Vol.2017-OS-141 No.11
2017/7/27

隊列走行可能な交通システムの実稼働可能なシミュレータの開発

秋谷龍太朗[†1]　加藤和彦[†2]　阿部洋丈[†2]　長谷部浩二[†2]

概要：隊列走行可能な新交通システムの実稼働可能なシミュレータをコンテナ型仮想化技術を用いることで開発した．ここで言う新交通システムは，電子連結によって隊列走行が可能であり，路線網の分岐点で隊列を再編成することで，旅客が路線を乗り換えずに目的地に到着できる．また，本交通システムは旅客から乗降需用をリアルタイムで中央サーバに収集し，それらを基に，運行スケジュールを最適化する．中央サーバに障害が発生した場合でも，運行を継続的に行えるように，分散管理型の制御方式を採用し，中央サーバ以外のサーバに動作を切り替え，運行を継続する．我々は，中央サーバが通常動作時と障害発生時の運行管理の詳細設計・開発を行った．そして，シミュレータの動作確認をし，運行管理方式の有効性を確認した．

キーワード：交通システム，コンテナ型仮想化，デプロイ, DevOps

1. はじめに

超高齢社会への突入や障害者の社会進出に伴い，身体的困難がある人々が交通機関を利用することが増えている．2006 年末に施工された「高齢者，障害者等の移動等円滑化の促進に関する法律」（通称「バリアフリー新法」）によって，鉄道に代表される基幹的な公共交通機関や大規模施設へ移動の利便性は高まった．しかし，それらをつなぐ端末交通機関の整備はまだされていない．そのため，身体的困難がある高齢者や障害者などの旅客にも使いやすい端末交通機関が求められている．

すでに海外や日本において，端末交通システムは実用や実験的運用がなされている[1][2][7]．しかし，端末交通機関の停留所は各所に分布しているため，従来の交通機関や運行システムを流用することは難しい．端末交通機関の停留所を接続するためには,路線を増やさなければならない．しかし，近年の公共交通機関の運転手の減少傾向や過疎地域では採算が合わないことが理由に全ての地域で運行することは困難である．また，旅客の乗降需用のある全ての停留所に停車しなければならない問題が生じる．

それらの問題を解決するため，隊列を編成して走行可能な電子連結車両を用いたデマンド型端末交通システムが検討されている[10]．電子連結車両は，前後の車両と情報を共有して隊列を編成することのできる車両である．電子連結車両には，運転手が操縦する先導車と自動運転で先導車を追従する後続車がある．先導車は複数台の後続車両を接続して隊列を編成することができる．その隊列を路線網の分岐点で再編成することで，旅客は車両の乗換をすることがなく，目的地へ到着することができる．乗換をすることがないため車両への乗降動作は最小限になり，身体の不自由な高齢者や障害者の負担が軽減される．また，本交通システムは，リアルタイムで旅客の乗降需用を収集し，運行計画を最適化するデマンド型の交通システムである．旅客の需用のある停留所のみに停車することができるので，停留所の数を増やすことや過疎地域での運行が可能である．

今後，超高齢社会やそれに伴う仮想地域の拡大による移動弱者や買い物弱者は増加することが予想されるため，運行可能な電子連結車両を用いた端末交通システムの開発が求められている．交通システムは運行が停止することが許されないため，耐故障性を備えなければならい．そのようなシステムは複雑であり，また，シミュレーションと実地試験を繰り返す可能性がある．そのため，システムソフトウェアは肥大化や複雑化し，開発費や開発期間が増加する可能性がある．そこで，本交通システムの開発に，短い期間でシステムを開発するためのアジャイル開発や開発環境と運用環境の壁を取り払うDevOpsと呼ばれる概念がある．これらの概念を取り入れることで，開発環境であるシミュレータと運用環境である交通システムの実稼働の移行が円滑に行えるようになり，開発期間の短縮することが可能である．

本研究では，電子連結車両を用いた公共交通システムの運行制御方式をアジャイル開発やDevOpsなどソフトウェア開発の考えを取り入れ，詳細設計・開発をした．コンテナ型仮想化ソフトウェアDocker[3]を使用し，交通システムの運用環境への移行を円滑に行えるシミュレータの設計・開発を行った．また，機能をコンテナと呼ばれる単位に分割し，開発することでシステムの複雑さを緩和している．我々は，このシミュレータをデプロイアブル・シミュレータと呼んでいる．デプロイアブル・シミュレータは，単一のマシン上で分散システムを再現するシミュレータの役割をし，実際に分散システムへのデプロイが可能なシミュレータである．そのため，開発環境から運用環境への迅速な切り替えが可能となり，研究・開発速度が向上することが期待できる．

本論文の構成は以下の通りである．2 章で電子連結車両を用いた交通システムの概要とその制御方式について述べ

†1 筑波大学大学院システム情報工学研究科 コンピュータ・サイエンス専攻
†2 筑波大学システム情報系

る．3 章では，シミュレータの設計方針とその構成について，また，デプロイの手順について述べる．4 章では，関連研究について述べ，最後に 5 章で結論と今後の課題について述べる．

2. 交通システム

2.1 電子連結車両

電子連結車両は，隊列を成して走行することが可能な車両である．電子連結車両の隊列は，物理的な接触を持たない．電子的な制御によって車両感で情報を共有することで隊列を編成する（図 1）．電子連結車両には，先導車と後続車の２種類の車両がある．先導車は運転手が操縦することで走行する．後続車は自動運転により先導車を追従する．各種の電子連結車両は，行先を示す行先票を備えている．先導車は，後続車を予め設定した数だけ接続し，隊列を成すことができる．隊列は路線の分岐点である駅でのみ，再編成することが可能である．旅客が分岐点にて乗換動作をする代わりに隊列が再編成を行うため，旅客は目的地の行先票を提示している車両に乗ることで目的地に乗換動作なしで到着することが可能である．

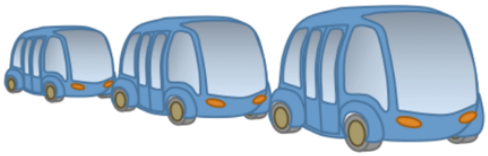

図 1　電子連結車両の概念図

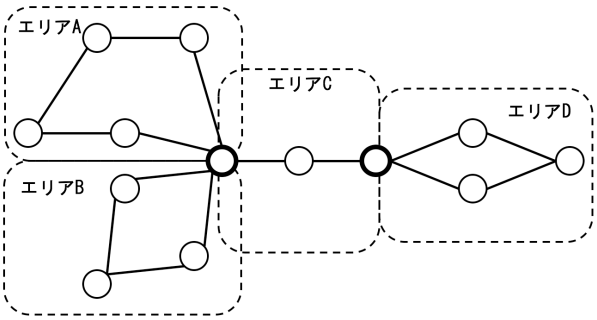

図 2　路線網の例

2.2 路線網

本交通機関の路線網は，旅客が乗り降りする停留所とそれらを結ぶ静的な運行経路から成る．路線網上の停留所は，地理的な要因によりエリアに分類される．停留所は一つ以上のエリアに必ず属す．そして，二つ以上のエリアに属す停留所を結節点と呼ぶ．電子連結車両の隊列は，結節点で行先票を変更や隊列の再編成を行う．結節点には車両プールがあり，再編成時に先導車に接続されなかった後続車を停車しておくことが可能である．

図 2 は，路線網の例である．図中の丸が停留所であり，それをつなぐ線が運行経路である．図の中央あたりにある太丸は結節点を表す．エリア A，エリア B，エリア C，エリア D の 4 つのエリアが存在する．中央左にある結節点は，エリア A，エリア B，エリア C に属しているが．このように 3 つ以上のエリアに属する結節点も存在する．また，エリア C のように結節点が 2 つ以上あるエリアがあっても構わない．

2.3 運行制御

本交通システムは旅客の乗降需用を収集し，それを用いて運行を最適化するフルデマンド型のシステムである．旅客は，スマートフォンなどの通信機器を用いて，乗降需用を本交通システムの中央サーバへ送る．中央サーバは，電子連結車両の（1）編隊，（2）運行経路，（3）行先票を旅客の乗降需用により決定する命令を発行する．複数の命令を組み合わせ，スケジュールとして扱う．

デマンド型の交通システムでは，中央サーバで情報を一元管理し，そこから各車両に指示を出す中央集中型の制御方式を採用することが多い．しかし，本交通システムでは，複数台のサーバを用いる分散管理型の制御方式を採用する．その理由は，以下の２つである．

- 中央サーバが停止した場合，交通機関全体が運行を停止する．
- 携帯回線を使用したサーバ・車両間通信のパケット通信コストが運用コストの 30%以上を占める可能性がある[8].

各結節点にサーバが置き，これを結節点サーバと呼ぶ．中央サーバが発行するスケジュールは，結節点サーバを経由して車両へと配信される．また，結節点サーバは，車両の発車時刻・出発時刻などの情報を中央サーバへと送信する．中央サーバはこの情報から車両の位置を推測する．

中央サーバが何らかの原因で故障した場合は，結節点サーバが予め決められた運行スケジュールに従って運行することで，交通機関が完全に停止することを避ける．

2.4 スケジュール

複数の命令を組み合わせることでスケジュールを表現することが可能である．電子連結車両を制御するために必要な命令セットを表 1 に示す．また，スケジュールの例を図 3 に示す．全ての先導車と後続車には，それらを一意に識別できる識別子を付け，命令内では，識別子を使い，車両

表 1　命令セット

命令	説明
SetLabel(V, A)	車両 V の行先票を A に変更する.
Connect(V)	先導車に後続車 V を接続する.
Disconnect(V)	隊列内の後続車 V の接続を解除する
GoTo(BS)	停留所 BS へと向かう.

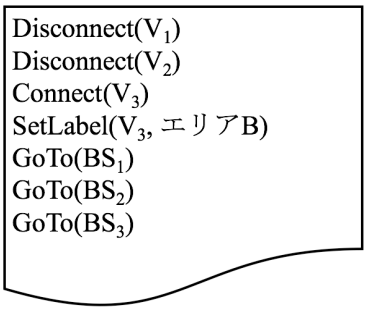

図 3　車両識別子を用いた車両のスケジュール

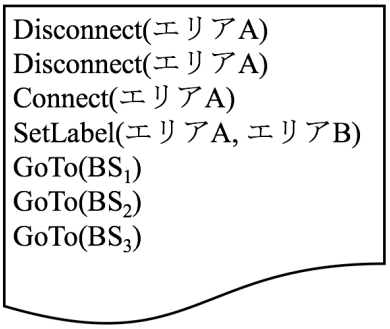

図 4　行先票を用いた車両のスケジュール

を指定する．しかし，現実の運行は車両の故障や交通渋滞などが存在する．この影響により車両が結節点に到着することが遅れる．このような事態が起こると，スケジュール内の命令の実行の完了が遅れる．また，車両が故障する可能性もある．車両が故障した場合は，その車両の代車が投入される可能性もある．しかし，故障した車両の識別子と代車の識別子は違うため，スケジュールは実行不可能である．そこで，中央サーバから発行されるスケジュールは，車両の識別子の代わりに車両の行先票で車両を指定する．そして，行先票により車両を指定したスケジュールを結節点サーバが受け取り，識別子により車両を指定したスケジュールに変換する．変換時に，車両プール内に停車している車両を選ぶことで，交通渋滞などで遅延している車両以外を指定することが可能である．スケジュールを実行する先導車もまた，同様に行先票で指定する．結節点に到着した時点で，自身の行先票へのスケジュールを受け取る．

2.5 スケジュールの変換の具体例

図 4 は，行先票で車両を指定したスケジュールの例である．これを図 3 の識別子で車両を指定したスケジュールに変換するまでの流れを示す．

次の前提を設ける. スケジュールを受け取る隊列内には，少なくともエリア A を行先票に持つ後続車 V_1 と後続車 V_2 がいる．また，結節点には，エリア A を行先票に持つ後続車 V_3 が停車中である．隊列内と結節点には，それ以外の車両が存在しても構わない．

隊列が結節点に到着すると，隊列を構成する車両の情報を結節点サーバへと送信する．その情報から結節点サーバは，接続を解除する後続車を決める．該当する後続車が複数ある場合は，隊列に接続されている時間が長いものから接続を解除するなどの方法で車両を絞る．そして，それぞれの Disconnect 命令内のエリア A をそれぞれ識別子 V_1 と V_2 に変更する．次に Connect 命令について行う．結節点車両プール内に停車中の後続車の中からエリア A を行先票に持つ車両を選ぶ. ここで, 該当する車両が複数ある場合は，待ち時間の長いものを選ぶ．そして，Connect 命令内の行先票を選ばれた V_3 に変更する．次に，SetLabel 命令内の車両を識別子によって指定する．Disconnect 命令と Connect 命令を終えた後の隊列の中から，行先票がエリア A である車両を選ぶ．ここでは，接続された V_3 の行先票がエリア A なので，V_3 を選ぶ．以上で行先票によって車両を指定したスケジュールの変換が完了した．

3. システム構成

3.1 設計方針

ソフトウェアの性能の計測や動作確認をするために，シミュレータを開発環境で作成して動作させる．そして，良い結果が得られた場合は，本番環境で動作させる事があるだろう．しかし，シミュレータに変更を加えることなく，本番環境で動くことは少ない．開発環境から本番環境に移行する場合，それらの環境の違いやソフトウェアに求められる機能の違いから，再実装やインフラストラクチャ（ハードウェア，OS，ミドルウェア，以下インフラ）の再整備が必要となる．実際のソフトウェア開発の現場でも，開発環境であるソフトウェア開発者のパソコンと本番環境であるサーバの間でインフラの違いからソフトウェアが動作しないことがある．そのようなことを避けるために，DevOps というソフトウェア開発手法がある．DevOps は，ソフトウェア開発者（Development）と保守・運用担当者（Operation）の意識ギャップをなくして開発と運用の一体化を目指し，生産性を向上させることが目的である[9]．本研究においても DevOps を取り入れ，シミュレーションから実地試験への円滑な移行を可能にするデプロイアブル・シミュレータ

の開発に取り組んだ．

DevOps を可能にする技術の一つにコンテナ型仮想化がある．コンテナ型仮想化とは，ソフトウェアの実行環境をコンテナと呼ばれる論理的な区画で隔離する技術である．それによって，開発環境内に本番環境のインフラ構成を再現したコンテナを作り，本番環境と同じ環境で開発に臨むことが可能になる．コンテナ型仮想化は，ホスト OS 上にコンテナ仮想化ソフトウェアを起動して使用する．コンテナの中でゲスト OS を実行してソフトウェアが実行する．ゲスト OS は，自身のカーネルを持たずにホスト OS のカーネルを用いて，ゲスト OS を模倣する．そのため，ホスト型仮想化やハイパーバイザー型仮想化などの他の仮想化技術に比べて，実行時オーバーヘッドが少ないというメリットがある．

Docker は，コンテナ型仮想化ソフトウェアのである（図 5）. Docker には (1) Docker イメージを作る機能，(2) Docker コンテナを動かす機能，(3) Docker イメージを公開・共有する機能の 3 つの機能がある. Docker イメージは，ルート・ファイルシステムに対する変更のレイヤであり，コンテナの元になるものである．Docker はこのイメージを共有することで，別の環境へのソフトウェアの移行を簡単にする．また，Docker はクラスタを生成する機能が存在する．これらの機能により，中央サーバと結節点サーバ群によるクラスタシステムのデプロイが容易になると考えられる．

本システムを構成するにあたり，サービス指向アーキテクチャ（SOA）[6]を採用する．SOA は，ソフトウェアの機能をサービスと呼ばれる粒度にシステムを分解して実装し，それらをネットワークで接続し，一つのシステムを構築する手法である．SAO によって機能の追加，削除，編集を容易になると想定される．SAO における一つのサービスを一つの Docker コンテナに対応して交通システムにおける各サーバを構成した．

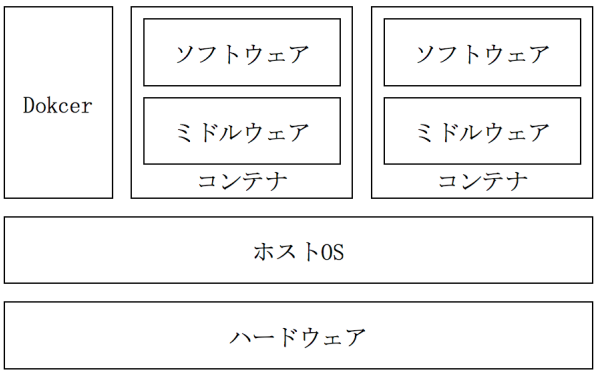

図 5　Docker によるコンテナ型仮想化

3.2 アーキテクチャ

本シミュレータのコンテナとそれらを接続する仮想ネットワークの構成を図 6 に示す．交通システムでは，結節点サーバや先導車，後続車は複数台存在するのが普通であるが，図中では便宜上，それぞれ 1 つずつで表現する．また，コンテナが接続されているネットワークは，全て仮想ネットワークである．ローカルネットワークと交通システム全体のネットワークの 2 種類のネットワークが存在する．各サーバや各車両内の点線で囲まれたコンテナを接続するネットワークは，ローカルネットワークである．また，図中中央のネットワークハブに接続されて構成されるものが交通システム全体のネットワークである．全てのコンテナはサーバであり，クライアントでもある．データの保存，コンテナ間でデータの共有をするためのデータベースとして Redis を用いる．Redis は，キーバリューストア型のインメモリデータベースである．

中央サーバは，以下の 4 つのコンテナから構成される．

- 制御コンテナ
- スケジューラコンテナ
- web コンテナ
- データベースコンテナ

制御コンテナは交通システムの交通システム全体の情報を受け取ってデータベースに保存したり，スケジュールを結節点サーバに送信したりする．スケジューラコンテナは，データベースに保存されている旅客の乗降需用や交通システムの情報からスケジュールを作成して制御コンテナへと渡す．web コンテナは，インターネット経由で旅客が本交通システムの情報を得たり，乗降需用を送信したりするためのコンテナである．

結節点サーバは，制御コンテナとデータベースコンテナの 2 つのコンテナから構成される．制御コンテナは中央サーバや先導車，後続車と情報をやり取りする．そして，結節点の車両プール内の車両の情報を保存する．

先導車は，制御コンテナ，ドライバーコンテナ，データベースコンテナの 3 つのコンテナから構成される．制御コンテナは結節点サーバや後続車との情報のやり取りをする．シミュレータ内では，ドライバーコンテナは，仮想的な先導車の運転手であり，隊列のスケジュールを実行に見せかけるだけである．しかし，本番環境に移行時に運転手への情報提供サービス用コンテナなどに置き換えることを想定している．

後続車は，制御コンテナとデータベースコンテナの 2 つのコンテナから構成される．制御コンテナは，先導車とのみ通信を行い，先導車への接続・接続解除や行先票の変更を行う．

各サーバを構成するコンテナと各車両の制御コンテナは本番環境への移行時に変更なく利用可能である．また，機能を増やすことは，コンテナを増やして制御コンテナを変

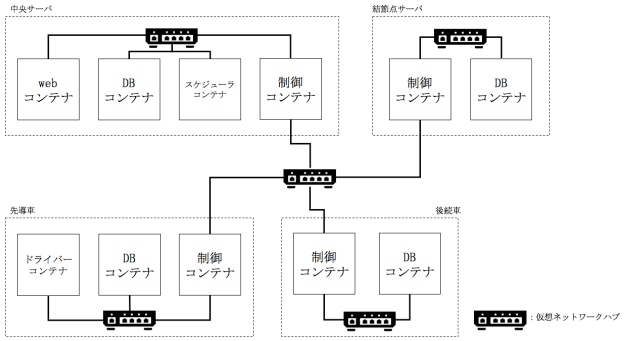

図 6　シミュレータ内のコンテナのアーキテクチャ

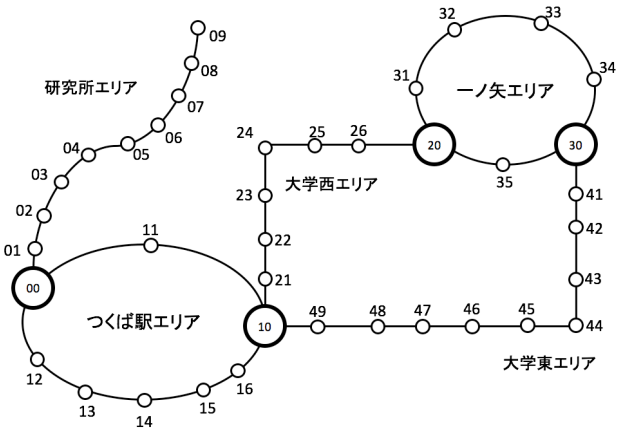

図 7　筑波大学周辺の路線網

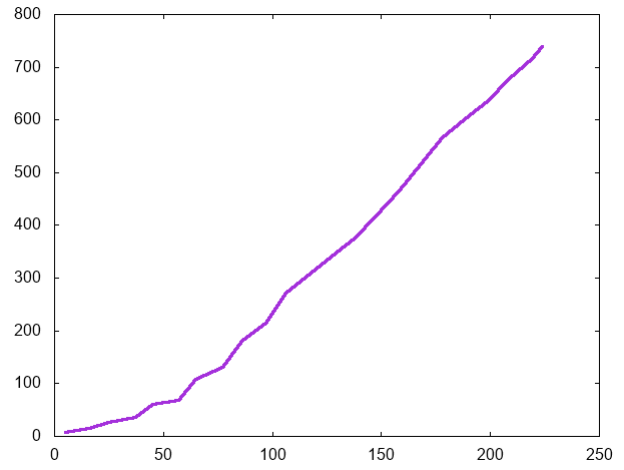

図 8　車両の総移動距離

更することで可能である．本番環境へ移行すると，車両は移動体であるため，結節点から離れるとネットワークは途切れる．しかし，仮想ネットワークはそのままに接続状態とし，次の結節点で結節点内の LAN に接続されて通信が可能な状態へと復帰することが可能である．

3.3 動作確認

中央サーバ 1 台と路線網に応じた数の結節点サーバ，そして，任意の台数の先導車と後続車のコンテナを作り，動作を確認したい．しかし，コンテナが多くなると手動でコンテナを起動することは，手間がかかり現実的ではない．そこで，簡単な路線網の情報と先導車，後続車の台数を設定すると，それに応じたコンテナを自動生成するスクリプトを作成した．そして，筑波大学周辺の路線バスの路線網をエリア分けし，本交通システムへと適用した（図 7）．路線バスの路線網を 5 つのエリアへと分割した．図中 10 番の停留所が主要駅であり，30 番の停留所が大学の中央の停留所である．

図 8 は，先導車と後続車の総移動距離の推移である．グラフは単調増加になっている．スケジュールの実行が行われて運行が継続していることが確認できる．

4. 関連研究

IoT や自動運転の研究が盛んに行われ，デマンドバスや自動運転を用いた新たな公共交通システムのデモ運行も各所で行われるようになった．函館市内では SAVS（Smart Access Vehicle System）[7]の実証実験が行われた．SAVS は，世界で初めてコンピュータシステムによる複数台完全リアルタイム配車サービスを実現した．最大 16 台の車両を用いて，旅客の乗降需用に対応している．また，自動運転による公共交通システムとしては，オランダ・ロッテルダム近郊を走る ParkShuttle[1]やフランス・ラロシェルの CityMobil2[2]がある．デマンド型配車システムの例として，また，自動運転のタクシーの配車システム[5][11]が考案されている．本交通システムのように電子連結車両を用いた交通機関として，Next Future Transportation[4]が挙げられる．本交通システムと違い，この交通システムは完全自動運転車両を用いたフルデマンド型の交通機関であり，走行中に公道で隊列の連結・連結解除が行われる．

5. 結論

本論文では，電子連結車両を用いた交通システムの制御システムの詳細設計を行った．また，ソフトウェア開発の手法である DevOps とサービス指向アーキテクチャを取り入れ，本番環境への円滑な移行や容易な機能追加が可能なデプロイアブル・シミュレータを，Docker を用いて設計・開発した．そして，動作確認を行い，運行管理方式の有効性を確認した．Docker を用いることで，シミュレータに必要なミドルウェアなどを本番環境で簡単にインストールすることが可能となる．また，コンテナ型仮想化による分散システムとして，シミュレータを作成することで本番環境

と同等のアーキテクチャを予め表現することができるようになった．今後，物理的に独立するコンピュータ上に分散システムを移行し, 移動体への Docker の適用が実際に可能であるかを検討する必要がある．

シミュレータ内では，中央サーバが正常に動作している場合の運行制御方式のみを実装した．中央サーバが故障した場合は運行が停止してしまう．中央サーバが故障した場合の耐故障性の確保があるが課題の一つである．また，交通システムの制御を実現した本研究では，スケジュールの最適化を行っていない．そのため，電子連結車両を用いた交通機関のスケジュールの最適化手法の開発がまだ行われていない．

謝辞 本研究の一部は、トヨタ自動車(株) と筑波大学との特別共同研究事業「高度アクセシブル社会実現に向けた研究基盤」の助成を受けて実施された。

参考文献

[1] “2get there”. http://www.2getthere.eu/projects/rivium-grt/, (参照 2017-6-25)
[2] “CityMobil2”. http://www.citymobil2.eu/en/, (参照 2017-6-25)
[3] “Docker – Build, Ship, and Run Any App, Anywhere”. https://www.docker.com/, (参照 2017-6-25)
[4] “Next Future Transportation inc.”. http://www.next-future-mobility.com/, (参照 2017-6-25)
[5] Albert, Y.S. L., Yiu-Wing L. and Xiaowen C. Autonomous Vehicle Public Transportation System: Scheduling and Admission Control. IEEE Transactions on Inteligent Transportation Systems, 2016, vol.17, no.5, p.1210-1226.
[6] Laskey, Kathryn B., and Kenneth Laskey. Service oriented architecture. Wiley Interdisciplinary Reviews, Computational Statistics, 2009, vol.1, p.101-105.
[7] 中島秀之, et al. バスとタクシーを融合した新しい公共交通サービスの概念とシステムの実装. 土木学会 D3（土木計画学）, 2015, vol.71, no.5, p.875-888.
[8] 大谷達彦. バスロケーションシステムの運用に関する検討. JICE report: Report of Japan Institute of Construction Engineering, 2006, vol.9, p.33-38.
[9] 榊原彰. 開発と運用の融合 −DevOps の波−. 日本アイビーエム, Provision, 2012, no.75, p.26-31.
[10] 長谷部浩二, 加藤和彦, 安倍洋丈, 川本雅之. 隊列走行可能な端末交通システムにおける旅客の輸送方式. 日本ソフトウェア科学会第 33 回大会, 2016.
[11] Yuqi Wang, Hui Qi. Research of Intelligent Transportation System Based on the Internet of Things Frame. Wireless Engineering and Technology, 2012, vol.3, no.3, p.160-166.