情報処理学会論文誌 Vol.63 No.2 588–596 (Feb. 2022)

Vehicular Edge Computingにおける コンテナ応答時間を削減するためのコンテナ再配置手法

豊田 睦[1,a)] 佐竹 颯太[1] 武藤 晟[1,b)] 重野 寛[1,c)]

受付日 2021年5月10日, 採録日 2021年11月2日

概要：Vehicular Edge Computing（VEC）によって車両と路側のインフラ設備が連携し，応答の即時性が重要な車両周辺の認識が可能になる．また，近年，サービスを 1 つのパッケージとして提供するコンテナ技術が登場し，軽量性や移植性などの利点から注目を浴びている．コンテナ技術を VEC へ導入した関連研究では，車両移動と時間推移により変化する計算資源，ネットワーク状態への対応が欠如しており，それによるコンテナ応答時間の増大が課題である．そこで，本論文ではコンテナ応答時間の削減を目的とした，車両と Road Side Unit（RSU）が通信できない範囲である RSU 通信ギャップを利用した車両移動性を考慮するコンテナ再配置手法を提案する．提案手法のプロトタイプシステムを実装し，提案手法の性能評価を行う．動作確認の結果より，RSU 通信範囲内において，コンテナとの接続性を確保しながら応答時間を最小化する計算資源へコンテナを配置/再配置することを示す．

キーワード：Vehicular Edge Computing（VEC），コンテナ技術，車両移動性

A Container Relocation Method to Reduce Container Response Time in Vehicular Edge Computing

Mutsumi Toyoda[1,a)] Hayata Satake[1] Akira Muto[1,b)] Hiroshi Shigeno[1,c)]

Received: May 10, 2021, Accepted: November 2, 2021

Abstract: In Vehicular Edge Computing (VEC), vehicles and roadside infrastructures recognize together the around the vehicle, which immediate response is important. Recently, container technology, which provides services as a single package, has been gaining attention due to lightweight and portability. In the related studies that have introduced container technology to VEC, the issues are that vehicles are a lack of response to changing computational resources and network conditions due to vehicle movement and time transition. In this paper, we propose a container reallocation method that considering vehicle mobility by using the RSU communication gap, which is the range where vehicles and RSUs cannot communicate. We implement a prototype of the proposed method that shows the proposed method suppress network traffic and maintain comparable location accuracy with the case when the transmission control was not performed.

Keywords: Vehicular Edge Computing (VEC), container technology, vehicle mobility

1. はじめに

近年，車載コンピュータの計算能力の向上にともない，衝突回避や周辺環境認識などの計算負荷が高く，遅延に対する要求が厳しいサービスが開発されている [1], [2]．それらのサービスは車両と路側のインフラ設備と連携し，認識に使用する情報を集約することで認識の範囲拡大が可能になる．路側インフラの一種である Road Side Unit（RSU）から提供されるサービスは特に応答の即時性が重要な要件である [3]．現在，車載コンピュータは，処理，ストレージなどに制限があるため必ずしもすべてのサービスの実行に適しているといえない．そのため，遅延に敏感でリソースを多く消費するアプリケーションは車両近辺にある計算

[1] 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University, Yokohama, Kanagawa 223–8522, Japan
[a)] toyoda@mos.ics.keio.ac.jp
[b)] muto@mos.ics.keio.ac.jp
[c)] shigeno@mos.ics.keio.ac.jp

リソースへとサービスのホストを要求することが効果的である．

このため，ネットワークエッジにクラウドの機能を提供するモバイルエッジコンピューティング（MEC）[4], [5]が注目されている．MEC ではユーザ近傍のネットワークエッジに配置された MEC サーバを利用し，ユーザと MEC サーバ間の通信距離を短縮することにより厳しい遅延要求を満たすことが可能になる [6]．また，車両ネットワークに MEC を導入した研究は Vehicular Edge Computing（VEC）[7] と呼ばれる．MEC サーバが備わる RSU は分散的に配置されており，RSU の通信範囲は限られているため，車両と RSU はつねに通信が可能ではない．そのため，VEC においてサービスを実行するためには車両の移動性を考慮する必要がある．

サービスの待ち時間を減らし，ユーザの QoE を満たすためにサービスの要求はリソース仮想化環境によって処理される．仮想化はサービスを管理可能な独立したユニットとして提供するために最適である．近年，Docker [8] などのコンテナ技術が注目されている．サービスを 1 つのパッケージとして提供するコンテナ技術は軽量性や移植性などの利点によりクラウドやエッジベースのサービスに主に使用されている．そのコンテナの管理を容易にするコンテナオーケストレーションプラットフォームはコンテナの展開や管理の自動化を提供する．

関連研究として，路側のエッジサーバから車両に継続的にパッケージとしてサービスを提供するコンテナに関する研究が進んでいる [9]．これらの研究ではネットワーク状況を考慮したコンテナスケジューリングが提案されている．しかし，これらの研究には計算資源が置かれている場所やコンテナをリクエストする車両の位置を把握できないという問題がある．また，車両は RSU とつねに通信可能ではないこと，車両移動性が考慮されていないこと，車両移動と時間推移により変化する計算資源，ネットワーク状態への対応が欠如していることが課題である [10]．

本論文では，VEC におけるコンテナ応答時間を削減するためのコンテナ再配置手法を提案する．本論文の目的は，RSU 通信範囲内で車両とコンテナの接続性を維持しながらコンテナ応答時間の削減をすることである．提案手法では，VEC のための階層的なコンテナベースのフレームワークを構築する．車両移動による位置関係の変化・時間推移による計算資源状態とネットワーク状況の変化を考慮し，応答時間が最小になるようなコンテナの MEC サーバへの配置を決定する．車両移動により変化するコンテナと車両の位置関係に対応するために RSU 通信ギャップでコンテナを再配置を完了させる．性能評価では，提案手法のプロトタイプシステムを実装し，交通流シミュレータによって車両の動きを再現した環境において実験を行う．実験では，RSU 通信範囲内における車両とコンテナの接続性やコンテナの応答時間について提案手法の有効性を評価する．これにより，本論文で提案する手法が車両とコンテナの接続性を確保しながら，コンテナ応答時間を最小化するコンテナ配置を実現しているか確認する．

本論文の構成は以下のとおりである．2 章では背景である VEC および，VEC 環境へのコンテナ技術の導入に関する関連研究について述べる．3 章では提案手法について述べる．4 章では提案手法の実装と性能評価について述べる．5 章では本論文のまとめについて述べる．

2. 関連研究

本章では，VEC と VEC 環境へのコンテナ技術の導入に関連する研究について述べる．

2.1 Vehicular Edge Computing（VEC）

図 1 に VEC の概要を示す．VEC は MEC サーバを車両ネットワークに導入することで，車両上のデバイスよりも強力な計算資源でアプリケーションを代わりに処理すること，センシングデータを高速に処理および分析すること，周辺車両のセンシングデータを集約し利用することが可能になる．また，MEC サーバはネットワークのエッジ部分に分散的に配置されており，MEC サーバが備わる RSU は通信可能な範囲が限られて，車両と RSU には通信不可能な範囲（通信ギャップ）が存在する．したがって，VEC では車両の位置把握（移動性）を考慮しなければならないという課題がある [10]．

2.2 VEC 環境へのコンテナ技術の導入

路側のエッジサーバから車両に継続的にパッケージとしてサービスを提供するコンテナに関する研究が進んでいる．Kubernetes [11] に代表されるコンテナオーケストレーションフレームワークは，軽量性や移植性などの利点によりクラウドベースやエッジベースのサービスに主に使用されている．しかし，コンテナオーケストレーションでは計算資源の地理的位置を認識することができないため，要求ユーザから離れた計算資源にコンテナを配置することがある．たとえば，文献 [9] ではネットワーク状況を考慮した

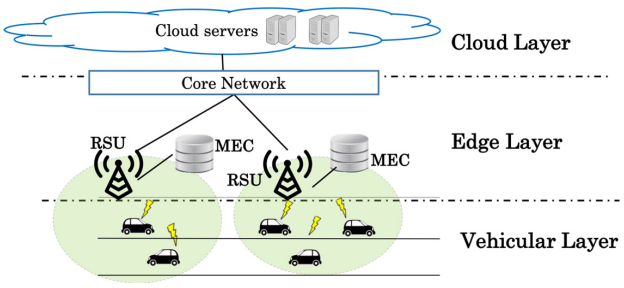

図 1 VEC の概要
Fig. 1 Overview of VEC.

コンテナスケジューリングが提案されているが，コンテナが配置される計算資源の地理的位置は考慮されていない．

Tang ら [12] は自動運転のためのコンテナベースのオフローディングを提案した．最寄りエッジサーバへアプリケーションカテゴリと実行データのみをオフロードし，エッジサーバのコンテナからサービスを受ける．最寄りサーバが混雑しているときは近隣のエッジサーバからオフロード候補となるエッジサーバを選択する．1 台の車両から複数のカテゴリのサービスの要求が来ることも想定しており，ビンパッキング問題を高速な GA で解くことでオフロード先を決定する．

Maheshwari ら [13] はコンピューティングとネットワーク対応の分散リソース移行アルゴリズムの ShareOn を提案した．アプリケーションの遅延合計が大きいコンテナを候補リストに入れ，それらを別のエッジサーバに移行する．各エッジサーバがコンテナを分散して制御・移行する．コンピューティングリソース使用率とユーザの地理的位置によって決定する．

以上の研究では VEC において遅延を最小化するコンテナ配置問題について研究がされているが，車両とコンテナをホストする車両の位置関係を継続的に考慮していないという点，車両移動性により変化する計算資源，ネットワークの状態への対応が欠如しているという点があげられる．よって，VEC 環境において計算資源の地理的位置や車両移動性を考慮したコンテナ配置手法が必要となる．

3. 提案

本章では，VEC におけるコンテナ応答時間を削減するためのコンテナ再配置手法を提案する．

3.1 提案概要

提案手法は RSU 通信可能時間を最大限活用し，車両とコンテナの接続性を確保したうえでコンテナ応答時間を削減することを目的とする．提案手法では，都市部のように要所にスポットとして車両と通信可能な RSU が複数存在する環境を想定する．車両とコンテナの接続性，応答の即時性の確保は通信ギャップにおいてコンテナを再配置することで実現する．コンテナの応答時間の削減はコンテナをホストする候補となる MEC サーバから応答時間が最小となる MEC サーバを選択することで実現する．提案手法の主な特徴は以下の 3 点である．

- VEC のための階層的なコンテナベースのフレームワークを構築する．コンテナオーケストレーションフレームワークを導入し，MEC サーバをクラスタ化し，コンテナの管理・運用を簡潔にする．
- Mobility-Aware Container Scheduler（MAC Scheduler）は応答時間を最小化する MEC サーバを決定し，決定された MEC サーバのコンテナ配置を制御する．車両移動による位置関係の変化・時間推移による計算資源状態とネットワーク状況の変化を考慮したコンテナ配置決定を行う．配置決定のパラメータとして，計算資源の状態，ネットワーク状況，通信遅延を導入する．
- 車両と RSU が通信できない範囲である通信ギャップでコンテナを再配置する．車両の移動を予測し，車両が次の RSU 通信範囲内に入るまでにコンテナの再配置を完了させる．

以上 3 点の特徴により，RSU 通信範囲内において，コンテナとの接続性を確保しながら応答時間を最小化する計算資源へコンテナを配置/再配置することを可能にする．最後に，実機を用いたプロトタイプシステムを実装し，実験を行う．

3.2 コンテナベースの VEC フレームワーク

図 2 にコンテナベースの VEC フレームワークを示す．提案手法では，カバーエリアの異なる M の RSU を道路脇に配置し，その集合を $\{RSU_1, \cdots, RSU_m, \cdots, RSU_M\}$ とし，RSU_m の通信範囲を r_m と表記する．RSU_m には計算資源を提供するための MEC サーバ MEC_k が備わっており，MEC サーバの集合を $\{MEC_1, \cdots, MEC_k, \cdots, MEC_K\}$ とする．車両の集合を $\{V_1, \cdots, V_n, \cdots, V_N\}$ と表す．各車両 V_n に対して，道路に到着した際にコンテナ要求を生成し，このコンテナを車両 V_n の固有コンテナ c_n として指定する．

MEC Controller

RSUM（Road Side Unit Manager）とともに設置されており，提案手法におけるコントローラの役割およびシステム全体の制御を担う．コンテナオーケストレーションの Master として機能し，MEC Servers をクラスタ化し，管理する．コンテナの起動状態，複数の MEC server の位置・計算資源の状態，ネットワークの状態，車両情報などの情報の集約も行う．それらの情報に基づいて，車両移動の予

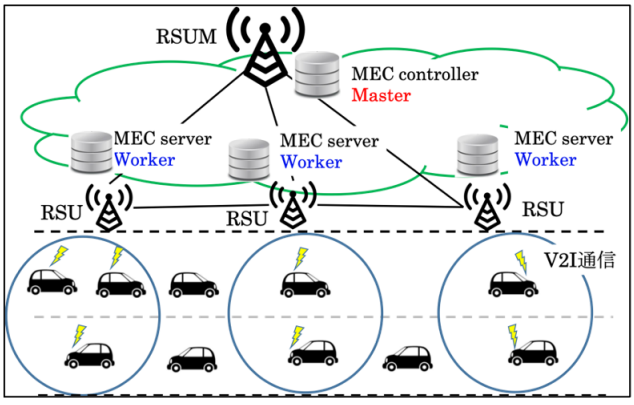

図 2 コンテナベースの VEC フレームワーク
Fig. 2 VEC framework based on container.

測やコンテナの配置場所を MAC Scheduler によって決定し，配置の命令を MEC Servers へ送信する．

RSUM（Road Side Unit Manager）

RSU よりも高レイヤに配置されており，RSU の管理を行う．また，車両からの車両情報，コンテナリクエストの受信，MEC Server からの計算資源の情報の受信を行う．

MEC Server

RSU（Road Side Unit）とともに設置されており，コンテナオーケストレーションの Worker として機能し，コンテナのホストを担当する．コンテナが情報を利用できるように周辺車両情報を集約・管理する．MEC Server どうしで車両情報の受け渡しも行う．自計算資源の状況に関する情報を MEC Controller へと定期的に送信する．

RSU（Road Side Unit）

通信範囲内の車両と通信を行い，車両情報の受信やコンテナの計算結果の送信を行う．RSUM と通信を確立し，MEC Controller が MEC Server へコンテナの配置命令などを送信可能にする．

車両（Client）

車両は自車両情報を通信可能な RSU へ定期的に送信する．RSU 経由で Master ノードへとコンテナの実行に必要な情報を含んだコンテナのリクエストを送信する．その後，コンテナをホストする MEC Server から計算結果を受信する．

3.3 Mobility-Aware Container Scheduler（MAC Scheduler）

MAC Scheduler は MEC Controller 上に実装されたコンポーネントであり，応答時間を最小にするコンテナを配置する MEC Server を決定する．配置決定のパラメータとして，計算資源の状態，ネットワーク状況，通信遅延を導入することで，車両移動による位置関係の変化・時間推移による計算資源状態とネットワーク状況の変化をコンテナ配置決定に反映させる．このコンポーネントが配置するコンテナは各車両専用のコンテナとする．車両ごとにコンテナを起動することで，リソースを分離することが可能になり安全性を確保できる．

3.4 コンテナ応答時間を最小化する MEC Sever の決定

総応答時間は車両がコンテナを要求してから計算結果を受信するまでの時間差であり，コンテナ処理遅延 T_{com}，伝送遅延 T_{trans}，通信遅延 T_{dis} の和で表すことができる．そこで，総応答時間を最小化する最適なコンテナ配置先 MEC_k を決定する問題を以下のように定式化する：

$$min \sum_{k=1}^{K} (T_{com} + T_{trans} + T_{dis}) \tag{1}$$

式 (1) において，応答時間が最小となる MEC サーバのインデックス k を決定し，その MEC_k 上にコンテナ C_n を配置する．以下に式 (1) の各要素について示す．

車両からの計算処理は要求ごとにコンテナにより処理される．コンテナの処理にはコンテナの起動，処理，終了があるが，ここでは起動から終了した時間までをコンテナ処理時間とする．車両 n の要求を処理するコンテナ C_n を，ある MEC サーバ MEC_k で処理するために要する時間 T_{com} を式 (2) に示す．

$$T_{com} = COM_{k,C_n}/(1 - \delta_{t,k}) \tag{2}$$

ここで，COM_{k,C_n} は，MEC_k においてコンテナ C_n が要求する計算時間，$\delta_{t,k}$ は時刻 t における MEC_k の CPU 使用率，つまり，$1 - \delta_{t,k}$ は時刻 t における MEC_k の CPU 使用可能率を表す．

MAC Scheduler によって車両 n の要求を処理するコンテナ C_n を配置する MEC サーバ MEC_k が決定する．伝送遅延 T_{trans} とは，コンテナ C_n が配置されている MEC_k から次に車両が進入する MEC_{k+1} へコンテナを伝送する時間であり，式 (3) に示す．

$$T_{trans} = S_{C_n}/B_{t,m+1,k} \tag{3}$$

ここで，S_{C_n} はコンテナ C_n の実行に必要なデータサイズ，$B_{t,m+1,k}$ は時刻 t における次に車両が進入する RSU_{m+1} とコンテナ配置先 MEC_k の間の帯域幅を表す．

通信遅延 T_{dis} は，コンテナ C_n の配置先 MEC サーバ MEC_k と次に車両が進入する RSU_{m+1} の間の距離による通信遅延であり，式 (4) に示す．

$$T_{dis} = D_{t,m+1,k} \tag{4}$$

ここで，$D_{t,m+1,k}$ は時刻 t における次に車両が進入する RSU_{m+1} とコンテナ配置先 MEC_k の間の距離による通信遅延を表す．

図 3 に車両がコンテナを要求し，総応答時間を最小化する最適なコンテナ配置先 MEC_k を決定するアルゴリズム

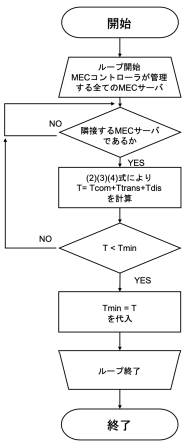

図 3 コンテナ配置アルゴリズムフローチャート

Fig. 3 Flowchart of the container placement algorithm.

を示す．MEC コントローラは車両情報および MEC サーバ情報を収集する．車両情報とは車両 ID，車両の地理的位置，車両速度，車両角度であり，MEC サーバ情報とは時刻 t における CPU 使用率，MEC サーバの地理的位置，コンテナ C_n が要求する計算時間，コンテナ C_n のデータサイズを指す．車両コンテナ配置アルゴリズムでは，MEC コントローラが管理するすべての MEC サーバのうち通信範囲外になる前の MEC サーバおよびその隣接 MEC サーバを対象に式 (2)，(3)，(4) によりコンテナ推定総応答時間を計算する．対象の MEC サーバに対してコンテナ総応答時間を計算し，最小となる MEC サーバを決定する．

3.5 RSU 通信ギャップにおけるコンテナ再配置

MEC Controller が車両と RSU が通信できない範囲である通信ギャップでコンテナを再配置する．車両の移動を予測し，車両が次の RSU 通信範囲内に入るまでにコンテナの再配置を完了させる．車両が RSU の範囲外に出る時刻を算出し，その時刻に MAC Scheduler を起動する．まず，車両が RSU 通信範囲内のどの場所に位置するかを把握するために，車両 V_n と MEC_k の位置関係を求める．車両と MEC サーバとの距離 $L_{n,k}$ は以下のように算出する：

$$L_{n,k} = |P_k - {x_n}^t| \tag{5}$$

ここで，P_k は MEC_k の地理的位置であり，RSU_m の中心位置 P_m と同義と仮定する．${x_n}^t$ は時刻 t における車両 V_n の位置を表す．また，車両が RSU の範囲外に出る時刻 ${t_n}^{out}$ は以下のように算出する：

$${t_n}^{out} = t_{now} + (r_m \pm L_{n,k})/v_n \tag{6}$$

ここで，t_{now} は現在時刻，r_m は RSU_m の通信半径，v_n は車両 V_n の速度を表す．

図 4 はコンテナ再配置の全体像を示す．図 4 において車両 V_n〜V_{n+3} の 4 台が存在し，V_{n+1} と V_{n+2} は RSU の通信範囲に位置し，V_n と V_{n+3} は RSU の通信範囲外に位置する．車両 V_{n+1} と V_{n+2} は RSU と通信し，コンテナから結果を受信する．一方，車両 V_n と V_{n+3} は通信ギャップに存在する．その通信不可能な範囲において，MEC Controller がコンテナの再配置を行う．たとえば，車両 V_n 専用のコンテナ C_n は車両 V_n が RSU_{m+1} の通信範囲内に位置していたときは MEC_{k+1} に配置されていた．車両 V_n が RSU_{m+1} の通信範囲外に出た時刻 ${t_n}^{out}$ に再配置を開始し，コンテナ C_n を MEC_{k+2} に配置する．車両 V_{n+3} 専用のコンテナ C_{n+3} も同様に配置する．

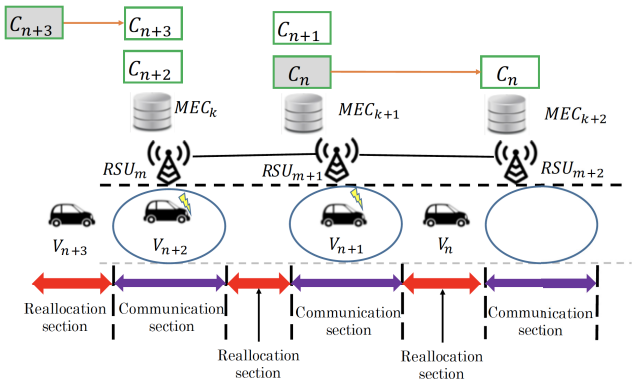

図 4 コンテナ再配置の全体像
Fig. 4 Overview of container relocation.

4. プロトタイプシステム実装と評価

本章では，提案手法のプロトタイプシステムの詳細と評価項目および性能評価について述べる．

4.1 プロトタイプシステム実装

提案手法の動作確認および性能評価を行うために，提案手法のプロトタイプシステムを実装した．**図 5** は提案手法のプロトタイプシステムの概要を示している．プロトタイプシステムにおいて，コンテナの管理や運用にはコンテナオーケストレーションである Kubernetes を導入し，MEC Controller が Master Node として動作し，3 つの MEC サーバが Worker Node として動作している．MEC Controller が kubeadm を使用して 3 つの MEC サーバをまとめた Kubernetes クラスタを作成した．車両は物理マシン上に VM を 8 台起動し，車両情報の送信やコンテナのリクエストを行う．RSUM は MEC Controller 内に，RSU は MEC サーバ内に構成されているものとし，すべての通信は UDP によるソケット通信で行われる．

4.2 評価方法

実装したプロトタイプシステムを用いて評価実験を行った．**表 1** は実験パラメータをまとめたものである．実験時間は 500 秒で，総流入車両台数は 40–50 台となるようにした．車両は実装したプロトタイプシステムに車両のエミュ

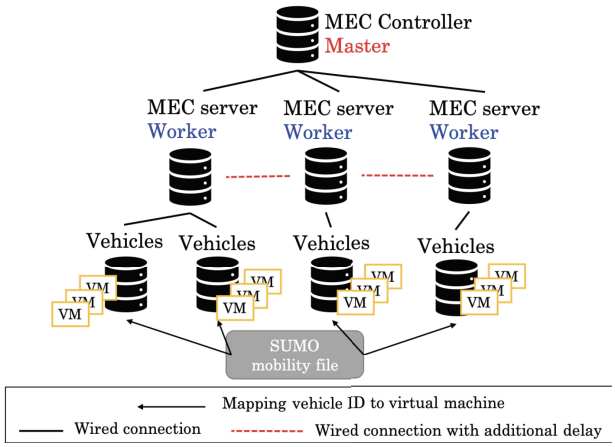

図 5 プロトタイプシステムの概要
Fig. 5 Overview of prototype system.

表 1 実験パラメータ
Table 1 Experimental parameters.

パラメータ	値
交通流シミュレータ	SUMO（ver.1.6.0）
車両移動モデル	Krauss
車両速度	50–60 km/h
実験時間	500 sec
車両総数（密度：低）	40–50 台
車両情報の送信間隔	100 msec
RSU の通信範囲	半径 100 m 円内
MEC サーバ情報収集間隔	100 msec
コンテナリクエスト間隔	200 msec
道路モデル	片側 1 車線の直線道路
MEC サーバの付与遅延	5–10 msec

レーションと交通流シミュレータ SUMO（ver.1.6.0）[14] を加え，仮想的に動作するものとした．車両の走行モデルは前方車両と衝突しない車間距離の維持に基づいたモデルである Krauss ら [15] のモデルを使用した．車両情報の送信間隔は 100 msec とした．RSU，MEC サーバは路側に設置されているものと想定し，半径 100 m の通信範囲内で車両と通信できるようにした．また，MEC Controller における MEC サーバの情報取得間隔は 100 msec とした．MEC サーバ間の通信経路に tc コマンド [16] を用いて，意図的に 10 msec の遅延を付与し，通信距離による遅延を再現した．

動作コンテナは周辺環境情報共有サービスを提供するものとした．車両専用のコンテナを作成し，コンテナ内で動作するアプリケーションは周辺車両情報共有アプリケーションと画像に写っている人物をカウントするアプリケーションで構成されている．MEC サーバにコンテナが起動した時点から車両が RSU 通信範囲外へ移動する時点まで，車両カメラによって撮影された画像データがコンテナの配置された MEC サーバへ送信され，その画像データを処理する．そのため，データサービスへのアクセスがコンテナ応答時間へ与える影響は小さい．人物カウントアプリケーションは OpenCV ライブラリ [18] を用いて実装した．車両は RSU 通信範囲内において，200 msec の間隔で，RSU 経由で MEC Controller へコンテナのリクエストを送信する．

帯域幅は Iperf [17] を用いて周期的に測定を行い，その実測値を実験パラメータとして使用した．3.4 節でも述べたとおり，対象となる MEC サーバは通信範囲外になる前の MEC サーバおよびその隣接 MEC サーバであり，数台程度と想定されるため，帯域幅の計測は性能評価において影響を及ぼさないと考える．

4.3 評価項目

実験における評価項目は以下のとおりである．

- 車両 1 台における応答時間の推移

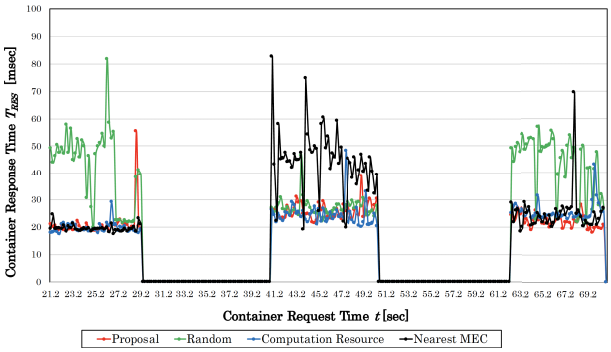

図 6 車両 1 台における応答時間の推移
Fig. 6 Response time for one vehicle.

応答時間は車両がコンテナをリクエストしてから，コンテナにアクセスし，結果を得るまでの時間を計測した値とする．提案手法が応答時間を最小にする MEC サーバを選択できていることを確認する．比較対象は Random と Computation Resource，Nearest MEC である．Random とは MEC コントローラによってランダムにコンテナ配置先を決定する手法である．Computation Resource とは MEC サーバの計算負荷のみを考慮した手法であり，対象となる MEC サーバ k の時刻 t における CPU 使用率 $\delta_{t,k}$ を比較し，低い順にコンテナを配置する手法である．Nearest MEC とは車両と MEC サーバの物理的距離のみを考慮した手法で，車両の近隣 MEC サーバにのみコンテナを配置する手法である．

- 平均応答時間
  平均応答時間は実験時間におけるすべてのコンテナ応答時間の平均値である．提案手法が車両ネットワーク全体において，応答時間を削減できていることを確認する．車両 1 台における応答時間の推移と同様である．
- コンテナライフタイム
  コンテナライフタイムは車両とコンテナが接続されている間の時間である．提案手法が RSU 通信範囲内において車両とコンテナが接続性を維持し続けることを確認する．比較対象は Allocating into RSU である．Allocating into RSU とは車両が次の RSU 通信範囲内に進入するたびにコンテナを再配置する手法である．

4.4 車両 1 台における応答時間の推移

図 6 に車両 1 台における応答時間の推移を示す．図 6 に示すように，車両が通信ギャップを通過するごとにコンテナが配置されている MEC サーバが再配置によって変化していることが確認できる．また，提案手法はそれぞれの RSU 接続時に応答時間が最小となるコンテナ配置場所を決定できていることが分かる．特に，提案手法は Nearest MEC と比較して，混雑している MEC サーバや計算能力の低い MEC サーバを避けたコンテナの配置が可能であ

る．これは提案手法が車両移動と時間推移にともなう計算資源・ネットワーク状態を考慮し，応答時間を最小化するMEC サーバを選択できていることを示している．

4.5 平均応答時間

図 7，**図 8** に MEC Server 間に距離による通信遅延を 10 msec 付与した場合，通信遅延を付与しない場合の平均応答時間を示す．通信遅延を 10 msec 付与した理由は通信距離による遅延を考慮するためと帯域圧迫による遅延を考慮するためである．実験では，アプリケーションへ送信するデータサイズは 100–400 KB であり，5 fp で動作しているため，各車両の必要帯域は 0.5–2 Mbps 程度だと考えられる．車両と RSU 間の通信を有線接続の 1 Gbps で動作確認をしているため，帯域圧迫は発生していない．本論文はコンテナの配置に焦点を当てたものであり，MEC が車両環境に導入される今後の展望として無線通信の帯域幅の拡張が見込まれるため，本論文ではその点を考慮しない．図 7 に示すように，MEC Server 間に距離による通信遅延を 10 msec 付与した場合，提案手法の平均応答時間は Random と比較して，14.11 msec の削減，Computation Resource と比較して，9.15 msec の削減，Nearest MEC と比較して 4.10 msec の削減を実現した．図 8 に示すように，提案手法の平均応答時間は Random と比較して，9.21 msec の削減，Computation Resource と比較して，6.11 msec の削減，Nearest MEC と比較して 4.99 msec の削減を実現した．

以上より，提案の配置手法は車両ネットワークの全体的なコンテナ応答時間の削減を達成し，リアルタイムなサービスに対して効果的であるといえる．MEC コントローラにリクエストが集中した場合においても 3.4 節で述べたアルゴリズムの計算量が評価に与える影響は小さいと考える．また，応答時間は車両が MEC コントローラにリクエストを送信してからコンテナが起動し，アプリケーションの実行結果が車両に転送されるまでの時間を指しており，車両，路側機，MEC サーバ，コントローラ間のシグナリングなどの遅延を含めた時間で評価を行っている．評価において制御のためのオーバヘッドを含めても他のコンテナ配置手法と比較して有効であると考えられる．

4.6 コンテナライフタイム

図 9，**図 10** に RSU 通信ギャップが 100 m の場合，30 m の場合の車両 1 台のコンテナライフタイムの実験結果を示す．図 9 と図 10 に示すように，コンテナの起動時間は 2 秒程度かかる．3.3 節で述べたように提案手法では初期起

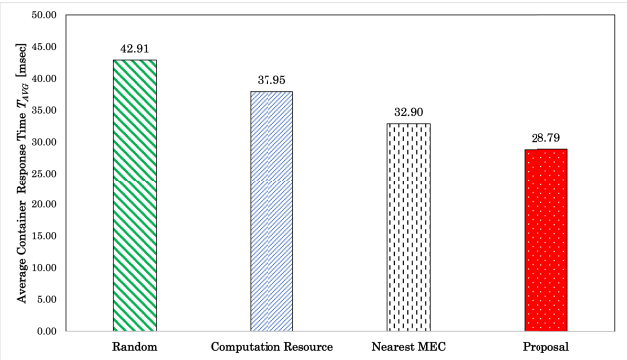

図 7 通信遅延を 10 msec 付与した場合の平均応答時間
Fig. 7 Average response time when communication delay is 10 ms.

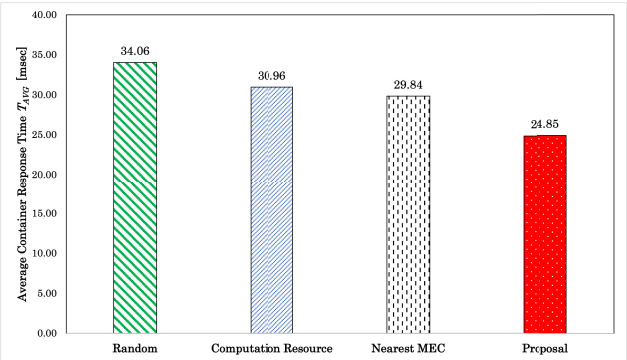

図 8 通信遅延を付与しない場合の平均応答時間
Fig. 8 Average response time when no communication delay is added.

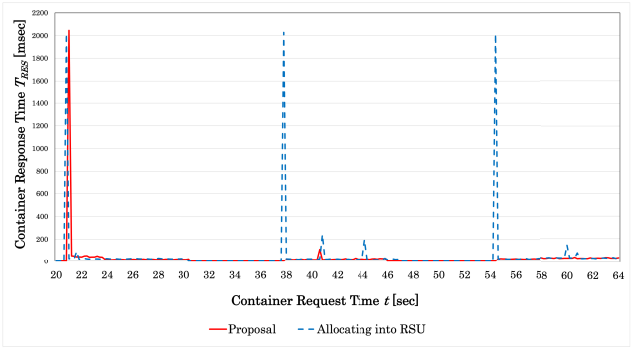

図 9 RSU 通信ギャップが 100 m の場合のコンテナライフタイム
Fig. 9 Container lifetime when the RSU communication gap is 100 m.

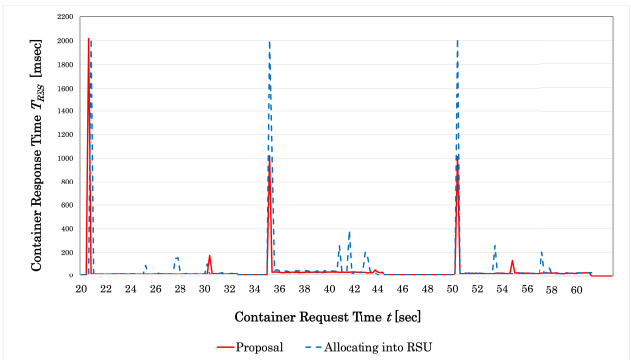

図 10 RSU 通信ギャップが 30 m の場合のコンテナライフタイム
Fig. 10 Container lifetime when the RSU communication gap is 30 m.

動を除いて，コンテナの起動を RSU 通信ギャップで行っているため，RSU 通信範囲内ではコンテナの起動時間を削減できている．つまり，RSU 通信範囲内では低遅延でのコンテナ応答を維持することができている．よって，車両移動にともない，接続する RSU が変化するが，コンテナリクエスト時におけるコンテナ起動状態を維持することができ，接続性，応答の即時性を確保可能であるといえる．図 10 に示すように，RSU 間の通信ギャップが 30 m のとき，提案手法でも 1 秒程度のコンテナ起動時間が RSU 通信範囲内でかかる．それはコンテナの起動を終える前に次の RSU 通信範囲内に車両が進入するからである．車両速度が 50–60 km/h であり，コンテナ起動時間が 2 秒であるとすると，RSU 間の通信ギャップが 30 m 程度のときが本システムが有効に機能する上限である．

5. おわりに

本論文では，VEC におけるコンテナ応答時間を削減するためのコンテナ再配置手法を提案した．提案手法では，路側の RSU に接続された MEC サーバへコンテナを最適に配置し，通信ギャップを利用したコンテナの再配置も行う．配置決定のパラメータとして，計算資源の状態，ネットワーク状況，通信遅延を導入することで，車両移動による位置関係の変化・時間推移による計算資源状態とネットワーク状況の変化をコンテナ配置決定に反映させる．また，車両の移動を予測し，車両が次の RSU 通信範囲内に入るまでの通信ギャップでコンテナの再配置を完了させる．

性能評価のために，提案手法のシステムを実装し，交通シミュレータを用いて実験を行った．実験では，コンテナライフタイムと車両 1 台における応答時間の推移，平均応答時間について評価した．評価の結果，提案手法が RSU 通信範囲内において，コンテナとの接続性を確保しながら応答時間を削減する計算資源へコンテナを配置/再配置することを実現したと結論づけられる．

今後の課題として，本論文では片側 1 車線の直線道路という想定で実験を行ったが，現実には交差点のような複雑な道路環境も存在する．提案手法では車両の移動性を車両の角度 θ および車両速度によって予測しているが，車両がある方向へ走行している際に，次の MEC サーバが一意に決定できる場合に有効であり，交差点や分岐点のような進行先が複数に候補となるような道路形状には，このままでは対応できない．より正確の移動予測手法と組み合わせて，改良することが課題である．車両密度の高い都市部において適用するためにも，車両の移動予測精度を向上させ，同一の RSU 通信範囲に移動する車両をクラスタ化して処理することでアルゴリズムの再計算を抑えるなどの検討も必要である．また，車両の移動予測が外れた場合に，コンテナ起動時間が応答時間に大きく影響を及ぼすことが考えられる．実際の車両環境に適用することを考えると，車両のオフロードサービスの種類やその数を特定し，どのサービスのコンテナが標準化されるかなども考慮する必要がある．また，車載処理もコンテナベースの管理が検討されている．このような技術を利用して車載コンピュータを含む，コンテナベースの処理の管理や最適化は今後の課題である．

謝辞 本研究は JSPS 科研費 JP20H04180 の助成を受けたものです．

参考文献

[1] Eiter, T., Fureder, H., Kasslatter, F., Parreira, J.X. and Schneider, P.: Towards a semantically enriched local dynamic map, *International Journal of Intelligent Transportation Systems Research*, Vol.17, pp.32–48 (Jan. 2019).

[2] 内閣府：ダイナミックマップの概念/定義および, SIP-adus における取り組みに関する報告 (2017)（オンライン），入手先 ⟨https://www8.cao.go.jp/cstp/gaiyo/sip/iinkai/jidousoukou30/siryo30-2-1-1.pdf⟩（参照 2021-01-08）.

[3] Kitazato, T., Tsukada, M., Ochiai, H. and Esaki, H.: Proxy cooperative awareness message: An infrastructure-assisted v2v messaging, *2016 9th International Conference on Mobile Computing and Ubiquitous Networking* (*ICMU*), pp.1–6 (Oct. 2016).

[4] ETSI – European Telecommunications Standards Institute: ETSI GS MEC-IEG 004 V1.1.1 (2015-11) – Mobile-Edge Computing (MEC); Service Scenarios (2015) (online), available from ⟨https://www.etsi.org/deliver/etsigs/MEC-IEG/001099/004/01.01.0160/gsMEC-IEG004v010101p.pdf⟩ (accessed 2021-01-08).

[5] ETSI – European Telecommunications Standards Institute: ETSI GS MEC 011 V2.1.1 (2019-11) – Multi-access Edge Computing (MEC); Edge Platform Application Enablement; Service Scenarios (2019) (online), available from ⟨https://www.etsi.org/deliver/etsigs/MEC/001099/011/02.01.0160/gsMEC011v020101p.pdf⟩ (accessed 2021-01-08).

[6] 佐竹颯太，谷遼太郎，豊田 睦，重野 寛：車載器非搭載車両を考慮した協調認識のためのモバイルエッジコンピューティング支援型車両情報共有システム，情報処理学会論文誌，Vol.62, No.2, pp.475–483 (2021).

[7] Raza, S., Wang, S., Ahmed, M. and Anwar, M.R.: A survey on vehicular edge computing: Architecture, applications, technical issues, and future directions, *Wireless Communications and Mobile Computing*, Vol.2019, 19 pages (Feb. 2019).

[8] Docker containerization unlocks the potential for dev and ops, Docker (online), available from ⟨https://www.docker.com/⟩ (accessed 2021-01-08).

[9] Xu, C., Rajamani, K. and Felter, W.: Nbwguard: Realizing network QoS for Kubernetes, *Proc. 19th International Middleware Conference Industry* (*Middleware '18*), pp.32–38, Association for Computing Machinery (Dec. 2018).

[10] Raza, S., Wang, S., Ahmed, M. and Anwar, M.R.: A Survey on Vehicular Edge Computing: Architecture, Applications, Technical Issues, and Future Directions, *Wireless Communications and Mobile Computing*, Vol.2019, 19 pages (Feb. 2019).

[11] Kubernetes, automated container deployment, scaling, and management (online), available from ⟨https://kubernetes.io/⟩ (accessed 2021-01-08).

[12] Tang, S.L.J., Yu, R. and Gaudiot, J.: A container based edge offloading framework for autonomous driving, *IEEE Access*, Vol.8, pp.33713–33726 (Feb. 2020).
[13] Maheshwari, S., Choudhury, S., Seskar, I. and Raychaudhuri, D.: Traffic-aware dynamic container migration for realtime support in mobile edge clouds, *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems* (*ANTS*), pp.1–6 (May 2018).
[14] Krajzewicz, D., Erdmann, J., Behrisch, M. and Bieker-Walz, L.: Recent development and applications of sumo – Simulation of urban mobility, *International Journal on Advances in Systems and Measurements*, Vol.3-4 (Dec. 2012).
[15] Krauss, S., Wagner, P. and Gawron, C.: Metastable states in a microscopic model of traffic flow, *Phys. Rev. E*, Vol.55, pp.5597–5602 (May 1997).
[16] Manpage maintained by Bert Hubert (ahu@ds9a.nl), tc - show / manipulate traffic control settings (online), available from ⟨http://man7.org/linux/man-pages/man8/tc.8.html⟩ (accessed 2021-01-08).
[17] Iperf.fr: iperf - the tcp/udp bandwidth measurement tool (online), available from ⟨https://iperf.fr/⟩ (accessed 2021-01-08).
[18] Intel Corporation: OpenCV (open source computer vision library) (online), available from ⟨https://opencv.org⟩ (accessed 2021-01-08).

豊田 睦 （学生会員）

2020 年慶應義塾大学理工学部情報工学科卒業．現在，同大学大学院理工学研究科修士課程在学中．

佐竹 颯太

2019 年慶應義塾大学理工学部情報工学科卒業．2021 年同大学大学院理工学研究科修士課程修了．

武藤 晟 （学生会員）

2021 年慶應義塾大学理工学部情報工学科卒業．現在，同大学大学院理工学研究科修士課程在学中．

重野 寛 （正会員）

1990 年慶應義塾大学理工学部計測工学科卒業．1997 年同大学大学院理工学研究科博士課程修了．現在，同大学理工学部教授．博士（工学）．情報処理学会論文誌編集委員，同 DPS 研究会主査，同理事，電子情報通信学会英文論文誌 B 編集委員，Secretary of IEEE ComSoc APB 等を歴任．現在，同 ITS 研究会主査，Co-Chair of IEEE ComSoc APB MCC．ネットワーク・プロトコル，ITS 等の研究に従事．著書『ユビキタスコンピューティング』（オーム社），『情報学基礎第 2 版』（共立出版）等．電子情報通信学会，IEEE，ACM 各会員．本会シニア会員．