Vol.2013-SLDM-160 No.7
Vol.2013-EMB-28 No.7
2013/3/13

車載ネットワークにおける CAN-Ethernet プロトコル変換アルゴリズム

松村 潤[1,a)]　松原 豊[1]　高田 広章[1]　大井 正也[2]　豊島 真澄[2]　岩井 明史[2]

概要：車載ネットワークのゲートウェイに用いられる CAN-Ethernet プロトコル変換アルゴリズムの研究がこれまでに行われている．従来の手法は CAN メッセージの遅れ時間とゲートウェイの処理量の低減に着目している．しかし，どちらかを低減させる手法なので，CAN メッセージのデッドライン制約とゲートウェイの処理量のトレードオフを考慮して，適切にネットワークを設計することが困難である．そこで本論文では，CAN メッセージのデッドライン制約を満たすことに着目したデッドライン型アルゴリズムを提案する．提案手法は，メッセージの送信周期内に通信が間に合うようにゲートウェイ内での待機時間を調整する手法である．提案手法の評価を行うために，離散イベントシミュレータ OMNeT++をベースにシミュレーション環境を開発した．評価実験の結果，提案手法ではデッドライン制約を守りつつ，従来手法に比べてネットワーク負荷を低減できることを確認した．

キーワード：CAN，Ethernet，プロトコル変換アルゴリズム，ゲートウェイ，車載ネットワーク

CAN-Ethernet protocol convert algorithm for automotive networks

Jun Matsumura[1,a)]　Yutaka Matsubara[1]　Hiroaki Takada[1]　Masaya Oi[2]　Masumi Toyoshima[2]
Akihito Iwai[2]

Abstract:
CAN-Ethernet protocol convert algorithms for gateways in automotive control networks have been studied so far. These algorithms can reduce either delay time of CAN messages or the processing amount of a gateway by changing parameters such as conversion timing and buffer size. Due to a tradeoff between them, determination of proper parameters for each network is difficult. To meet deadline constraints of CAN messages, we proposed a new algorithm, called deadline algorithm. The deadline algorithm adjusts the wait time at the gateway according to transmission period of each CAN message. To evaluate the proposed algorithm, we have developed a simulation environment based on OMNeT++ which is an open-source discreate network simulator. The results of evaluations present that the deadline algorithm can save the network resources with satisfying the deadline constraint.

Keywords: CAN, Ethernet, protocol convert algorithm, gateway, automotive network

1. はじめに

近年，DoIP（Diagnostics on Internet Protocol）技術を用いた診断ツールや，画像処理を伴う車線維持機能など自動車の機能が高度化している．自動車を制御するシステム（以下，車載制御システム）の複雑化が進むに従って，車載ネットワークを流れるデータ量が増加しており，業界標準の通信プロトコルである CAN（Controllar Area Network）を用いると，ネットワーク帯域が不足するという問題が出てきている．CAN は最大通信速度が 1Mbps であり，画像などの大きなデータを扱う場合には適していない．

そこで，世界中のオフィスや家庭で使用されている Eth-

[1] 名古屋大学
Nagoya Univercity, 464–8601, Japan
[2] 株式会社デンソー
DENSO CORPORATION, Japan
[a)] j_matsu@ertl.jp

ernet が，その帯域幅の拡張性・柔軟性から注目を集めるようになってきた．具体的には，マルチメディア系 ECU が接続するネットワークや，車外のインターネットと制御系ネットワークを接続するバックボーンネットワークなどに Ethernet を採用することが検討されている．故に，今後の車載ネットワークとして，CAN と Etheret が混在するネットワーク構成（以下，CAN-Ethernet 混在ネットワーク）が採用されてゆくと考えられる．CAN-Ethernet 混在ネットワークを含む車載制御システムの開発が盛んになると，設計工程において，容易にネットワーク構成を評価するための手法が必要となる．

CAN-Ethernet 混在ネットワークにおいて重要となるのが，CAN と Ethernet の各プロトコル間を中継するゲートウェイである．ゲートウェイは，PDU（Protocol Data Unit）の変換を行いデータを中継するが，変換方法によっては，データの中継に時間がかかり，データの応答時間が要求を満たせなくなる可能性がある．その結果，車載制御に問題が生じ，事故へと発展する恐れがある．コスト面から，効率よく変換を行いつつ，ゲートウェイの資源（CPU，バッファサイズ等）を低減することも重要である [1]．

本論文では，デッドラインに着目したデッドライン型アルゴリズムを提案する．優先度に着目した既存のアルゴリズムでは，平均遅れ時間の短縮や CPU 負荷率の低減には貢献できるが，デッドラインについては考慮していないため，要求されるデッドライン制約を満たせない可能性がある．それに対して，提案手法では，CAN メッセージのデッドライン制約を守るようにゲートウェイでの待機時間を調整しながら，ネットワーク負荷を妥当な値に抑えることを目標にする．ネットワーク構成の評価を行うために，既存シミュレータ OMNeT++をベースにシミュレーション環境を開発した．そのシミュレーション環境を用いて最大遅延率やバス負荷率などの値を評価し，デッドライン制約を守りつつネットワーク負荷を低減できることを確認する．

本論文の構成は以下の通りである．2 章で関連研究を述べたあと，3 章で対象とするネットワーク環境について説明する．4 章で既存研究の問題点を示し新たなアルゴリズムを提案し，詳細を述べる．5 章で自動車メーカから提供されたメッセージセット（一部修正）を用いた評価を行い，本手法の有効性を示し，6 章で本論文をまとめる．

2. 関連研究

Kern らは，CAN-Ethernet プロトコル変換アルゴリズムを 4 種類提案した [2]．提案しているアルゴリズムは車載ネットワーク内に設置されるゲートウェイで使用されること想定している．提案アルゴリズムの評価として，単一ノードを 2 つのゲートウェイを中継して接続したネットワーク構成で，メッセージの平均遅れ時間やゲートウェイの CPU 負荷率などの項目を評価している．結論として，

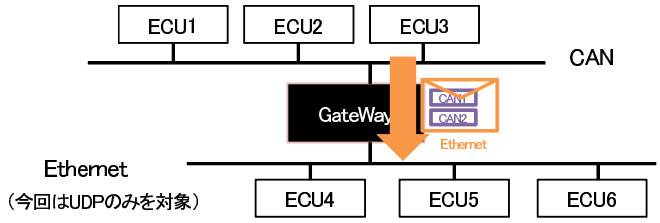

図 1　対象とするネットワーク構成と変換方向
Fig. 1　Network configuration and conversion direction

ゲートウェイを用いるネットワークの要求に応じて，適切なアルゴリズムを選択する必要があるとまとめている．

実際の製品にも組み込まれているアルゴリズムがある．ADFweb 社の製品 HD67048[3] では，受信した CAN メッセージをバッファに格納していき，CAN メッセージがバッファに一定数格納されるか，最初に到着したメッセージから一定時間後が経過するかの 2 つのタイミングで Ethernet パケットへ変換を行う．

また，ネットワーク構成を評価するシミュレータが多く開発されている．ネットワークシミュレーションソフト OPNET[6] は CAN モデルのシミュレーション精度がビット単位と非常に高いが，シミュレーション時間が長くなる傾向にある．Lund 大学にて開発されている TrueTime[7] は，Matlab/Simulink ベースのシミュレータであり，設計段階におけるネットワーク構成の評価にはふさわしくないということが調査の結果明らかになった．

3. 対象とするネットワーク環境

図 1 は本論文で対象とするネットワーク構成と変換方向を示した図である．本論文では CAN → Ethernet 方向のプロトコル変換アルゴリズムに注目する．通信される各 CAN メッセージの送信周期をデッドラインとし，特に次周期時刻を絶対デッドラインと呼ぶ．前周期で送信された CAN メッセージは絶対デッドラインまでに目的地となる ECU に到着することが求められる．

4. デッドライン型アルゴリズム

4.1 既存アルゴリズムの問題点

ここで，既存研究 [2] で提案されている 4 種類のアルゴリズムとその問題点について説明する．まず 1 種類目が基本型である．これは CAN メッセージ 1 つに対して 1 つの Ethernet パケットへ変換するアルゴリズムである．2 種類目がバッファ型である．これは，ゲートウェイのバッファに CAN メッセージを格納していき，バッファが満杯になる（バッファフル）か，あらかじめ定めた変換周期が経過する（タイムアウト）かのどちらかのタイミングで Ethernet パケットへ変換するアルゴリズムである．3 種類目が時間型である．バッファ型の拡張であり，変換周期を管理する際に，CAN メッセージの優先度を考慮するアルゴリズム

Vol.2013-SLDM-160 No.7
Vol.2013-EMB-28 No.7
2013/3/13

表 1　変換アルゴリズムのパラメータ設定
Table 1　Parameters of convert algorithms

	バッファサイズ	変換周期	最大前倒し時間	緊急メッセージ	CAN見積もり遅れ時間
basic	1	-	-	-	-
buffer	○	○	-	-	-
time	○	○	○	-	-
urgency	○	○	○	○	-
deadline	○	○	-	-	○

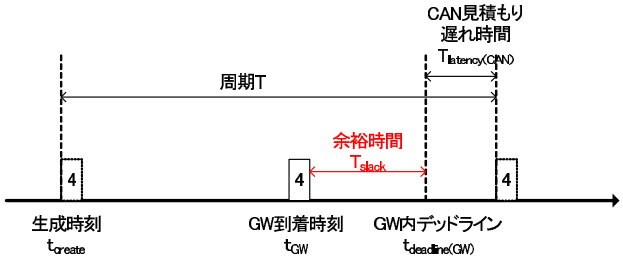

図 2　デッドライン型アルゴリズムに関する用語
Fig. 2　Words of the deadline algorithm

である．優先度の高い CAN メッセージが到着したとき，次回変換予定時刻を前倒し，ゲートウェイでの待機時間を短くする．前倒す時間を求める式は様々な記述が可能である．最後に 4 種類目が緊急型である．緊急メッセージを設定し，そのメッセージが到着したときには他の状況を一切無視してその時点で Ethernet パケットへ変換するというアルゴリズムである．緊急メッセージではないメッセージの取り扱いは，時間型と同じである．

これらのアルゴリズムは，優先度を考慮してメッセージごとに遅れ時間を調整する手法であるといえる．これらの手法では，平均遅れ時間の短縮や CPU 負荷率の低減には貢献できるが，デッドラインについては考慮していないので，要求されるデッドライン制約を満たせない可能性がある．

4.2　提案手法

本節では，提案するデッドライン型アルゴリズムについて述べる．本手法では，全てのメッセージがデッドライン制約を守れるようにしつつ，バス負荷率などのネットワーク資源を妥当な使用量に抑えることを目標とする．

変換アルゴリズムのパラメータ設定を表 1 に示す．上から順に，基本型，バッファ型，時間型，緊急型，デッドライン型のパラメータ設定を表している．最大前倒し時間とは時間型アルゴリズムで用いるパラメータであり，この時間をある CAN メッセージの優先度で割った値がその CAN メッセージ到着時の前倒し時間となる．デッドライン型で新しく定義するパラメータとして CAN 見積もり遅れ時間がある．CAN 見積もり遅れ時間とは，ある CAN メッセージがゲートウェイを通過したあとの中継先 CAN サブネットワーク内で遅れ得る時間の見積もり値である．この値には，すべてのメッセージに対して一定値を用いたり，メッセージごとにその送信周期の一定割合値を用いたりする．

Ethernet パケットへの変換タイミングは，既存アルゴリズムと同様に，バッファフルとタイムアウトの 2 つを用いる．図 2 は，デッドライン型の動作の仕組みを説明する図である．周期 T の CAN メッセージ 4 の送信状況を表しており，横軸はシミュレーション時間の進行を表す．まず，時刻 t_{create} に生成・送信され，ゲートウェイに時刻 t_{GW} に到着する．CAN 見積もり遅れ時間 $T_{latency(CAN)}$ と周期 T を用いて，CAN メッセージが GW 内で待機可能な限界時刻である GW 内デッドライン $t_{deadline(GW)}$ を求める．そして，$t_{deadline(GW)}$ がゲートウェイ到着時刻 t_{GW} より前の時刻を示しているならば，即座に Ethernet パケットへ変換を行う．即座に変換する必要がなければ次に，$t_{deadline(GW)}$ と次回変換予定時刻 $t_{nextconv}$ を比較し，$t_{deadline(GW)}$ のほうが近い場合には $t_{nextconv}$ を $t_{deadline(GW)}$ の値に更新する．$t_{nextconv}$ のほうが近い場合にはなにもしない．この処理を，ゲートウェイに到着した CAN メッセージ 1 つ 1 つに対して行い，ゲートウェイでの待機時間を調整する．

4.3　動作例

図 3 は優先度 4 の CAN メッセージがゲートウェイに到着したときの様子を示している．優先度 4 の CAN メッセージの絶対デッドラインから CAN 見積もり遅れ時間を引いた時刻が GW 内デッドラインとなる．ここで，次回変換予定時刻がこのメッセージの GW 内デッドラインよりも後の時刻であるので，次回変換予定時刻まで待つとデッドラインに間に合わない可能性がある．このような場合，次回変換予定時刻を前倒し，優先度 4 の CAN メッセージの GW 内デッドラインを新たに次回変換予定時刻として設定する．時刻が進み，変換予定時刻となったら，バッファに格納されている優先度 6 の CAN メッセージと優先度 4 の CAN メッセージを Ethernet パケットへ変換する．

4.4　CAN 見積もり遅れ時間の決定方法

本論文では CAN 見積もり遅れ時間として 2 種類の決定方法を用いている．1 つ目がメッセージの優先度に関係なくすべてのメッセージに対して一定値（例：5ms）を割り当てる方法，2 つ目がメッセージの送信周期の一定割合値（例：周期の 20%）を各メッセージに割り当てる方法である．一定値を用いる場合，すべての CAN メッセージに対して同じ時間だけ CAN ネットワークで遅れ得ると考えることになる．しかし，実際は優先度に応じて CAN メッセージの待ち時間は異なるため，これは暫定的な手法である．一方で周期の一定割合値を用いる場合は，メッセージの周期に応じて CAN 見積もり遅れ時間を割り当てることにな

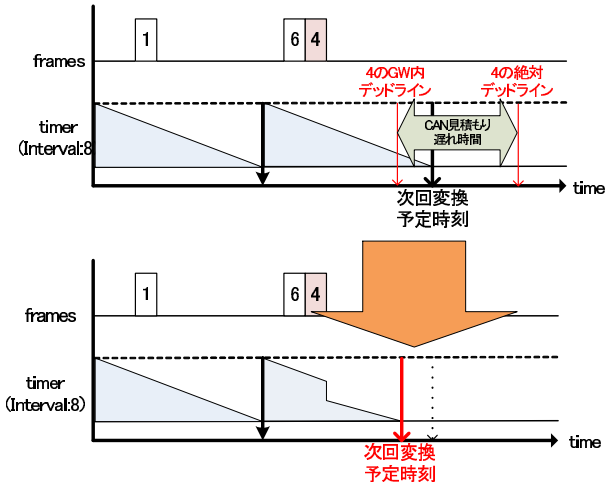

図 3　デッドライン型アルゴリズムの動作例
Fig. 3　Behavior of the deadline algorithm

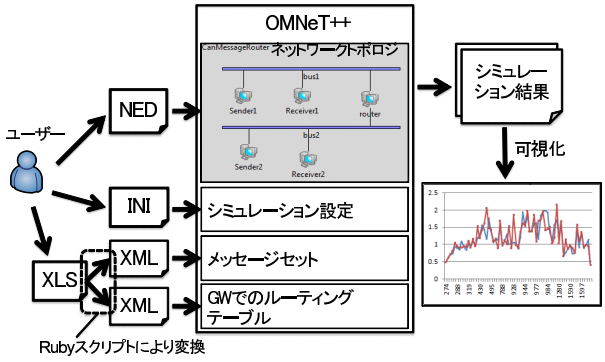

図 4　シミュレーション環境の全体像
Fig. 4　Overview of the simulation environment

る．そのため，優先度に応じて周期を設定したメッセージセット設計であれば，優先度と周期を考慮した手法といえる．優先度とは無関係に周期が設定されたメッセージセットに対しては，周期のみを考慮していることになる．

5. 評価実験

5.1 評価環境

評価環境として，離散イベントシミュレータ OMNeT++ を拡張したシミュレーション環境を開発した．これはネットワーク構成（メッセージの遅れ時間や負荷率など）を評価することに適したシミュレータである [4] [5]．図 4 に，拡張したシミュレーション環境の全体像を示す．OMNeT++ にはシミュレーションモデルとして Ethernet モデルが既に開発されていたが，CAN モデル及び CAN-Ethernet ゲートウェイモデルが開発されていなかったため，本研究で追加で開発した．さらに，各 ECU のモデルとしてアプリケーションモデルも追加で開発した．図 5 に，開発したモデルを含むシミュレーションモデルを示す．また，通信プロトコルのモデルだけでなく，メッセージセット等のデータである XLS から XML へ変換するツールや，シミュレーション結果を解析するツールなどを開発した．使用した OMNeT++のバージョンは 4.2.2 で，Ethernet モデルを開発した INETFramework[8] のバージョンは 1.99.4 である．

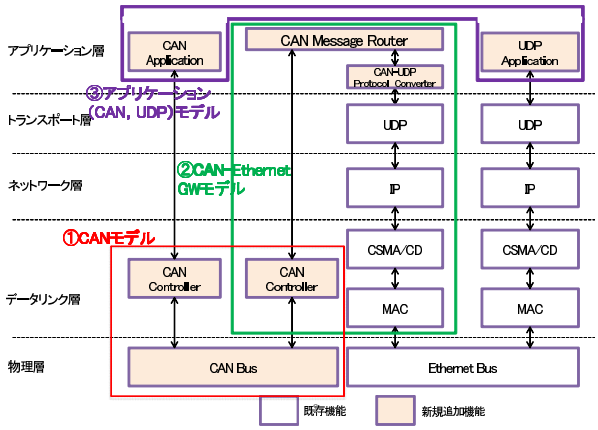

図 5　シミュレーションモデル
Fig. 5　Simulation model

開発した CAN モデルの正当性を確認するために，既存の CAN ネットワークシミュレータである Venet[9] との比較実験を行い，平均遅れ時間の比較誤差が高々 0.2%以内に収まることを確認した．0.2%の誤差は，平均値を求める際の丸め方の違いによるもので，CAN モデルの本質的な不具合はないことを確認した．

5.2 評価項目

実験では，以下の 3 つの評価項目について評価する．

- CAN メッセージごとの遅れ時間（最大，最小，平均，最大遅延率）
- CAN → Ethernet 変換回数
- バス負荷率

遅れ時間とは，CAN メッセージが生成されてから，目的地となる ECU に到着するまでにかかる時間である．最大遅延率とは，最大遅れ時間をメッセージの送信周期で割った値である．これは周期メッセージのみに適用できる評価項目であり，この値が 1 を超えるということは，メッセージ通信が周期に間に合っていないということを意味する．この値を計測することで，各メッセージがデッドライン制約を守っているかどうかを評価することができる．CAN → Ethernet 変換回数は，ゲートウェイにおける CAN メッセージから Ethernet パケットへの変換処理が行われた回数を示す．この回数が多いことは，Ethernet パケットが頻繁に送信されることを意味し，同じ Ethernet バスを共有する他の機能（マルチメディア系の通信など）を妨げる可能性が高くなる．ゲートウェイの CPU 負荷率をシミュレーションで計測することができないため，この評価項目を用いる．バス負荷率は単位時間内に伝送されたデータ量を伝送容量で割った値である．車載ネットワークでは，およそ 60%〜70%程度が目安とされている．

Vol.2013-SLDM-160 No.7
Vol.2013-EMB-28 No.7
2013/3/13

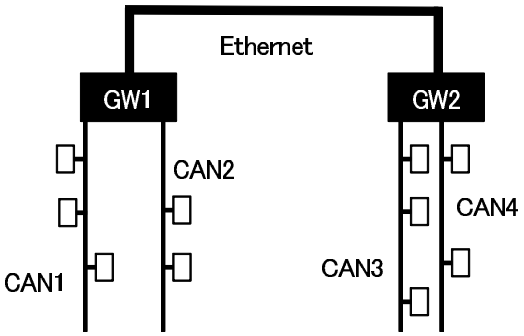

図 6　評価対象のネットワーク構成
Fig. 6　Network configuration for the evaluation

表 2　CAN-Ethernet 混在ネットワークの設定値
Table 2　Parameters of CAN-Ethernet network

CAN 通信速度	500kbps
Ethernet 通信速度	100Mbps
CAN ノードの数	90
Ethernet バスの数	1
CAN-EthernetGW の数	2
CAN バスの数	4
CAN メッセージ数	216（周期メッセージ） 194（イベントメッセージ）
ドリフト	± 1%
キューの種類	フル優先度

5.3　実験方法

図 6 に，評価対象となるネットワーク構成を示す．4 本の CAN バスと 1 本の Ethernet バスが CAN-Ethernet ゲートウェイで接続されたネットワークである．表 2 に，CAN-Ethernet 混在ネットワークを構成する要素の設定値を示す．

表 3 に示すパラメータのプロトコル変換アルゴリズムを CAN-Ethernet ゲートウェイに適用した．デッドライン型のパラメータは 2 通り設定した．具体的には，CAN 見積もり遅れ時間として一定値 5ms を使用する deadline(t=5ms) と，CAN メッセージの送信周期の 50%値を使用する deadline(r=0.5) である．シミュレーション時間は，ワンドライビングサイクルとして妥当な時間を考え，7200 秒とした．

5.4　結果

アルゴリズムごとの最大遅延率の比較結果を図 7 に示す．バッファ型，時間型では最大遅延率が 1 を超えるメッセージが存在することが確認できる．さらに，実験の平均バス負荷率，CAN → Ethernet 変換回数の比較結果を表 4 に示す．CAN → Ethernet 変換回数では，空パケットへの変換回数（empty）と，CAN メッセージを含む Ethernet パケットへの変換回数（not empty）を示す．CAN バスの平均バス負荷率はアルゴリズムごとに大きな変化はなく，Ethernet バス負荷率には若干の変化があることが確認できる．また，Ethernet パケットへの変換回数では，バッファ型の GW1 での変換回数が最少となっている．

表 3　実験のパラメータ
Table 3　Parameters of the experiment

	バッファサイズ	変換周期 [ms]	最大前倒し時間 [ms]	緊急メッセージ割合 [%]	CAN 見積もり遅れ時間
basic	1	-	-	-	-
buffer	40	20	-	-	-
time	40	20	20	-	-
urgency	40	20	20	20	-
deadline (t=5ms)	20	5	-	-	5ms
deadline (r=0.5)	20	5	-	-	周期の 50%

5.5　考察

図 7 から，最大遅延率の変化が著しい部分がいくつか存在することが確認できる．最大遅延率が突出している部分とその周辺のメッセージの送信周期を調査したところ，周辺のメッセージの送信周期に対して突出している部分のメッセージの送信周期が短くなっており，優先度に比例せずに送信周期を設定されているメッセージが存在することが明らかになった．つまり，同程度の優先度のメッセージの送信周期に比べて送信周期が短いメッセージの最大遅延率が比較的大きくなる傾向にある．これは，周期が短いにもかかわらず低い優先度が割り当てられているからであると考えられる．故に，優先度に応じて周期が設定されているメッセージセットであれば，送信周期の一定割合値を用いる提案手法で対応可能であるが，優先度とは関係なく周期が与えられているメッセージセットの場合，送信周期の一定割合値のみでは対応が困難である．優先度とは関係なく周期が与えられているメッセージセットの場合に，すべてのメッセージの最大遅延率を妥当な値に抑えるために，周期だけでなく優先度も考慮した CAN 見積もり遅れ時間の設定が必要であると考えられる．

6.　おわりに

本論文では，CAN-Ethernet プロトコル変換アルゴリズムの拡張について述べた．優先度に着目した既存アルゴリズムにおいて，デッドラインを考慮していないことにより生じる課題について指摘し，CAN メッセージのデッドライン制約を満たすことに着目したデッドライン型アルゴリズムを提案した．具体的には，CAN メッセージのデッドラインからゲートウェイでの待機時間を調整する手法である．さらに，提案手法の評価を行うため，OMNeT++をベースにシミュレーション環境を開発した．既存アルゴリズムとの比較評価実験を行った結果，デッドライン制約を守りつつ，従来手法に比べて Ethernet パケットへの変換回数を削減できることを確認した．

今後の課題としては，CAN 見積もり遅れ時間の決定方

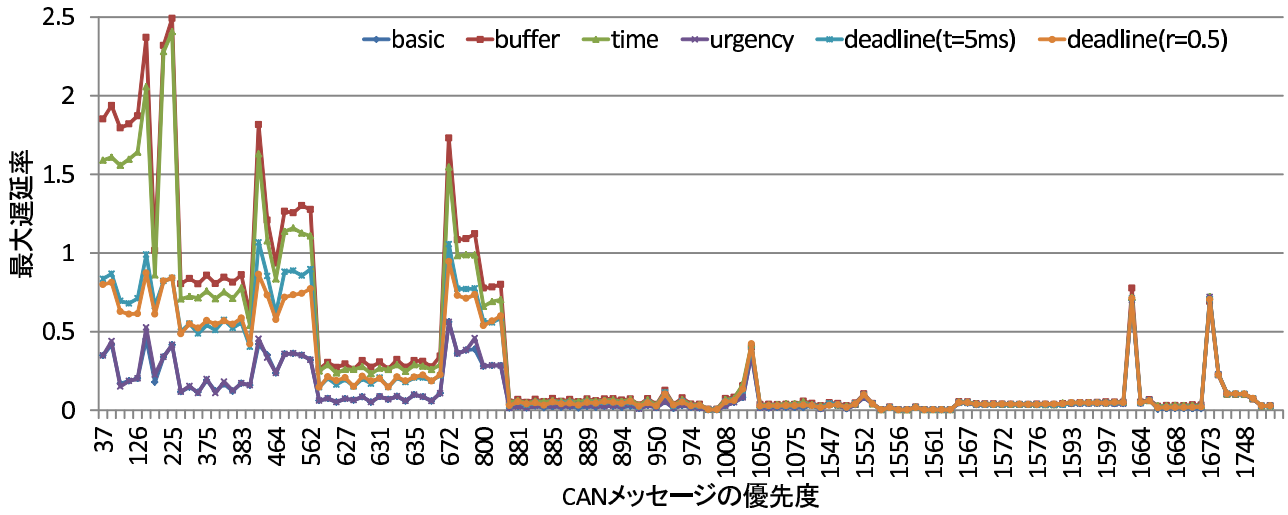

図 7　最大遅延率

Fig. 7　Maximum delay ratio

表 4　平均バス負荷率，CAN → Ethernet 変換回数

Table 4　Average workload of each bus and the number of conversions to Ethernet packets

	平均バス負荷率 [%]					CAN → Ethernet 変換回数			
						GW1		GW2	
	CAN1	CAN2	CAN3	CAN4	Ethernet	empty	not empty	empty	not empty
basic	32.78	69.61	45.31	50.81	2.14	1	1753806	1	11453219
buffer	32.78	69.61	45.31	50.81	0.34	1	359999	1	363375
time	32.78	69.61	45.31	50.81	0.35	1	366819	1	426005
urgency	32.78	69.61	45.31	50.81	2.01	1	1667816	1	10637389
deadline(t=5ms)	32.78	69.61	45.31	50.81	0.45	1	718992	1	824190
deadline(r=0.5)	32.78	69.61	45.31	50.81	0.48	1	718992	1	978810

法と Ethernet → CAN 方向のプロトコル変換アルゴリズムが挙げられる．本研究で用いた CAN 見積もり遅れ時間の決定方法では，メッセージの生成時刻を用いているため使用可能なデータ帯域は減少してしまう．そのため，GW を含む CAN ネットワークの最悪応答時間解析結果を用いた静的な CAN 見積もり遅れ時間の決定が必要であると考える．また本論文では，CAN → Ethernet 方向の変換のみに注目していた．今後は，Ethernete → CAN 方向の変換にも取り組んでいかなければならないと考える．課題としては，到着先の CAN サブネットワーク内のフロー制御問題が挙げられる．

謝辞　本研究の一部は，JSPS 科研費（24700027）の助成を受けたものです．

参考文献

[1] J. Sommer and R. Blind, “Optimized Resource Dimensioning in an embedded CAN-CAN Gateway,” in *Industrial Embedded Systems, 2007. SIES ’07. International Symposium on*, pp. 55 –62, july 2007.

[2] A. Kern, D. Reinhard, T. Streichert, and J. Teich, “Gateway Strategies for Embedding of Automotive CAN-Frames into Ethernet-Packets and Vice Versa,” in *Architecture of Computing Systems - ARCS 2011*, vol. 6566 of *Lecture Notes in Computer Science*, pp. 259–270, Springer Berlin / Heidelberg, 2011.

[3] ADFweb, “http://www.adfweb.com,”.

[4] A. Varga and D. History, “OMNeT++ Discrete Event Simulation System, version 3.1 edition,”, 2005.

[5] A. Varga and R. Hornig, “An overview of the OMNeT++ simulation environment,” in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, Simutools ’08, ICST, Brussels, Belgium, pp. 60:1–60:10 .

[6] X. Chang, “Network simulations with OPNET,” in *Simulation Conference Proceedings, 1999 Winter*, vol. 1, pp. 307 –314 vol.1, 1999.

[7] D. Henriksson, A. Cervin, and K.-E. Arzen, “TrueTime: Real-time Control System Simulation with MATLAB/Simulink,” in *Proceedings of the Nordic MATLAB Conference*, 2003.

[8] K. Wehrle, J. Reber, D. Holzhausen, V. Boehm, V. Kahmann, and U. Kaage, *INET Framework for OMNeT++*.

[9] Venet, “http://www.kumikomi.net/archives/2004/11/31et04/interd/vs.pdf,”.