「マルチメディア，分散，協調とモバイル
(DICOMO2021) シンポジウム」 令和3年6月-7月

車両エッジコンピューティングにおける アプリケーション分割オフロード手法の提案

武藤 晟[1]　豊田 睦[1]　重野 寛[1]

概要：現在，Internet of Vehicles (IoV) の発展により，遅延に敏感なアプリケーションが登場している．車両エッジコンピューティング (VEC) における既存のオフロード手法では，オフロード先の VEC サーバの内，一番計算資源が豊富なサーバへオフロードを実行する．しかし，既存手法ではアプリケーションベースのオフロードによって VEC サーバに負荷が集中し，応答時間が増加する可能性がある．そこで，車両エッジコンピューティングにおけるアプリケーション分割オフロード手法を提案する．タスクをオフロードした際の推定応答時間の合計が最小となるようなオフロード先を求めるオフロード決定式を定義する．各計算資源はタスク完了のために要求される計算能力に基づいて推定応答時間を算出する．そして，オフロード決定式を用いて推定応答時間が最小となるようにオフロード先を決定する．本研究では提案したタスクオフロード手法のプロトタイプを実装して実験を行い，動作確認と評価を行なった．実験により，実装したオフロード手法によってオフロード決定式で決定したオフロード先で各タスクは実行され，実際の応答時間も小さくなるように実行されることを確認した．

Proposal of Application Split Offloading in Vehicular Edge Computing

AKIRA MUTO[1]　MUTSUMI TOYODA[1]　HIROSHI SHIGENO[1]

1. はじめに

現在，IoV の発展により，自動運転や人物検出などの様々な遅延に敏感なアプリケーションが登場している．車載計算リソースには限界があり，車両だけではその遅延制約を達成できない可能性があるため，車両の計算負荷を軽減するために，タスクオフローディングが注目されている．車両はアプリケーション実行に必要なタスクを，計算資源が豊富なサーバへ転送し実行することで車両ネットワークの負荷を解消する．また，モバイルユーザが実行可能なアプリケーションを，ネットワークに分散配置されたサーバを用いて分散処理を行うコンピューティングモデルである，モバイルエッジコンピューティング (MEC) [1] が注目されている．特に，車両ネットワークにおける MEC を車両エッジコンピューティング (VEC) と呼ぶ．本稿では VEC におけるタスクオフローディングについて検討する．

車両ネットワークにおいていくつかのオフロード手法が提案されている．ロードバランシング・タスクオフロード手法 [2] では，複数の VEC サーバが存在した場合に異なる VEC サーバ間の負荷を均等にすることを考慮し，車両における計算タスクの処理遅延の最小化を達成することを目的としている．また，VEC ネットワークにおけるタスクオフロード手法 [3] では，車両の計算負荷軽減を目的にモバイルエッジコンピューティングを車両ネットワークに導入している．これらの手法ではアプリケーションを単位としてオフロードすることを検討しているが，このような粒度の大きいアプリケーションベースのオフロード手法では VEC サーバに負荷が集中して処理遅延が増加する可能性があるため，オフロード対象の粒度の大きさを考慮したオフロード手法が必要であると考える．

本稿では，車両エッジコンピューティングにおけるアプリケーション分割オフロード手法の提案を行う．提案手法において構築するシステムでは VEC サーバの計算リソー

[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

スとアプリケーションの応答時間に焦点を当てる．実装するオフロードシステムでは VEC サーバの 1 つがシステムを制御し，各計算資源はタスク完了のために要求される計算能力に基づいて推定応答時間を算出する．そして，推定応答時間が最小となるようにオフロード先を決定し，実行する．アプリケーションを複数タスクに分割し，タスク完了のために要求される計算能力に基づいてオフロード先の決定を行うことで，アプリケーション分割オフロード手法を実現する．

本稿の構成は以下の通りである．第 2 章では VEC とタスクオフロード手法の関連研究について述べる．第 3 章では提案手法について述べる．第 4 章では提案手法の実装と評価について述べる．第 5 章では本稿のまとめとする．

2. 関連研究

本章では，車両ネットワークにおけるモバイルエッジコンピューティングについて述べ，既存の車両ネットワークにおけるタスクオフロード手法の問題点について述べる．

2.1 車両エッジコンピューティング (VEC)

車両アプリケーションではリアルタイムに生成されたデータや大規模データを処理する必要があるため，計算能力，ストレージ容量，電力消費の要求も高まっている．しかし，車載コンピュータの計算資源，ストレージ容量ではこの要求を満たすことができず，電力消費が増加すると燃料消費が増加する．そこで，車両の計算負荷軽減を目的にモバイルエッジコンピューティングを車両ネットワークに導入した VEC [3] が提案されている．VEC では，道路を動く車両は遅延に敏感な計算タスクを路側機 (RSUs) を経由してエッジサーバにオフロードすることでタスク処理遅延の最小化を図る．

図 1 に VEC のシステムモデルを示す．システムは，車両，RSU，VEC サーバ，クラウドサーバ，コントローラから成る．車両は車車間通信，路車間通信を用いてタスクを VEC サーバ，またはクラウドサーバにオフロードする．RSU は路側インフラの一種であり，車両や RSU と通信を行う．また，RSU 間はバックホールネットワークで接続されている．VEC サーバは RSU 上に配置されており，車両アプリケーションの分散処理を行う．クラウドサーバは遠隔に配置されており，強力な計算資源を提供する．コントローラは VEC サーバのリソース情報を管理する．

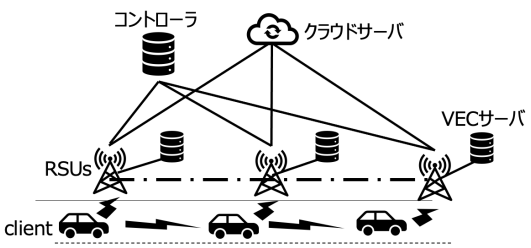

図 1 VEC のシステムモデル

2.2 VEC におけるタスクオフロード手法

VEC 環境におけるタスクオフロード手法の研究が進んでいる．

S.H.Song ら [4] は，無線リソース割り当てと計算リソース管理問題について考察している．車両と VEC サーバの最適な CPU サイクル周波数を得るために閉形式を用いるとともに，ガウスーザイデル法を用いることによって最適な伝送パワーと帯域幅割り当てを獲得する．

S.Maharjan ら [5] は，車両ネットワークと MEC サーバの統合に焦点を当てている．複数ユーザ・複数サーバでの VEC におけるタスクオフローディング問題を調査し，混合整数非線形計画問題として定式化し，2 つの問題に分割して考えることで，VEC サーバの選択とタスクオフロード決定を同時に最適化するアルゴリズムを提案している．

J.Zhang ら [2] は，VEC ネットワークにロードバランシングを導入することで特定の VEC サーバに負荷が集中することを回避する．集中管理によって車両やネットワーク・VEC サーバの情報を管理し，車車間通信と路車間通信を用いることでオフロード先を複数 VEC サーバに拡張することで，タスクを各 VEC サーバに均等に分散し，ロードバランシングを達成する．

以上の研究では，車両アプリケーションをオフロードすることによって車載計算資源の負荷を軽減する．しかし，既存手法ではアプリケーションの分割を考慮せず，アプリケーションベースのオフロードを行うためオフロード対象の粒度が大きく，VEC サーバに負荷が集中して処理遅延が増加する可能性がある．そのため，既存の VEC におけるロードバランシングオフロード手法で考慮がなされていないアプリケーションの分割を考慮し，VEC サーバの負荷を軽減して処理遅延を小さくするオフロード手法が必要であると考えられる．

3. 提案手法

本章では，車両エッジコンピューティングにおけるアプリケーション分割オフロード手法を提案する．

3.1 タスクオフロード手法の概要

提案するタスクオフロード手法では，オフロード決定式に基づいて推定応答時間の合計が小さくなるようにオフロード先を決定し，各タスクを各 VEC サーバにオフロードして実行する．ここで，推定応答時間とはオフロード決定式において算出する各タスクの推定の応答時間である．

タスクオフロードの要求があると，各 VEC サーバにおいて推定応答時間を算出する．その結果を VEC サーバの 1 つが集約し，オフロード決定式に基づいてオフロード先を決定し，各タスクを各 VEC サーバで実行する．オフロード決定式では全てのオフロード先の内，タスクをオフロードした際の推定応答時間の合計が最小となるようにオ

フロード先を選択する．推定応答時間は全タスクの推定実行時間と推定転送時間の和で算出される．推定実行時間は各タスクを実行するのに必要なサイクル数と各 VEC サーバの CPU 動作周波数に基づき算出を行う．推定転送時間は各タスクにおける入力データサイズ，出力データサイズ，データ伝送レートに基づき算出を行う．

3.2 オフロードシステム構成

提案するオフロードシステムの構成についてを述べる．図 2 に本稿で提案するオフロードシステムの構成を示す．クライアント - VEC サーバの 2 層環境を想定している．クライアントは車両であり，タスクはクライアントに配置されている．VEC サーバは，VEC サーバ 1 と VEC サーバ 2 が配置されている．クライアント と VEC サーバは接続がある．クライアントからのリクエストが VEC サーバ 1 に送られることによってオフロード手順が開始され，どの VEC サーバへタスクをオフロードし，タスクを実行するかについては VEC サーバ 1 が最終的に決定する．決定結果はクライアントに送信されると，クライアントからタスクが実行される VEC サーバへ画像とタスクを転送する．各コンポーネントは異なる計算資源の間でのリクエストや応答，自計算資源の状態の情報を取得，自計算資源における各タスクの推定応答時間の算出を行う．以降では車両システム，VEC サーバにおける各コンポーネントの役割について述べる．

3.2.1 車両システム

クライアント車両は Run Manager，Client Task Manager で構成されており，処理したい画像を入力としてタスクをオフロードし，処理結果として人物を検出した座標を取得する．

- Run Manager
  アプリケーションをタスクに分割してオフロードし，実行を要求する．
- Client Task Manager
  Run Manager からタスク実行要求を受け取ると，要求に則してタスクを車載コンピュータ上で実行し処理結果の転送を行う．

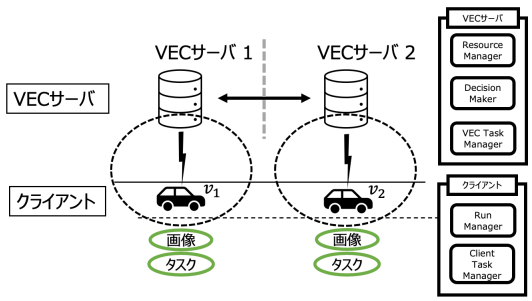

図 2 オフロードシステムの構成コンポーネント

3.2.2 VEC サーバ

VEC サーバは Decision Maker, Resource Manager, VEC Task Manager で構成されており，要求に則したタスク実行，推定応答時間を用いたオフロード先の決定，処理結果の転送を行う．

- Decision Maker
  オフロード先の決定を行う．Run Manager からオフロード決定の要求を受け取ると，リソース情報を Resource Manager に要求する． Resource Manager からリソース情報を受け取ると，受け取ったリソース情報を用いて推定応答時間を計算する．そして，リソース情報を基にオフロード決定を行い，決定結果を Run Manager に転送する．
- Resource Manager
  Decision Maker からリソース情報の要求を受け取ると，Decision Maker にリソース情報を渡す．
- VEC Task Manager
  Run Manager からタスク実行要求を受け取ると，要求に則してタスクを VEC サーバ上で実行し処理結果の転送を行う．

3.3 オフロード決定式

各タスクをオフロードして実行した際の推定応答時間の合計が最小となるオフロード先 VEC サーバ決定式を式 (1) に示す．また，オフロードの制約を式 (2) に示す．

$$min_{\lambda_i} \sum_{i=1}^{N} t_i(\lambda_i) \tag{1}$$

$$\lambda_i \in \{0, 1, 2\}, \forall i \in N \tag{2}$$

$t_i(\lambda_i)$ は λ_i にタスク i をオフロードした時の推定応答時間である．λ_i はタスク i のオフロード先を示しており，0 の場合はクライアント，1 の場合は VEC サーバ 1，2 の場合は VEC サーバ 2 にそれぞれオフロードすることを示す。i はタスク数であり，最大数を N で示す．t_i はタスク i の推定応答時間を示す．

オフロード決定式に用いる推定応答時間は式 (3) に示すように，データ転送に掛かる推定転送時間とアプリケーションの推定実行時間に基づいて算出を行う．

$$t_i(\lambda_i) = \frac{c_i}{f^{VEC}} + \frac{d}{r_{\lambda_i,\lambda_{i+1}}} \tag{3}$$

式において，c_i はタスク i の実行に必要な計算リソース，f^{VEC} は VEC サーバの CPU サイクル周波数， d は出力データサイズ，そして $r_{\lambda_i,\lambda_{i+1}}$ は λ_i から λ_{i+1} へのデータ転送レートをそれぞれ示す．

各計算資源における CPU 動作周波数を用いることで，すでに負荷が高い VEC サーバへとオフロードを行うことを回避し，データサイズ，データ転送レートを用いることで，転送に時間がかかる VEC サーバへのオフロードを回避する．

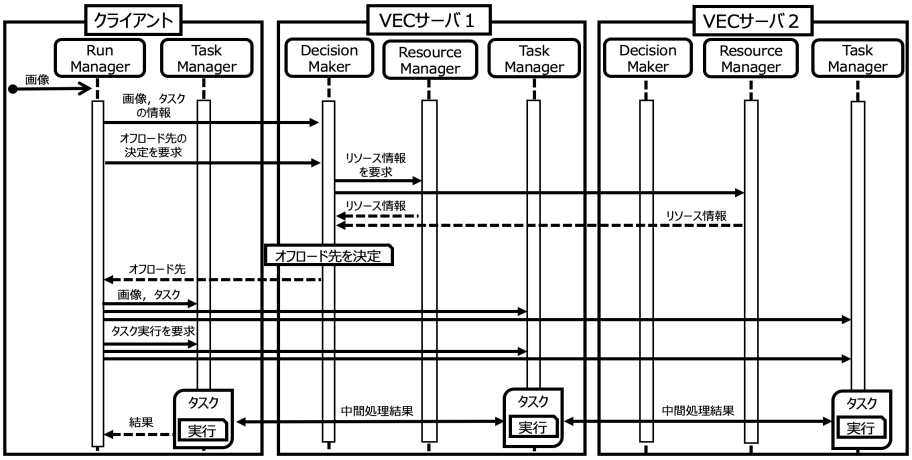

図 3 提案機構におけるタスクオフロード手順

3.4 タスクオフロード手順

提案手法のタスクオフロード手順について述べる．図 3 はタスクオフロード手順を示している．

(1) クライアントにおいて，画像とタスクの実行要求が Run manager に渡される．

(2) Run manager が実行要求を受け取ると VEC サーバ 1 の Decision Maker にタスク情報を送信するとともにオフロード先の決定を要求する．

(3) Decision Maker はオフロード先の決定を要求されると各 VEC サーバの Resource Manager にリソース情報を要求する．

(4) Resource Manager はリソース情報の要求を受け取ると，リソース情報を取得して Decision Maker に転送する．

(5) Decision Maker は受け取ったリソース情報を用いて推定応答時間を算出する．そして，推定応答時間を基にオフロード先の決定を行い，決定結果をクライアントの Run manager に転送する．

(6) Run manager はオフロード先の決定結果を受け取ると、決定結果に基づいて画像とタスクを各 Task Manager に転送するとともに、実行を要求する．

(7) 各 Task Manager で実行が行われ、最終的に処理結果として人物を検出した座標がクライアントに転送され，格納される．

4. 実験・評価

本章ではプロトタイプシステムを用いた実験による動作確認と評価について述べる．

4.1 実験環境

図 4 に実装環境を示す．今回，実験では 2 台のコンピュータを用いて，それぞれを VEC サーバと想定した．車両は仮想マシンとして実装し，今回は VEC サーバ 1 上で実行した．そして，図 4 のように有線接続した．実環境における遅延の影響を想定し，tc コマンド [6] を用いてクライアントと VEC サーバの間の帯域を制限した．表 1 に使用機材の性能を示す．また使用した OS は Ubuntu 20.04.1 LTS である．

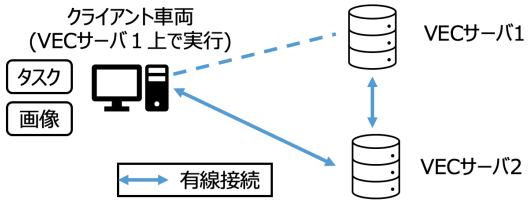

図 4 実験環境

表 1 使用機材

	CPU	Memory
client	VEC サーバ 1 上の仮想マシン (1 CPU)	1 GB
VEC 1	Intel(R) Core(TM) i7 - 6700 K CPU @ 4.00 GHz	8 GB
VEC 2	AMD Ryzen 3 3200 G with Radeon Vega Graphics	8 GB

実装するオフロードシステムにおいて各コンポーネントの実装には Python 3.8.5 を用いた．データの転送は Socket による UDP コネクションを用いて行われた．また，各サーバは有線で接続されており，サーバ間のルーティングテーブルは事前に設定されているものとした．また，実行の自動化のためにシェルスクリプトを用いた．

実行するアプリケーションは例として人物検出アプリケーションを用いた．今回実装する人物検出アプリケーションは Python で記述した．また，人物検出には OpenCV [7] ライブラリに含まれている HoG 特徴量とサポートベクタマシン (SVM) [8] を用いる．初めに人物画像を受け取ると，処理可能な形式に変換して取得する．次に，取得した画像の HoG 特徴量を計算する．その後，得られた HoG 特徴量を SVM によって識別することで人物を検出し，人物座標を取得する．最後に，取得された人物座標を結果としてクライアントに返す．それぞれの動作を画像取得タスク，特徴量計算タスク，特徴量取得タスク，結果取得タスクの最大 4 つに分割し，それぞれを別々に実装した．

4.2 実験パラメータ

それぞれのタスクを実行するのに必要なサイクル数を lscpu [9] を用いて測定した．また，クライアント，VEC サーバの CPU 動作周波数をオフロードシステムの実行直前に perf [10] を用いて測定し，クライアント - VEC サーバ間や VEC サーバ同士の間の帯域幅をオフロードシステムの実行前の数十秒間において Iperf 3 [11] を用いて測定した．それぞれの実測値を実験パラメータとして利用した．実験パラメータを表 2 に示す．今回，実環境における遅延の影響を想定した実験を行うために，cpulimit [12] を用いることで人物検出アプリケーションの実行に使用可能な CPU 使用率を任意の値に制限し，クライアントや VEC サーバに負荷が掛かった状態を再現する．評価では既存手法 [2] との比較を行なった．タスクごとの処理時間のばらつきを示すために，VEC サーバ 1 でタスクを実行した際の実行時間を表 3 に示す．

表 2 実験パラメータ

画像データサイズ	1.25[Mbit]
CPU 動作周波数 (VEC サーバ)	1600 - 3500[MHz]
CPU 動作周波数 (クライアント)	480 - 1050[MHz]

表 3 タスクごとの処理時間のばらつき

計算資源	処理タスク [msec]			
	画像読込	特徴量計算	特徴量抽出	結果取得
VEC サーバ 1	310.25	0.46	127.91	23.30

4.3 評価結果

図 5 にクライアント - VEC サーバ間の帯域制限を行わず，各計算資源における使用可能 CPU 使用率をクライアントでは 25%，VEC サーバ 1 では 100%，VEC サーバ 2 では 100% とした場合の応答時間を示す．応答時間は人物検出アプリケーションの実行時間と，タスクやデータの転送時間の和である．図 5 は，実行を 10 回行なった時の平均の応答時間である．提案手法で実行した場合，既存手法で実行した時よりも応答時間が小さくなった．

図 6 にクライアント - VEC サーバ間の帯域を 200Mbps に制限し，各計算資源における使用可能 CPU 使用率をクライアントでは 25%，VEC サーバ 1 では 100%，VEC サーバ 2 では 100%とした場合の応答時間を示す．図 6 は提案手法の応答時間および既存手法の応答時間である．応答時間は人物検出アプリケーションを実行するのに掛かった実行時間と，タスク転送やデータ転送に掛かった転送時間が含まれている．図 6 に示した応答時間は，実行を 10 回行なった時の平均の応答時間を示している．提案手法で実行した場合，既存手法で実行した時よりも応答時間が小さくなった．

以上の評価より，本稿で提案したオフロードシステムによって，オフロード決定式に基づき，推定応答時間が最も小さくなるオフロードにおいてタスクが実行されることを確認した．また，提案手法で実行した場合，既存手法で実行した時よりも応答時間が小さくなった．これは，既存手法で行なったアプリケーションベースのオフロードよりも提案手法で行なったアプリケーションをタスクに分割したオフロードの方が各オフロードにおける負荷が小さくなったためであると考えられる．

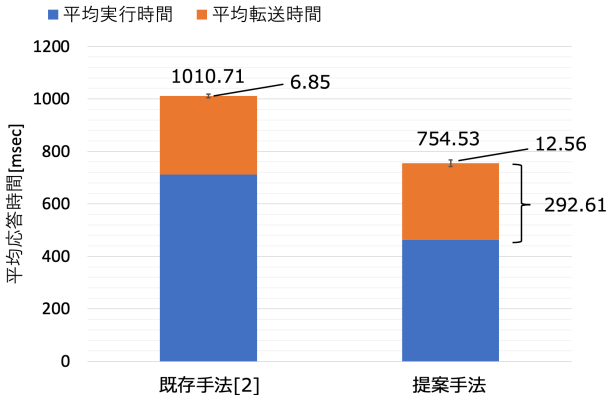

図 5 使用可能 CPU 使用率をクライアントでは 25%，VEC サーバ 1 では 100%，VEC サーバ 2 では 100%とした時における平均応答時間

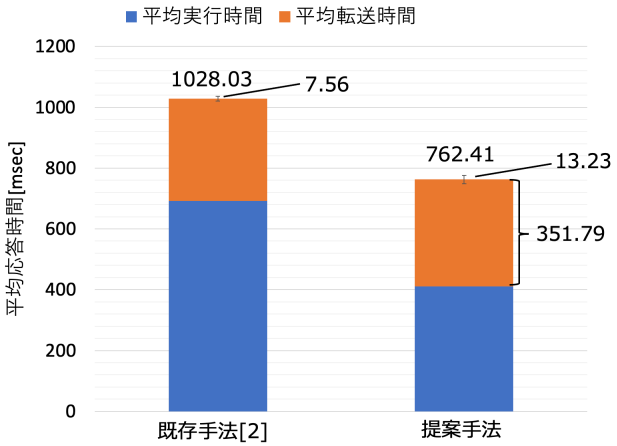

図 6 クライアント - VEC サーバ間の帯域制限時の使用可能 CPU 使用率をクライアントでは 25%，VEC サーバ 1 では 100%，VEC サーバ 2 では 100%とした時における平均応答時間

5. おわりに

本稿では，車両エッジコンピューティングにおけるアプリケーションの応答時間の減少を目的として車両エッジコンピューティングにおけるアプリケーション分割オフロードシステムの実装を行った．本稿で提案したタスクオフロード手法では，VEC サーバの 1 つがシステムを制御し，各計算資源はタスク完了のために要求される計算能力に基づいて推定応答時間を算出する．そして，推定応答時間が最小となるようにオフロードを決定し，実行する．これにより，オフロード対象の粒度の大きさを考慮したオフロー

ド手法を実現した．

また，本稿で提案したタスクオフロード手法を実装して実験を行い，動作確認と評価を行なった．実験では 2 台のコンピュータを用いてクライアント - VEC サーバの 2 層環境を構成した．実験により，既存手法と比較して応答時間が減少していることを確認した．

以上より，本稿で提案したタスクオフロード手法は車両エッジコンピューティングにおけるアプリケーションの応答時間を減少させるオフロード手法であることを示した．

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] C. M. Huang, M.S. Chiang, D. T. Dao, W. L. Su, S. Xu, and H. Zhou, ”V2V data offloading for cellular network based on the software defined network (SDN) inside mobile edge computing (MEC) architecture, ” IEEEAccess, vol. 6, pp.17741-17755, 2018.

[2] J. Zhang, Hongzhi Guo, Jiajia Liu, and Yanning Zhang, ”Task offloading in vehic- ular edge computing networks: A load-balancing solution,” IEEE Transactions on Vehicular Technology., vol.69, no.2, pp.2092-2104, Feb. 2020.

[3] S. Leng Y. He K. Zhang, Y. Mao and Y. Zhang. ”Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading, ” IEEE Vehicular Technology Magazine, 12(2):36-44, 2017.

[4] S. H. Song Y. Mao, J. Zhang and K. B. Letaief. ”Stochastic joint radio and compu- tational resource management for multi-user mobile-edge computing systems, ” IEEE Trans. Wireless Commun., 16(9):5994-6009, 2017.

[5] S. Maharjan Y. Dai, D. Xu and Y. Zhang. ”Joint load balancing and offloading in vehicular edge computing and networks, ” IEEE Internet Things J., 6(3):4377-4387, 2019.

[6] Manpage maintained by bert hubert (ahu@ds9a.nl). ”tc - show / manipu- late traffic control settings”. [Online]. Available: http://man7.org/linux/man-pages/man8/tc.8.html, [Online]; accessed 26-Jan-2019.

[7] Intel Corporation. ”opencv (open source computer vision library) ”. [Online]. Avail- able: https://opencv.org, [Online]; accessed 21-Jan-2021.

[8] Jianlong yuan. Hog-svm-python. [Online]. Available: https://github.com/jianlong- yuan/HOG-SVM-python, [Online]; accessed 21-Jan-2021.

[9] Manpage maintained by bert hubert (ahu@ds9a.nl). ”lscpu - display information about the cpu architecture”. [Online]. Available: https://man7.org/linux/man-pages/man1/lscpu.1.html, [Online]; accessed 20-Jan-2021.

[10] Vince Weaver. ”linux perf event features and overhead ”. 2013 fastpath workshop ”. [Online]. Available: http://web.eece.maine.edu/ vweaver/projects/perf events/overhead/fastpath2013 perfevent slides. [Online]; accessed 20-Jan-2021.

[11] Iperf.fr. ”iperf - the tcp/udp bandwidth measurement tool”. [Online]. Available: https://iperf.fr/, [Online]; accessed 20-Jan-2021.

[12] Angelo Marletta. ”cpulimit”. [Online]. Available: https://github.com/opsengine/cpulimit, [Online]; accessed 23-Jan-2021.