BLE のランダム MAC アドレスを用いた OD データ推定に向けた初期検討

Initial Study on OD Data Estimation Using BLE Random MAC Addresses

吉村 太斗 † 林 虎太朗 † 新井 イスマイル ‡ 松田 裕貴 †,*
Taito Yoshimura Kotaro Hayashi Ismail Arai Yuki Matsuda

1. はじめに

国土交通省の資料「地域の公共交通を取り巻く現状と検討の視点・課題」[1] によると，地域の路線バスの問題として，路線バスの利用者が減少していること，運転手の人材不足などが挙げられている．こうした現状から，路線バス事業者は，限られた労力と車両数で効率的な運行を行う必要がある．路線バスの効率的な運行のためには，利用実態の把握が不可欠である．利用実態の把握のためには，利用者ごとの乗車駅（Origin）と降車駅（Destination）の情報を集計したデータ（OD データ）が有効である．OD データを把握することで，利用者の多い区間では増便を行い，利用者の少ない区間では減便を行い，乗り換えを行う利用者が多い区間に乗り換えが不要な路線を追加したりすることで，利便性，採算性の向上を図ることができる．

従来，OD データの取得方法は調査員による乗客へのカード配布や，運転士又は添乗調査員による目視記録，IC カードによる取得手法が主流であった [2]．しかし，調査員によるカードの配布や目視での OD データの収集手法は人件費をかけすぎないために，調査できる日数に限りがあり，限定的な期間のデータしか集めることができず，長期間のデータを取集しようとするとコストがかかってしまう．また，IC カードを用いた手法は，IC カードの利用履歴を使用することで，利用客の OD データの把握ができるが，定額区間や現金支払いの利用者が含まれないといった問題がある. また他にも，カメラを用いた手法なども存在するが乗客の顔を撮影するためにプライバシーの問題が生じる．こうした，状況からプライバシーに配慮したコストがかからない OD データの取得手法が必要となる．

そこで本研究では，人々が持つスマートフォンなどの電子機器から発せられる BLE（Bluetooth Low Energy）のアドバタイズパケットをスキャンしそこから得られる情報をもとに路線バスの OD データの推定を行いデータ収集を目指す．

本稿では，BLE のアドバタイズパケットについての調査，iPhone15 Pro Max（iOS 17.0）の MAC アドレスの平均持続時間の調査，バスを想定した OD 推定実験を行った．その結果，ランダマイズされた BLE の MAC アドレスを用いたとしても OD 推定を実現できる可能性が示唆された．

本稿の構成は以下のとおりである．2 章では，OD データ推定に関する関連研究について述べる．3 章では，事前実験について述べる．4 章では，提案手法の概要を述べる．5 章では，まとめと今後の展望について述べる．

2. 関連研究

本章では，OD データ推定に関する関連研究，および Bluetooth デバイスの追跡に関する研究について述べる．

2.1 カメラを用いた OD データ推定

山田ら [3] は，路線バス内に設置した単一カメラを用いて高い精度で人物をトラッキングし OD データの推定を行っている．しかし，このようなカメラを用いたシステムは，プライバシー侵害の恐れがあるため容易に導入することはできない．

2.2 Bluetooth・WiFi を用いた OD データ推定

スマートフォンが発するパケットを用いて OD データを収集する様々な研究がなされてきた [4][5][6].

Ryu らは，WiFi パケットのみを用いて OD データの推定を行なっている [4]．Ziyuan らは，WiFi パケットと Bluetooth パケットを用いて OD データの推定を行っている [5]．これらの研究では，バス車内にセンシングデバイスを設置し，スマートフォンが発する WiFi と Bluetoth のパケットから MAC アドレスを取り出し，MAC アドレスの出現回数や平均 RSSI などをもとにバスの乗客以外の MAC アドレスをフィルタリングし，MAC アドレスの出現時間と消失時間を特定し，これらの時間と合致する出発時刻と発射時刻をもつバス停を特定し OD データを推定する．

しかし，これらの手法は現在のスマートフォンに用いることができない．現在のスマートフォンはプライバシー保護の観点から，乱数生成したアドレスであるランダム MAC アドレスが使用され，この MAC アドレスは定期的に書き換わるためである．

Kawashima らは，Google と Apple が開発した機能である Exposure Notification が発する BLE のアドバタイ

† 岡山大学，Okayama University
‡ 奈良先端科学技術大学院大学，
Nara Institute of Science and Technology
* 理化学研究所革新知能統合研究センター（AIP），
RIKEN Center for Advanced Intelligence Project

ズパケットを用いて OD データの推定を行なっている [6]. Exposure Notification とは，新型コロナウイルス感染症（COVID-19）の感染拡大を抑えるために開発された技術で，Bluetooth を使用して他のアプリユーザーとの接触を匿名で追跡するシステムである. MAC アドレスがランダム化されているデバイスを追跡するために，Exposure Notification に含まれる Rolling Proximity Identifer と RSSI を用いてバスの乗客の BLE デバイスを追跡し，OD データの推定を行っている．この手法は，バスの乗客が Exposure Notification を利用した COCOA（新型コロナウイルス接触確認アプリ）[7] などのアプリケーションをダウンロードしている必要があるが，新型コロナウイルス感染症の取り扱いの変化に伴い,2022 年 11 月に機能を停止していることから, この手法は利用できないと考えられる.

2.3 Bluetooth デバイスの追跡

MAC アドレスがランダム化されたスマートフォンを追跡する様々な研究がなされてきた．Becker らは，BLE のアドバタイズパケットに含まれる MAC アドレスとパケットのペイロードに含まれる識別子が同期されて変更されないことを利用し，MAC アドレスが変更されたときに，識別子をもとに新しい MAC アドレスと結びつける，アドレスキャリーオーバーアルゴリズム [8] を提案した．しかし，現在この脆弱性は改善されており，デバイスの追跡に利用することはできない.

秋山らは，MAC アドレスがランダム化されたスマートフォンを同定するために，アドバタイズパケットの受信時刻と RSSI の変化傾向を利用し回帰モデルを用いて，同一機器の推定を行っている [9]．高い精度で同定できているが，MAC アドレスが同じタイミングで変化すると精度が著しく低下する，研究室内で実験をおこなっているため実際の環境で同定できるかは不明といった問題が存在する.

2.4 本研究の位置づけ

バスの OD データ収集には様々な手法が存在するが，人件費が高コストであることやプライバシーの問題，ランダム MAC アドレスに対応していない手法や，ランダム MAC アドレスに対応した手法 [6] も存在するが，今後は使えなくなる可能性が高い.

MAC アドレスがランダム化された Bluetooth デバイスを追跡する手法 [8][9] は，現在使用できない，あるいは実際の環境で使用できるかは不明といった問題がある.

本研究では，以上の問題を解決するために BLE を用いて MAC アドレスのランダム化に対応した OD データの推定を目指す.

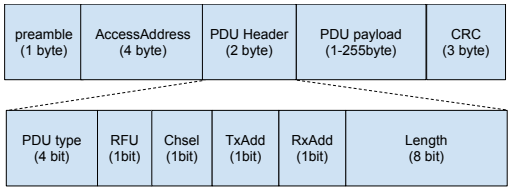

図 1: BLE アドバタイズパケットのフォーマット [10]

3. 事前実験

3.1 実験概要と使用デバイス

まず，電波暗箱を用いてどのようなデバイスがアドバタイズパケットを発信しているのかを調査を行った．次に，車を用いてバスを想定した実験を行い，OD データの推定が可能であるかを検証する実験を行った.

BLE アドバタイズパケット収集デバイス（以降，スキャナ）として, Bluetooth 4.0+EDR/LE Class1 対応 USB アダプタ（BUFFALO 社製，BSBT4D100）を取り付けた Raspberry Pi 4 Model B を使用した．Raspberry Pi 4 Model B は Bluetooth 通信モジュールを内蔵しているが, スペックを統一するため外付けの Bluetooth 通信モジュールを使用することとした．アドバタイズパケットの収集には，Python 用ライブラリの bluepy[*1] を使用した.

BLE のスキャンとは，BLE デバイスが周囲にある他の BLE デバイスを検出することを指し, その方法には，パッシブスキャンとアクティブスキャンの 2 種類が存在する．パッシブスキャンは，スキャナがアドバタイズパケットを受信可能な状態にしアドバタイズパケットを取得するスキャン方法である．アクティブスキャンは，スキャナがアドバタイズパケットを受信後に，アドバタイズしているデバイスにスキャン要求を行い追加情報を取得するスキャン方法である.

本研究で使用する bluepy については, どちらの方式も利用可能[*2] であるが，本稿での実験ではアクティブスキャンの方式を採用した.

3.2 BLE アドバタイズパケットのフォーマット

BLE のアドバタイズパケットの構造と詳細な内容を説明する．まず，アドバタイズパケットのフォーマットを図 1 に示す．プリアンブル，アクセスアドレス，PDU ヘッダー，PDU ペイロード，CRC が含まれている．図 1 に示すように，PDU ヘッダーのフォーマットはアドバタイジングやスキャン要求といったタイプを示す PDU タイプ，アドレスがランダムかランダムでないかを示す,

[*1] `https://github.com/IanHarvey/bluepy`

[*2] 公式ドキュメントには明記されていないが,scan 関数の引数に `passive=True` または `False` を与えることで, パッシブ・アクティブを変更することが可能である．なおデフォルトは `passive=False`（つまり, アクティブスキャン）が設定されている.

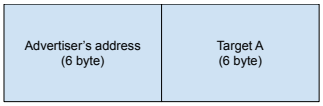

図 2: 特定のデバイスに向けたアドバタイズパケットの PDU ペイロード [10]

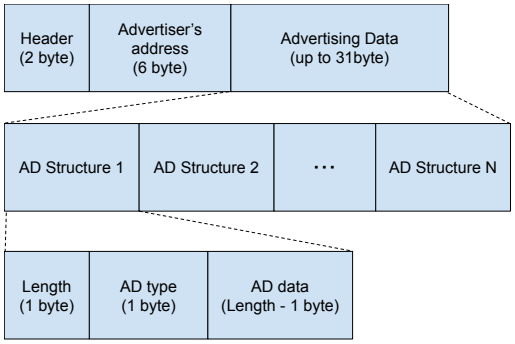

図 3: 不特定多数のデバイスに向けたアドバタイズパケットの PDU ペイロード [10]

TxAdd，RxAdd，PDU の長さを示す Length 等で構成される．

アドバタイジングには 2 種類あり，特定のデバイスに向けたアドバタイズと不特定多数に向けたアドバタイズが存在する．この 2 つは PDU ペイロードのフォーマットが異なる．特定のデバイスに向けたアドバタイズパケットの PDU ペイロードのフォーマットを図 2 に示す．図に示すように，Advertiser's address と Target Address を含むフォーマットである．Target Address と一致する MAC アドレスを持つデバイスのみが接続要求を送信することで接続を開始できる．不特定多数デバイスに向けたアドバタイズパケットの PDU ペイロードのフォーマットを図 3 に示す．図に示すように，Advertiser's address と Advertising Data を含み，Advertising Data には一つまたは，複数個の AD Structure が含まれている．

最後に AD Structure に含まれているアドバタイジングデータ（AD data）の説明をする．AD Structure のフォーマットを図 3 に示す．図に示すように，AD data と AD data の種別を表す AD type が含まれている．表 1 に主な AD type の一覧を示す．

表 1 に示すように，BLE のアドバタイズパケットに含まれる AD data には BLE デバイスの MAC アドレスを追跡する際の識別子として有効に働く可能性のある識別子が存在すると考えられる．実際に BLE のアドバタイズパケットを収集し，どのようなデバイスが BLE のアドバタイズパケットを発信しているのか，MAC アドレスの持続時間，AD data が MAC アドレスを追跡する際の識別子として有効であるのかを検証する．

表 1: AD type の主な例

AD type	名前	内容
0x01	Flags	デバイスがもつ発見や接続の機能を示す
0x0a	TX Power Level	値の単位は dBm で, 送信電力を表す
0xff	Manufacturer Specific Data	先頭 2byte に Bluetooth SIG が企業に発行した ManufactureID [11] そして任意長のバイナリ・データが続く

3.3 電波暗箱を用いた実験

どのようなデバイスが BLE のアドバタイズパケットを発信しているかを調査するために，数種類のデバイスを電波暗箱に入れ実験を行った．電波暗箱は，MICRONIX 社製の MY1510 を使用した．

3.3.1 調査対象のデバイス

本実験では，iPhone，Android スマートフォン，AppleWatch，MacBook を調査対象のデバイスとした．以下ではそれぞれのデバイスの詳細について述べる．

Android デバイス

実験を行った Android デバイスは 4 台（Google Pixel 6 Pro OS 12，Google Pixel 4 OS 10，Google Pixel 3a OS 10, Oppo A55s 5G OS 11）である．いずれのデバイスも Bluetooth を ON にし，スリープ状態でスキャナと共に電波暗箱に入れた．実験の結果，今回実験を行った Android デバイスはアドバタイズパケットを発信していないことが判明した．

iOS デバイス

実験を行った iOS デバイスは 5 台（15Pro max iOS 17.0，13 iOS 16.2，XR iOS 17.5.1，SE2 iOS 16.11，13 mini iOS 15.3.1）である．いずれのデバイスも Bluetooth を ON にし，スリープ状態でスキャナと共に電波暗箱に入れた．今回実験を行った iOS デバイスはいずれのアプリもダウンロードしていない新品の iOS デバイス（13 mini，iOS 15.3.1）以外は 1 〜 3 個のアドバタイズパケットを発信していることが判明した．

表 2 に，具体例として 3 台のデバイスのアドバタイズパケットの AD data を示す．Flags と TxPowerLevel はデバイスごとに固定でランダム MAC アドレスが変更されてもこれらに変化はなかった．また，Manufacture Specific Data はランダム MAC アドレスと同時に書き換わるが，上位 6 byte は変化しなかった．

Manufacture Specific Data の上位 2byte は ManufactureID で `0x004c` は Apple を表す (実際に取得できたデータでは `0x4c00` となっており，取得されるデータの

表 2: デバイスの Advertising Data

iPhone Model	パケット	MACアドレス	アドバタイジングデータ
15proMax (iOS 17.5.1)	パケット1	59:c5:18:7b:44:b8	Flags:0x1a TxpowerLevel:0x07 Manufacturer Spacific Data:0x4c0010062619333ebc78
	パケット2	fb:ab:b6:c2:bc:0f	Manufacturer Spacific Data:4c001219003ae04230f 977e5490bcdd8210ca88abc7a693ba528d00000
10R (iOS 15.4.1)	パケット1	4b:bd:c8:be:ca:5d	Flags:0x1a TxpowerLevel:0x0c Manufacturer Spacific Data:0x4c0010074b1f6da9f3d568
	パケット2	f4:aa:17:47:00:92	Manufacturer Spacific Data:0x4c0012020000
	パケット3	49:e0:ad:82:91:98	Flags:0x1a Manufacturer Spacific Data:0x4c00160800155bfaed63b792
13 (iOS 16.2)	パケット1	42:84:ae:ad:19:ed	Flags:0x1a TxpowerLevel:0x07 Manufacturer Spacific Data:0x4c0010063f1a4228ceb3
	パケット2	c5:74:8e:54:f8:a1	Manufacturer Spacific Data:0x4c0012020001

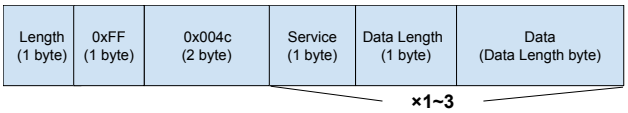

図 4: iPhone の Manufacture Specific Data の推定されるフォーマット

上位 2byte は前後の 1byte が入れ替わった状態で取得される)．この上位 2byte は BLE の仕様で定められているが，それ以降に続くデータについてどのように扱うかは企業に委ねられているためデバイスや企業ごとに異なるものとなる（非公開の場合は第三者は内容を知ることができない）．しかし，収集したパケットの Manufacture Specific Data を観察すると，iPhone の場合 Manufacture ID に続く 1 byte がサービスを表し，次に続く 1 byte がデータの長さである可能性が考えられた．これを iPhone が発する Manufacture Specific Data に当てはめてみると，図 4 のようになってることが確認できた．

実験を行った結果，新品の iOS デバイス以外は BLE のアドバタイズパケットを発信していて，アドバタイズパケット発信しているすべての iOS デバイスが，Manufacture Specific Data が `0x004c10` で始まるアドバタイズパケットを発信することが判明した．また，位置情報を無効化することで，Manufacture Specific Data が `0x004c12` から始まるパケットの発信が停止することが判明した．`0x004c10` から始まるパケットは Bluetooth を無効化しない限り発信されることが判明した．

AppleWatch と MacBook

AppleWatch2 台（Series 9，watchOS 10.3；Series 5. watchOS 10.3）と MacBook Pro1 台（macOS 14.5）についても同様の実験を行った．実験した MacBook はアドバタイズパケットを発信しておらず，AppleWatch は 2 台とも Manufactue Specific Data が `0x004c10` から始まるアドバタイズパケットを発信していた．

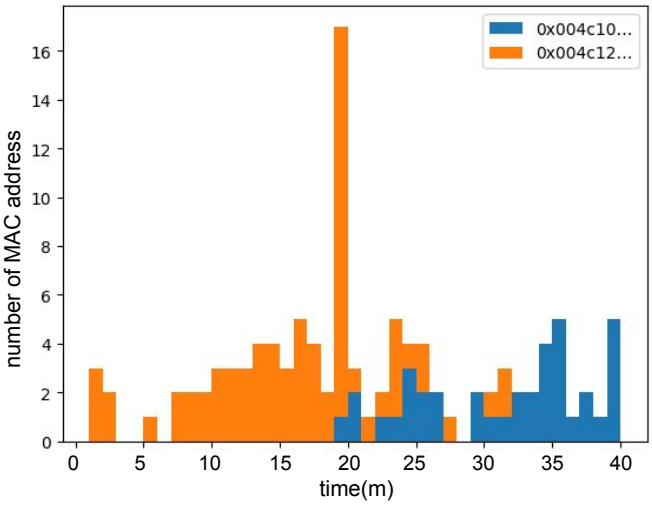

図 5: MAC アドレスの持続時間のヒストグラム

表 3: MAC アドレス持続時間の統計値

MSD	平均	最大	最小
0x004c10	31 分 37 秒	39 分 32 秒	19 分 58 秒
0x004c12	16 分 12 秒	31 分 51 秒	1 分 6 秒

3.3.2 MAC アドレスの継続時間の分析

今回実験対象としたデバイスは，Apple 社製 iPhone15 pro max とし，OS のバージョンは iOS 17.0 である．実験は研究室内で行い，実験対象のスマートフォン以外の電波を遮断するために，電波暗箱に実験対象のスマートフォン 1 台とスキャナ 1 台を入れた．約 20 時間電波暗箱に入れアドバタイズパケットを収集した．今回実験対象とした iPhone は，Manufacture Specific Data が `0x004c10`，`0x004c12` から始まる 2 つのパケットを発信していた．それぞれのパケットのアドレスの持続時間を調べた結果を（図 5）に示す．青色のグラフが Manufacture Specific Data が `0x004c10` の MAC アドレス，オレンジ色のグラフが Manufacture Specific Data が `0x004c12` のグラフを表す．また，MAC アドレス持続時間の統計値を表 3 に示す．

`0x004c10` と `0x004c12` から始まるパケットの MAC アドレス平均持続時間に差があることがわかり，比較的 `0x004c10` の方が持続時間が長いことが分かった．

実験結果から，iPhone と AppleWatch に関してはアドレスの持続時間が比較的長く，設定から Bluetooth を OFF にしない限り，新品を除く全てのデバイスがアドバタイズしていると考えられる，Manufacture Specific Data が `0x004c10` から始まるアドバタイズパケットの MAC アドレスを追跡することでデバイスの追跡は可能であると考えられる．Android デバイスは，スリープ状態ではアドバタイズパケットを発信しておらず追跡は不可能である．

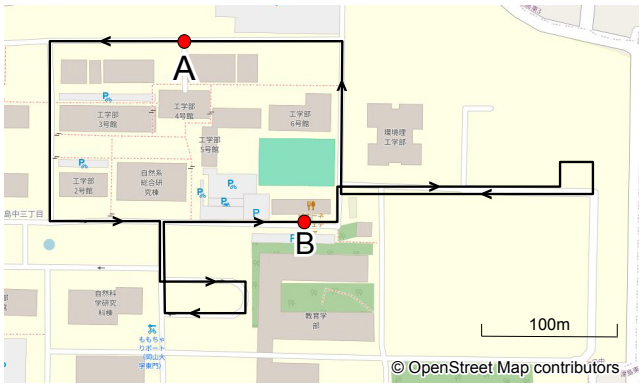

図 6: 周回コース

3.4　バスを想定した環境での実験

3.4.1　実験設定

岡山大学津島キャンパスの構内に図 6 のように設定した 1 週約 1.3km・約 5 分のルートをバスの運行コースと見立て，道中に停車駅 A・B を設置して実験を行った．始点・終点を A 地点とし，途中で A 地点・B 地点に停留しつつ，合計 6 周走行した．

BLE スキャンデバイスは，車の左前方ダッシュボード内に 1 台（スキャナ 1），右後方荷台に 1 台（スキャナ 2）を設置しデータを収集した．BLE のスキャン間隔は 1 秒とし，アドバタイズパケットの収集時刻，ランダム MAC アドレス，RSSI，AD data を記録した．使用デバイスは iPhone 7 台と AppleWatch 1 台を用い，図 7 に示すようなスケジュールで乗降車させた（内，ID-3 のデバイスについては Bluetooth を無効化した）．

3.4.2　データ解析

データ収集により得られたデータを可視化したものを図 8 に示す．図 8 は時間軸に沿って観測された MAC アドレスを表す．横軸は観測時間，縦軸は異なる MAC アドレス，横に並んだデータは同一の MAC アドレスであることを示している．さらに，デバイスの乗降車があった時間帯を青の縦破線で示している．このデータについてフィルタリングを施すことによって，乗車した 8 台のデバイスの検出を試みる．

まず，車内に存在するデバイスはスキャナ 1（車内前方），スキャナ 2（車内後方）の両方で観測できていると考えられることから，両方で観測できた MAC アドレスを抽出する．さらに，3.3.2 節の考察をもとに，Manufacture Specific Data が `0x004c10` から始まる MAC アドレスを抽出する．これを可視化すると図 9 に示すとおりとなった．車外のデバイスもスキャンしていることが想定されるため，それらを除外するために，スキャナ 1，2 での検知回数の合計が 100 回以上であること，という条件を設定した．最後に，本実験では同じコースを短い周期で周回するため，同じ場所に留まっているデバイスを定期的にスキャンすることが想定される．`0x004c10` から始まる MAC アドレスは毎秒スキャンできることが確認されているため，スキャン回数と（最終観測時刻－初回観測時刻）秒の値が 5 倍以上の差があるデバイスを除去した．最終的に得られた結果は図 10 に示すとおりである．

解析結果から同時に変化した MAC アドレスを紐づけることによるアドレスキャリーオーバーを手作業で行ったところ，図 10 に示す破線矢印のように紐づけが可能であることが分かった．例えば，最初から最後まで乗車した ID-1 と ID-7 のデバイスは MAC アドレスが，

`76:b4:..` ⇒ `76:d5...` ⇒ `4f:d1...`
または
`54:72:..` ⇒ `69:de...` ⇒ `47:11...`

と変化したもののいずれかであると推定される．いずれも iPhone であるが個体の識別については，現時点では有効な手がかりは見つかっていない．その他，途中下車して再乗車した ID-2 のデバイスについては，再乗車までの間に MAC アドレスの変化がなかったため，

`56:88:...` ⇒ `65:c8:...` ⇒ 降車
⇳ 一致
再乗車 ⇒ `65:c8:...` ⇒ `65:c8:...`

のように，同一デバイスとして追跡可能であった．一方で，同じく途中下車して再乗車した ID-5 については，再乗車するまでの間に MAC アドレスの変更が生じており，

`71:08:...` ⇒ 降車
⇳ 不一致
再乗車 ⇒ `5c:5d:...`

というように，追跡は不可能であるため，別のデバイスとして識別されるものと考えられる（ID-8 のデバイスについても同様）．なお，ID-3 は Bluethooth を無効化した状態としていたため，全体を通して観測できていない．

乗降車スケジュール（図 7）と分析結果（図 10）から，Bluethooth が有効化されているデバイスに関する OD 推定の実現可能性が示唆された．

3.4.3　考察

今回の実験では，乗降車（青の縦破線）の前後にも MAC アドレスが見えているサンプルが確認された．今回の実験において，駅に停車した際に分単位で記録を取っていたため，これらの MAC アドレスが降車した後もしばらく見えているのか，降車時間がずれているのか原因を特定することはできなかった．この点については，今後の検証が必要と考えられる．

また，アドレスの出現回数で車外のデバイスをフィルタリングする際，条件値を大きく設定しすぎると，バス

		1周目		2周目		3周目		4周目		5周目		6周目		終点
		A	B	A	B	A	B	A	B	A	B	A	B	A
機種名（OS version）	ID	17:47	17:50	17:53	17:55	17:58	18:00	18:03	18:06	18:09	18:11	18:14	18:16	18:18
iPhone 15Pro (17.5.1)	1	乗車	—	—	—	—	—	—	—	—	—	—	—	降車
iPhone 15Pro Max (17.5.1)	2	乗車	—	—	—	—	—	降車		乗車	—	降車		
iPhone 14 Pro (16.0)	3	乗車	—	—	—	降車				乗車	—	—	—	降車
iPhone XR (15.4.1)	4					乗車	—	—	—	—	—	降車		
iPhone XR (14.6)	5				乗車	—	—	降車		乗車	—	—	降車	
iPhone 13Pro (17.5.1)	6								乗車	—	—	—	降車	
iPhone 13 (16.2)	7	乗車	—	—	—	—	—	—	—	—	—	—	—	降車
AppleWatch series 7 (watchOS 10.5)	8	乗車	—	—	—	—	—	—	—	降車		乗車	—	降車

図 7: デバイスごとの乗降車スケジュール

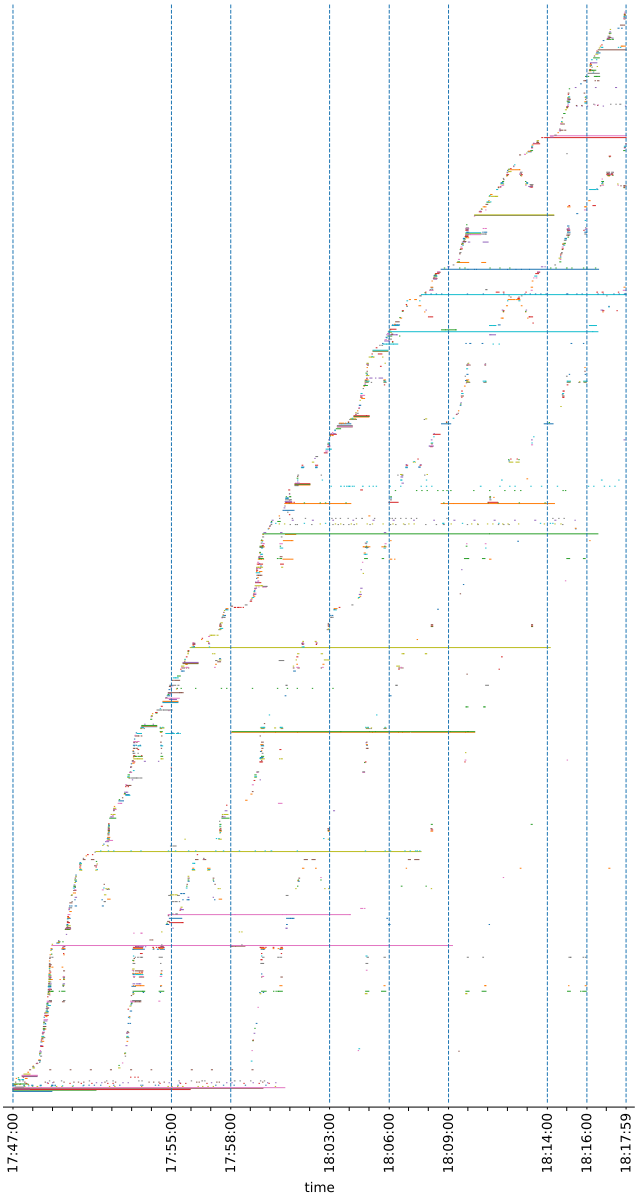

図 8: 無加工のデータ

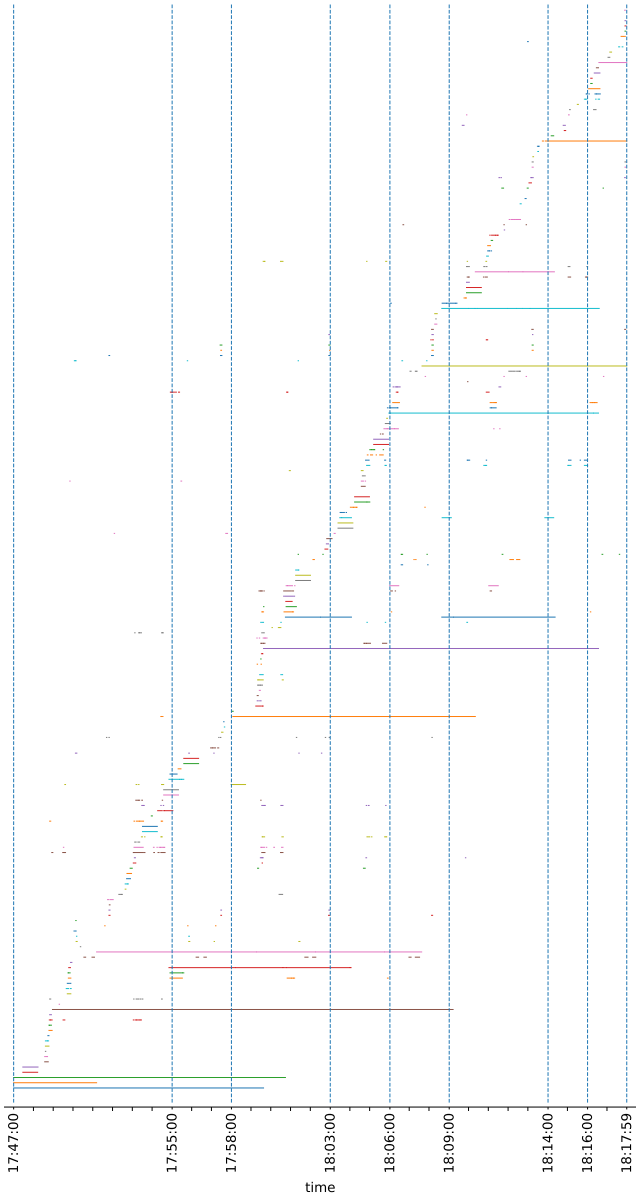

図 9: Manifacture Specific Data が `004c10` で始まる MAC アドレスを抽出したデータ

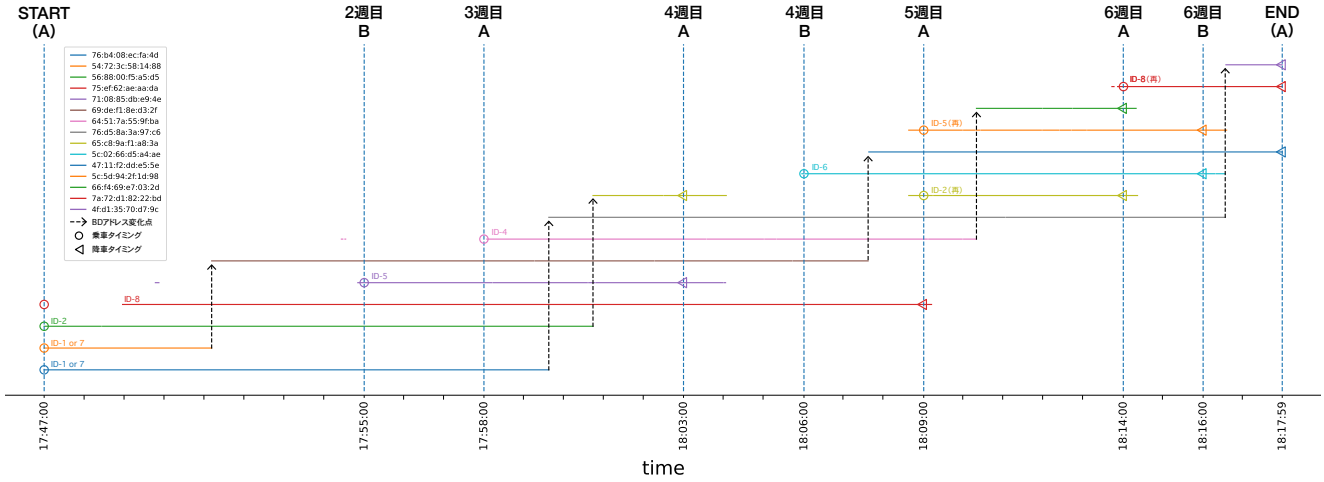

図 10: 出現回数でフィルタリングしたデータ．MAC アドレスが同時変化した箇所はを破線矢印で示している．

に乗車した直後に変更された MAC アドレスもフィルタリングしてしまう可能性があるため，この値は適切に定める必要がある．今回は両方のスキャナでスキャンできていることと，出現回数が一定数以上であること，という条件によって車外のデバイスの MAC アドレスをフィルタリングできたが，実環境ではより多くのデバイスが乗降車することが想定されるため，RSSI を用いたフィルタリング方法などについても今後検討する必要がある．

乗降車するデバイスが多い環境においては，バスが停車したタイミングで「MAC アドレスの変化」と「他のデバイスの降車」が同時に発生することが想定される．この場合は，アドレスキャリーオーバーに失敗する可能性があると考えられるため，RSSI や AD data などの追加情報を用いたアルゴリズムの改良が必要と考えられる．

ID-8 のデバイス（AppleWatch）に関して，原因は不明であるが最初の 2 分間 MAC アドレスが観測されていない．考えられる原因としては，AppleWatch は iPhone とアドバタイズパケットを発信する条件が異なる，または常に Manufacture Specific Data が `0x004c10` で始まるパケットを発信していないことなどが挙げられるため，今後の調査によって明らかにする必要がある．

4. 提案手法

3 章で行った実験の結果を踏まえて，BLE のランダム MAC アドレスを用いた路線バスの OD データ推定手法を提案する．提案手法の概要を図 11 に示し，各手順について以下に述べる．

ステップ（1）パケット取集

事前実験で使用したスキャナをバスの車内に設置し BLE のアドバタイズパケットをスキャンする．この時，スキャン時刻，RSSI，ランダム MAC アドレス，AD data を記録し保存する．

ステップ（2） フィルタリング

まず，iPhone または，AppleWatch を追跡するために Manufacture Specific Data が `0x004c10` で始まるアドレスのみを抽出する．次に，すべてのスキャナでスキャンできている MAC アドレスのうち出現回数や，平均 RSSI などからバスの乗客のデバイスのものと考えられる MAC アドレス以外のものを消去する．

ステップ（3）アドレスキャリーオーバー

BLE デバイスの MAC アドレスが書き換わる時間が非同期であること，RSSI の値が近いこと，AD data などの条件を用いてアドレスキャリーオーバーを行い，乗客の BLE デバイスの追跡を行う．

ステップ（4）OD 推定

アドレスキャリオーバーの結果から同一デバイスのものと思われる MAC アドレスの出現時刻と，消失時刻を特定し，これらと出発時刻と到着時刻が閾値以内のバス停があれば乗客とみなし OD データを生成する．

検討事項としては，実際の路線バスでは iPhone を 2 台所持しているなど複数デバイスを持っている乗客が乗り降りした際に，実際の人数よりも多く推定してしまうことが考えられる．複数デバイスを所持している利用者をどのように検出するかを検討する必要がある．また，事前実験では Android などのデバイスからのパケットは観測出来なかったため，iPhone や AppleWatch に限定した調査を行ったが，実環境での OD データ推定にあたっては，それらのユーザをどのように追跡できるかについて検討を行う必要がある．