FlexRay のダイナミックセグメントにおけるメッセージの最大遅れ時間解析

村上靖明[†] 村上尚彦[†] 高田広章[†]

現在，自動車内の制御系ネットワークにおいては，新しいプロトコルである FlexRay が注目を浴びている．FlexRay を用いてメッセージの送信を行う場合には，スタティックセグメントとダイナミックセグメントのどちらかが用いられる．そのうち，後者を用いる場合には，バスを有効に使えるというメリットがあるが，リアルタイム性を保証するのが難しい．そこで，本研究では，ダイナミックセグメントを安全に使用するための解析を行う．最初に，遅れ時間がより長くなる条件を整理する．それをもとに，メッセージの最大遅れ時間の解析手法を提案する．この手法をランダムに作成した複数のメッセージセットに対して適用し，この手法の評価を行った．

Response Time Analysis for Messages in Dynamic Segment of FlexRay

Yasuaki Murakami,[†] Naohiko Murakami[†] and Hiroaki Takada[†]

Nowadays, FlexRay attracts attention as a new protocol in automotive networks for control. Messages are transmitted in a static segment or a dynamic segment. In the latter case, it is difficult to guarantee real-time requirements though there is an advantage that the bus can be effectively used. In this research, we analyze for using dynamic segment safely. Firstly, we organize the conditions that response time is longer. And we proposed a method to analyze the worst-case response time of messages under these features. We applied it to message sets which are made randomly, and evaluated it.

1. はじめに

現在，自動車内の制御系ネットワークにおいては，1990 年代に提案されたバス型ネットワークの CAN (Controller Area Network)[1] が，事実上の標準となっている．しかし，自動車に搭載される機能の増加にともない，ネットワークトポロジの複雑化や通信データ量の増大などの問題が表面化してきた．このような問題に対応するため，FlexRay というプロトコルが新たに提案され注目を浴びている．

FlexRay の特徴としては，あらかじめ定義した送信時刻でデータを転送するタイムトリガ方式，高い柔軟性，冗長性，高信頼性，高帯域であることがあげられる．この FlexRay を用いて，メッセージ（データ）の送信を行う際には，高信頼性の実現を目的としたスタティックセグメントと高い柔軟性の実現を目的としたダイナミックセグメントの 2 つの時間が用意されており，メッセージごとにどちらかの時間が用いられる．スタティックセグメントでは，メッセージの長さが固定されており，周期的な通信が保証されている．一方，ダイナミックセグメントでは，メッセージの長さが可変であり，リアルタイム性を検証することが難しい．そのため，スタティックセグメントでは，リアルタイム性が高く要求されるメッセージを送信し，ダイナミックセグメントでは，遅延をあまり気にしないメッセージを送信するということが考えられる．しかし，バスを有効に使いたい，CAN の設計資産を引き継ぎたいなどの要求があるために，ダイナミックセグメントを用いてリアルタイム性が要求されるメッセージを送信したい場合がある．そのため，スタティックセグメントとダイナミックセグメントの両方を使用することが重要であるが，スタティックセグメントのリアルタイム性検証については，すでに研究が行われている[2),3)]．そこで，本研究では，通信が動的にスケジューリングされるダイナミックセグメント内でのリアルタイム性を検証することを目的として，ダイナミックセグメントを用いた際の最大遅れ時間の解析を行う．

ダイナミックセグメントでは，既存の車載ネットワークである CAN における送信方法と同様に，優先度を

† 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

持った可変長のメッセージを扱う．そのため，解析では，最初に，CAN におけるメッセージの最大遅れ時間解析手法[4),5)] を適用できないか検討する．次に，検討結果をふまえて，遅れ時間がより長くなる条件の整理を行う．最後に，メッセージの最大遅れ時間を求めるアルゴリズムを提案し，評価する．

本論文では，2 章で本研究対象である FlexRay の概要，ならびにダイナミックセグメントを用いてメッセージを送信する際の仕組みや送信条件について述べる．3 章では問題点を整理し，4 章でその問題点を考慮したアルゴリズムの説明を行う．5 章では 4 章で提案したアルゴリズムを用いた結果を示し，考察する．最後に 6 章で本論文をまとめる．

2. FlexRay の概要

2.1 FlexRay プロトコルの概要

FlexRay の最新仕様 ver2.1 が，2005 年 5 月に FlexRay コンソーシアム[6)] によって一般に公開されている．本節では，そのプロトコルの概要のうち，本研究に関する部分について述べる．

FlexRay プロトコルでは，メディアアクセス制御は，0〜63 からなる通信サイクルが繰り返されることを基準にしている．**図 1** に示すように，通信サイクルは，スタティックセグメント，ダイナミックセグメント，シンボルウィンドウ，ネットワークアイドルタイムで構成される．このうち，直接メッセージの送信に使用されるのが，スタティックセグメントとダイナミックセグメントの 2 つである．スタティックセグメントはスタティックスロット，ダイナミックセグメントはダイナミックスロットで構成される．スロットとは，各ノードに割り当てられたバスを占有できる時間である．各スロットには，スロット ID がスタティックセグメント，ダイナミックセグメント全体に対して，時間に沿って小さい順に割り振られる．また，スタティックセグメントでは，スロット長が固定されており，送信が静的にスケジューリングされる．一方，ダイナミックセグメントでは，スロット長の変更を自由にすることができ，送信が動的にスケジューリングされる．

2.2 ハードウェア構成

FlexRay は，ローエンドからハイエンドの車種に対応できるように，バス型，スター型，バス型とスター型の混在したハイブリッド型に対応している．また，フォールト・トレランス（耐故障性）を高めるために，デュアルチャネルにも対応している．**図 2** にシステム構成例を示す．図のように車載ネットワークは，複数のノード（ECU☆）により形成される．また，各ノードは，主に，ホスト CPU と通信コントローラにより構成される．メッセージの送受信の際には，メッセージは，通信コントローラのメッセージバッファを経由する．

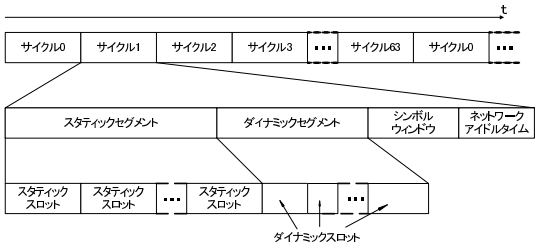

図 1 FlexRay のメディアアクセス
Fig. 1 Media access scheme within FlexRay.

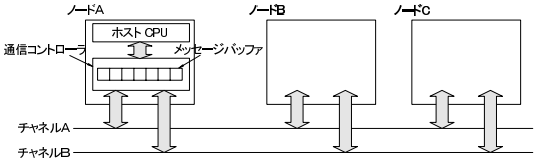

図 2 FlexRay のシステム構成例
Fig. 2 System architecture example of FlexRay.

2.3 ダイナミックセグメントにおける送信

本節では，ダイナミックセグメントを用いた際の送信方法や送信条件などについて述べる．

メッセージは，フレームに格納されて送信される．このフレームには，それぞれフレーム ID が割り当てられている．フレーム ID を割り当てる際，送信ノードごとに使用できるフレーム ID の値はシステムの設計時に静的に決まっている．このフレーム ID により，フレームが送信されるべきスロットを明確にしており，フレーム ID と現在のスロット ID が一致したときに，フレームが送信される．現在のスロット ID を示すために，チャネルごとにスロット・カウンタが用意されている．

ただし，本論文では，簡単化のため，ダイナミックセグメントの開始のスロット・カウンタの値を 1 とし，メッセージのフレーム ID もそれに合わせて表記する．また，フレーム ID の値が i であるフレームに格納されているメッセージをメッセージ i と呼ぶ．

図 3 にダイナミックセグメントにおける送信のモデルを示す．ダイナミックスロットは，1 つないし，複数のミニスロット（固定長）で構成される．ダイナミックスロットの長さは，フレームの長さに応じて動的に変化する．フレームの送信がない場合には，1 つのミニスロット（空きスロット）に対応するアイドル時間

☆ Electronic Control Unit：電子制御装置

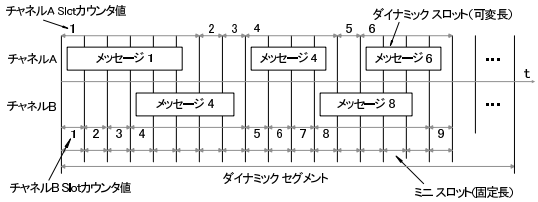

図 3 ダイナミックセグメントにおける送信
Fig. 3 Transmission in the dynamic segment.

の後，次のスロットに切り替わる．また，ダイナミックスロットの長さはチャネルごとに決められる．

さらに，2.1 節で述べたように，FlexRay ではスタティックセグメントとダイナミックセグメントにより交互にメッセージの送信が行われるために，送信を開始したメッセージは，そのダイナミックセグメント内で送信を完了しなければならない．そのため，ノードごとに，最終送信可能スロット（$pLatestTx$）が決められている．$pLatestTx$ とは，各ダイナミックセグメントにおいてメッセージの送信を開始できる最後のミニスロット番号を表しており，メッセージ長の上限値に基づいて決定される．各ノードは，そのノードの $pLatestTx$ 以降では，いずれのメッセージの送信も開始することができないため，ダイナミックセグメント内に送信が完了するようなメッセージ長の短いメッセージであっても送信は不可能である．そのため，送信するフレームに対して十分なミニスロットが確保できないとき，送信が行われないことになり，フレーム ID の大きいメッセージは必ずしも通信が保証されない．したがって，ダイナミックセグメントにおけるメッセージの送信では，フレーム ID がそのフレームの優先度を表しており，フレーム ID の小さいものほど優先度が高いといえる．

このため，CAN では，バス上にデータが流れておらず，かつ，高優先度メッセージの送信要求が発生していなければ，いつでも送信を開始できたのに対し，ダイナミックセグメントの場合には，送信を開始するのに制約がある．

3. 問題点の整理

3.1 前 提 条 件

本研究で扱うメッセージモデルについて述べる．本研究では，ダイナミックセグメントを用いて周期メッセージの送信を行う．**図 4** に，本研究で扱うメッセージモデルを示す．図 4 に示したように，各メッセージの送信要求は，それぞれある一定の周期で発生可能となる．ただし，発生可能になると，すぐに送信要求が発生するとは限らず，遅れて発生する場合もある．こ

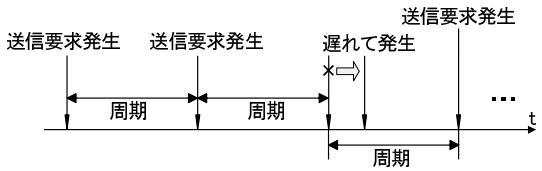

図 4 本研究で扱うメッセージモデル
Fig. 4 Message model that we treat in this research.

のようなメッセージの送信要求が遅れて発生する場合を考慮に入れるのは，遅れて発生した場合の方が，遅れ時間が長くなる場合があるためである．詳細は，次節以降で示す．また，遅れ時間を過大に求める必要があるために，送信要求の発生は，どれだけでも遅れることがあると仮定する．

また，本研究では，すべてのノードにおいて，メッセージの送信に十分なメッセージバッファがあると仮定する．すなわち，メッセージバッファが優先度の低いメッセージで満たされており，メッセージがメッセージバッファに格納できず，送信が行われないといった状況は起こらない．そのため，メッセージがより優先度の低い同一ノードから送信されるメッセージに邪魔されない．

さらに，問題を簡単化するために，以下のように前提条件を設定する．

- 各メッセージの周期はサイクル長の整数倍である．
- 各メッセージの送信要求はサイクルの最初にしか発生しない．
- 扱うチャネルは 1 つとする．
- 目的メッセージより優先度の高いメッセージは，そのメッセージの送信が完了する前に次の送信要求発生可能時刻に到達することがない．

3.2 関 連 研 究

FlexRay と同様に近年注目されているプロトコルに，TTP（Time Trigger Protocol）[7] がある．しかし，TTP では，FlexRay におけるスタティックセグメントに対応する部分しか存在しない．

ダイナミックセグメントを用いて，非周期メッセージの送信を行う場合についての解析は，Pop らによって行われている[8]．Pop らの手法では，あるメッセージを邪魔することのできるメッセージ数が固定であるため，各サイクルをビン，各メッセージをビンに詰める物としてとらえることにより，ビンパッキング問題に帰着させている．しかし，本論文では，周期メッセージを扱うため，各メッセージの送信タイミングを考慮する必要があり，かつ，各メッセージに複数回邪魔されることが起こるため，適用できない．

ダイナミックセグメントでは，CAN における送信

表 1 遅れ時間が長くなる条件の比較
Table 1 Comparison of the condition that the response time is longer.

	同時発生のとき遅れ時間が最大か? (I)	最大長のとき遅れ時間が最大か? (II)	残りメッセージ数が多いとき遅れ時間が長いか? (III)	残りメッセージのメッセージ長が大きいとき遅れ時間が長いか? (IV)
(a)	No	Yes	Yes	Yes
(b)	No	No	No	No

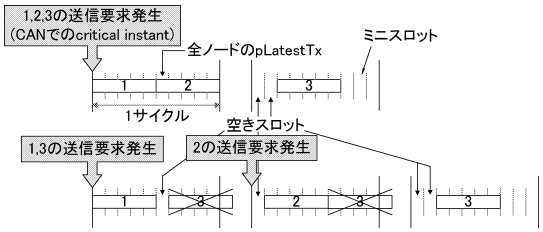

図 5 CAN の場合の手法が適用できない具体例
Fig. 5 An example of not applying the method for CAN.

方法と同様に，優先度を持った可変長のメッセージを扱う．CAN を用いてメッセージの送信を行う場合については，*critical instant* 定理[4),5)] が証明されている．ここで，*critical instant* とは，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のことである．この定理によれば，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る．しかし，FlexRay のダイナミックセグメントの場合には，そういった状況のときに必ずしも遅れ時間が最も長くなるとは限らない．その例を**図 5** に示す．図 5 において，メッセージ 3 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について考える．なお，1 サイクル目でメッセージ 1，2，3 の送信要求は発生可能になったとする．図の上段の場合（CAN の場合の *critical instant*）には，メッセージ 3 は 2 サイクル目で送信される．一方，図の下段の場合のようにメッセージ 2 の送信要求の発生が遅れた場合には，空きスロットの影響により，3 サイクル目で送信される．したがって，CAN における *critical instant* 定理が適用できない．そのため，CAN において最大遅れ時間を解析する際には，*critical instant* のときのみを探索すればよかったが，ダイナミックセグメントの場合には，空きスロットの影響により，送信要求が発生するすべての組合せについて探索する必要がある．

3.3 遅れ時間が最大となる条件

$pLatestTx$ の値は，ノードごとに設定できる．仮に，$pLatestTx$ の値をノードごとにばらばらに設定したとすると，あるメッセージが，より優先度の高いメッセージよりも先に送信されるといった状況が起こりうる．したがって，$pLatestTx$ の値が全ノードで同じである場合と同じでない場合，つまり，ノードごとのメッセージ長の上限値が全ノードで同じである場合と同じでない場合によって遅れ時間が長くなる条件が異なるため，これらを区別して考える．

まず，以下のように状態を定義する．

状態定義 メッセージ i の最大遅れ時間について考えるとき，メッセージセットの状態を (Q,T) と表す．ただし，$Q = (q_1, q_2, .., q_i)$，$T = (t_1, t_2, \cdots, t_i)$ とする．q_k とはメッセージ k について，送信要求の発生が可能な個数である（送信が行われる前に，次の送信要求発生可能時刻になると，q_k の値は 2 になる）．また，t_k とは，次回の送信要求発生可能時刻までにかかるサイクル数である．

以下では，この状態を用いる．全ノードでメッセージ長の上限値が同じ場合（a）と同じでない場合（b）についての遅れ時間が長くなる条件を**表 1** に整理する．ただし，表 1 の “残りメッセージが多いとき，遅れ時間が長いか” という項目は，$q_k > 0$ であるメッセージからなる集合 A と B が存在したときに，$A \supset B$ が成立するならば，A の方が遅れ時間が長くなるか，ということである．また，“残りメッセージのメッセージ長が大きいとき，遅れ時間が長いか” という項目は，先ほどと同様の集合 A と B を考えたときに，$|A| = |B|$ かつ $\forall i$，$L_{a_i} \geq L_{b_i}$（a_i（または b_i）:= 集合 A（または B）の中で i 番目にメッセージ長が大きいメッセージ，L_k := メッセージ k の長さ）という条件が成り立つならば，A の方が遅れ時間が長くなるか，ということである．

以下，表の各枠について見ていく．まず，No の部分について，具体例をあげて説明する．ただし，以下の具体例では，メッセージセットを ((メッセージ ID, 最大長),(メッセージ ID, 最大長), ···, (メッセージ ID, 最大長), サイクル長) のように表記する．また，メッセージの最大長ならびにサイクル長の単位はミニスロット数とする．

（a）において，(I) の条件が成立しない例としては，先ほど示した図 5（メッセージセット：((1,5),(2,5),

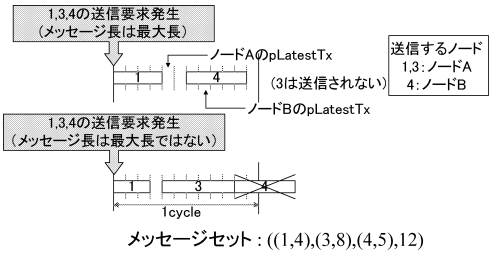

図 6 （b）において，（Ⅱ）の条件が成立しない例
Fig. 6 In (b), an example that the condition (Ⅱ) does not hold.

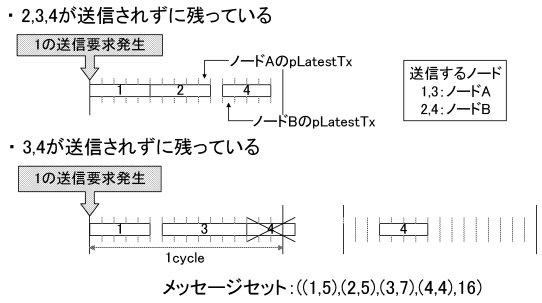

図 7 （b）において，（Ⅲ）の条件が成立しない例
Fig. 7 In (b), an example that the condition (Ⅲ) does not hold.

(3,5),10)）があげられる（$pLatestTx$ は，6 個目のミニスロット）．また，（a）は（b）の特殊な場合であると考えることができるため，（b）においても，（I）の条件が成立しないのは，自明である．

次に，（b）において，（Ⅱ）の条件が成立しない例として，**図 6** に具体例を示す．メッセージ 4 の遅れ時間について考える．メッセージ 1，3 がメッセージ 4 と同時に発生したとすると，図の上段の場合（すべてのメッセージが最大長で発生）には，メッセージ 4 は 1 サイクル目で送信できる．しかし，図の下段の場合（メッセージが最大長ではない）には，メッセージ 1 が最大長でないことにより，メッセージ 3 が送信されるため，メッセージ 4 が 1 サイクル目では送信されない．したがって，（Ⅱ）の条件が成立するとは限らない．

また，（b）において，（Ⅲ）の条件が成立しない例として，**図 7** に具体例を示す．メッセージ 4 の遅れ時間について考える．残りメッセージが多い図の上段の場合には，メッセージ 4 は 1 サイクル目で送信される．しかし，図の下段の場合には，メッセージ 3 が送信されるため，メッセージ 4 は 2 サイクル目で送信される．したがって，（Ⅲ）の条件が成立するとは限らない．

最後に，（b）において，（Ⅳ）の条件が成立しない例として，**図 8** に具体例を示す．メッセージ 4 の遅れ

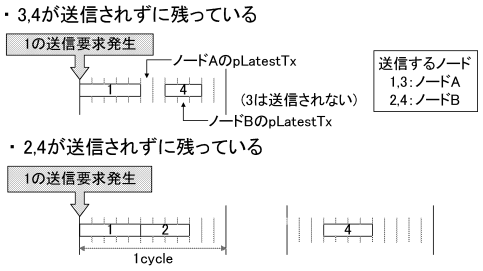

図 8 （b）において，（Ⅳ）の条件が成立しない例
Fig. 8 In (b), an example that the condition (Ⅳ) does not hold.

時間について考える．図の上段の場合には，メッセージ 4 は 1 サイクル目で送信される．一方，図の下段のように，メッセージ 3 よりもメッセージ長の短いメッセージ 2 が送信されずに残っている場合を考える．この場合には，メッセージ 2 が送信されることにより，1 サイクル目では，メッセージ 4 が送信されない．したがって，（Ⅳ）の条件が成立するとは限らない．

次に Yes の部分について見ていく．全ノードでメッセージ長の上限値が同じ場合には，メッセージの遅れ時間について以下の定理が成り立つ．

定理 メッセージ i がいつかは送信される場合において，状態 (Q,T) と状態 (Q',T') について，$\forall j, t_j = t'_j$ のとき，$\forall j, q_j \geq q'_j$ ならば，状態 (Q,T) の方が送信完了までにかかる時間が同じかより長くなる

証明 各状態において送信要求の発生が可能なメッセージは，$\forall j, q_j \geq q'_j$ であるため，以下のように表すことができる．

- 状態 (Q',T')：メッセージ $a_1, a_2, a_3, \cdots, a_m$
- 状態 (Q,T)：メッセージ $a_1, a_2, a_3, \cdots, a_m, b_1, b_2, b_3, \cdots, b_n$
  $(a_1 < a_2 < \cdots < a_m = i, b_1 < b_2 < \cdots < b_n < a_m)$

このメッセージの中で，以下のように送信要求が発生したとする．

- 状態 (Q',T')：メッセージ $a'_1, a'_2, a'_3, \cdots, a'_{m'}$
- 状態 (Q,T)：メッセージ $a'_1, a'_2, a'_3, \cdots, a'_{m'}, b'_1, b'_2, b'_3, \cdots, b'_{n'}$

（ただし，$a'_1 < a'_2 < \cdots < a'_{m'} = a_m, b'_1 < b'_2 < \cdots < b'_{n'}, \forall s \in \{1, 2, \cdots, m'\}(\exists t \in \{1, 2, \cdots, m\}, a_t = a'_s), \forall s \in \{1, 2, \cdots, n'\}(\exists t \in \{1, 2, \cdots, n\}, b_t = b'_s), m' \leq m, n' \leq n$）

全ノードでメッセージ長の上限値が同じ場合，全ノードで $pLatestTx$ も同じになるため，あるメッセージが送信されなければ，それ以降のメッセージ ID の大

きいメッセージは送信されない．すると，このサイクルで送信されるメッセージは，

- 状態 (Q',T')：メッセージ $a'_1, a'_2, a'_3, \cdots, a'_p$（$a'_1 \sim a'_p$ のメッセージはすべて送信される）
- 状態 (Q,T)：メッセージ $a'_1, a'_2, a'_3, \cdots, a'_q, b'_1, b'_2, b'_3, \cdots, b'_r$（$a'_1 \sim a'_q$，$b'_1 \sim b'_r$ のメッセージはすべて送信される）

（ただし，$p \geq q$，$1 \leq p \leq m'$，$0 \leq q \leq m'$，$0 \leq r \leq n'$）
と表せる．

（ア）$p = m'$ のとき

状態 (Q',T') では，メッセージ $i(= a'_{m'})$ が送信される．一方，状態 (Q,T) では，メッセージ i が送信されるとは限らない．よって，状態 (Q,T) の方が送信完了までにかかる時間が同じかより長くなる．

（イ）$p \neq m'$ のとき

このサイクルでは，状態 (Q,T) も状態 (Q',T') もメッセージ i は送信されない．次サイクルでは，送信要求発生が可能であるが送信要求が発生しなかったメッセージ，送信要求が発生したが送信されなかったメッセージ，新たに送信要求発生が可能になったメッセージが存在する．送信要求発生が可能であるが発生しなかったメッセージは，状態 (Q,T) の方が状態 (Q',T') よりも $n - n'$ 個分多くなる．また，送信要求が発生したが，送信されなかったメッセージは，$p \geq q$ であるため状態 (Q,T) の方が状態 (Q',T') より多いか等しくなる．ここで，送信要求発生が可能であるが送信要求が発生しなかったメッセージと送信要求が発生したが送信されなかったメッセージは，ハードウェアの観点からするとどちらも送信されなかったという意味で同じであるため，送信要求が発生したが送信されなかったメッセージを送信要求が発生しなかったと見なす．すると，次サイクルでも，今回のサイクルのような関係になる．このようなことを繰り返すと，メッセージ i が送信されるとするなら，いずれは（ア）になるため，状態 (Q,T) の方が送信完了までにかかる時間が同じかより長くなる．

したがって，状態 (Q,T) の方が送信完了までにかかる時間が同じかより長くなる． □

この定理より，（Ⅱ）の条件が成立する．また，送信要求の発生が可能なメッセージは多いほうが，送信完了までにかかる時間が同じかより長くなっている．つまり，それ以前のサイクルでは，送信されずに残るメッセージが多いほうが，いい換えると，1 サイクルで送信されるメッセージは少ないほうが，送信完了までにかかる時間が同じかより長くなる．メッセージが最大長のときに，送信するのに多くのミニスロットを使用するため，1 サイクルで送信されるメッセージ数が少なくなる．したがって，（Ⅲ）の条件は成立する．

最後に，（a）において，（Ⅳ）の条件が成立することの証明を示す．

証明 メッセージ i について考えたときに，メッセージ $a_1, a_2, a_3, \ldots, a_k$ が送信されたとする（$a_1 < a_2 < a_3 < \ldots < a_k < i$）．このとき，メッセージ i の送信が開始されるミニスロット番号 N は，以下の式で表される．

$$N = \sum_{j=1}^{k} \{(a_j - a_{j-1} - 1) + L_j\} + (i - a_k - 1) + 1$$
$$= \sum_{j=1}^{k} (L_j - 1) + i \quad (1)$$

（$L_j :=$ メッセージ j の長さ，$a_0 = 0$）
したがって，以下の式が成立するとき，メッセージ i は送信されない．

$$\sum_{j=1}^{k} (L_j - 1) + i > pLatestTx$$

この式において，可変なのは，各メッセージの長さと送信されるメッセージの個数であり，メッセージ ID の値は関係ない．今，送信されるメッセージの個数を固定して考える．この際，次サイクル以降でもこの式を満たすことを考えたとき，各サイクルでは，より短いメッセージでメッセージ i を邪魔し，メッセージ長が大きいものを次サイクル以降に回したほうが遅れ時間が長くなる．よって，（Ⅳ）の条件は成立する． □

なお，$pLatestTx$ の値をノードごとにばらばらに設定したとすると，優先度の低い（ID が大きい）メッセージが優先度の高い（ID が小さい）メッセージよりも先に送信されるということが起こりうるため，上記の定理や証明が成り立たない．

4. 最大遅れ時間を求める解析手法

4.1 扱うモデル

全ノードでメッセージ長の上限値を同じに設定した場合には，通信効率が低下するが，メッセージ長の上限値の変更が生じた場合の最大通信遅延の変化を低減させることができるというメリットがある．また，前章で，遅れ時間が長くなる条件を整理した結果，全ノードでメッセージ長の上限値を同じに設定した場合には，遅れ時間が長くなる条件が複数存在した．そこで，遅れ時間が長くなる条件をもとに，全ノードでメッセージ長の上限値が同じ場合について解析を行う．この場

```
/*メッセージiの最大遅れ時間を求めるアルゴリズム(全数探索)*/
cycle(*q,*t){
  for(j=1;j<i;j++){
    if(tj>0) tj --; else qj ++,tj = メッセージjの周期;
  }
  while(送信されるメッセージの組み合わせをすべて探索していない){
    qk>0であるメッセージの中から送信されるメッセージを選択;
    if(目的メッセージを邪魔できる){
      送信されたメッセージのqの値を-1する;
      cycle(q,t);
      qの値を元に戻す;
    }else{
      得られた解に応じて解の更新;
    }
  }
}
```

図 9 全数探索アルゴリズム
Fig. 9 Exhaustive search algorithm.

合には，ノードごとの違いがないため，各メッセージがどのノードから送信されるかは考慮しない．

4.2 全数探索アルゴリズム

基本的な解析手法として，送信されるメッセージの組合せに対して，全数探索を行う．本研究では送信要求が遅れて発生する場合を考慮しているため，各サイクルでは，送信要求発生可能なメッセージの中から送信されるものを選択する．そして，各探索を目的メッセージが送信されるまで行い，全探索の中で，目的メッセージが送信されるまでにかかった時間が最も長いものを最大遅れ時間とする．なお，3 章で遅れ時間が長くなる条件を整理した際に，メッセージが最大長のとき，遅れ時間が最大になっていたため，すべてのメッセージは最大長で発生するものとする．ただし，ダイナミックセグメントの 1 サイクルのミニスロット数，各メッセージの周期（送信要求が発生する最小の間隔），各メッセージについてメッセージが最大長のときに送信に必要なミニスロットの数はあらかじめ分かっているものとする．全数探索アルゴリズムを**図 9** に示す．また，探索をする際，初期状態は $\forall i, q_i = 0$, $t_i = 0$ とする．初期状態 $\forall i, q_i = 0$, $t_i = 0$ から探索を開始して，最大遅れ時間を求めることができる理由を以下に示す．この証明の考え方は，Lehoczky が提案した *level-i-busy period*[9] という概念に基づいている．

証明

(1) $\forall i, q_i = 0$ のとき
送信要求発生可能時刻までの時間が短いほうが，メッセージ i を邪魔できるので $\forall i, t_i = 0$ のとき遅れ時間が最大になる．

(2) (1) 以外のとき（メッセージ i より優先度の高いまだ送信されていないメッセージが存在するとき）
メッセージ i の送信要求が発生する時刻を t とする．また，メッセージ i よりも優先度が高いメッセージの中に送信が完了していないメッセージが存在しない時刻を t_0 とする．ただし，t_0 と t の間では，つねに送信されずに残っているメッセージが存在するものとする．時刻 0 においては，どのメッセージも送信要求が発生していないので，t_0 は必ず存在する．このとき，メッセージ i の送信要求発生時刻を t_0 に移動しても，メッセージ i が送信される時刻は変化しない．そのため，メッセージ i は時刻 t_0 に発生したほうが遅れ時間がより長くなる．時刻 t_0 とは，$\forall i, q_i = 0$ のときである．

(1)，(2) より，$\forall i, q_i = 0$，$t_i = 0$ から探索を開始すればよい． □

4.3 枝刈り手法

全数探索を行うと，計算時間が膨大になるため，枝刈りを行い，効率的な解析の実現を目指す．送信されるメッセージの組合せに対して全数探索を行うと，1 サイクル目の探索だけで，$2^{\text{目的メッセージ } ID - 1}$ 回の探索が必要である．そのため，メッセージ数が増加すると，この 1 サイクル目の探索だけで計算時間が膨大になる．したがって，根本的なアルゴリズムの改良が必要である．

目的メッセージを邪魔するための送信されるメッセージ集合を考える．このとき，この集合からどの要素（メッセージ）を取り除いても，目的メッセージを邪魔できなくなる（目的メッセージが送信される）とき，この集合は，目的メッセージを邪魔できる極小集合であるという．ここで，枝刈り手法として，送信されるメッセージ集合が，極小集合のときのみ探索する．3 章で遅れ時間が長くなる条件の比較を行った際に，全ノードでメッセージ長の上限値が同じ場合には，残りメッセージが多いときに，遅れ時間が長くなっていた．残りメッセージが多くなるためには，1 サイクルで送信されるメッセージが少ない必要がある．そのため，送信されるメッセージ集合が目的メッセージを邪魔できる極小集合のときに遅れ時間が長くなる．このとき，極小集合を動的に求めるアルゴリズムを**図 10** に示す．このアルゴリズムでは，リストにメッセージを追加したり削除したりすることにより，極小集合を求めている．リストは，現在の送信されるメッセージ集合を表している．また，メッセージをリストに追加する際には，メッセージ長の大きいものから追加している．これは，極小集合のみを探索するようにするためである．極小集合であるためには，メッセージ集合の中からメッセージ長が最小のメッセージを取り除い

```
/*メッセージiの最大遅れ時間を求めるアルゴリズム(極小集合のみ探索する)*/
cycle(*q,*t){
 n=0;
 リストを空にする;
 for(j=1;j<i;j++){
  if(tj>0) tj --; else qj ++,tj = メッセージjの周期;
 }
 qk>0であるメッセージ(p個あるとする)をメッセージ長の降順にソートし,
 配列arrayに格納する(array[0]>array[1]>array[2]>・・・>array[p-1]);
 while(探索が終了していない){
  array[n]に格納されているメッセージxをリストの末尾に追加する;
  if(n==p-1){
   if(リストにあるメッセージで目的メッセージを邪魔できる){
    リストにあるメッセージのqの値を-1する;
    cycle(q,t);
    qの値を元に戻す;
   }else 得られた解に応じて，解の更新;
   リストの末尾からメッセージxを削除;
   (リストが空になると，探索終了(while文から抜ける))
   do{
    リストから末尾のメッセージを削除;
    (リストが空になると，探索終了(while文から抜ける))
   }while(最近削除した2つのメッセージ間にqk>0であるメッセージがない);
   n = 最後に削除したメッセージがarrayに格納されていた場所 + 1;
  }else{
   if(リストにあるメッセージで目的メッセージを邪魔できる){
    リストにあるメッセージのqの値を-1する;
    cycle(q,t);
    qの値を元に戻す;
   }
   n ++;
  }
 }
}
```

図 10 極小集合のみを探索するアルゴリズム
Fig. 10 Algorithm that search only for a minimal set.

ても目的メッセージを邪魔できなくなる必要がある．メッセージ長の大きいものから，リストに加えていき，あるメッセージを加えたときに，目的メッセージを邪魔できるようになったとすると，このメッセージ集合からどのメッセージを取り除いても目的メッセージは邪魔できなくなる．

また，このアルゴリズムに改良を加える．表 1 のセルの中で Yes であった箇所のうち，アルゴリズムに使用されていない性質，すなわち，“残りメッセージのメッセージ長が大きいときに，遅れ時間が長くなる” という性質を用いて，さらなる枝刈りを行う．送信されるメッセージの個数が同じ場合において，メッセージ長が大きいものを残す（次サイクル以降に回す）ためには，なるべくメッセージ長の小さいもので，目的メッセージを邪魔する必要がある．そのため，目的メッセージを邪魔できる極小集合が求まった際に，その集合で次サイクル以降の探索をする必要があるか判断し，必要なときだけ次サイクル以降の探索をし，そうでない場合には，次サイクル以降の探索を省くようにする．具体的には，極小集合 A が求まった際（図 10 の “cycle(q,t)” の直前）に，その集合 A に対して，以下の条件が成り立つならば，その極小集合では，現サイクルで探索を終了し，次サイクルへは進まないものとする．

条件 $\exists j \in A(\exists i \notin A(((L_j > L_i) \vee (L_j = L_i \wedge i < j)) \wedge$ (集合 $(A \backslash \{j\} \cup \{i\})$ が目的メッセージを邪魔できる)))
（ただし，i，j は，送信要求発生可能なメッセージの ID とし，L_k は，メッセージ k のメッセージ長とする）．

なお，極小集合を求める際に，送信要求発生可能なメッセージをメッセージ長の降順にソートしているため，この条件を満たすか否かの判定は容易に行うことができる．

4.4 近似解法

別のアプローチとして，厳密解を求めるのではなく，短い計算時間で近似解を求める方法を提案する．ここでは，メッセージ数が増えても，計算時間にほとんど影響が出ない方法を示す．具体的には，探索が枝分かれせず，1 本道であるような方法，すなわち，計算量が，厳密解を求める場合には指数時間かかるのに対して，$O(n)$（n：メッセージ数）である方法を 2 つ提案する．ただし，近似をする際には，必ず近似解法で得られた遅れ時間の方が，実際の最大遅れ時間よりも長くなるように計算する必要がある．

まず，1 つ目の近似では，すべてのメッセージの長さを最大のメッセージ長と見なす．式 (1) から分かるように，あるメッセージを邪魔するための送信されるメッセージ集合を考えたとき，メッセージ集合に含まれるメッセージの ID の値は関係がなかった．そのため，メッセージ長をすべて同一に見なすということは，メッセージの区別がなくなることになる．そのため，目的メッセージを邪魔するメッセージの組合せを考えなくてよく，送信されるメッセージの個数だけを考慮すればよい．さらに，この近似では，1 サイクルで送信されるメッセージの個数を固定する．1 サイクルのミニスロット数を N，メッセージ ID の最大値を M，最もメッセージ長が長いメッセージのメッセージ長を L とすると，少なくとも $\lfloor \frac{N-M}{L-1} \rfloor$ 個のメッセージが，1 サイクルで送信されるはずである．そのため，1 サイクルで送信されるメッセージの個数を $\lfloor \frac{N-M}{L-1} \rfloor$ に固定する．この近似では，メッセージ長を過大に見積もっているため，この近似により得られる解は，厳密解より大きいものとなる．この近似方法を近似 1 とする．

しかし，この近似 1 では，すべてのメッセージ長を同一に見なしているため，メッセージの長さのばらつきが大きいときに，実際の遅れ時間との誤差が大きくなることが推測される．そこで，それぞれのメッセージ長を考慮した近似を提案する．あるメッセージの送信が行われないためには，それよりも優先度の高いメッセージと空きスロットにより，$pLatestTx$ 個のミニスロットを使用すればよい．ここで，目的メッセー

ジは，それよりも優先度の高いメッセージが送信されなかったとしても，“目的メッセージ $ID-1$” 個のミニスロットの後でしか，送信が開始されない．各メッセージ長を “メッセージ長 -1” と見なし，空きスロットの影響を最初に考慮する．すると，“$pLatestTx-$(目的メッセージ $ID-1$)” 個のミニスロットを高優先度のメッセージで使用すればよいことになる．この際，メッセージの分割を許し，各サイクルではつねにミニスロット “$pLatestTx-$(目的メッセージ $ID-1$)” 個分だけメッセージが使用されるものとする．そのため，送信可能なメッセージの “メッセージ長 -1” の合計から “$pLatestTx-$(目的メッセージ $ID-1$)” を引いた数分のメッセージが次サイクルに回されることになる．この近似では，実際には起こりえないメッセージの分割を許し，次サイクル以降に余った分を回しているため，この近似により得られる解は，厳密解より大きいものとなる．この近似方法を近似 2 とする．

近似 2 では，メッセージ数が少ない場合に，送信要求発生可能なメッセージの個数に対する分割されたメッセージの個数の割合が大きくなるため，分割されたメッセージの影響が大きくなり，誤差が大きくなってしまうことが推測される．

5. 実　　験

5.1 適 用 対 象

50 個のメッセージセットを用意し，それらのメッセージセットに対して，提案したアルゴリズムを用いたプログラムを作成し，最大遅れ時間の解析を行った．なお，メッセージセットを生成する際には，乱数を用いることとし，1 サイクルのミニスロット数，各メッセージの周期ならびに送信するのに必要なミニスロット数を正の整数で求めている．

5.2 実 験 結 果

結果を示す前に負荷率という概念について説明する．あるメッセージの負荷率とは，そのメッセージを送信するのに使用される時間の割合であり，複数のメッセージ（メッセージセット）に対する負荷率は，それぞれのメッセージの負荷率を合計したものとなる．負荷率は，バスの詰まり具合を表しており，RMA（Rate Monotonic Analysis）[10] において，重要な指標である．なお，遅れ時間の単位はサイクル数とし，最大遅れ時間が 100 サイクル以上かかる場合には，近似解が求めれないものとして，その時点で探索を終了させている．今回は，遅れ時間をサイクル数単位で求めたが，より詳細に求めることも可能である．

表 2，**表 3** にメッセージセットに対して実行した結果の一部を示す．ただし，メッセージセットの中で最大の ID を持つメッセージの最大遅れ時間を求めている．表 2 は，メッセージ長のばらつきが小さいメッセージセットに対して，表 3 は，メッセージ長のばらつきが大きいメッセージセットに対しての適用結果である．ここで，メッセージ長のばらつきが大きいメッセージセット

表 2 適用結果（メッセージ長のばらつきが小さいメッセージセット）
Table 2 Result of applying to message sets that the dispersion of message length is large.

メッセージ数	負荷率	最大遅れ時間（厳密解）	全数探索		枝刈り		近似 1		近似 2	
			探索回数	計算時間	探索回数	計算時間	最大遅れ時間（近似解）	計算時間	最大遅れ時間（近似解）	計算時間
5	38.54%	5	299	0.000s	5	0.000s	100 以上	0.000s	100 以上	0.000s
10	36.39%	5	98819	0.044s	8	0.000s	18	0.000s	6	0.000s
14	37.65%	3	217968	0.116s	53	0.000s	3	0.000s	3	0.000s
20	30.16%	3	293201035	3m16.968s	197	0.080s	4	0.000s	3	0.000s
30	36.66%	2	486114246035	7243m0.724s	9	2m59.483s	3	0.000s	3	0.000s
38	31.64%	3	-	-	32954	364m29.907s	4	0.000s	3	0.000s

表 3 適用結果（メッセージ長のばらつきが大きいメッセージセット）
Table 3 Result of applying to message sets that the dispersion of message length is small.

メッセージ数	負荷率	最大遅れ時間（厳密解）	全数探索		枝刈り		近似 1		近似 2	
			探索回数	計算時間	探索回数	計算時間	最大遅れ時間（近似解）	計算時間	最大遅れ時間（近似解）	計算時間
5	32.75%	4	175	0.000s	4	0.000s	6	0.000s	100 以上	0.000s
10	38.64%	6	305619	0.140s	143	0.000s	8	0.000s	8	0.000s
14	42.55%	4	4346699	2.300s	735	0.008s	100 以上	0.000s	5	0.000s
20	32.69%	4	9427408991	102m32.208s	60462	1.056s	24	0.000s	4	0.000s
30	29.77%	3	-	-	315431417	1968m2.116s	12	0.000s	3	0.000s
38	29.27%	-	-	-	-	-	15	0.000s	7	0.000s

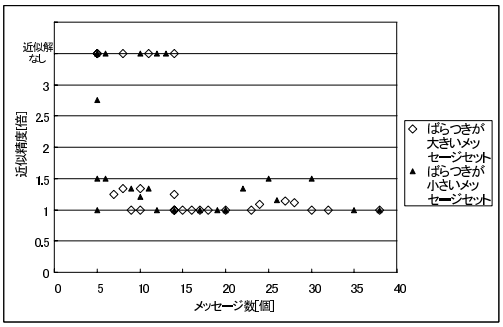

図 11 近似 2 の精度
Fig. 11 Accuracy of approximation method 2.

とは，メッセージ長の変動係数（= 標準偏差/平均値）が 0.5 以上であるものである．

以下の説明は，表に掲載したものだけでなく，すべてのメッセージセットの実験結果にもあてはまる．全数探索により最大遅れ時間を求める場合には，メッセージ数が多くなると，指数関数的に探索回数ならびに計算時間が増加する．また，枝刈りを行うことにより，大幅に探索回数ならびに計算時間を短縮することができ，効率的な探索ができていることが分かる．しかし，枝刈りによって，探索空間を狭めたとはいえ，メッセージ数が大変多くなると，現実的な時間で解を求めることは難しいといえる．実際，表 3 のメッセージ数が 38 の場合には，厳密解を求められていない．

次に，近似の効果について検討する．近似では，計算量が $O(n)$ であるため，メッセージ数が増えることにより計算時間が大幅に増えることはない．近似 1 では，メッセージ長のばらつきが小さい場合には，実際の解との誤差が少なく効果的に解を求めることができている．しかし，メッセージ長のばらつきが大きい場合には，懸念されていたとおり，実際の解との誤差が大変大きくなっている．一方，近似 2 では，各メッセージ長を考慮しているため，メッセージ長のばらつきが大きい場合にも誤差が小さくなっている．また，メッセージ長のばらつきが小さい場合でも，近似 2 の方が良い解を得られることが多い．

次に，近似 2 に着目し，**図 11** にメッセージ数と近似精度の関係を示す．近似精度は，近似解を厳密解で割ることにより求められている．図 11 から，近似 2 ではばらつきの大小に影響されず，近似解を求められていることが改めて確認できる．近似 2 を提案した際に懸念していたように，メッセージ数が少ない場合には，近似解を得られない場合がある．しかし，実験した範囲内では，メッセージ数が 15 個以上のメッセージセットにおいては，近似解を必ず求めることができている．また，つねに近似解は厳密解の 1.5 倍以下になっており，近似精度も高いといえる．考えられるすべてのメッセージセットについて適用することはできないので，メッセージ数が 15 個以上であれば，確実に精度の高い近似解を得ることができるとはいいきれないが，大概の場合には，メッセージ数が 15 個以上のメッセージセットについては近似 2 は有効であるといえる．

したがって，メッセージ数が少ない場合（今回の実験範囲内では，30 個以下）には，枝刈りを用いたアルゴリズムによって，厳密解を求め，メッセージ数が多い場合には，近似解法を用いて近似解を求めることが有効である．また，近似解を求めるとき，多くのメッセージセットの場合には，近似 2 の方が良い解が得られるが，なかには近似 1 の方が良い解を得られることもあった．そのため，近似解を求める場合には，近似 1，2 の両方を用い，各近似により得られる解の中で，より良い解を近似解として採用することで，厳密解に近い解を得ることができる．

6. おわりに

本研究では，FlexRay のダイナミックセグメントを用いた際のリアルタイム性を検証することを目的として，メッセージの最大遅れ時間の解析を行った．

まず，CAN の解析手法が適用できないことを指摘した．また，全ノードでメッセージ長の上限値を同じに設定するかしないかによって，遅れ時間が最大となる条件が異なってくるため，両者を区別して考え，遅れ時間がより長くなる条件を整理した．

そのうえで，本研究では，全ノードでメッセージ長の上限値が同じ場合について解析を行った．解析手法としては，まず，送信されるメッセージの組合せに対して，遅れ時間がより長くなる条件を考慮した全数探索アルゴリズムを提案した．また，枝刈り手法を提案し，現実的な時間での探索を目指した．一方，別のアプローチとして，時間をかけて厳密解を求めるのではなく，短い時間で近似解を求める方法を提案した．そして，これらの方法を複数のメッセージセットに対して適用した．その結果，メッセージ数が少ない場合には，枝刈りを用いたアルゴリズムにより効率的に厳密解を求め，メッセージ数が多い場合には，2 つの近似を併用することにより精度の良い近似解を求めることが効果的であるといえる．

本論文では，問題の複雑さに対して簡略化された近似手法を提案したが，演算量は増すが近似精度を高める他の近似手法の存在の有無の検討については今後の

課題である．

謝辞 本研究は，トヨタ自動車との共同研究である．本研究を行うにあたり，有益なご助言をいただいた，トヨタ自動車の関係者の皆様に深く感謝いたします．

参 考 文 献

1) ISO 11898, Road Vehicles — Interchange of digital information — Controller area Network (CAN) for high speed communication (1993).
2) 村上尚彦，飯山真一，高田広章，城戸正利，細谷伊知郎：自動車制御分散システムの静的スケジューリング手法，組込技術とネットワークに関するワークショップ ETNET 2005 (2005).
3) Ding, S., Murakami, N., Tomiyama, H. and Takada, H.: A GA-Based Scheduling Method for FlexRay Systems, *Proc. 5th ACM int'l Conference on Embedded Software*, pp.110–113 (2005).
4) Tindell, K., Burns, A. and Wellings, A.: Calculating Controller Area Network (CAN) Message Response Times, *Control Engineering Practice*, pp.1163–1169 (1995).
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) FlexRay Consortium. http://www.flexray.com/
7) Kopetz, H. and Bauer, G.: The time-triggered architecture, *Proc. IEEE*, Vol.91, No.1, pp.112–126 (2003).
8) Pop, T., Pop, P., Eles, P., Peng, Z. and Andrei, A.: Timing Analysis of the FlexRay Communication Protocol, *Proc. 18th Euromicro Conference on Real-Time Systems*, pp.11–21 (2006).
9) Lehoczky, J.P.: Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadlines, *Proc. Real-Time Systems Symposium*, pp.201–209 (1990).
10) Liu, J.W.S.: *Real-Time Systems*, Prentice Hall (2000).

(平成 19 年 1 月 9 日受付)
(平成 19 年 6 月 5 日採録)

村上 靖明

2006 年 3 月名古屋大学工学部電気電子情報工学科卒業．現在，同大学院情報科学研究科情報システム学専攻博士前期課程に在学．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．

村上 尚彦

2005 年 3 月名古屋大学工学部電気電子情報工学科卒業．現在，同大学院情報科学研究科情報システム学専攻博士前期課程に在学．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．

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．