社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2005－ＳＬＤＭ－119（16）
2005／3／18

自動車制御分散システムの静的スケジューリング手法

村上 尚彦† 飯山 真一‡ 高田 広章§ 城戸 正利* 細谷 伊知郎*

†名古屋大学工学部 〒464-8603 名古屋市千種区不老町
‡豊橋技術科大学情報工学系 〒441-8580 愛知県豊橋市天伯町雲雀ヶ丘1-1
§名古屋大学大学院情報科学研究科 〒464-8601 名古屋市千種区不老町
*トヨタ自動車株式会社統合システム開発部第1開発室 〒471-8572 愛知県豊田市トヨタ町1番地
E-mail: †murakami@ertl.jp, ‡shin@ertl.jp, §hiro@ertl.jp, *{kido@msg, hosotani@n1hs}.tec.toyota.co.jp

あらまし 近年，次世代車載LANプロトコルの1つとして，FlexRayが注目されている．しかしながら，FlexRayはいまだ実用段階にないため，システム設計に関する具体的な評価手法に乏しい．本研究ではまず，将来的にFlexRayシステムを導入するにあたり，そこで必要となる条件や制約を分析・整理した．そして，そこで具体化されたモデルに対し，システムの処理や通信におけるスケジュールを静的に求める手法を考案した．また，考案した手法を実際の自動車内で実現されることを想定したアプリケーションモデルに対して適用し，その有効性を確認した．

キーワード FlexRay，タスクグラフ，スケジューリング，設計制約，シミュレーティド・アニーリング

A Static Scheduling Method for Distributed Automotive Control Systems

Naohiko MURAKAMI† Shinichi IIYAMA‡ Hiroaki TAKADA‡
Masatoshi KIDO* and Ichiro HOSOTANI*

†School of Engineering, Nagoya University Hurou-tyo, Chikusa-ku, Nagoya, 464-8603 Japan
‡Department of Information and Computer Sciences, Toyohashi University of Technology
1-1, Hibarigaoka, Tenpaku-tyo, Toyohashi, Aichi, 441-8580 Japan
§Graduate School of Information Science, Nagoya University Hurou-tyo, Chikusa-ku, Nagoya, 464-8601 Japan
*Development dept. No.1, Integrated system engineering div., Toyota Motor Corporation
1, Toyota-tyo, Toyota, Aichi, 471-8572 Japan
E-mail: †murakami@ertl.jp, ‡shin@ertl.jp, §hiro@ertl.jp, *{kido@msg, hosotani@n1hs}.tec.toyota.co.jp

Abstract Recently, FlexRay attracts attention as a next generation LAN protocol for automotive network systems. However, FlexRay systems are still not in the stage for practical application, and there are not enough concrete methods for designing them. First, in this research, we analyzed and adjusted the necessary conditions and constraints for designing FlexRay systems. And then, we devised a static scheduling method for processing and communication in the systems under the model we materialized. We also applied our method to an application model likely to be used for an actual automotive system, and confirmed the method to be effective.

Keyword FlexRay, Task graph, Scheduling, Design Constraints, Simulated Annealing

1. はじめに

近年，自動車制御システムは，省エネルギー，低排気ガス，安全性の向上，利便性の向上などの実現に伴い，複雑化・大規模化している．そのため，車両に搭載されるECU(Electronic Control Unit)の数は増加し，その間でやり取りされるデータも増加の一方にある．また，最近ではX-by-wireへの移行に伴い，通信における高信頼性の要求も高まってきている。

このような中で，次世代車載LANプロトコルの1つとして注目されているのが，FlexRayである．本研究では，FlexRayを車載ネットワークシステムに導入することを考慮し，そこで行なわれる通信や処理のスケジュールに着目した評価・設計手法を考案することを目的とする．しかしながら，FlexRayの車載ネットワークシステムへの導入はいまだ実用段階にいたっておらず，リアルタイム分野においても実際の適用を考慮した評価・設計手法に関する研究に乏しい．

そこで本研究ではまず，自動車メーカとの協力のもとで要求分析を行い，FlexRayシステムを設計する上で必要となる制約の整理，および想定するシステムや

その動作に対するモデル化を行う．そして，そこで具体化された問題に対し，設計制約を満たすようなスケジュールを静的に求める手法を考案する．

本論文では，2 章で FlexRay システムの概要について述べる．3 章では，要求分析の中でモデル化した問題について説明する．4 章では，実際に考案したシステムのスケジューリング手法について説明する．5 章では，本手法をある問題に対して適用した結果を示し，考察する．最後に 6 章で，本論文のまとめと今後の課題について述べる．

2. FlexRay システム概要

2.1. プロトコル概要

FlexRay の仕様 ver2.0 が，2004 年 7 月に FlexRay コンソーシアムに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について述べる．

FlexRay は，最大 10Mbps の転送速度を実現するタイム・トリガ方式のプロトコルである．

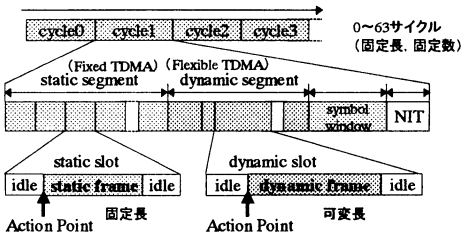

図 1：FlexRay 通信サイクル

FlexRay の通信は，0～63 からなる通信サイクルを基準に行われる．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（NIT）で構成される．スタティック・セグメントはデータ長が固定の Fixed TDMA 方式，ダイナミック・セグメントはデータ長が可変の Flexible TDMA 方式でおそれぞれ通信が行なわれる．シンボル・ウィンドウはバスのテストなどに使用するシンボルを送信するための時間である．NIT は，すべてのデータ送受信が停止されるアイドル時間であり，各ノードのタスク処理，クロック補正などが行われる．

各セグメントは複数のスロットで構成される．スロットはノードに割り当てられるバスの占有時間であり，アクション・ポイント・オフセットとよばれるアイドル時間，およびデータ送受信の単位となるフレームから構成される．

FlexRay の送信スケジュールはシステムの設計時に静的に決定される．フレームにはそれぞれフレーム ID が割り振られ，これがスロット番号と一致したときにフレームを送信する．スロット番号はスタティック・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．

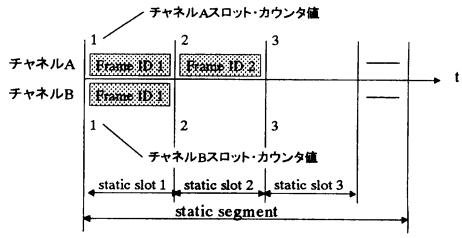

図 2：スタティック・セグメント送信モデル

図 2 に，スタティック・セグメントにおける送信のモデルを示す．スタティック・セグメントは，スロット長がすべて同じである複数のスロットで構成され，1 つのスロット内で 1 つの固定長フレームが送信される．チャネルごとに現在のスロット番号を示すスロット・カウンタがあり，フレーム ID とスロット・カウンタが一致したときにフレームを送信する．両方のチャネルでスロット・カウンタの値は一致するが，必ずしも同じフレームを両チャネルに送信しなくてもよい．

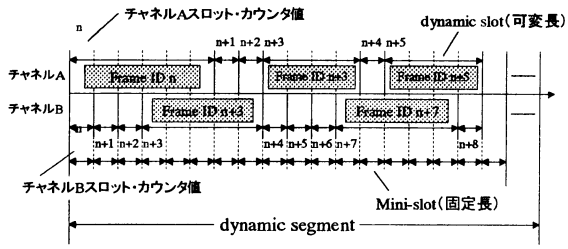

図 3：ダイナミック・セグメント送信モデル

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・セグメントは，チャネルごとに複数のミニ・スロットに分割される．送信したいフレームの長さにあわせて，そのフレーム ID に対応するダイナミック・スロットが複数のミニ・スロットによって動的に構成される．スタティック・セグメントの場合と同様に，フレーム ID とスロット・カウンタが一致したときにフレームを送信する．ただし，スロット・カウンタの値は 2 つのチャネルで異なり，送信されるフレームも異なる．

ダイナミック・セグメントでは，フレームの送信が無い場合，ミニ・スロット長のアイドル時間の後，次のスロットに切り替わる．また，送信するフレームに対して十分なスロットが確保できない場合，送信が行

われないため，IDの大きいフレームは必ずしも通信が保証されるとは限らない．

各ノードは、フレームIDと0〜63の通信サイクル・カウンタ値から，受信フレームをフィルタリングする．

2.2. システム構成

FlexRayはバス型・スター型の両方のトポロジに対応しており，さらにそれらを組み合わせたハイブリット型のネットワーク・トポロジも可能となっている．ネットワークは完全に2重化され，ノードごとにシングル・チャネルかデュアル・チャネルかを選択できる．ネットワーク内のノード群は優れた同期アルゴリズムによってグローバルな同期が実現される．

本論文では，1つのバス型ネットワークに複数のノードがすべてデュアル・チャネルで接続されたシステムを考える．図4に示すように，各ノードはホスト・プロセッサ，通信コントローラ，バス・ドライバ，バス・ガーディアンから構成される．

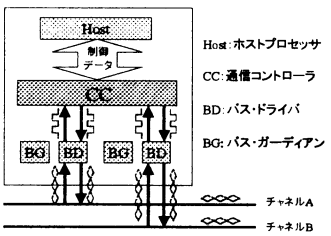

図4：FlexRayノード構成

ホスト・プロセッサがノード全体を制御し，送信データの作成や受信データの処理，および通信状態の管理を行う．通信コントローラは主に送受信を制御し，送信フレームの生成，受信データのフィルタリングなどを行う．バス・ドライバは，実際に各ノード間のデータ送受信を行う．バス・ガーディアンは，不正なタイミングでのデータ送信を検出し，バス・ドライバの動作を停止させることによって通信を遮断する．

ホスト・プロセッサでは，各ノードに割り付けられたタスクの処理や送信データの作成を行うアプリケーション・プログラムが動作している．ホスト・プロセッサで作成された送信データは，コントローラ・ホスト・インタフェースを介して通信コントローラに送られる．この際，送信データの受け渡しはホスト・プロセッサ側のタスクによって行われる．通信コントローラに送られた送信データは，そこでフレームに変換され，バス・ドライバによってバス上に送信される．

受信されたフレームは，バス・ドライバを介して通信コントローラに送られ，そこでフィルタリングされる．フィルタリングの結果，受信条件を満たしたフレームが通信コントローラによってプロセッサで扱える形に変換されて格納される．

このように，各送受信データは，フレーム単位でそれぞれが割り当てられるスロットを通してノード間でやり取りされる．そのため，送信データを生成するホスト・プロセッサのタスクは，そのデータがフレームとして送信されるスロットの開始以前に処理が完了していなければならない．また，受信データを受け取って起動されるホスト・プロセッサのタスクは，その受信データがフレームとして送信されるスロットの終了時点以降にはじめて処理可能となる．

3. 問題のモデル化

3.1. 想定するシステム

本節では，本手法が扱う問題を簡単化するために与えたシステムに対する仮定条件について説明する．

まず，ネットワークに接続されている各ノードのCPU性能はすべて均一とした．これは，スケジューリングの際にノードごとに処理されるタスクの時間を考慮する手間を省くためである．

また，本手法では標準的な機能のみに着目し，通信が確実に保証されるスタティック・セグメントにおいて，それらをスケジューリングする．できるだけ送信されるフレームを通信サイクルの前半のスロットに割り当てることにより，残った後半部分をダイナミック・セグメントで置き換えるように考慮している．この詳細については本章の4節，および次章で説明する．

スタティック・セグメントでは両チャネルに送られるフレームが同一なので，バスをシングル・チャネルとみなすことができ、スケジュールが簡単になる．

3.2. タスクグラフ

本手法では，車載ネットワーク上で動作するアプリケーションを「タスクグラフ」として表現する。これは，センサから受け取った入力データをもとに，複数の処理単位である「タスク」が「メッセージ」とよばれるデータのやり取りを通して強調して動作し，その結果求められた出力データをアクチュエータに与えるという一連の処理を表している．

図5にタスクグラフの例を示す． タスクは周期と処理時間をもち，メッセージは周期とデータサイズ，および，その送受信タスクをもつ．ここでいう周期とは，各タスク，メッセージが実行される時間の間隔を表しており，それぞれが独立してもつものとする。

ここで本手法における仮定として，これらタスク，メッセージの周期はFlexRayの通信サイクル周期の2^n (n=1,2,3,...)倍の値であるものとする．これは，ダイナ

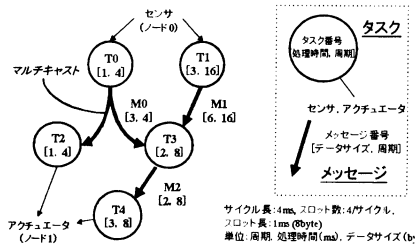

図 5：タスクグラフの例

ミック・セグメントの領域も考慮しつつ、通信を保証するためである．また，通信サイクルの数は $64=2^6$ であることから，これ以外の周期に設定するとフィルタリングにおいてソフトウェアに余分な付加が生じてしまうからである．

メッセージの送信タスクは 1 つであるが，受信タスクはここでは複数である場合も想定し，そのようなメッセージは「マルチキャスト」であると呼ぶことにする．さらに，柔軟性を考慮し，メッセージのフィードバックによるループにも対応できるように配慮した．

また，センサからの入力データを受け取るタスクとアクチュエータへの出力データを送るタスクにはその情報が含まれているものとする。

3.3. スケジューリング

本手法では、タスクグラフ中の各タスク，メッセージを次のようにスケジューリングする．

タスクについては，その処理ノード，およびそのノードにおける実行時間を割り当てる．ただし，センサやアクチュエータに接続されているタスクはあらかじめその処理ノードは固定されているものとし，実行時間のみを割り当てる．また，ノード内でのタスクのプリエンプションはないものとし，タスクの実行時間はその開始から終了まで連続的に割り当てる．

メッセージについては，それを送信するスロットを割り当てる．ただし，同じノード間でやりとりされるメッセージについては，ネットワークを介して送信される必要が無いため,割り当ての対象外とする.また，すべてのメッセージは 1 つのスロット内に収めるものとし,複数のスロットへの分割は行わないものとする．

このとき，同ノードから送信される複数のメッセージは，同じフレームにまとめて同じスロットに送信することができる．そのため、各スロットにフレームを送信するノードはここで決定される。

これらのスケジューリングは，タスクグラフ中のすべてのタスク，メッセージの周期の中で最大となる周期の長さだけ求めればよい．なぜなら，前節で述べたように，タスクグラフ中の周期はすべて通信サイクル周期の 2^n (n=1,2,3,...)倍の値をもつので，各タスクやメッセージは倍々関係になるからである．

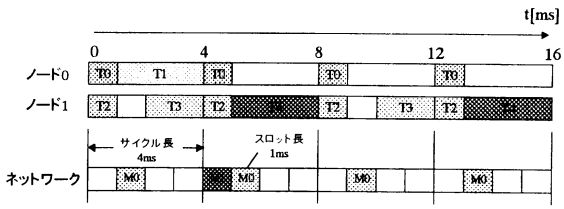

図 6：スケジュールの例

図 6 に，図 5 のタスクグラフをスケジューリングした例を示す。最大周期は 16ms なので，0～16ms までのスケジュールを求めている．

3.4. 設計制約

設計制約にはスケジュールに対する 4 つの時間制約を与えた．はじめに，ここで新たに定義する用語について説明しておく．まず，これらの時間制約は，次で定義されるタスクグラフ上の経路に対して与えられる．

- **経路**

  タスクグラフにおいて，あるタスクからタスクとメッセージを交互に辿って到達できるあるタスクまでの流れを表す部分グラフ．経路の始点，終点のタスクをそれぞれ入力タスク，出力タスクとよぶ．

ループなども考慮して，経路中には同じタスクやメッセージを複数入れることができる．ただし，経路中で連続するタスクとメッセージは送信→受信の形でつながりをもち，その順序は守られているものとする．

次に，実際の制約に関する用語を定義しておく．

- **レスポンス**

  入力タスクに対し，その処理が開始されてから，与えられた経路を通り，その処理結果を反映した出力タスクの処理が終了するまでの時間．

- **フレッシュネス**

  出力タスクに対し，与えられた経路を通り，その処理に反映される入力タスクの処理が開始されてから，当該出力タスクの処理が終了するまでの時間．

経路中のタスクやメッセージの周期の関係から，レスポンスは入力タスク，フレッシュネスは出力タスクの処理タイミングごとにそれぞれ異なる値をとる可能性がある.そこで,新たに次の 2 つの用語を定義する．

- **最大レスポンス**
  すべての入力タスク処理タイミングに対する最大のレスポンス.
- **最大フレッシュネス**
  すべての出力タスク処理タイミングに対する最大のフレッシュネス.

これらを用いた時間制約は次のように定義される.

- **レスポンス制約**
  経路に対する最大レスポンスが，与えられた時間以下でなければならないという制約.
- **フレッシュネス制約**
  経路に対する最大フレッシュネスが，与えられた時間以下でなければならないという制約.
- **入力同期制約**
  出力タスクが共通な複数の経路に対し，その出力タスクのすべての処理タイミングに対するフレッシュネスの差の最大値が，与えられた時間以下でなければならないという制約.
- **出力同期制約**
  入力タスクが共通な複数の経路に対し，その入力タスクのすべての処理タイミングに対するフレッシュネスの差の最大値が，与えられた時間以下でなければならないという制約.

本手法ではこれらの他，次の性質を定義する.

- **スロット余裕度**
  通信サイクルの後ろから連続して残された未使用スロットの数で表されるスロットの空き具合. 未使用スロットの数が大きいほど余裕度は高い.

これは，通信サイクルの後方に残された未使用スロットをダイナミック・セグメントに使用することを考慮した性質である. 本手法では，上の4つの時間制約を設計にあわせて任意の経路，または経路の集合に設定し，それらをすべて満たした上で，よりスロット余裕度の高いスケジュールを求めることを目的とする.

4. スケジューリング手法

4.1. シミュレーティド・アニーリング(SA)

本手法では、基本アルゴリズムとしてシミュレーティド・アニーリング（SA）を採用している.

SAは、問題の変化に対する柔軟性が高いという特徴を持つため、本研究で扱った問題のように、それを解く上での条件や制約が要求分析の中で頻繁に変化していく場合に有用であるといえる.

SAでは,ある初期解から近傍をたどりながら良い解を探索していく手法であるが、その探索過程は複数のパラメタで決定され,「冷却スケジュール」とよばれる.

4.2. 解の表現とスケジュール生成

解は，タスク，メッセージのそれぞれの割り当てに関するパラメタで構成され，これらによってスケジュールが一意に生成される.

タスクの割り当てに関するパラメタは，割り当て順序，処理ノード，割り当て基準，割り当てオフセットの4つである.

割り当て順序は，タスクの割り当て優先度を表しており，この順番によってタスクは実行時間を割り当てられる. 処理ノードは，各タスクが処理されるノードを指す. ただし，センサ・アクチュエータとの入出力を含むタスクについては，この値は固定されている.

本手法では設計制約として，スケジュールから求められる経路の実行時間を対象としているため，それが短くなるようにスケジューリングすることが必要になる. そのためには，各タスクにおいて送・受信されるメッセージは，そのタスクの終了後，開始前にそれぞれ配置されるのがよい. また，それらのメッセージが同ノードで処理されるタスク間にある場合には，そのメッセージの接続先のタスクが,そのタスクの終了後，開始前にそれぞれ配置されるのがよい. 割り当て基準は，そのようにタスクを配置するときの基準を決めるパラメタである。具体的には，当タスクとメッセージを介して接続されるいずれかのタスクを与える。

しかし，この割り当て基準によってタスクが配置される位置が一意に求められるわけではなく、割り当てるタスクに対して周期の異なるタスクやメッセージを基準とした場合，複数の割り当てパターンを持つことがある. このとき，どのパターンをとるのかを決めるのが,割り当てオフセットである.オフセットの値は，スケジュール範囲の中で時間の早い方から,0,1,2･･･の順で各割り当てパターンに対応している.

メッセージの割り当てに関するパラメタは，割り当て順序のみであり，その意味はタスクと同様である. ただし，その中には（タスクグラフの最大周期分のスロット数−1)の数だけ「空メッセージ」が挿入される. メッセージは，タスクのように割り当て基準の概念を持たず，前のスロットから順番にメッセージを詰めていくため，そのままだと前半のスロットだけを用いたスケジュールしか求められない. よって，空メッセージでスロットを擬似的に埋めることにより，実際のメッセージを後半にも配置できるよう考慮している.

以上のようにスケジュールを生成するが、これらのタスク・メッセージはほかのタスク・メッセージに邪

魔されることにより，割り当てられないことが起こりうる．そのような場合は，次節に述べるコストの計算時にペナルティとして処理される．

4.3. コストの設定

以上のようにして得られたスケジュールに対して、設計制約を与えた経路をたどって実行時間を求めていく．各値とそれらのデッドラインとの間に，本手法では次のようなコストを設けた．

デッドライン以下では，値に比例したコストを与える．デッドラインを超える場合には，ペナルティとして，デッドライン時点のコストより大きな定数を与えた上で，デッドライン以下での傾きよりも大きな傾きを設定する．これによって，デッドラインを超えにくくし，かつ，より小さな値を得られるようになる．

スケジュール生成時に割り当てられないタスクやメッセージが経路中に含まれると，実行時間が求められない．そのような場合には，割り当てられなかったタスクやメッセージを含む経路ごとにペナルティとして定数を与えることとした．この際，割り当て不能時のペナルティをデッドライン超過時のペナルティよりも大きく見積もることが必要である．

スロット余裕度は，通信サイクルの後ろから連続して残された未使用スロットの数が多いほど良いので，通信サイクルの中で最後に使用されているスロット番号に比例したコストを与える．その係数は，スロット数や経路数に応じて決定されるべきである．

5. 適用

5.1.適用対象

本論文では，車載制御アプリケーションとして必要とされる機能を表現したタスクグラフ[1]に対し，本手法が想定する FlexRay システムでの実現を考慮して改良を加えたものを，適用対象とした．

これは，タスク数が 28，メッセージ数が 31 で，周期は 2ms，4ms，8ms の 3 つで構成されている．ノード数は 6 で,全タスク中の 17 タスクがこれらに割り振られる。通信サイクル長は 2ms,スロット数は 20 とした。

時間制約は，各センサからアクチュエータまでの全ての経路に対し，あるデッドラインを設定してレスポンス，およびフレッシュネス制約を与えた．

5.2.複数の冷却スケジュールによる実行

はじめに,SA の 4 つのパラメタを 3 通りずつ変化させた 81 パターンの冷却スケジュールに対し、同じ 10 パターンの初期解による探索を行った．終了条件は，150 万回目の探索終了時とし、各パターンにおいて制約を満たしたかどうか，および，制約を満たしたものはスロット数がいくらになったのかを記録した．

結果では，全 810 パターン中 32 パターンで制約を満たす解が得られ，その中でスロット数を最小で 11 まで抑えることができた。冷却スケジュールごとに見ると、制約を満たした数は最大で 10 パターンの初期解のうち 5 パターンであった。

5.3.冷却スケジュールの絞込みと調査

上の結果に用いた冷却スケジュールの中から 9 つのパターンを無作為に選び，それぞれに対して探索回数を 1,000 万回に増やして実行してみた。

結果，8 つの冷却スケジュールで制約を満たす解が得られたパターンが増加し，最大では 9 パターンにまで達した．この結果からすると，このような問題に対して本手法は十分に有効であるといえる。

ただし，残りの 2 つの冷却スケジュールでは制約を満たす解の数が変化せず，この数は温度の冷却速度に依存することがわかった。よって，本手法を適用する上では，各問題に対し適切な冷却スケジュールを求めることが必要であり，そのためには，より厳密な実験と調査を進めていくことが必要である．

6. 結論

本研究ではまず，自動車メーカとの要求分析のなかで，想定されるシステムの条件と，そこで実現されるアプリケーションに対する 4 つの時間制約，およびスロットの使用率に対する制約を定義した．

具体化された問題に対し，基本アルゴリズムに SA を用いて，処理や通信における静的なスケジュールを求める手法を考案した．

本手法を，実際の自動車内での実現が想定されるアプリケーションを模したモデルに適用した．冷却スケジュールを複数のパターン用意することで，それらと得られる解との関係を探るとともに，本手法の有効性を確認することができた．

さらなる実験と調整により，得られる解と冷却スケジュールの関係を把握し，問題に適したパラメタのチューニングを行うことが，今後の課題として必要になる．また，将来的に FlexRay が実用されるようになれば，その実際のシステムに対して本手法が有効かどうかを確認することが必要であり，それも今後の課題として残されている．

文　献

[1] N.Kandasamy, J.P.Hayes, and B.T.Murray, “Dependable Communication Synthesis for Distributed Embedded Systems," Proc. 22nd Int'l Conf. on SAFECOMP, Lecture Notes in Computer Science, vol.2788, pp. 275-288, Edinburgh, UK, Sept.2003.