2004－ＳＬＤＭ－114　(14)
2004／3／19

CANにおけるメッセージスケジューリング手法

松谷元気[†]　飯山真一[††]
冨山宏之[†]　高田広章[†]

現在，自動車の車載 LAN として CAN が幅広く採用されている．従来から CAN（Controller Area Network）のスケジューリングについての研究は行われてきているが，衝突の発生を回避するためにどのようにメッセージスケジューリングをするかという研究はされていない．そこで本論文では実際のシステムを考慮した上で，CAN に接続されているノード内での衝突を回避するようなスケジューリング手法を提案する．そして，このスケジューリング手法に有効なメッセージセットの例を示し，また，実際に使用検討されているメッセージセットに対してもこのスケジューリング手法をつかいスケジューリングをし，その評価も同時に行う．
キーワード　CAN，車載 LAN，リアルタイムスケジューリング

Message Scheduling Method for CAN

GENKI MATSUTANI ,[†] SHINICHI IIYAMA ,[††] HIROYUKI TOMIYAMA [†]
and HIROAKI TAKADA[†]

CAN (Controller Area Network) has been widely used in in-vehicle LAN. Until now, a number of research on message-scheduling for CAN have been conducted. However, they do not address how to schedule messages to avoid collisions. In this paper, we propose a scheduleing method to reduce collisions in nodes connected to CAN . Then we show an example of message set effectively scheduled by our method. We also apply our method to a actual message set to evaluate the effective of our method.
Key Words CAN, in-vehicle LAN, realtime-scheduling

1. はじめに

近年の自動車の車両設計では環境を配慮した燃料消費量と排ガスの削減，安全機能の向上，居住性の向上，エンターテイメント性の向上が求められるようになってきた．このような様々な要求は電子制御により実現されており，そのため ECU(Electronic Control Unit) の数も増加する傾向にある．このような状況では ECU 間を結ぶワイヤハーネスの使用量が増えるため，車両重量の増加し，燃料消費量と排ガスの削減の妨げとなる．このワイヤハーネスの量を減らすために車載 LAN の導入されている．車載ネットワークはその応用ごとにグループ分けがされ，そのうちでもリアルタイム性が重視されるパワートレイン系のネットワークで CAN(Controller Area Network)[1] が幅広く採用されている．

従来から CAN におけるスケジューリングに関する研究はリアルタイムスケジューリング理論の分野において行われている．これらの従来手法では，あらかじめ決定されているオフラインスケジューリングされたものをどのように CAN システムで実現するかというものである．従って，従来の手法を用いて衝突を回避する，その発生回数を軽減するといったことはできない．

そこで本論文では，CAN システムの各ノードで与えられているメッセージセットに対して，メッセージの衝突が起こらないスケジューリング手法を提案する．具体的にはメッセージセットのメッセージのそれぞれにオフセット（初回送信要求時刻）を与え，必要に応じてメッセージの送信要求周期を変更することを用いたスケジューリング手法である．

CAN のメッセージスケジューリングの流れは本手法で仕様を満たすメッセージスケジューリングのパターンを数個得た後，それらに対して最悪遅れ時間解析を行う．そして，その中のどのパターンが最適であるか

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

を評価することにより，最適なものを得るというものである．従って，この研究は最悪遅れ時間解析の前段階に位置するものである．

本論文では，まず 2 章では，想定するシステムの構成について述べる．3 章では，CAN に接続されたノードで送信要求がされるメッセージセットに対するスケジューリング手法を提案する．4 章では，そのスケジューリング手法を有効なメッセージセットと実際に使用検討されているメッセージセットに適応し，スケジューリングをした事例について述べ，その評価を行う．

2. 関連研究

CAN システムに関する研究はこれまでに数多く行われてきた．

CAN の最大遅れ時間の評価に関する研究として，Tindell らの研究がある [2)3)4)]．この研究では CAN に接続されているノード内で，1 つのメッセージにつき 1 つの送信メッセージボックスが与えられているような状況での議論がなされている．しかし，実際のシステムではこのように充分に資源を与えられることは少なく，実用的なシステム構成で考える必要がある．

また，Tindell らの研究を発展させたものとしては，文献 5) がある．この論文では実際の資源を考慮した CAN を用いたシステムにおける最大遅れ時間解析の手法が提案されている．

CAN におけるメッセージスケジューリングに関する研究では文献 6) がある．こちらでは CAN においてメッセージの優先度を考慮したスケジューリング手法を提案しているが，決定したオフスケジューリングどおりに CAN 上でメッセージ送信を行う手法について述べているが，そのオフスケジューリングの決定方法までは述べられていない．また，こちらも資源という意味で実用的なシステム構成を考慮するまでには至っていない．

3. システム構成

本章では CAN の概要と本研究で想定した CAN を用いたシステム構成をハードウェア，ソフトウェアの点から述べる．なお，本研究で想定したシステムは文献 5) と同じ構成である．

3.1 CAN(Controller Area Network)

CAN(Controller Area Network) はリアルタイム性が重視される制御系ネットワークで用いられている．CAN の通信速度は最大で 1Mbps である．また，バスが解放されているときは，全ノードがメッセージ送信を行うことができるマルチマスタ方式で，各ノードのバス使用の優先権を決めるために，アービトレーション (調停) が必要となる．このアービトレーションを行うために各メッセージ・フレームには 11 ビットの識別子 (ID) があり，データは最大で 8 バイトを載せることが可能である．詳細は文献 1) を参照のこと．

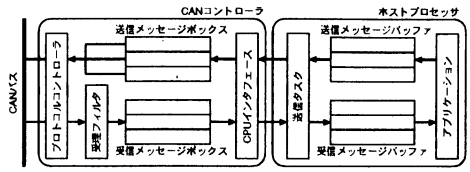

図 1 CAN ノードの構成

3.2 ハードウェア構成

本研究では複数のノードが 1 つの CAN バスにつながっているシステムを考える．また，図 1 に示すようにこの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ている．

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，ホストインターフェイスから構成される．コントローラには送信用と受信用のメッセージボックスがある．

送信用のメッセージボックスが複数個あり，それぞれのボックスのメッセージに対して同時に送信要求がされた場合は，それらのメッセージの優先度によりノード内でのアービトレーションが行われ，1 つのメッセージがそのノードの送信メッセージとして CAN バスに載せられる．このとき CAN バス上で複数のノードが送信要求をしている場合にはここでも優先度によりアービトレーションがされる．メッセージの送信が完了後に，コントローラは送信が完了したことをホストプロセッサへ割込みにより通知する．受理フィルタは，バス上に流れている全てのメッセージの中からそのノードが受信すべきメッセージかどうかを判断し，受理すべきメッセージの場合は受理用のメッセージボックスに格納し，受理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．

3.3 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．

他のノードに対して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高い優先度を持つメッセージ送信用メッセージボックスに格納する．メッセージボックスに格納されたメッセージは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．

また，一定の時間間隔で割込みがかかり，送信メッセージボックスにメッセージが存在するかが調べられている．従って，1つのメッセージの送信が終わっても，次の割込みで送信メッセージボックスを調べるまでは，次のメッセージが送信されることはない．設計時にこの時間間隔にあわせて各ノードで送信されるメッセージの送信要求される周期が決められる．この割込みの間隔は各ノードで異なる．

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，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．

受理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理メッセージボックスに格納され，受理したことを割込みにより受理タスクに通知する．通知を受けた受理タスクは，受理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理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理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. 提案手法

本論文ではノード内でメッセージの送信要求が同時に発生し，衝突が起こることを回避するために各ノードで送信要求されるメッセージに対して適切なオフセットを持たせ，また，必要に応じて各メッセージの送信要求の周期を変更するスケジューリング手法を提案する．

この手法でノード内での衝突を回避することにより，ノード内の他のメッセージが原因で送信が遅延されることがなくなり，ネットワークのピーク負荷を抑えることができる．また，優先度逆転現象[7]という状態に陥る可能性も軽減できると考えられる．優先度逆転現象とは高優先度のメッセージが低優先度のメッセージに邪魔されてしまう優先度逆転現象という状態である．この現象が起こるのは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く，同時に複数のメッセージの送信要求がされる可能性がある場合である．従って，同時

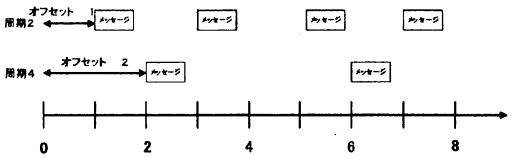

図 2 オフセット付きメッセージの概念

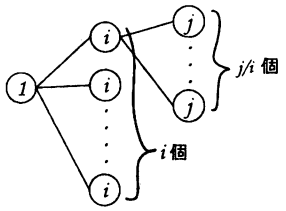

図 3 スケジュール木の概念

発生する頻度を抑えることにより，この現象に陥る可能性を低くできる．よって，本スケジューリング手法でノード内における同時発生の可能性を無くすことで，優先度逆転現象に陥る可能性を軽減することができる．

4.1 オフセット

CAN の各ノード内では全メッセージが周期的に送信要求されるメッセージとなっている．もし，全メッセージがシステム開始時に同時に送信要求されると，ノード内で衝突が起こり，ネットワークのピーク負荷が上がる．このようになることを防ぐために実際のシステム設計ではシステム開始時から最初にメッセージが送信要求されるまで時間つまり，オフセットを各メッセージに持たせることが多い．

図 2 は送信要求の周期が 2 と 4 のメッセージにそれぞれオフセットを 1 と 2 を持たせた例である．この場合はオフセットを持たせたことにより，周期が 2 のメッセージと周期が 4 のメッセージが同じタイミングで送信要求されことがない．つまり，2 つのメッセージ衝突を回避できている．

各メッセージごとにオフセットを持たすことでこの場合のようにメッセージの衝突を防ぐことが可能になる．

4.2 スケジュール木

各メッセージに対してそのオフセットを求め，かつ，そのメッセージセットがスケジューリングできるかどうかの判定をする方法としてスケジュール木を用いる．スケジュール木は周期が j のメッセージが j/i 個あればそのメッセージはまとめて 1 つの周期が i のメッセージであると解釈できる概念から構成される．図 3 にスケジュール木の概念図を示す．

スケジュールの木の節点にはそれぞれ数が付けられ

ており，メッセージの周期に対応するものである．また，これとは別に各節点には $offset$ というものが与える．以下，節点 p に対する $offset$ を $offset(p)$ と表す．以下に示すスケジ ュール木のルールに従い，木を展開し，各メッ セージとその周期と同じ数字のスケジュ ール木の節点とを対応付ける．その対応付けされた節点に与えられている $offset$ がそのメッセージのスケジ ューリングで与えられるオフセッ トとなる．

(1) 根は 1 が付けられている節点とする．また，この節点の $offset$ は 0 である．

(2) ある節点の数字が i であればその子として j という節点を j/i 個だけ作ることができる．このとき，i, j, j/i は整数でなければならない．

(3) 1 つの節点につき 1 つのメッセージを対応させることができ，また，そのメッセージに対応付けられた節点はそこから新たな節点を展開することはできない．

(4) ある節点 p がその子の節点 q を作るとき，この節点 q の $offset(q)$ は以下のようにして決められる

$$offset(q) = offset(p) + t(p) \cdot a \quad (1)$$
$$a = 0, \cdots, n-1$$

ここで，$t(p)$ は節点 p の数字，n は節点 q の兄弟の数を表している．例えば，ある節点が親に対して 2 番目の子であるときは $a = 1$ となる．

このようにして，スケジュール木の各節点にノード内のメッセージセットの全メッセージを割り振ることができれば，オフセットを与えることにより衝突を回避することができる．

メッセージセットが {4, 4, 4, 8} の周期のメッセージセットを例に例えると，**図 4** のようなスケジュール木が構成され，メッセージとその周期と同じ節点とが対応付けされる．この図において，円内の数字は節点に与えられた数であり，円外の数字はその節点の $offset$ である．また，図中の二重丸の節点はメッセージに対応付けされた節点を表している．このようにスケジュール木を用いることより各メッセージはその周期と同じ数が与えられている節点に対応付けられ，各メッセージのオフセットを一意に決めることができる．

4.3 スケジュール木を用いたスケジューリング手法

スケジュール木を用いたスケジューリング手法を提案する．

スケジュール木で新たに展開できる節点のリストを expandable，これからスケジュール木の節点に対応させるメッセージ周期のリストを remain，スケジュール木に既に対応付けられているメッセージのリストを

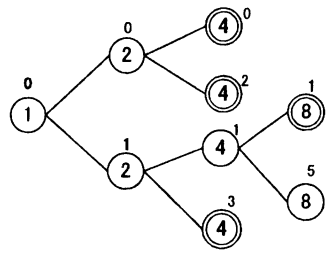

図 4 スケジュール木の例

messagein とする．スケジュール前のリストはそれぞれ expandable は {1}, remain はスケジューリングすべきメッセージセット, messagein は空集合となっている．また，前節で述べたようにリスト expandable 中の節点はそれぞれ $offset$ を持っており，子を作るときは式 (1) に従って，その子にも $offset$ を与える．リスト remain，リスト expandable に対して以下の操作を行うことによりメッセージにオフセットを与える．

(1) リスト remain を昇順に並べる．

(2) リスト expandable の各要素と比べて

(a) リスト remain の先頭の要素と等しいとき
対象となっているリスト expandable の要素を expandable から，remain の先頭の要素を remain からそれぞれ除き，除かれた expandable の要素とそのオフセットとを組にしてリスト messagein に入れる．

(b) リスト remain の先頭の要素と等しくないとき
対象となっているリスト expandable の要素とリスト remain の先頭の要素との間に倍数関係があれば，リスト remain の先頭の要素と同じ数の要素を子として対象となっているリスト expandable の要素が作る．倍数関係にない場合は何もしない．

(3) リスト remain が空であれば終了．そうでなければ (2) に戻る．

以上の操作でリスト messagein にメッセージセットに対応する数とそのオフセットの組合せが数パターン得られる．

4.4 周 期 変 更

スケジューリングするメッセージセットではどのように各メッセージに対してオフセットをつけても，衝突が起こってしまう組が存在する．例えば，周期が 5 と 7 を含むメッセージセットなどがそうである．

スケジュール木で言い換えると与えられたメッセージセットに対してどのようにスケジュール木を構成したとしても，そのスケジュール木のどの節点にも対応

付けできないメッセージが出てくることがある．このようにスケジュール木を使うことで，そのメッセージセットがどのようにオフセットをつけても衝突を回避できないものかどうかがわかる．

このようにオフセットを付けても衝突を回避できないときには，そのメッセージセットのいくつかのメッセージの送信要求の周期を変更する必要がある．

メッセージの周期を変更するときは決められたその周期よりも小さい周期へと変更することが望ましい．これは設計で求められた制御周期よりも大きくなるような周期変更は，制御の品質を落とすことになりかねないからである．

また，周期の変更は変更の幅が少ないことが望ましい．これは設計の要求に対してできるだけ忠実にスケジューリングするためである．

4.5 提案スケジューリング手法

先に述べたスケジュール木を用いて各メッセージのオフセットの決定及び，メッセージの送信要求の周期変更を行うメッセージスケジューリング手法を提案する．

スケジュール木で展開できる節点のリストをexpandable, これからスケジュール木の節点に入れるメッセージ周期のリストをremain, スケジュール木に既にいれたメッセージのリストをmessageinとする．スケジュール前の各リストはそれぞれexpandableは{1}, remainはスケジューリングすべきメッセージ集合, messageinは空集合となっている．また，前節で述べたようにリストexpandable中の節点はそれぞれ$offset$を持っており，子を作るときは式(1)に従って，その子にも$offset$を与える．リストremain, リストexpandableに対して以下の操作を行うことによりメッセージにオフセットを与え，必要に応じてその周期の変更を行い，スケジューリングをする．

(1) リストremainを昇順に並べる．

(2) リストexpandableの各要素に対して

(a) リストremainの先頭の要素との比が2未満のとき

対象となっているリストexpandableの要素をexpandableから，remainの先頭の要素をremainからそれぞれ除き，除かれたexpandableの要素とそのオフセットとを組にしてリストmessageinに入れる．

(b) リストremainの先頭の要素との比が2以上のとき

リストexpandable内のその要素の子をその比を構成する数だけ作る．そして，子を作った要素をリストexpandableから除く．

(3) リストremainの要素でリストexpandableのどの要素の倍数にもなっていない要素があるとき，その要素に対して以下の操作を行う．リストexpandableの各要素の倍数で，対象となるremainの要素に対して小さいかつ最も近い数の集合を作り，その集合で最も対象の要素に近い数を探索し，対象のremainの要素をその数に変更する．

(4) リストremainが空であれば終了．そうでなければ2へ戻る．

以上の操作を行って，最終的にリストremainが空になったときのリストmessageinに入っているメッセージの周期及び，それと組になっているオフセット値がメッセージスケジューリングのパターンである．以上の操作では与えられるメッセージセットによって複数個のスケジューリングパターンが得られる可能性がある．

5. 適用と評価

提案したスケジューリング手法をこの手法が有効であるメッセージセット及び，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適応し，そのスケジューリングを行い，評価を行った．

5.1 評価指標

与えられたメッセージセットに対してスケジューリングをした結果を次の2つの指標で評価を行う．1つ目はノード内でのメッセージの衝突を回避してスケジューリング可能かどうかであり，2つ目はメッセージの周期変更を行ったスケジューリングではその周期の変更により送信スロットの充足率がどのように変化したかを検証した．ここで送信スロットの充足率とはノード内で送信可能なタイミングに対してメッセージ送信がどれくらいの割合で占めているかを示すものである．

5.2 提案手法によりスケジューリング可能なメッセージセット

メッセージの送信要求の周期がそれぞれ3, 6, 7, 13, 24, 28のメッセージセットに対してのスケジューリングを提案メッセージスケジューリング手法を用いて行った．この結果，24個のスケジューリングのパターンを得られた．24個の全てのスケジューリングパターンがメッセージ間の衝突を回避できるものになっていた．また，この得られた24個のスケジューリングパターン全てで5つのメッセージのうちの周期が7, 13, 28のメッセージがそれぞれ6, 12, 24の周期へと変更された．これにより，この周期変更による送信スロッ

表 1　スケジュール結果の 1 つ

周期	変更後の周期	オフセット
3	3	2
6	6	4
7	6	1
13	12	9
24	24	15
28	24	3

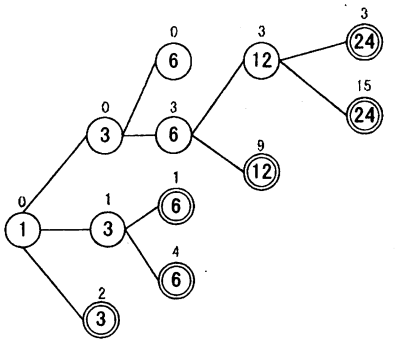

図 5　構成されたスケジュール木

トの充足率の変化は以下より約 3.8%となることがわかる.

$$0.038\left(\simeq\left(\frac{1}{6}-\frac{1}{7}\right)+\left(\frac{1}{12}-\frac{1}{13}\right)+\left(\frac{1}{24}-\frac{1}{28}\right)\right) \quad (2)$$

表 1 に得られた 24 個のスケジューリングパターンのうちの 1 つを示す.

このスケジューリングを行ったときに構成されたスケジュール木を図 5 に示す. この図において二重丸の節点はメッセージが対応付けされた節点で, 節点の上の数字はそのメッセージのオフセットを表す.

5.3　実際のメッセージセットに対する適用

適用対象としたのは, 実際のシステムでの使用が検討されている, あるノードにおいて送信要求がされる約 20 個のメッセージから構成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したスケジューリング手法でスケジューリングを行った. その結果, 大半のメッセージの送信要求周期を変更し, スロット充足率は約 4%上昇した. 衝突を回避するように各メッセージにオフセットをつけたスケジューリング結果を得ることができた.

6. 結　　論

本論文では自動車で実際に使用検討されている CAN を用いたシステムについて述べた. そして, そのシステムのノード内で送信要求されるメッセージセットに対して, オフセット及び, その送信要求される周期の変更によってノード内で衝突が発生しないスケジューリング手法を提案した. そのスケジューリング手法を用い, 本手法が有効なメッセージセット及び, 実際に自動車で使用の検討されているメッセージセットに対してスケジューリングを行った結果, バス負荷率を数%上昇させ, 各メッセージに対して適切なオフセットを与えることによりノード内でメッセージ衝突を回避するスケジューリング結果を得ることができた.

また、本論文では 1 つのノードに与えられたメッセージセットに対して, そのメッセージ間での衝突を回避するスケジューリング手法を提案した. しかし, 実際のシステムでは CAN バスに複数のノードが接続されている. よって, システム全体で各ノードが送信するメッセージ間の衝突が最小になることを考えなければならない. そこで, 提案したスケジューリング手法で得られたスケジューリングパターンのうちでどのパターンが各ノード間の送信メッセージの衝突回数を抑えることができるかを考察し, それを考慮した全ノードのメッセージに対するスケジューリング手法を提案することが今後の課題である.

謝　　辞

適応評価において, データを提供していただいたトヨタ自動車 (株) に感謝いたします. また, 研究に対してご意見をいただきました関係各位に御礼を申します.

参 考 文 献

1) International Standards Organization: ISO11898, Road Vehicles – Interchange of digital infomation – 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 Sience, 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 System Symposium(1994)
4) Tindell, K.W., Burns, A. and Wellings, A.J.:Calculating Controller Area Network (CAN) Message Response Times (1995).
5) 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, Vol. 45, No. SIG1(ACS4), pp. 66–76(2004).
6) Radu Dobrin, Gerhard Fohler: Implementing Offline Message Scheduling on Controller Area Network(CAN),In 8th IEEE Int. Conf. on Emerging Technologies & Factory Automation Nice (2001).
7) Sha, L., Rajkumar, R.and Lehoczky, J. : Priority Inheritance Protocols: An Approach to Real-Time Syncronization, IEEE Trans. Comput, Vol.9, No.39, pp. 1175-1185(1990)