Vol.2014-ITS-58 No.7
2014/9/19

車々間通信を用いた安全運転支援のための リアルタイムストリーム処理

山口 晃広[1]　佐藤 健哉[1,2]　中本 幸一[1,3]　渡辺 陽介[4]　高田 広章[1]

概要：近年，自動車には複数のセンサが搭載され，自動運転や衝突回避などが研究開発されている．このようなセンサ情報処理では，最大遅延時間が許容時間を超えないリアルタイム性が要求される．しかし，周辺車両が車両情報をブロードキャストする車々間通信では，周辺車両の増加により，車両がリアルタイム性を満たして車々間通信からの入力を常に全て処理することは難しい．この問題に対応するため本稿では，動的なリアルタイムスケジューリングアルゴリズムである Earliest Deadline First を適用し，デッドラインの早いデータを優先して処理するリアルタイムストリーム処理を提案する．これにより，データ量が増加しても優先度の高い情報を待たさずに処理できる．車々間通信からの車両情報が増加する見通しの悪い交差点で出会い頭衝突警告をシミュレーションにより評価し，提案手法がリアルタイムスケジューリングを行わない従来のストリーム処理と比較して，リアルタイム性を満たして多くの車両情報を処理することで，出会い頭衝突事故を削減できることを確認した．

1. はじめに

自動車には，車速センサ，レーダ，レーザ，カメラなど，車両の状態や周辺状況を監視するセンサを車両に搭載した安全運転支援システムが普及し始めている [1]．加えて，Google に代表されるように複数のセンサを車両に搭載し，これらのセンサ情報を融合し高度な制御を行うことで，ドライバが操作することなく市街地を自動走行する研究も行われている [2]．

このような自動走行も含めた安全運転支援は，センサにより車両の周辺環境を認識し，その情報をもとに車両の制御やドライバへの警告を行う．これらのセンサ情報処理には，平均的に低遅延なだけでなく，センサからデータが発生してからその処理が完了するまでの最大遅延時間（許容最大遅延時間）があらかじめ定められた一定時間を超えないリアルタイム性が求められる．このセンサ情報処理のリアルタイム性が満たせないと，車両の前方に危険な事象を発見してから車両を停止するなどの遅延時間を保証できず，車両が衝突するなどの事故が発生しやすい．

一方で，前述のように車両に搭載されるセンサ（搭載センサ）のみを用いた安全運転支援では，見通しの悪い交差点からの飛び出しなど，走行中に搭載センサで監視できない状況への対応が難しい．このような問題に対して，搭載センサに加えて車々間通信や路車間通信を利用して相互に情報を交換することで，より安全な走行を目指す協調 ITS の活用が進められている [3]．協調 ITS の仕様化を進めている欧州標準化機構（ETSI）では，協調 ITS を活用した交差点衝突警告（Intersection Collision Risk Warning, ICRW）や前方衝突警告などの仕様の策定を進めている [4], [5]．

しかし，車々間通信では，周辺車両が各々のタイミングにより 100 ミリ秒周期でブロードキャストする可能性があるため [6]，車々間通信から受信する情報の量や到着タイミングは走行状況により変わる．特に，走行する車両で混雑する大規模な交差点などでは一度に大量の情報が送られてくるため，一時的に処理が間に合わずリアルタイム性を満たせない可能性もある．協調 ITS では周辺車両から情報を受信して処理を行うが，これらの処理には車両一台につき数十ミリ秒の計算時間がかかる場合があることに対して [7], [8]，ETSI の仕様 [4] では 1 秒間に 1000 台分の車両情報を受信できることを要求している．そのため，状況によっては車々間通信から受信する情報を全て処理することはできない．このような状況では，データの処理順序を適切に決定し，デッドラインをミスしないように重要なデータを処理することが求められる．

[1] 名古屋大学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
Center for Embedded Computing Systems, Nagoya University
[2] 同志社大学モビリティ研究センター
Mobility Research Center, Doshisha University
[3] 兵庫県立大学大学院応用情報科学研究科
Graduate School of Applied Informatics, University of Hyogo
[4] 名古屋大学 未来社会創造機構
Institute of Innovation for Future Society, Nagoya University

本稿では，このような状況においても，リアルタイム性を満たしてセンサ情報を効率的に処理する方法を述べる．ストリーム処理は，データ管理にストリームキューを用いるので車々間通信のようにデータ量の増加に対応しやすく，センサ情報のように連続的なデータを低遅延に処理することにも適しているため [9]，本手法にはストリーム処理を用いる．本手法では，デッドラインを満たすように処理順序を決定するリアルタイムスケジューリング [10] をストリーム処理に導入し，リアルタイムスケジューリングを行うストリーム処理（リアルタイムストリーム処理）を安全運転支援に適用可能な方式として提案する．これにより，デッドラインの早い情報を優先して処理し，データ量が増加した場合でも優先度の高い情報を待たさずに処理できる．

本稿の構成は以下のとおりである．まず，2 章で本稿が解決する課題を述べ，3 章でそれを解決するリアルタイムストリーム処理を提案する．そして，4 章でシミュレーションによる提案手法の評価を行う．最後に，5 章で関連研究を紹介し，6 章でまとめを述べる．

2. 車々間通信を活用する安全運転支援の課題

2.1 ストリーム処理を用いた安全運転支援のデータ処理

安全運転支援を行う車両では，複数のセンサや車々間通信から得られた情報を複数のアプリケーションが利用する場合が多い．我々は，このデータ処理をアプリケーションプログラムとセンサから分離してストリーム処理で統合管理することで，低遅延な処理を実現しながらソフトウェアの開発効率を向上するプラットフォームを研究開発している [11], [12]．ストリーム処理システムに登録されたデータ処理は，オペレータと呼ばれる処理ブロックをストリームで繋いだデータフローとして表現される．そのため，センサやアプリケーションの追加/変更に対して，処理ブロックを追加/置換することにより，比較的容易に対応できる [13], [14].

ストリームからのデータの入出力の仕組みはキューとして実現されるため，車々間通信のようにデータ量が状況により変動しても柔軟に対応できる．オペレータは，入力として繋がれたストリームからデータを取り出して処理し，出力として繋がれたストリームへデータを流す．オペレータの種類には，ユーザがパラメータのみを指定する基本的な演算（結合, 射影, 合流など）と，ユーザがプログラミング言語で処理を記述するものがある．センサや車々間通信など，ストリーム処理システムの外部からデータを入力するストリームを入力ストリームと呼ぶ．パネルの表示やアクチュエータの制御などを行うアプリケーションプログラムにデータを配信するストリームを出力ストリームと呼ぶ．

2.2 車々間通信データのリアルタイムスケジューリング

提案手法が解決しようとしている問題を**図 1**，**図 2**，**図 3** を用いて説明する．図 1 は，見通しの悪い交差点における車々間通信を用いた ICRW を表しており，以下のケースを想定する．自車両 A は，ICRW により車両 X を早期に発見してブレーキをかけなければ車両 X と衝突してしまうが，見通しが悪いためにレーダやカメラなどの搭載センサでは車両 X を早期に検知できない．一方，車両 E は，それに搭載されたセンサにより，車両 X を早期に検知できる．車両 B,D,E は，自車両 A と車々間通信ができるが，車両 X,C,F とは車々間通信できない．ETSI では，衝突警告に対する許容最大遅延時間を 300 ミリ秒と提示しており [4]，ここでは，車両 E が車両 X を検知してから 300 ミリ秒以内に，自車両 A が車両 X の情報を受信し ICRW を完了できなければ，自車両 A と車両 X は衝突する状況を想定する．なお，図中の Y_Z は，車両 Y をセンシングした結果を車両 Z が送信するメッセージを表す．タイムスタンプはセンサからデータが発生した時刻として付与される．説明を簡単にするために，各車両は同時にセンシングを行い，車両 E はその 95 ミリ秒後にそのデータを送信し，他の車両はセンシングの 5 ミリ秒後にそのデータを送信した

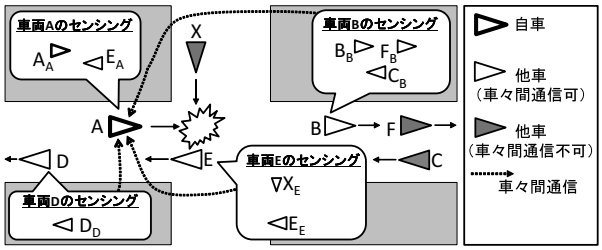

図 1 本提案で用いるユースケースシナリオ
Fig. 1 Usecase scenario used in the proposal.

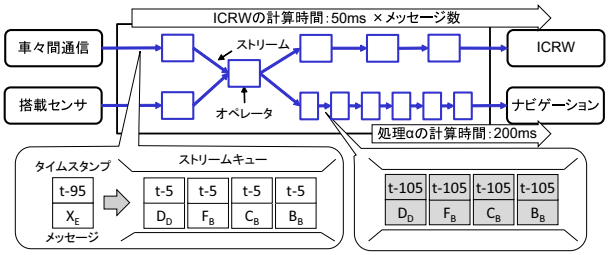

図 2 ユースケースシナリオのストリーム処理
Fig. 2 Stream processing for the usecase scenario.

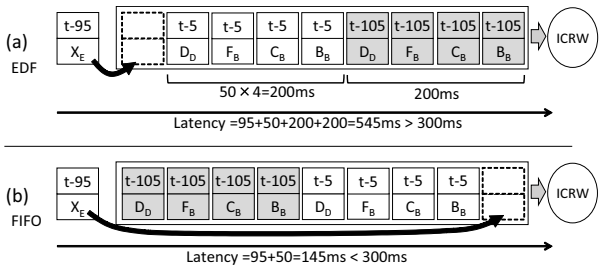

図 3 車々間通信から得られるストリームデータの処理順序
Fig. 3 Processing order of the data stream from V2V comm.

場合を考える．

図 2 は，ICRW とナビゲーションを行うストリーム処理の概略を表す．衝突警告とナビゲーションに対する許容最大遅延時間を，それぞれ 300 ミリ秒と 3 秒とする．現時刻 t において，ナビゲーションのみに用いられる処理の一部（処理 α）が残っており，車々間通信から得られたメッセージが B_B, C_B, F_B, D_D, X_E の順で到着した状況を考える．処理 α を完了するのにかかる残り時間を 200 ミリ秒として，ICRW の処理に対して 1 メッセージあたりの処理時間を 50 ミリ秒とする．図 3(a) の従来のストリーム処理のように，データを FIFO（処理 α → B_B → C_B → F_B → D_D → X_E）の順で処理すると，X_E の遅延時間は $95 + 200 + 50 \times 5 = 545$ ミリ秒かかるため，X_E はデッドラインをミスしてしまう．その結果，自車両 A は，ICRW が間に合わないことで車両 X と衝突してしまう．処理 α または車々間通信で先に到着したメッセージを先に処理しても，同様にデッドラインをミスし車両 X と衝突する．

リアルタイムスケジューリングのアルゴリズムとして良く知られている Earliest Deadline First（EDF）は，デッドラインの早い順に処理順序を決定する [10]．ICRW はナビゲーションよりも許容最大遅延時間が短く，X_E は先に入力ストリームに到着したメッセージ B_B,C_B,F_B,D_D よりもタイムスタンプが古い．そのため，EDF に基づきデータを処理すると図 3(b) のように X_E → B_B → C_B → F_B → D_D →処理 α の順となり，全てのデータをデッドラインを満たして処理することが可能になる．これにより，300 ミリ秒以内に X_E の処理を完了し，車両 X との衝突を回避できる．このように，安全運転支援を対象としたストリーム処理では，リアルタイムスケジューリングが必要となる．

ETSI は，1 秒間に最大 1000 メッセージを受信できるように要求している [4]．車々間通信のメッセージとして用いられる CAM の場合，1 メッセージに車両の 1 台分の情報が含まれるため [6]，1 秒間に最大 1000 台の車両情報が受信される想定となる．しかし，この台数をリアルタイム性を満たして全て処理することは難しい．そのため，リアルタイム性を満たすように，車々間通信からの入力データを効率的にフィルタリングすることが必要となる．

3. リアルタイムストリーム処理

3.1 アーキテクチャ

提案手法によるストリーム処理システムのアーキテクチャを**図 4** に示す．2.2 節の例のように，車々間通信から入力ストリームに入力されるストリームデータでは，到着タイミングやデッドラインが実行時に変動するため，リアルタイムスケジューリングアルゴリズムとして，それらに対応できる EDF を用いる．ユーザは，クエリの各出力ストリーム s に対して，許容最大遅延時間 l_s を指定する．各出力ストリームに到着するストリームデータのデッドライン d_s は，センサからそのデータが発生した時刻 ts を用いて，式 (1) で計算される．

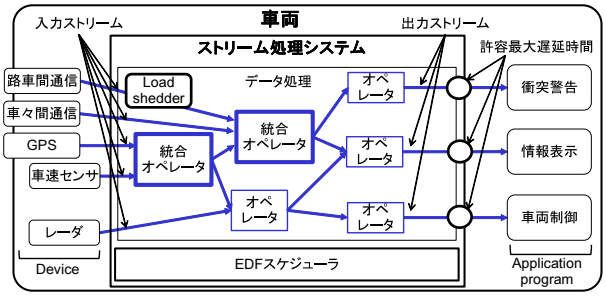

図 4 提案手法によるシステムのアーキテクチャ
Fig. 4 Architecture of proposed system.

$$d_s = l_s + ts \tag{1}$$

車々間/路車間などとの通信は一時的に切断する可能性がある．これらを入力に用いる処理は，搭載センサのみを用いる処理と多重化/冗長化して，データ処理を設計する．これにより，通信を用いるデータ処理がデッドラインを満たせない場合，搭載センサのみを用いた最低限の処理でリアルタイム性を確保する．多重化/冗長化したデータ処理の統合には，3.2 節で説明する統合オペレータを用いる．

車々間通信のようにフィルタリングが必要な入力ストリームに対して，ユーザは load shedder を設定する．Load shedder は，ユーザが登録したフィルタリングの条件に基いて，設定された入力ストリームから，デッドラインを満たす分だけ価値の低いデータを削除する．

3.2 統合オペレータ

統合オペレータは，多重化/冗長化した複数のストリームデータを 1 つに統合するためのオペレータである．これは，ある入力にストリームデータが到着した後，一定時間待っても他の入力にストリームデータが到着しない場合，到着した入力データのみを用いて処理を実行する．このタイムアウトの機構により，通信が利用できない場合でも，搭載センサのみを用いた最低限の処理でリアルタイム性を満たす．なお，タイムアウトする時間は，データ処理を設計する時にユーザが指定する．

統合オペレータの処理は，ユーザがプログラミング言語を用いて記述できる．統合オペレータの処理で，複数の入力に対してセンサフュージョンを行うことで，統合された結果出力されるストリームデータの信頼性を高められる．

3.3 リアルタイムスケジューリングのパラメータ

EDF スケジューリングを行うには，スケジューリングにおける並行実行の単位であるタスクを定義し，各タスクに対して，生成（リリース）されるタイミング，デッドライン，計算時間のパラメータを設定する必要がある．タスク

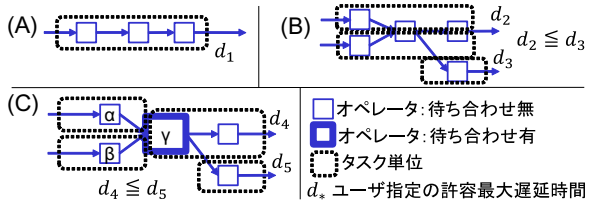

図 5 タスク単位のパターン
Fig. 5 Patterns of task units.

は，処理順序が設計時に確定できる一連のオペレータが単位となる．そのリリースのタイミングは，タスクの入力ストリームにデータが到着した時か，または，タイムアウトによってタスクの一部である統合オペレータが実行可能になる時である．タスクの計算時間は，タスクを構成するオペレータの計算時間の合計として見積もられる．

図 5 では，様々なデータ処理のパターンに対するタスクの単位を表している．基本的には，(A) のように，ある入力ストリームからある出力ストリームまでの一連のオペレータのパス (オペレータパス) をタスクの単位とする．(B) は，オペレータの出力を複数に分岐する場合や，複数の入力を待ち合わせることなく単純に合流させる場合を表す．(C) は，通信により到着時刻に揺らぎのある入力データがあり，複数の入力を待ち合わせてからオペレータを実行する場合である．この場合は，一連のオペレータを α,β,γ と処理するか，β,α,γ と処理するか，入力データの到着タイミングに応じて動的に変更する必要があるため，タスクの単位を分割する．統合オペレータも，このタイプ (C) の待ち合わせをするオペレータに分類される．

(C) のように，先行の複数のタスクと後続のタスクの間に順序依存があり，先行のタスクのデッドラインが分からない場合に対応するため，提案手法ではリアルタイムスケジューリングの既存手法である EDF*[15] を用いる．EDF*では，順序依存を持つタスク間で，後続のタスク $\tau_j,(j=1,\cdots,n)$ のデッドライン d_j と計算時間 c_j から，先行のタスク τ のデッドライン d' を式 (2) で導出できる．なお，式 (2) の CaseA は，ストリーム処理の出力ストリーム s に配信するオペレータがタスクの単位に含まれる場合であり，d_s は式 (1) により計算される．

$$d' = \begin{cases} \min\{d_s, d_j - c_j; j=1,\cdots,n\} & \text{(CaseA)} \\ \min\{d_j - c_j; j=1,\cdots,n\} & \text{(その他)} \end{cases} \quad (2)$$

後続のタスクから先行のタスクへ式 (2) を再帰的に用いてデッドラインを決めていくことで，順序依存を持つ全てのタスクのデッドラインを適切に決めることができる．EDF* を適用することで，順序依存の無い通常の EDF に修正できることが証明されている [10], [15]．以上の方法により，安全運転支援のデータ処理で必要となる広範のストリーム処理にリアルタイムスケジューリングを適用できる．

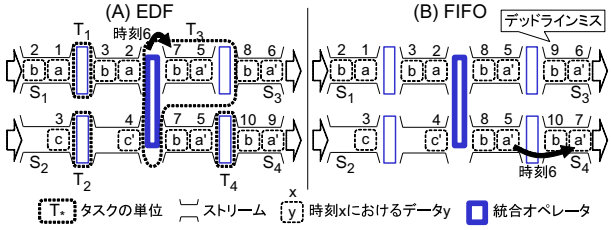

図 6 EDF/FIFO に基づくデータの処理順序
Fig. 6 Processing order of data based on EDF/FIFO.

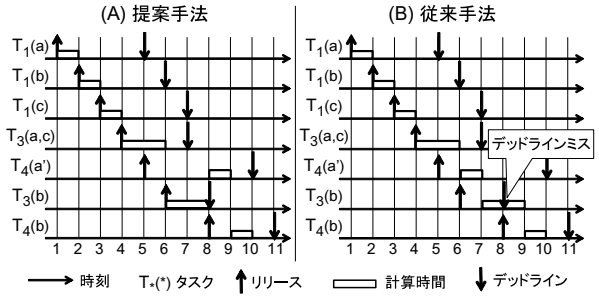

図 7 タスクのスケジューリングとしての比較
Fig. 7 Comparison as task scheduling.

3.4 ストリームデータのスケジューリング

センサや通信機器などから，ストリーム処理の入力ストリームにデータが到着した時に，それに対応するタスクをリリースする．タスク間に順序依存がある場合は，後続のタスクの入力ストリームにデータが到着した時に，そのタスクの全ての入力が揃っていれば，そのタスクをリリースする．後続のタスクがタイムアウトの機構を持っている場合は，そのタスクの入力ストリームに待ち合わせを行う最初のデータが到着した時にタイマーを起動し，そのタスクをリリースする時にタイマーを解除する．タイムアウトの機構を持つタスクは，タイマーがタイムアウトした場合には，全ての入力が揃っていなくてもリリースされる．式 (1) に用いる ts は，そのタスクのリリースに関与する入力データの ts を用いる．但し，複数の入力データが関与する場合は，入力データの中で最古（最小）の ts を用いる．リリースされたタスクは，EDF に基づきデッドラインの早い順に処理する．

図 5(C) を例に用いて，ストリームデータの処理の流れを説明する．ここでは，(C) のオペレータ γ は，時間 4 でタイムアウトする統合オペレータであり，出力ストリームに指定された許容最大遅延時間 d_4,d_5 は，それぞれ 6,9 とする．**図 6** のように，データ a,b が時刻 1,2 に入力ストリーム S_1 に到着し，データ c が時刻 3 に入力ストリーム S_2 に到着する場合を考える．各オペレータの計算時間は単純に全て 1 とし，データ上の数字はそこにデータが到着した時刻を表している．$T_j(*)$ の計算時間を c_j とする．

図 6(A) は，EDF スケジューリングを適用した場合である．時刻 1 で入力ストリーム S_1 に a が到着すると，タスク

$T_1(a)$ がリリースされ実行される．出力ストリーム S_3 と S_4 のデッドラインが式 (1) により $6+1=7$ と $9+1=10$ となることから，$T_1(a)$ のデッドラインは式 (2) を用いて $\min\{7-c_3, 10-c_3\} = \min\{7-2, 10-2\} = 5$ となる．時刻 2 で統合オペレータの入力に a が到着したことで，a に対するタイムアウトの時刻が $2+4$ と設定される．また，その時刻に入力ストリーム S_1 に b が到着するため，タスク $T_1(b)$ がリリースされ実行される．時刻 4 では，統合オペレータの 2 つの入力にデータが到着したため，a をタイムアウトする前に $T_3(a,c)$ がリリースされ実行され，入力 a と c による統合オペレータの出力結果 a' を時刻 5 に出力する．このように複数のストリームデータを結合する場合，出力データの ts には，結合するストリームデータの中で最古（最小）の ts を指定する．時刻 5 では，タスク $T_4(a')$ がリリースされるが，デッドラインが $T_3(a,c)$ より遅いため，$T_4(a')$ は未だ実行されない．時刻 6 では，統合オペレータで b に対してタイムアウトが発生し $T_3(b)$ がリリースされる．$T_4(a')$,$T_3(b)$ のデッドラインは 9+1,6+2 であるため，後からリリースされた $T_3(b)$ が，EDF に基いて先に実行される．

このようにスケジューリングすることで，図 6(A) では全てのストリームデータがデッドラインをミスせずに処理される．一方，リアルタイム性を考慮しない従来の FIFO スケジューリングでは，図 6(B) のように，時刻 6 で先に入力されたデッドラインの遅い a' を先に処理してしまう．その結果，デッドラインの早い b の処理が間に合わず，b は時刻 9 で出力ストリーム S_3 に到着した時にデッドラインをミスする．

データストリームのスケジューリングは，**図 7** のようにリアルタイムシステムにおける通常のタスクのスケジューリングとみなせる．提案手法では，EDF に基づいて図 7(A) のようにデッドラインの早いタスクを先に処理することで，デッドラインミスを削減する．一方，FIFO に基づきデータを処理する従来手法は，図 7(B) のように単純にリリースされた順にタスクを処理することに対応しており，提案手法よりデッドラインをミスしやすい．

以上の方法を直接実現すると，データがタスクの入力ストリームに到着する度に，タスクがリリースされスケジューリングされる．これにより，車々間通信のように入力データ量が増大すると，スケジューリングのオーバーヘッドがかかり効率的ではない．そのため，本提案でも従来手法 [16], [17] のように複数のデータをバッチして 1 つのデータとしてスケジューリングする．車々間通信やレーダから得られる周辺車両の情報のように，走行状況により動的にデータ量が変動する入力データに対しては，一定周期[*1]でデータをバッチしてスケジューリングする．バッチされたデータのデッドラインは，式 (1) の ts に対して，バッチ内の各データの中で最古の ts を用いる．

[*1] 本実験では車々間通信は 100 ミリ秒, レーダは 10 ミリ秒とした．

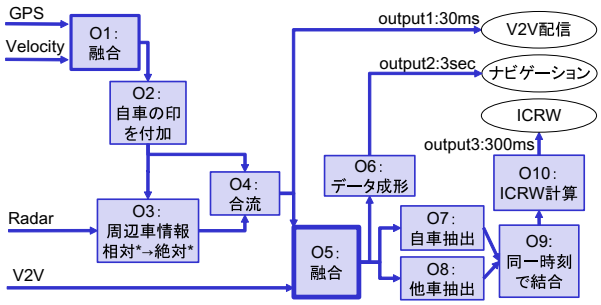

図 8 評価で用いるストリーム処理
Fig. 8 Stream processing for the evaluation.

表 1 評価で用いるタスクの単位
Table 1 Task units for the evaluation.

タスクの単位	構成するオペレータ	デッドライン (msec)
T_1	O_1, O_2, O_4	$d_1 = \min(30+ts, d_2 - c_2)$
T_2	O_3, O_4	$d_2 = 30 + ts$
T_3	O_5	$d_3 = \min(d_4 - c_4, d_5 - c_5)$
T_4	O_7	$d_4 = d_6 - c_6$
T_5	O_8	$d_5 = d_6 - c_6$
T_6	O_9, O_{10}	$d_6 = 300 + ts$
T_7	O_6	$d_7 = 3000 + ts$

4. シミュレーションによる評価

4.1 評価に用いるストリーム処理

本評価では，車々間通信を活用する安全運転支援として，2.2 節で述べた ICRW を対象とし，それを実現するストリーム処理として**図 8** を用いる．なお，O_1〜O_{10} はストリーム処理のオペレータである．図 8 のタスクの単位は**表 1** のようになる．$d_j, c_j (j = 1, \cdots, 7)$ は，T_j がタスクとしてリリースした時のデッドラインと計算時間を表す．

出力ストリームとしては，自車両でセンシングした情報を車々間通信可能な周辺車両にブロードキャストする output1 と，ナビゲーションやログ出力などのために自車と周辺車両の情報を配信する output2 と，自車と周辺車両との ICRW を通知する output3 がある．各出力ストリームに指定する許容最大遅延時間については，output3 は ETSI の提示に合わせて 300 ミリ秒とした [4]．output1 は周辺車両の ICRW の入力に利用されるため最も短い 30 ミリ秒を設定した．output2 にはリアルタイム性が求められないアプリケーションとして十分に長い 3 秒を設定した．

搭載センサから得られる入力は，GPS と車速センサから得られる自車の位置と速度，レーダから得られる周辺車両の相対位置/相対速度である．車々間通信から得られる入力は，自車両が output1 からブロードキャストした情報に相当し，自車と周辺車両の位置や速度が含まれる．

O_1 は，統合オペレータであり，自車両における位置と速度を用いたカルマンフィルタが実装されている．O_2 は，O_1 から出力された自車両の位置/速度からなるストリーム

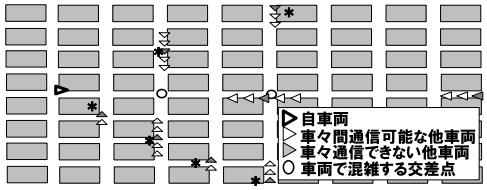

図 9 車両の初期位置
Fig. 9 Initial positions of vehicles.

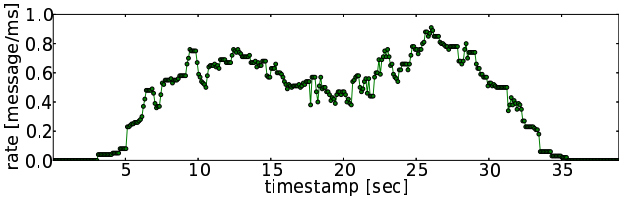

図 10 車々間通信からの入力レートの推移
Fig. 10 Input rate from V2V communications.

データに，自車の搭載センサで生成された印を付けた結果を O_3 と O_4 に配信する．O_3 は，レーダから得られた周辺車両の相対位置/相対速度を自車の位置/速度と加算し，絶対位置/絶対速度に変換する．O_4 は，O_2 と O_3 の出力結果を待ち合わせることなく合流し，output1 として車々間通信でブロードキャストする．この合流結果は，O_5 の入力にも用いられる．O_5 は，統合オペレータであり，搭載センサと車々間通信から得られる車両の位置情報をセンサフュージョンで融合するよう実装されている．O_5 の出力結果は，O_6 で表示用に成形されて ouput2 に出力される．O_7 と O_8 も，O_5 の出力結果を入力し，自車と周辺車のストリームデータをそれぞれ抽出する．O_9 は，同一のタイムスタンプを持つ自車と周辺車の情報を結合する．O_{10} は，O_9 の結合結果から，自車と交差する周辺車両に対して Time To Collision （TTC）を計算し，その結果を ICRW として output3 に出力する．

4.2 評価方法

本評価では，車々間通信から入力する車両情報が，ETSI の要求する 1 秒あたり最大 1000 台受信される状況に近づけて，2.2 節で述べたシナリオを評価する．そのため，多数の車両の車々間通信を模擬することが可能であり，電波の強度や特性，車両のモビリティなどを模擬することも可能である，ネットワークシミュレータ Scenargie を用いた[*2]．

各車両は，碁盤目の道路（マンハッタンモデル）に**図 9** のように初期配置し，矢印の方向に時速 60km で直進する．交差点間の距離は 100m である．**図 10** は，車々間通信からの入力レート（1 ミリ秒あたりのメッセージ数）の推移を表しており，図 9 の円でマークした 2 つの交差点付近で，車々間通信の入力レートが増加する．車々間通信やレーダ

[*2] Scenargie: https://www.spacetime-eng.com/

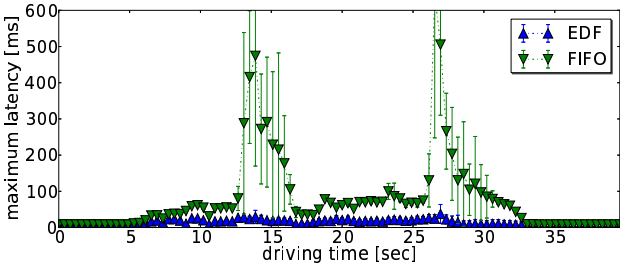

図 11 走行時間における最大遅延時間の変化
Fig. 11 Changes in maximum latency with driving time.

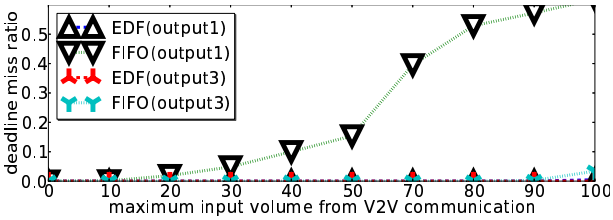

図 12 デッドラインミス率
Fig. 12 Deadline miss ratio.

は，100 ミリ秒周期で 200m 程度の範囲まで到達するように，電波強度を設定した．車々間通信は 700MHz とし，障害物があってもある程度屈折して電波が届く．レーザやレーダは 70GHz 以上の高周波を設定し，障害物で殆ど屈折しない．評価マシンには，CPU:800MHz，メモリ:1024MB，OS:Linux (Fedora10) を用いた．

4.3 評価結果

4.3.1 リアルタイム性

提案手法である EDF と load shedder のリアルタイム性への効果を確認する．**図 11** は，EDF と FIFO スケジューラを用いた場合で，許容最大遅延時間が最も短い output1 の最大遅延時間の推移を表しており，エラーバーは同一の走行シナリオで 100 回試行した場合の標準偏差を表す．遅延時間は，センサからデータが発生してから，output1 にそのストリームデータが到着するまでの時間として測定した．EDF により，output1 の最大遅延時間を平均 65%削減できた．これは，EDF ではデッドラインの早いストリームデータを待たせずに優先して処理したためである．

図 12 は，output1 と output3 のそれぞれにおいて，出力データの中でデッドラインをミスしたデータの割合（デッドラインミス率）を表している．横軸は，load shedder が車々間通信から 100 ミリ秒あたり最大通過させるデータ量（メッセージ数）である．図 12 のように，load shedder による入力データのフィルタリングや，リアルタイムスケジューリングにより，デッドラインミスを削減できる．

図 12 により，EDF によるリアルタイムスケジューリングの効果は特に大きい．EDF を適用した場合には，リアルタイム性を満たして 100 ミリ秒あたり最大 80 個の車々間通信からの入力データを処理できるが，適用しない (FIFO)

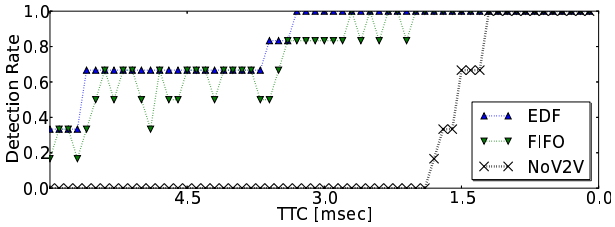

図 13 TTC に対する検出率の変化
Fig. 13 Changes in detection rate with TTC.

表 2 交差点での車両衝突による事故率
Table 2 Vehicle crash rate at intersections.

EDF	FIFO	NoV2V
0%	8.3%	100%

場合は，最大 10 個しか処理できない．これにより，提案手法では，リアルタイム性を満たしながら従来より多くの車々間通信からの入力データの処理が可能となった．これは，EDF により output1 と output3 の最大遅延時間が削減されたためである．以降では，EDF と FIFO で 100 ミリ秒あたりの最大処理可能な車々間通信の入力データ数をそれぞれ 80 個と 10 個とする．

4.3.2 ICRW への影響

車々間通信を活用した ICRW の提案手法における有効性を確認するため，output3 から出力される周辺車両との Time To Collision（TTC）から，本走行シナリオにおける出会い頭衝突への影響を確認する．自車と出会い頭衝突する交差点侵入車両は，図 9 の＊印の付いた車々間通信できない 6 台の車両（車両＊）である[*3]．ここでは，時速 60km で乾いた路面を走行する場合の停止距離から [18]，自車両の停止時間を 2.8 秒と算出し，最初に車両＊を検出した時点で TTC が 2.8 秒以内なら衝突すると仮定する．ここで，停止しない場合に衝突する車両の台数 N のうち，衝突を回避できなかった台数 M の割合 M/N として，出会い頭衝突の事故率を定義する．本実験では，同一の走行シナリオで 100 回試行（$t = 1, \cdots, 100$）した場合で事故率を測定した．つまり，$N = 6 \times 100$ であり，試行 t で衝突する車両＊の台数を M_t とすると $M = \sum_{t=1,\cdots,100} M_t$ である．

EDF を用いた場合 (EDF), 従来手法の場合 (FIFO), EDF を用いるが車々間通信を利用しない場合 (NoV2V) における測定結果を**図 13** と**表 2** に示す．図 13 は，ある試行において TTC を変化させた時，6 台の車両＊の内検出できた台数の割合（検出率）を表しており，表 2 は，出会い頭衝突の事故率をまとめている．まず，車々間通信を利用しない場合（NoV2V），事故率は 100%であり衝突を回避できなかったが，全ての車両＊を衝突前には検知できた．これは，車々間通信を利用できない場合でも，統合オペレータの冗長化/多重化により搭載センサのみを用いたストリーム処理がリアルタイム性を満たして動作しているためである．次に，EDF を用いることで，特定のシミュレーション環境において，従来手法の事故率が 8.3%であるのに対して，0%まで削減できた．これは，EDF を用いてより多くの車々間通信からの車両情報を処理することで，車両＊を早期に検出し，TTC を大きくしたためである．これにより，車々間通信から受信する車両情報が増加する場合に，提案手法が見通しの悪い交差点における出会い頭衝突事故を削減できることを示した．

[*3] 全ての車両＊は車々間通信できないため，車両＊を早期に検出するには，車々間通信可能な他の周辺車両が検出した車両＊の情報を受信する必要がある．

5. 関連研究

連続的なデータを低遅延に管理する技術として，ストリーム処理が研究されており，センサネットワーク，金融，インターネットなどの分野で主に用いられてきたが [9], [19]，車両に搭載してセンサ情報を扱うストリーム処理システムも幾つか提案されている．[20] では，車載システムを診断や検査するためのストリーム処理が提案された．StreamCars では，ストリーム処理を用いた安全運転支援のデータ管理を提案している [21]．しかし，これらの研究では安全運転支援の重要な要件であるリアルタイム性については述べられていない．また，搭載センサを主な対象としており，車々間通信の課題については検討されていない．

ストリーム処理のスケジューリングや load shedder は，主要なトピックとして研究されてきたが [9], [19]，リアルタイム性に着目したストリーム処理としての研究は未だあまり行われていない．まず，リアルタイムスケジューリングのアルゴリズムとして良く知られている rate monotonic に基づく方法が提案された [22]．しかし，データレートや到着時刻の変動が大きい入力を扱うことができないため，車々間通信の性質に合わない．この性質に合うストリーム処理のリアルタイムスケジューリングとして， EDF に基づく方法が幾つか提案されてきた [16], [17], [23]．しかし，これらの方法には，適用できるストリーム処理に制限があり，安全運転支援で用いられるデータ処理への適用は難しい．[16], [23] では，データ処理を表す連結したデータフローを 1 つのタスクと暗にみなすために，図 5(B) のように分岐した出力ストリームを持つ場合に対応できない．一方，[17] では，提案手法と同様に，オペレータパスに基づいてタスクを定義することで，分岐した出力ストリームを持つ場合にも対応できる．しかし，図 5(D) のように，デッドラインを式 (1) から直接計算できないタスクや，タスク間に順序依存がある場合に，対応できない．また，いずれの従来手法も，車外との通信を入力する場合に必要となるタイムアウトの機構に対応しない．提案手法では，3.3 節で述べたようにタスクの定義を拡張することで，従来手法では扱えなかった広範なストリーム処理を扱うことができる．また，[16], [17] に基づき，リアルタイム性を満たす

QoS 管理や load shedder も研究されているが [24], [25]，適用できるストリーム処理に同様の制限があるため，安全運転支援で用いることは難しい．

6. まとめ

本稿では，データレートや到着タイミングが動的に変化する車々間通信からの入力データをリアルタイム性を満たして有効に活用する方法として，EDF に基づくリアルタイムスケジューリングを行うストリーム処理方式を提案した．本提案では，タスク間に順序依存がある場合や，タイムアウトを持つオペレータにも対応できるため，安全運転支援のデータ処理を含めた広範のデータ処理を扱うことができる．車々間通信を活用する ICRW をストリーム処理で実現し，ETSI の仕様に合わせて 1 秒間に最大 1000 台程度の車両情報を車々間通信から受信する場合をシミュレーションにより評価した．その結果，本走行シナリオにおいて，提案手法が，FIFO スケジューリングを用いる従来のストリーム処理と比較して，最大遅延時間を平均 65%削減し，リアルタイム性を満たしてより多くの車両情報を処理することで，特定のシミュレーション環境のもと，見通しの悪い交差点における出会い頭衝突の事故率を 8.3%から 0%まで削減することを確認した．

謝辞：本研究の一部は SCOPE(121806015) と科研費 (25240007) の助成を受けたものである．

参考文献

[1] 藤田浩一，宇佐美祐之，山田幸則，所　節夫：衝突危険性のセンシング技術，自動車技術，Vol. 61, No. 2, pp. 62–67 (2007).
[2] Erico, G.: How Google's Self-Driving Car Works, Google (online), available from ⟨http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works⟩ (accessed 2011-10).
[3] 国土交通省自動車交通局先進安全自動車推進検討会：先進安全自動車（ＡＳＶ）推進計画報告書 (2011-6).
[4] ETSI: Intelligent Transport Systems; V2X Applications; Part 3: Longitudinal Collision Risk Warning application requirements specification (2013).
[5] ETSI: Intelligent Transport Systems; Vehicular Communications; Basic Set of Applications; Local Dynamic Map Rationale for and Guidance on Standardization (2011).
[6] ETSI: Intelligent Transport Systems; Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service (2011).
[7] Althoff, M., Althoff, D., Wollherr, D. and Buss, M.: Safety verification of autonomous vehicles for coordinated evasive maneuvers, *Intelligent Vehicles Symposium*, IEEE, pp. 1078–1083 (2010).
[8] Tian, Q., Guo, T., Qiao, S., Wei, Y. and wei Fei, W.: Design of Intelligent Parking Management System Based on License Plate Recognition, *Journal of Multimedia*, Vol. 9, No. 6, pp. 774–780 (2014).
[9] Golab, L. and Özsu, M. T.: Issues in Data Stream Management, *SIGMOD Rec.*, Vol. 32, No. 2, pp. 5–14 (2003).
[10] Buttazzo, G. C.: *Hard Real-time Computing Systems: Predictable Scheduling Algorithms And Applications (Real-Time Systems Series)*, Springer (2004).
[11] 佐藤健哉：自動車走行環境認識のためのセンサデータ処理機構，データ工学研究会信学技報， Vol. 110, No. 107, pp. 51–56 (2010).
[12] 勝沼　聡，山口晃広，熊谷康太，本田晋也，佐藤健哉，高田広章：車載組込みシステム向けデータストリーム管理システムの開発，電子情報通信学会論文誌，Vol. 95, No. 12, pp. 2031–2047 (2012).
[13] 山田真大，鎌田浩典，佐藤健哉：データストリーム管理機構を利用した車載データ統合モデルの提案と評価，自動車技術会論文集，Vol. 41, No. 2, pp. 419–424 (2010).
[14] 山口晃広，本田晋也，佐藤健哉，高田広章：車載システム向けデータストリーム管理システムにおけるクエリ自動構築手法，情報科学技術フォーラム講演論文集，Vol. 11, No. 2, pp. 7–14 (2012).
[15] Chetto, H., Silly, M. and Bouchentouf, T.: Dynamic Scheduling of Real-time Tasks Under Precedence Constraints, *Real-Time Syst.*, Vol. 2, No. 3, pp. 181–194 (1990).
[16] Wei, Y., Son, S. and Stankovic, J.: RTSTREAM: real-time query processing for data streams, *Proc. of the 9th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, pp. 141–150 (2006).
[17] Li, X., Jia, Z., Ma, L., Zhang, R. and Wang, H.: Earliest Deadline Scheduling for Continuous Queries over Data Streams, *Proc. of International Conference on Embedded Software and Systems*, pp. 57–64 (2009).
[18] South Australia. Department for Transport, E., Infrastructure, SA., T. and SA, S. A. G. T.: *The Driver's Handbook*, Department for Transport, Energy and Infrastructure (2005).
[19] Chakravarthy, S. and Jiang, Q.: *Stream Data Processing: A Quality of Service Perspective Modeling, Scheduling, Load Shedding, and Complex Event Processing*, Springer (2009).
[20] Schweppe, H., Member, A. Z. and Grill, D.: Flexible On-Board Stream Processing for Automotive Sensor Data, *IEEE Transactions on Industrial Informatics*, Vol. 6, No. 1, pp. 81–92 (2010).
[21] Bolles, A., Appelrath, H., Geesen, D., Grawunder, M., Hannibal, M., Jacobi, J., Koster, F. and Nicklas, D.: StreamCars: A new flexible architecture for driver assistance systems, *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pp. 252–257 (2012).
[22] Schmidt, S., Legler, T., Schaller, D. and Lehner, W.: Real-time scheduling for data stream management systems, *Proc. of the 17th Euromicro Conference on Real-Time Systems*, pp. 167–176 (2005).
[23] Ou, Z., Yu, G., Yu, Y., Wu, S., Yang, X. and Deng, Q.: Tick Scheduling : A Deadline Based Optimal Task Scheduling Approach for Real-Time Data, *Proc. of Advances in Web-Age Information Management*, pp. 725–730 (2005).
[24] Wei, Y., Prasad, V., Son, S. and Stankovic, J.: Prediction-Based QoS Management for Real-Time Data Streams, *Proc. of IEEE Real-Time Systems Symposium*, pp. 344–358 (2006).
[25] Li, X., Ma, L., Li, K., Wang, K. and an Wang, H.: Adaptive Load Management over Real-Time Data Streams, *Proc. of the 4th International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 719–725 (2007).