持つメッセージ以外に邪魔されることはない．つまり，優先度の低い

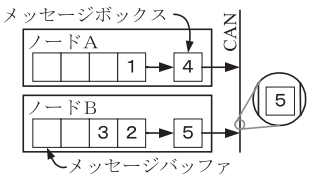

図 3 実際には起こりうる現象
Fig. 3 A realistic situation in CAN.

メッセージにはたかだか 1 回しか邪魔されないため，B_i は低優先度メッセージの中で最も長いメッセージを転送するために要する時間と一致し，次のように表される．

$$B_i = \max_{\forall j \in lp(i)} C_j \tag{4}$$

$lp(i)$ はメッセージ i よりも優先度の低いメッセージの集合を表している．また，τ_{bit} は，バス上で 1 ビット転送するのに必要な時間を表している．メッセージ i が最低優先度のメッセージである場合，$B_i = 0$ となる．

4.2 従来手法の問題点

従来手法は，図 2 に示すような送信メッセージボックスの数に制約のない理想的なモデルの下に議論が行われている．いい換えると，システム上のすべてのメッセージが，その送信要求と同時にメッセージボックスに格納され，優先度に従い送信されるモデルを用いている．

しかしながら，実際のシステムでは，送信メッセージボックスが十分にあるという従来手法での前提は必ずしも正しいとはいえない．一般的には，メッセージの受信漏れとリアルタイム性の弊害となる再送を抑えるために，コントローラ内のほとんどのメッセージボックスは受信用として使われ，送信用としては数個程度しか割り当てられないからである．ノード内の送信メッセージボックスの数が，そのノードから送信されるメッセージの数よりも小さい場合，送信メッセージボックスでの優先度逆転現象が発生する可能性がある．優先度逆転現象[7]は，高い優先度を持つメッセージの送信が低い優先度を持つメッセージに邪魔される状態であり，コントローラ内のバッファで発生することが知られている[8]．優先度逆転はメッセージの遅れ時間に際限のない遅れを生じさせ，その予測可能性を損なわせる．

図 3 は，メッセージボックスの数が 1 である場合について，従来手法が考慮していない現実のシステムで起こりうる状況を示している．メッセージボックスの数が 1 である場合に，5 つのメッセージの送信がほぼ同時に（最初に 5 が，続いて 4，1，2，3 が）要求された状況を考える．それぞれのバッファとボックス内の数字はメッセージの優先度（小さいほど優先度が高い）を表している．最初に送信要求のあったメッセージ 5 がバスに送信されているため，それよりも優先度の高いメッセージ 4 は 5 の送信が完了するまで待たされることになる（バスでの優先度逆転）．また，メッセージ 1 は 4 の送信が完了するまで送信することはできない（メッセージボックスでの優先度逆転）．メッセージバッファからメッセージボックスへの転送遅れがないとすると，結果的に 5，2，3，4，1 の順に送信され，最高優先度であるメッセージ 1 はすべての低優先度のメッセージに待たされることになる．

従来手法は，このようなメッセージボックスが十分にない状況を取り扱うことはできないため，従来手法を用いて，メッセージの時間制約を満たすために必要なメッセージボックス数を見積もることはできない．

5. アーキテクチャを考慮した解析手法

従来手法の問題を解決するためには，ノードのアーキテクチャを考慮し，また，対象とするメッセージを送信するノードとそれ以外のノードを明確に区別して取り扱う必要がある．

そこで，本論文では，メッセージボックス数が限られたより現実的なシステムモデルに対して，CAN メッセージの最大遅れ時間を求めるための解析手法を提案する．本手法は，メッセージが優先度ベースで，かつ，周期的に送信されるようなバス型ネットワークを前堤としており，この前堤が満たされていれば，CAN 以外のネットワークにも適用できる可能性はある．

本章では，3 章に示したシステム構成に対して，従来手法と同じメッセージモデルを用いて提案手法を構築する．本論文中では，メッセージの遅れ時間をメッセージの送信要求が発生してからその送信が完了するまでの時間とする．

5.1 通信モデルと表記法

通信エラーが発生した場合の挙動を含めた議論については本論文では対象とせず，つねに正常に通信が行われていると仮定し，CAN のデータフレームをメッセージとして扱う．

提案手法では，対象のメッセージを送信するノードとそれ以外のノードを区別するため，メッセージを送信するノードをそのメッセージに対する自ノードと呼び，それ以外のノードを他ノードと呼ぶ．また，メッセージ i の自ノードの中でより高い優先度を持つメッセージの集合を $shp(i)$ と表す．同様に，自ノードの

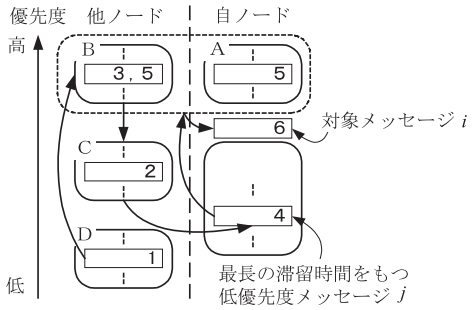

図 4 メッセージ i の最大遅れ時間
Fig. 4 A maximum responce time of a message.

中でより低い優先度を持つメッセージの集合を $slp(i)$，他ノードの中でより高い優先度を持つメッセージの集合を $ohp(i)$，他ノードの中でより低い優先度を持つメッセージの集合を $olp(i)$ と表記する．

5.2 メッセージボックス数が 1 である場合

まず，送信メッセージボックスの数を 1 とした場合において，メッセージの最大遅れ時間について考える．

5.2.1 Critical Instant 定理

あるメッセージに対する critical instant とは，その最大遅れ時間が最も長くなるような状況である．メッセージの遅れ時間が最も長くなるのは，バス上と自ノードのメッセージボックス内とで最も長く優先度逆転が発生するような場合である．たとえば，図 3 で示した状況は，メッセージ 1 にとって優先度逆転が最も長くなり，遅れ時間も最大となる critical instant を示している．

本項では，critical instant 定理を導くが，その前に**図 4** を用いて，より一般的なメッセージに対する直感的な critical instant について述べる．図中の長方形はメッセージを表しており，その中の数と矢印はバスに送信される順番と流れを示している．メッセージ i の遅れ時間が最大となるのは，その送信要求と同時に，自ノードの低優先度メッセージである j がメッセージボックスに格納された状態で，j よりも優先度の低いメッセージ（図 4 の D）のうちの 1 つがバスへの送信を開始した場合であると考えられる．最初のメッセージ（D）と j より優先度の高い他ノードのメッセージ（図 4 の B，C）のすべてに邪魔された後に，j は送信される．その送信が完了すれば，i よりも優先度高いメッセージ（図 4 の A，B）のすべてが送信された後に，メッセージ i が送信される．

Critical instant 定理を証明する前に，別の定理を示す．以下の定理は，あるメッセージがメッセージボックスに格納されてから送信を完了するまでの時間（以下，滞留時間）が最大となる状況を示している．

Tindell らが示した従来手法[2)~4)]は，定理 1で示すような状況が critical instant であることを前提として，メッセージの最大遅れ時間を求めている（ただし，自ノードと他ノードとの区別はしていない）．しかしながら，Tindell らは，このような状況でメッセージの遅れ時間が最大となることの証明を示していないため，以下で証明を行う．なお，ここで用いる証明手法は，類似の証明に広く用いられている方法である[9),10)]．

定理 1 メッセージボックス数が 1 の場合，メッセージ i の滞留時間は，メッセージボックスに格納されると同時に，次の条件をすべて満たすときに最大となる．（1）メッセージ i より優先度の高い他ノードのすべてのメッセージの送信要求が発生し，（2）それらの最初の送信要求は最大のジッタを持ち，それ以降に続く送信要求のジッタは 0 となる場合で，かつ，（3）メッセージ i よりも優先度の低い他ノードのメッセージの中で最も長い送信時間を持つメッセージのバスへの送信が開始された．

証明 1 いったんメッセージボックスに入ったメッセージは，その送信が完了するまで同じメッセージボックスに存在し続けるため，メッセージボックス数が 1 の場合，自ノードのメッセージに邪魔されることはない．つまり，メッセージ i の滞留時間は，そのメッセージ自身と他ノードのメッセージのみについて考えればよい．

以下では，（1）〜（3）の条件を満たすように，順に条件を課した場合のメッセージ i の滞留時間の変化を考える．それぞれの過程において，滞留時間が短くならないことを示すことにより，最終的に（1）〜（3）の条件を満たした場合において，滞留時間が最大となることを示す．

（1）メッセージ i がメッセージボックスに時刻 t^i で格納されたとき，時刻 t^i_{end} にその送信を完了すると仮定する．

t^i 以前で，より高い優先度を持つメッセージのいずれも送信されていない最も遅い時刻を t_0 とする．時刻 0 においては，どのメッセージも送信されていないので，t_0 は必ず存在する．メッセージ i がメッセージボックスに格納される時刻を t_0 に変更したとしても，t_0 と t^i の間は少なくとも 1 つのより高い優先度を持つメッセージが送信されているため，依然としてメッセージ i は t^i_{end} に終了する．メッセージ i より高い優先度を持つすべてのメッセージの t_0 以降の最初の送信要求を t_0 に移動したとする．このとき，これらのメッセージがメッセージ i を邪魔する回数は変わら

ないか増加するため，メッセージ i の送信完了時刻は，$t^i_{end}{}'(\geq t^i_{end})$ に遅延される可能性がある．

（2）さらに，これらの各メッセージ j の t_0 以降の最初の送信要求が最大のジッタ J_j を持ち，それ以降の送信要求にはジッタがないものと仮定する．ここで，最初の送信要求時刻を t_0 から $t_0 - J_j$ に移動したとする．このとき，これらのメッセージがメッセージ i を邪魔する回数は変わらないか増加するため，メッセージ i の送信完了時刻は $t^i_{end}{}''(\geq t^i_{end}{}')$ に遅延される可能性がある．

（3）メッセージは優先度に従って送信されるため，$(t_0, t^i_{end}{}'']$ の間では，i よりも優先度の低いメッセージが送信されることはない．

しかしながら，送信はノンプリエンプティブに行われるため，i よりも優先度の低い他ノードのメッセージは，時刻 t_0 に送信を開始した場合のたかだか 1 回に限り，送信することができる．メッセージ i は，少なくとも，その低優先度のメッセージの送信時間は遅れさせられるため，t_0 において，他ノードの低優先度メッセージの中で最も長い送信時間を持つメッセージのバスへの送信が開始された場合に，メッセージ i の送信完了時刻が最も遅れさせられる．

このとき，メッセージ i は任意に設定した最初の状況と同じかより長い遅れ時間を持つ．よって，メッセージ i の送信が時刻 t_0 で要求された場合に最大遅れ時間を持つ． □

定理 1より，各メッセージの最大滞留時間は，（1）〜（3）の条件から始めたスケジュールを作ることで求めることができる．

定理 2（Critical Instant 定理） メッセージボックス数が 1 の場合，メッセージ i に対する critical instant は，その送信要求と同時に次の 4 つの条件をすべて満たす状況である．（1）メッセージ i よりも優先度が低い自ノードのメッセージの中で，最長の滞留時間を持つメッセージ l がメッセージボックスに格納され（ただし，メッセージ l が存在しないときは，この条件はなくなる），（2）メッセージ l より優先度の高いすべてのメッセージの送信要求が発生し，（3）それらの最初の送信要求は最大のジッタを持ち，それ以降に続く送信要求のジッタは 0 で，（4）メッセージ l よりも優先度の低い他ノードのメッセージの中で最も長い送信時間を持つメッセージのバスへの送信が開始された．

証明 2 メッセージ i の送信要求が時刻 t^i で発生したとき，時刻 t^i_{end} にその送信を完了すると仮定する．また，メッセージ i より優先度の低い自ノードのメッセージ l が，t^i 以前の時刻 t^l でメッセージボックスに格納され，t^l_{end} にその送信を完了し，t^l から t^i の間には，l 以上の優先度のメッセージが送信されているものとする．t^i 以前で，どのメッセージも送信されていない最も遅い時刻を t_0 とする．時刻 0 においては，どのメッセージも送信されていないので，t_0 は必ず存在する．

（2）メッセージ l より優先度の高いメッセージを 3 つの場合（図 4 の A，B，C）に分けて，どの場合においても，メッセージ i の送信完了時刻が早くならないことを示す．

（2a）メッセージ i よりも優先度の高い自ノードのメッセージ（図 4 の A）の t_0 以降の最初の送信要求を t_0 に移動したとき，メッセージ i が邪魔される回数は変わらないか増えるため，メッセージ i の送信完了時刻は遅延させられる可能性がある．

（2b）メッセージ i よりも優先度の高い他ノードのメッセージ（図 4 の B）の t_0 以降で最初の送信要求を t_0 に移動したとき，メッセージ l や i が邪魔される回数は変わらないか増加するため，メッセージ l の滞留時間は長くなる可能性があり，また，メッセージ i の送信完了時刻は遅延させられる可能性がある．

（2c）メッセージ i よりも優先度が低く，メッセージ l よりも優先度の高い他ノードのメッセージ（図 4 の C）の t_0 以降で最初の送信要求を t_0 に移動したとする．メッセージ l を邪魔する回数は変わらないか増加するため，メッセージ l の滞留時間は長くなる可能性がある．また，メッセージ i はこれらのメッセージ（図 4 の C）に直接的に邪魔されないが，l の滞留時間が長いほど，C のメッセージに邪魔される回数は変わらないか増加するため，メッセージ i の送信完了時刻は遅延されられる可能性がある．

定理 1の証明で用いた同様の方法で，（3）と（4）の条件を満たすような状況において，メッセージ i の遅れ時間は短くならないことを証明できるため，ここでは省略する．

（1）ここで，メッセージ l をメッセージ i よりも優先度が低いメッセージの中で最長の滞留時間を持つメッセージに置き換える．このとき，メッセージ i の送信完了時刻が，少なくとも長くなった滞留時間の分，遅れさせられることはあっても早くなることはない．

このとき，メッセージ i は任意に設定した最初の状況と同じかより長い遅れ時間を持つ．よって，メッセージ i の送信が時刻 t_0 で要求された場合に最大遅れ時間を持つ． □

Critical instant 定理より，critical instant から始

まるスケジュールのみを考えることで，メッセージの最大遅れ時間が求めることができる．

5.2.2 解析手法

ここでは，前項で示した critical instant 定理を用いて，各メッセージの最大遅れ時間を求める手法を示す．

まず，メッセージ i の遅れ時間を最大にする自ノードのメッセージ l の最大滞留時間を R'_l を求める．R'_l が最大となる状況は，送信メッセージボックスに格納されたと同時に，より低い優先度を持つ他ノードのメッセージの中で，最も長い送信時間を持つメッセージがバスへの送信を開始し，かつ，他ノードのすべての高優先度メッセージの送信要求が発生した場合である．よって，R'_l は，メッセージ l のメッセージボックス内で滞留している時間 Q'_l と送信時間 C_l を用いて，次のように表される．

$$R'_l = Q'_l + C_l \tag{5}$$

メッセージボックス内のメッセージ l は自ノードの高優先度メッセージには邪魔されることはなく，他ノードの高優先度メッセージのみに邪魔されるため，Q'_l は次のように表すことができる．

$$Q'_l = B'_l + \sum_{\forall j \in ohp(l)} \left\lceil \frac{Q'_l + J_j + \tau_{bit}}{T_j} \right\rceil C_j \tag{6}$$

B'_l はメッセージ l がより優先度の低いメッセージに邪魔される時間である．自ノードのメッセージには邪魔されず，他ノードのメッセージにのみ邪魔されるため，式 (4) を次のように書き換えることができる．

$$B'_l = \max_{\forall j \in olp(l)} C_j \tag{7}$$

次に，メッセージ i の遅れ時間が最大となるのは，その送信要求が出されたと同時に，より優先度の高いすべてのメッセージの送信要求が発生した場合で，かつ，送信用メッセージボックスに，自ノードの最長の最大滞留時間を持つメッセージ l が格納された場合である．式 (2) と同様に，メッセージ i の最大遅れ時間 R_i は次式のように表される．

$$R_i = J_i + Q_i + C_i \tag{8}$$

また，キューイング時間 Q_i も式 (3) と同様に書ける．

$$Q_i = B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{Q_i + J_j + \tau_{bit}}{T_j} \right\rceil C_j \tag{9}$$

ここで，B_i は i よりも優先度の低いメッセージの存在により，そのメッセージも含めて他のメッセージに直接的，または間接的に邪魔される時間を表している．単純には，B_i はメッセージ l の送信要求バッファに格納されてからの遅れ時間 R'_l であると思われるが，そうではない．メッセージ i は，メッセージ l の存在に関係なく，メッセージ i よりも優先度の高いメッセージに邪魔される．そのため，Q'_l の時間にメッセージ i より優先度の高いメッセージがメッセージ l を邪魔した重複する時間を B_i から除外する必要がある．よって，B_i は次式のように表すことができる．

$$B_i = R'_l - \sum_{\forall j \in ohp(i)} \left\lceil \frac{Q'_l + J_j + \tau_{bit}}{T_j} \right\rceil C_j \tag{10}$$

メッセージ i 自身が自ノードの最低優先度メッセージである場合は，$B_i = B'_l$ として，式 (5) で重複する送信時間を除くいて計算する．

5.2.3 最悪シナリオシミュレーション

前項でのメッセージ i がいったんメッセージボックスに格納された後は，自ノードの高優先度メッセージに邪魔されることはないにもかかわらず，前項の式（以下，提案式）では，これを邪魔されるものと扱っている．そのため，提案式は，メッセージが最大遅れを持つための十分条件であり検証としては安全側にはあるが，必要十分条件ではないため悲観的に評価してしまう可能性がある．修正して必要十分な式を記述することは可能であるが，非常に複雑な形になり解析時の見通しが悪くなる．

そこで，本論文では必要以上に煩雑な式を記述することを避け，提案式がメッセージの最大遅れを求めるのに実用上問題ないことを示すために，CAN のネットワークシミュレータの結果と比較を行った．シミュレータを用いることで，式では表現しにくい条件を含めた場合についても，メッセージの最大遅れ時間を正確に求めることができる．

具体的には，シミュレータに 5.2.1 項で critical instant として示した最悪シナリオの初期値を与えて動作させ，各メッセージの最大遅れ時間を求めることを行う．シミュレーションの具体的な動作を図 3 を用いて説明すると，同図で示した状況は，メッセージ 1 にとっての critical instant であり，メッセージ 1 の最大遅れ時間を求めるために，シミュレータに初期状態としてこのような状況を与える．この状態から，シミュレータは各メッセージが順に送られていく様子をメッセージ 1 の送信完了までシミュレーションし，その最大遅れ時間を求める．

このような最悪シナリオの初期値から始めたシミュレーションを，以降，最悪シナリオシミュレーション

と呼ぶ☆.

このように，シミュレーションにより求めた最大遅れ時間との比較を行い，提案式で求めた最大遅れ時間が良い近似解となっていることを示せれば，提案式は実用上において問題ないといえる．6 章において，この最悪シナリオシミュレーションを行い，前項で示した式がメッセージの最大遅れ時間を求めるのに実用上問題ないことを示す.

5.3 メッセージボックス数が 2 以上の場合

以上では，メッセージボックスが 1 である場合の検証手法について述べた．ここでは，送信メッセージボックスの数が 2 以上の場合のより一般的な場合においても，提案式を用いてメッセージの最大遅れ時間を求めることができることを示す．また，提案手法が従来手法の一般化になっていることを示す.

あるメッセージ i の遅れ時間は，自ノードのより優先度の低いメッセージ数とメッセージボックス数の関係に影響されるため，それぞれの場合に分けて考える.

まず，メッセージボックス数が自ノードの低優先度メッセージ数以上である場合を考える．これは従来手法の前提であり，メッセージ i は，低優先度メッセージにより送信メッセージボックスへの格納を邪魔されることはなく，優先度逆転は発生しない．メッセージ i は自ノードの最低優先度メッセージと見なし，提案式を用いることでメッセージ i の最大遅れ時間を求めることができる．このとき，提案式は従来手法で示された式と一致し，より一般化されたものであるといえる.

一方，メッセージボックス数が自ノードの低優先度メッセージ数よりも小さいとき，つまり十分なメッセージボックスがないとき，最悪の場合においては，優先度逆転が発生し，メッセージ i はそれらのメッセージに送信メッセージボックスへの格納を邪魔される.

メッセージボックスの数を m とすると，メッセージ i の送信要求と同時に，すべてのメッセージボックスに（m 個の）低優先度メッセージが格納された場合に，i の遅れ時間は最大となる可能性がある．このとき，どの低優先度メッセージがメッセージボックスに格納されるかが問題となるが，どの組合せにおいても，優先度の低い方から $(m-1)$ 番目までのメッセージはメッセージ i を邪魔することはできない．なぜなら，優先度の低い方から m 番目以上のメッセージがメッセージボックスに存在し，メッセージ i はそのメッセージに邪魔されることはあっても，いったん送信された後は，i よりも優先度の低いメッセージに邪魔されることはないからである.

つまり，メッセージ i は，自ノードの優先度の低い方から $(m-1)$ 個を除いた残りの低優先度メッセージの中で，最大の滞留時間を持つメッセージ k により邪魔されたとき，最大遅れ時間を持つことがいえる．このときのメッセージ i の最大遅れ時間は，提案式のメッセージ l をここでのメッセージ k に置き換えることで求めることができる.

$m = 1$ とした場合，k は自ノードの最大の滞留時間を持つ低優先度メッセージを示しており，メッセージボックスの数が 1 である場合の式と一致する．つまり，これらの式はメッセージボックスの数による影響を示すより一般的な式となっているといえる.

従来手法との大きな違いとして，提案手法は自ノードの低優先度メッセージにより他ノードの低優先度メッセージに間接的に邪魔される影響が含まれていることがあげられる．そのため，各ノードの最低優先度メッセージの最大遅れ時間は両者で一致する.

6. 適　　用

自動車メーカから提供されたメッセージセットに対して提案手法を適用し，メッセージの最大遅れ時間を求め，それらが時間制約を満たすかどうかを評価した．また，従来手法や最悪シナリオシミュレーションとの結果を比較し，提案手法の有用性と妥当性を評価した.

6.1 適 用 対 象

CAN では最高 1 Mbps のデータ伝送が可能であるが，実際に高速な制御系のネットワークなどに使用される場合には，電磁放射ノイズや安定性などを考慮して，通常その半分の 500 Kbps 以下で用いられる☆☆．そのため，以降の適用においては，CAN の転送速度を 500 Kbps と設定する.

適用の対象としたのは，実際のシステムでの使用が検討されている，10 前後のノード間を合計で約 30 種類の周期的なメッセージが通信されるメッセージセットである.

このメッセージセットに対して，提案手法を用いて，すべてのノードのメッセージボックス数が 1 である場合について，各メッセージの最大遅れ時間を求めた.

☆ 一般的に，シミュレーションはシミュレータにさまざまなパラメータをランダムに与えて行われるが，5.2.1 項の critical instant 定理により最悪シナリオの初期値が分かるため，この場合だけをシミュレータに与えて動作させれば，メッセージの最大遅れ時間を求めることができる.

☆☆ 低速でも問題ない場合には，さらに遅い 250 Kbps や 125 Kbps で用いられる.

その結果，すべてのメッセージの時間制約が満たされることが確認できた．

6.2 従来手法との比較

全ノードのメッセージボックス数を 1 とした場合のメッセージの最大遅れ時間について，提案手法と従来手法との比較を行った．ただし，従来手法ではメッセージボックス数に制約を持たせることはできないため，メッセージボックスが十分にあるという前提で行っている．そのため，以下では，従来手法を用いて求めた最大遅れ時間は，提案手法に対する理想的な指標として用いている．

提案手法は優先度逆転による影響を考慮していることが従来手法とは異なる．そのため，メッセージの最大遅れ時間に対する優先度逆転の影響に注目して従来手法との比較を行った．具体的には，メッセージを大きく，優先度（高優先度，中優先度，低優先度）で 3 つのグループに分類し，それぞれの最大遅れ時間について比較を行った．簡単のため，以降の表中では，それらのグループをそれぞれ高メッセージ，中メッセージ，低メッセージと表記する．

各メッセージの最大遅れ時間を**表 1** に示す．以降では，従来手法の最大遅れ時間を 1 として正規化した結果を用い，提案手法での最大遅れ時間を従来手法での最大遅れ時間で割った値を伸び率と呼ぶ．

高い優先度のメッセージであるほど，低優先度メッセージに大きく影響され，同じノード内にあるメッセージの優先度が低いほど，最大遅れ時間が長くなる結果が得られた．最も大きい伸び率 8.45 となったのは，メッセージセットの中で最高優先度を持つメッセージである．これは，このメッセージと同じノード内に非常に優先度の低いメッセージがあり，そのメッセージによる優先度逆転の影響が大きいためである．

低優先度メッセージに関しては，優先度逆転の影響が小さく，従来手法と大きな違いは見られなかった，メッセージ全体としては約 2 倍の伸び率が得られた．

これにより，従来手法ではメッセージボックス数に制約があるような実際のシステムについて，メッセージの真の最大遅れ時間が求められておらず，実際のシステムを評価するために用いることはできないことが確認できた．

6.3 最悪シナリオシミュレーションとの比較

提案式の有用性を示すために，5.2.3 項で述べた最悪シナリオシミュレーションによりメッセージの最大遅れ時間を求め，提案式を用いて求めた結果との比較を行った．提案式は十分条件だけを満たしているため，一般には，最悪シナリオシミュレータを用いた最大遅れ時間は，提案式を用いた場合と同じか短くなる．しかしながら，今回用いたメッセージセットでは，すべてのメッセージの最大遅れ時間が両者の間で一致し，提案式を用いて良い近似解が得られることが示せた．これにより，ただちに提案式がすべてのメッセージセットに対して有用であるという結論は導けないが，少なくとも今回の適用においては提案式を用いて妥当な結果が得られることが示されたといえる．提案手法をより多くのシステムに適用して評価することは今後の課題である．

最悪シナリオシミュレーションを行った場合，正確にメッセージの最大遅れ時間を求めることができるが，各時刻でのシステム状態を更新しながら動作するため，提案式を用いて求める場合に比べて，評価に非常に多くの時間を要する．そのため，一次評価として提案式を用いメッセージがその時間制約を満たすかどうかの評価を行い，時間制約を満たせないときなどより詳細な評価が必要な場合に，シミュレーションを行うことが妥当であると考えている．

7. メッセージボックス数の最適化

ここでは，提案式を用いてメッセージの最大遅れ時間を小さく抑えるための方法について述べる．

7.1 メッセージボックス数と最大遅れ時間

メッセージボックス数によるメッセージの最大遅れ時間の関係を調べるために，全ノードのメッセージボックス数を増やした場合について，提案式を用いて最大遅れ時間を求めた．結果を**表 2** に示す．

結果より，メッセージボックス数を増やすことにより，全体的に最大遅れ時間が短くなり，特に高優先度

表 1 提案手法を用いた最大遅れ時間
Table 1 Maximum response times of messages using our method.

	伸び率の範囲	平均伸び率
高メッセージ	1.00〜8.45	2.77
中メッセージ	1.00〜1.67	1.24
低メッセージ	1.00〜1.06	1.01
全メッセージ	1.00〜8.45	1.84

表 2 メッセージボックス数の効果
Table 2 The effect of number of message boxes.

	メッセージボックス数			
	1	2	3	10
高メッセージ	2.77	2.04	1.38	1.00
中メッセージ	1.24	1.08	1.04	1.00
低メッセージ	1.01	1.01	1.00	1.00
全メッセージ	1.84	1.47	1.18	1.00

表 3 優先度に従ったメッセージ割当てによる効果
Table 3 The effect of message assignments according to their priorities.

	メッセージボックス数		
	1	2	3
高メッセージ	2.77	1.27	1.09
中メッセージ	1.24	1.12	1.12
低メッセージ	1.01	1.01	1.01
全メッセージ	1.84	1.16	1.09

メッセージに効果があることが確認できる．しかしながら，メッセージボックス数が 1 の場合に 8.45 の伸び率を持つ最高優先度メッセージは，メッセージボックスが 2 の場合でも 6.82 倍，3 の場合でも 3.41 倍と他の高優先度メッセージに比べて，効果が低い結果が得られた．これは，他のメッセージに比べて，同じノードに多くの低優先度メッセージが存在しており，いくつかのメッセージボックスを追加しても，低優先度メッセージによる影響が依然として強いためである．したがって，このような場合，単純にメッセージボックス数を増やすことは，メッセージの最大遅れ時間を平均的に短くすることはできるが，メッセージボックス数に比べて，多くの低優先度メッセージを同じノードに持つメッセージには，効果が現れにくいことが分かる．

7.2 メッセージボックスの分割による効果

高優先度メッセージの最大遅れ時間が従来手法に比べて大幅に長くなるのは，自ノードの低優先度メッセージに邪魔される影響が大きいからである．この影響を小さくするために，どのメッセージボックスに格納して送信するかをメッセージの優先度ごとに分割する手法が考えられる．たとえば，メッセージボックスが 2 つある場合，一方を高い優先度のメッセージ用として割り当て，もう 1 つを低い優先度のメッセージ用として割り当てる．この方法は，1 つのノードを 2 つの異なるノードに分割したことと等価である．これにより，高い優先度のメッセージが低い優先度のメッセージの影響を受けにくくなり，効果的に最大遅れ時間を小さく抑えることができる．

メッセージの最大遅れ時間が小さく抑えるようにメッセージ分割をした場合の最大遅れ時間の平均伸び率を**表 3** に示す．表中で，メッセージボックス数が 3 の場合は，単純にメッセージボックス数を増やしても効果が低かった最高優先度メッセージのみに専用のメッセージボックスを割り付けた結果であり，その他の条件は，メッセージボックス数が 2 の場合と同じである．

結果として，最大遅れ時間を各メッセージの平均で，理想の最大遅れ時間から 1 割増に，最大でも 3 割増に抑えることができた．このように優先度に従って格納するメッセージボックスを決定することで，メッセージボックスを効率的に利用できると同時に，メッセージの最大遅れ時間を小さく抑えることができる．また，本論文で提案した手法を用いることで，それぞれのメッセージをどのメッセージボックスに割り当てるかについて，最適な分割方法を与えることができる．

8. 今後の課題

本論文では，メッセージバッファからメッセージボックスへの転送時間は 0 として扱ってきたが，実際には，送信が完了しメッセージボックスが空になってから，次のメッセージが格納されるまでの時間がわずかながらでも必要である．そのため，この時間に他ノードのメッセージボックスにあるメッセージに先に送信される可能性がある．たとえば，図 3 のメッセージ 2 は，メッセージ 5 が送信されてからすぐには送信できず，メッセージ 4 が先に送信されることになる．

このように，転送時間を 0 としない場合，あるメッセージがメッセージバッファからメッセージボックスへ転送されるときに，より優先度の低いメッセージが先に送信される場合が問題となる．メッセージボックス数が 1 の場合，最悪の場合では，そのメッセージ自身を含め，より優先度の高い自ノードのメッセージの送信要求が発生するごとに，より優先度の低いすべてのメッセージに先に送信される可能性がある．ただし，これは転送時間がメッセージの最短送信時間よりも短く，転送している間に 1 メッセージにしか割り込まれないものとした場合であり，もしそうでない場合，より多くのメッセージに割り込まれる可能性がある．

メッセージボックスが複数ある場合は，空いたメッセージボックスに新しいメッセージを転送中に，同じノードの他のメッセージボックスにあるメッセージがアービトレーションに参加できる可能性がある．そのため，メッセージボックス数が 1 の場合に比べて，メッセージボックスへの転送中に割り込まれる影響は小さくなると考えられる．

このような時間的なすきまを考慮した問題は critical instant 定理が成立せず，一般的に難しいことが知られている．この時間を含めた，より現実的なモデルに対する解析手法は今後の課題としてあげられる．

また，今回適用したメッセージセット以外にも適用し，本手法が有効であることを示すことや，評価手法をベースとしたノードやメッセージの最適なパラメータ付け，およびスケジューリング方法に関しては今後の課題である．

9. 結　　論

本論文では，従来手法では実際のシステムを検証するためには不十分であることを指摘し，システム構成を考慮した場合のメッセージの最大遅れ時間を求める手法を新たに提案した．また，実際のシステムでの使用が検討されているメッセージセットに対して提案手法を適用し，その有効性を確認するとともに，すべてのメッセージの時間制約が満たされることが確認した．

そのメッセージセットを用いて，従来手法ではメッセージの真の最大遅れ時間が求められず，実際のシステムを評価するために用いることはできないことを示した．また，メッセージボックスの数を変化させた場合と，提案手法を用いて，優先度に従って格納するメッセージボックスを分割した場合の最大遅れ時間を求めた．これにより，本論文で提案した手法を用いることで，メッセージの最大遅れ時間を小さく抑えるような最適なメッセージの割当てを行い，限られたリソースを効率的に利用するための指針を与えうることを示した．

謝辞　適用評価において，データを提供いただいたトヨタ自動車（株）BR–制御システム開発室に感謝いたします．また，研究を進めるにあたりご意見をくださった関係各位に御礼申し上げます．

参 考 文 献

1) International Standards Organization: ISO 11898, Road Vehicles — Interchange of digital information — Controller area network (CAN) for high speed communication (1993).
2) 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).
3) Tindell, K., Hansson, H. and Wellings, A.J.: Analysing Real-Time Communications: Controller Area Network (CAN), *Proc. 15th IEEE Real-Time Systems Symposium* (1994).
4) Tindell, K.W., Burns, A. and Wellings, A.J.: Calculating Controller Area Network (CAN) Message Response Times (1995).
5) Nolte, T., Hansson, H. and Norström, C.: Probabilistic Worst-Case Response-Time Analysis for the Controller Area Network, *Proc. 8th IEEE Real-Time and Embedded Technology and Application Symposium* (2002).
6) Audsley, N.C., Buns, A., Richardson, M., Tindell, K. and Wellings, A.J.: Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling, *Software Engineering Journal*, Vol.8, No.5, pp.284–292 (1993).
7) Sha, L., Rajkumar, R. and Lehoczky, J.: Priority Inheritance Protocols: An Approach to Real-Time Synchronization, *IEEE Trans. Comput*, Vol.9, No.39, pp.1175–1185 (1990).
8) Marchok, T.E., Strosnider, J.K. and Tokuda, H.: Token-Ring Adapter-Chipset Architectural Considerations for Real-Time Systems, *Proc. 10th IEEE Real-Time Systems Symposium*, pp.79–91 (1989).
9) Liu, C.L. and Layland, J.W.: Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment, *J. ACM*, Vol.20, No.1, pp.46–61 (1973).
10) Mok, A.K. and Chen, D.: A multiframe model for real-time tasks, *Proc. Real-Time Systems Symposium*, pp.22–29 (1996).

(平成 15 年 5 月 9 日受付)
(平成 15 年 9 月 10 日採録)

飯山　真一

2002 年豊橋技術科学大学大学院情報工学専攻修士課程修了．現在，同大学院電子・情報工学専攻に在学．リアルタイムスケジューリング理論とその応用に関する研究に従事．修士（工学）．

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．同学科の助手，豊橋技術科学大学情報工学系助教授等を経て，2003 年 4 月より現職．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．ITRON 仕様の標準化活動に，中心的メンバとして参加．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．