2003 年度 (平成 15 年度) 慶應義塾大学卒業論文

インターネット自動車における車両情報管理および利用に関する研究

慶應義塾大学 総合政策学部
堀岡大輔
s00824dh@sfc.keio.ac.jp

指導教員
慶應義塾大学 環境情報学部
村井　純
徳田　英幸
楠本　博之
中村　修
南　政樹

平成 15 年 12 月 29 日

概 要

車両の情報化が進み、インターネットを利用し交通情報やメールなど外部のコンテンツを車両で受信できる。また車両からの情報発信も可能となっている。

車両から発信する情報の一つとして現在の速度やワイパの動作状態といった車両が生成する情報 (以下、車両情報) がある。車両情報を収集、加工することで社会的に価値のある情報を生み出そうという動きがあり、現在プローブ情報システムなどの研究が進んでいる。各車両のワイパの状態を収集し降雨状況を把握する等の車両情報の活用が可能となる。

プローブ情報システムでは各車両の情報を定期的にセンタへ収集するモデルとなっている。定期的に車両情報を収集するため、センタ-車両間の通信コストが大きい。また、多数の車両から取得した情報を蓄積・管理するためにセンタ側に規模の大きな設備が必要となる。車両情報を取得するための統一したインターフェースがないため独自にインターフェースを用意する必要がある。以上の問題点から、車両情報を利用したサービスを構築、提供するには困難な状況となっている。

車両情報を利用したサービスをより容易に構築、提供できる環境を作る必要がある。そのためにはサービス側が車両情報を取得するための通信コストの低減が求められる。また、サービスに必要な車両情報を取得するための統一したインターフェースが必要となってくる。

そこで本研究では、サービス側が必要とする車両情報を取得するための車両情報管理及び利用に関するモデルの提案を行った。本モデルではサービス側が指定した車両情報を車両内に蓄積するためのシステム及び蓄積された車両情報をアクセスするためのインターフェースからなる。車両内に情報を蓄積することで、サービス側は定期的な車両情報の収集が必要なくなった。また、蓄積された車両情報にアクセスするためのインターフェースを定義することで、蓄積された情報からサービスに必要となる情報のみセンタへ収集を行うことができ、通信コストの低減、センタ側システムの負担が軽減できた。

本モデルの実現可能性、有用性を検証するため、システムの検証、及びセンタに情報を定期的に収集するモデルとの通信量、データ量の比較を各種サービスを構築して行った。その結果、車両情報の取得にかかる通信量及びセンタ側で収集するデータ量の削減が確認できた。

以上の研究成果により、車両情報を利用したサービスを構築、提供しやすい環境ができた。

キーワード

1, インターネット ITS　2, 車両情報利用サービス　3, プローブ情報システム　4, 蓄積手法　5, API

abstract

Progress in computerization in vehicles allow passengers to retrieve traffic information and other contents as well as to send information through the Internet.

Information that can be sent out from vehicles, which we define as Vehicle Information, include information such as the current speed of the vehicle and the current status of sensors devices such as wipers. Currently, prove information system is seeking to collect these vehicle information to produce new and valuable information to our society. For example, collecting wiper status and location information of each vehicles can produce precipitation information, an effective example of utilization of vehicle information.

However, probe information system uses the model where information of each vehicle is collected periodically at a specific server. Since the server must periodically collect the vehicle information, its cost in communication with the vehicles is high. This model also requires scalable data center inorder to store and maintain the information collected from the vehicles. Furthermore, the model requires a common interface when collecting the vehicle inofrmation. Consequently, it is currently difficult to provide services using vehicle information.

This research propose a model to manage and utilize vehicle information needed by the service providers when collection vehicle information from vehicles. The proposed model is constructed of a system to store the specific vehicle information specified by each service inside the vehicle, and an interface to access the vehicle informatioon actually stored inside the vehicles. By storing information inside the vehicles, there is no need for the service providers to periodically collect the vehicle information. Moreover, by defining a common interface for accessing the stored vehicle information, only those information neccessary can be collected at the data center, reducing load at the data center and communication cost.

In order to verify the effectiveness and the reality of the proposed model, various services was built to compare communication cost with the model using in the prove information system. As a result, the proposed model verified to lower the amount of data needed to collect at the data center, and reduced communication cost when collecting vehicle information.

From this research, an environment to provide services using vehicle information with low cost is realized.

Keywords

1, Internet ITS　2, Service using Vehicle Information　3, Probe Information System　4, Storage System　5, API

目 次

図 目 次

表 目 次

第1章　序論

本章では、本研究の背景及び研究の目的について述べる。 また、本論文の構成を示す。

1.1　背景

無線技術の発展により携帯やPDAなど様々な移動体がネットワークに接続するようになった。自動車もネットワークにつながった移動体のひとつである [4][29]。

自動車がインターネットにつながることにより、ネットワークを経由して交通情報やメールなど外部のコンテンツから情報の受信ができる。一方、現在の速度やワイパの動作状態といった車両が生成する情報 (以下、**車両情報**) を外部に向けて送信できるようになった。車両には約 120 種類以上センサ情報が取得でき [3]、これら情報を収集、加工することで社会的に価値のある情報が生成できると期待されている。

車両情報を利用したサービス例としては、次のようなものがあげられる。

- 車両の位置やスピードを収集して車両に交通情報を提供するサービス
- ワイパの状態を収集することで降雨情報を生成するサービス
- エアバックの情報から事故を検知するサービス
- 車両の位置を常に把握することによって安全対策を施すサービス

その他、多種多様なセンサがあり、マーケティングや車両のメンテナンスに利用するなど新たなサービスの出現も考えられる。

このように車両情報を利用することで様々なサービスが可能となる。しかし現状では車両情報を収集するための通信コストが高いことや、収集するための統一したシステムがない。そのため車両情報を利用したサービスを構築するには収集システムを独自に構築する必要があるなどサービス構築に高いコストがかかり、サービスを容易に構築、提供できない環境となっている。

今後車両情報を収集、利用するためのモデルが共通化されれば、そのモデル上に車両情報を利用したサービスが容易に構築でき、車両情報を利用したサービスの拡大が期待できる。

1.2　目的

本研究では共通化した車両情報を収集するシステムを構築するため、サービスに応じた車両情報の収集ができる車両情報管理及び利用に関するモデルの提案を行う。

本モデルにより、サービスに必要となる車両情報のみを収集できるため、センタへ収集するため通信コストを削減できる。またセンタ側で収集したデータには冗長な情報が含まれないため車両情報の加工、提供が容易となる。

本モデルを構築することで、よりコストを抑えて車両情報を利用したサービスが提供できる環境が整い、車両情報を活用したサービスが増加し社会的な車両情報の活用ができる。

1.3 本論文の構成

本論文では、第２章で車両情報の特徴及びそれら情報を利用したサービス例を挙げた上で、今後車両情報を利用したサービスを構築する上で必要な要件、課題を述べる。

第３章では現状行われているサービスの現状、問題点を整理する。

第４章では現在のサービスの問題点、今後のサービスへの要件を元に今後の車両情報利用モデルの提案を行う。

第５章では第４章で述べたモデルを元にシステムの設計を行い、第６章ではシステムの実装を行う。

第７章では提案するモデルの実行可能性、有効性を検証するため行った実証実験について述べ、第８章でそれら実験の結果を受けモデルの評価を行う。

第 9 章では結論を述べるとともに今度の課題について述べる。

第2章　車両情報の利用

本章では、まず車両情報の分析を行う。次にそれら車両情報を利用したサービスの分類を行う。また、分類した各種サービスを構築するための必要となる環境を述べ、最後に現状の課題を述べる。

2.1　車両情報

車両側から発信する情報として、現在の速度やワイパの動作状態といった車両が生成する情報 (以下、車両情報) がある。これら車両情報は以前は車内でのみ活用されていたが、今後はインターネットを経由し各車両から車両情報を収集し、新たな情報を生成し提供するサービスが行われていく。

車両情報には様々な特徴があり、現在走行中の走行時速などリアルタイムに取得できる情報があるほか、走行履歴など過去に渡って持っている車両情報もある。

これら車両情報は図 2.1 のような特徴をもつ。現在の車両情報は各車両がリアルタイムに生成し情報量も大きい。また、最新の情報は高い価値をもつ。

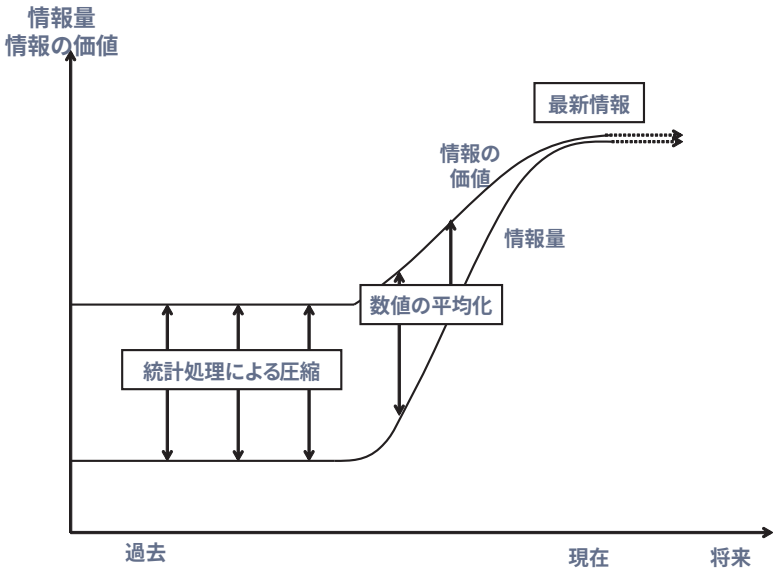

図 2.1: 車両情報の流れ

一方、時間が経つにつれ情報の価値は失われていく。全ての情報を保持する必要性が低くなるため、平均値や代表値を保存しておくことで情報の価値を維持させると共に、その他不必要な情報を削除していく。

さらに過去に遡ると情報は統計処理によって加工され情報価値を維持していく。情報はさらに

捨てられていき、情報量は下がり続けることになる。

以上で述べたように車両情報には大まかに現在、最近、過去の 3 種類の特徴を持ち、情報量、価値がそれぞれ異なるため活用方法も異なってくる。

2.2 車両情報を利用したサービス

車両情報をサービスの視点から分類する。サービスの観点から分類を行うことで、今後の車両情報を利用したサービスの要件の参考とする。

前節の車両情報の分類でで述べたように、現在の情報やここ最近の情報、過去の情報というように時間を軸とした車両情報の見方がある。そのため時間を軸としたサービスが考えられる。各車両は時間の幅を持って走行しており、○○月から△△月までの車両情報を活用したいといったサービス要求がある。

一方、車両の特徴の一つとして「移動する」ということが挙げられる。車両は各地点を移動しており、空間から車両を把握する見方がある。ある地点、ある路線にいる車両から情報を取得したい、ある地域にいる車両の情報が知りたいといったサービスが考えられる。

時間、空間という二つの大きな軸からサービスを観測という基準で表 2.1 に分類した。またそれぞれの観測手法を図 2.2 に示す。観測手法は以下の４つに分類した。

- 定点観測
- 経路観測
- 衛星観測
- 移動観測

空間を着目した観測として 0、1、2 次元観測が挙げられる。定点観測のように 0 次元観測では、ある１地点を通過する車両の状態を調べることができる。また、1 次元観測として経路観測が挙げられ、ある道路の混雑状況を調査するサービスが考えられる。衛星観測が例に挙げられるように 2 次元観測もある。ある地理的範囲の車両情報を取得することで地域の降雨情報を提供するといったサービスが考えられる。このように空間に基準にした際にも様々な車両情報の利用方法があることがわかる。一方、４つ目の観測例として時間に着目した移動観測が挙げらる。車両の時間幅を持った情報を利用することで動態管理などのサービスを行うことができる。

2.3 課題

前節で述べたように様々な観点から車両情報を利用したサービスが考えられる。現在車両情報を利用したサービスの構築には自動車メーカや自動車関連企業が取り組んでいるが、今後車両情報を利用したサービスが容易に構築できる環境となれば、他業種のコンテンツプロバイダなどがサービスを行うといったことも考えられ、新たな車両情報を利用したサービスの出現も期待できる。車両情報を利用したサービスの増加、多種多様化は今後の更なる自動車、情報関連市場の拡大を促し社会全体に利益をもたらすものとなる。

しかし、現状では車両情報を利用したサービスを構築、提供するには課題が以下の２点があげられる。

ひとつにはコストがある。車両情報を収集するためにセンタ-車両間で通信を行う必要がある。

表 2.1: サービス分類

観測名	説明文	基準	サービス例
空間に着目した分類			
0 次元観測 (例：定点観測)	ある地点を基準にして、その地点を通過する車両から情報を収集する	ある一つの地理的位置	旅行時間
			渋滞情報
1 次元観測 (例：経路観測)	ある二点間を走行した車両から情報を収集する	ある二つ以上の地理的位置	交通流調査
			道路混雑状況
2 次元観測 (例：衛星観測)	ある一定の範囲の車両から情報を収集する	地理的範囲	交通公害調査
			降雨情報
時間に着目した観測			
移動観測	移動体の動きを追い、時間の幅を持たせて車両から情報を収集する	時間	車両の状態 (バッテリー等) 管理
			車両の動態管理・運行管理

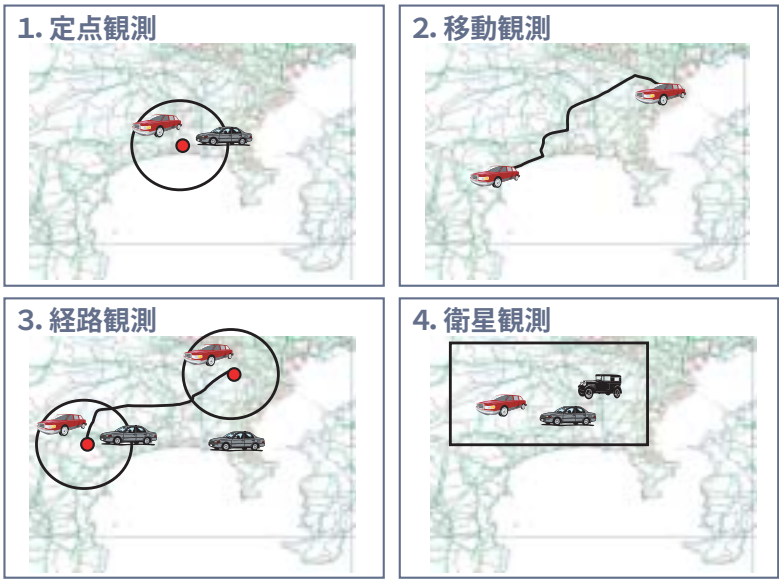

図 2.2: 観測手法一覧

車両情報を収集する際に通信が発生し、収集する対象となる車両が増加するごとに通信量も増加する。また、センタに各車両から情報を収集するデータ量も車両台数が増加するにつれデータ量も増加する。データ量が増大すれば通信設備、システムが必要となり、サービスを提供する車両台数が増加すればコストも増大していく。

二つ目の問題点として各サービスの要求にあった車両情報を取得するためのサービスに共通したインターフェースがないことがあげられる。車両情報を利用したサービスには空間や時間を軸とした利用方法がある。これらの軸を元に車両情報を取得することで、サービスにあった情報収集が可能となる。しかし、実証実験などで構築されるインターフェースでは各実験に特化しており、他のサービスを構築するには使いづらいものとなっている。そのため、各サービスが共通して利用できるインターフェースがなく、現段階でのサービス構築は困難となっている。

これらの問題を解決する共通した車両情報を利用するためのモデルを構築することで、サービス参入への障壁を減らし、サービスの多種多様化、しいては車両情報の社会的有効利用が可能となる。

第3章　車両情報利用サービスの現状及び問題点

本章では現在行われている車両情報を利用したサービスを官、民、研究の３つに分けそれぞれどのように車両情報を扱っているかについて述べる。現在の車両情報利用の現状をまとめ、今後の課題を探る。

3.1　官主導で行われているサービス

本節では官主導によって実施されているサービスについて述べる。

道路交通情報通信システム (VICS)

Vehicle Information and Communication System(VICS)[24] とは、VICS センターで編集、処理された渋滞や交通規制などの道路交通情報をリアルタイムに送信し、カーナビゲーションなどの車載機に、文字・図形で表示するシステムである。日本道路交通情報センタから受けた交通情報を VICS センタで処理・編集を行う。加工された交通情報は、主に FM 放送波を利用した FM 多重放送で車両に対して交通情報の提供が行われている。また、主要幹線道路では光（赤外線）を利用したり高速道路で電波（準マイクロ波）を使っている場所もあり、必要な地域の交通情報が取得できるようになっている。

道路交通情報センタからの交通情報を VICS では利用している。道路交通情報センタでは全国にあるセンター・駐在で情報収集を行っているほか、日本道路公団や首都高速道路公団、阪神高速道路公団、一部の都道府県警察の交通管制センターなどからの情報も収集している。これらの情報はセンタの職員や装置が決まった地点で交通情報を収集しているため、全国全てを把握することは困難である。

また、VICS は光ビーコンや FM 放送波を利用しているため、VICS 以外のサービス情報を外部から取得したり車両から情報を発信することができない。

自動料金収受システム (ETC)

Electronic Toll Collection(ETC)[25][26] は、車両のダッシュボード上などに設置する無線車載機と、IC カードリーダ、料金所に設置される路側アンテナにより、車載機と ETC 装置が無線通信を行う。高速道路などの料金所での一時停止をする事なく料金の徴収を可能にする。現在普及が進んでおり、全国 600 箇所ほどの料金所に設置されている。

ETC も VICS 同様、サービスを利用するためには専用の機器が必要となる。また、ETC では車両情報は利用されておらず、個人情報や ID を利用してサービスを行っている。

走行支援道路システム (AHS)

Advanced Cruise-Assist Hightway System(AHS)[28] は、スマートウェイの一環として運転中に人間が行う判断状況や自動車の機能を、道路インフラと情報通信システムが協調して支援するシステムである。利用例としては、道路に沿って設置されたセンサによって障害物が検知された場合に、無線通信によって車両へ警告を出したり、運動制御を行うことが

できる。AHS はITS の技術的に最も高度なシステムである。構想としては三段階のシステム (情報提供、制御支援、自動運転) を設定している。車両の位置情報と道路側のインフラ間で情報をやり取りすることで安全運転を目指している。このシステムでは道路にインフラを作るためのコストが高いほか、車両に搭載された車載センサで自分の車両の位置事故防止を主目的としているので他の車両情報を利用したサービスとの連携がない。

3.2 民主導で行われているサービス

本節では企業を主体として民間で行われているサービスについて述べる。

ATIS

Advanced Traffic Information Services(ATIS)[11] は、VICS と同じ情報ソースを用いた情報サービスを行うシステムである。VICS との最大の違いは、サービス主体の ATIS が民間企業であり、情報提供の対価として利用者からサービス料を徴収する点である。情報提供メディアはVICS が官主導の整備による無料の配信網と違い、課金システムが整備されたものを利用している。有線系ではパソコン通信やダイヤルQ2、無線系では携帯電話などがそれである。AITS 独自のシステムで処理したデータを配布しており、最新の地図情報もダウンロードできる。

また、ATIS では企業ユーザに対してもサービスを提供している。テレビによる道路交通情報もその一つである。ITGS 等の情報サービスは、ATIS の業務代行サービスと位置づけられている。このようにATIS の情報を加工・再販するコンテンツも展開されている。ATIS では交通情報がパソコンや携帯電話に情報が提供されるが車両情報と連動したサービスはない。

新地域交通システム (ICVS)

Intelligent Community Vehicle System(ICVS)[12] は、本田技研工業 (株) が提唱した新たな地域交通システムである。ICVS の概念は生活圏や限定された地域内で低公害車を相互利用するシステムである。ICVS と自家用車や公共交通機関の組み合わせにより利便性を損なうことなく交通問題を解消することを目的としている。

用いられている技術としては会員の認証に用いるIC カード、ポート (専用駐車場) 及び無線通信による車両管理、超広角レーザーによる無人隊列走行などがある。また、磁気ネイル、誘導ケーブル、超音波センサによる入出庫の自動化も行われている。現在、シンガポールで名称を「Honda DIRACC(ホンダ　ダイラック)」と変えサービスを開始している。

このサービスでは共同利用するための車両にIC カードや車両管理システムが利用されているため、他の車両が同様のシステムを利用できず、ひとつのサービスの中で車両情報が利用されている。

internavi

internavi[13] とは本田技研工業 (株) が行っているテレマティクスサービスで、携帯電話で通信することで最新の店舗や交通状況の情報を受信できるほか、走行距離、速度情報を元に燃費やメンテナンスに活用することができる。サービスを利用するにはインターナビ対応のハンドフリー通信キットが必要となる。また、サービスは自動車メーカやメーカと提携した企業から提供されており、自由にコンテンツプロバイダ等がサービスを提供できない。また、走行距離、速度情報といった車両情報もサービスに依存した形で利用されており、他

のサービス事業者が利用できない。

CARWINGS

CARWINGS[14] は、日産自動車 (株) から提供されているオペレータと話したりメールや最新の情報を取得できるテレマティクスサービスである。ナビの利便性を追求しており、ドライバの変わりにオペレータが対応するなどナビに特化したサービスとも言える。通信メディアとして携帯を利用している。車両の位置情報を携帯電話やパソコンへメールを送信できるが、他の車両情報は活用されていない。

G-BOOK

G-BOOK[15] とはトヨタ自動車 (株) が行っているサービスで、ドライブ中の情報の受発信を可能にするほか車を運転していない際にも携帯電話、パソコンを通じてサービスが利用できる。GPS を利用し位置情報を把握することでセキュリティサービスを行っているが、その他車両情報は活用されていない。また通信料は定額制のためユーザは意識せず利用できるが通信量が増えるとサービス側の大きな負担となる。他のテレマティクスサービス同様、専用の機器を購入する必要がある。位置情報や車両のオートアラームを利用したサービスは行われているが、その他車両情報は利用されておらず他のサービスから車両情報を取得できる環境となっていない。

OnStar

OnStar[16] とは OnStar 社が行っているサービスで、全地球測位システム (GPS) 網と通信技術を利用し、ドライバーと自動車をオンスターセンターで結ぶ。サービス内容としては事故におけるエアバック作動時の自動通報や盗難車両の追跡、緊急通報、遠隔診断などがある。衛星通信を利用することで全米の広大が地域に均等なサービスを提供している。しかし、衛星通信の通信容量が少ないためコンテンツの容量が制限されるほか、OnStar 社に車両情報を収集しており、他のサービスから利用できない。

NetworkCar

NetworkCar[17] は車両に搭載したデバイスとインターネット側のシステムで自動車の各種データを収集し、ワイヤレス通信を用いて通知するサービスである。サービス内容としては運転中にタイヤがパンクするなどの事故が生じた場合の支援サービスや、盗難車追跡、故障の診断などである。

車両情報の送信方法としては、絶えず車のデータを蓄積し、その情報をプロバイダに送信する。電波が弱い地域でも蓄積を続け、電波が通じる地域に入るとそれをまとめて送信する仕組みとなっている。システムをインストールすれば車種に関係なくサービスを利用することができる。車両情報は NetworkCar のセンタに収集されるため、それら情報を他のサービスが容易に利用することが困難である。また NetworkCar のシステムを利用し、他のサービスが独自に車両情報を収集できない。

3.3 研究段階のシステム

本節では、研究段階の車両情報提供システムについて述べる。

プローブ情報システム　プローブ情報システム [1][22][23][30] とは、車両によって生成されたデータを集めて、データを格納し、無線通信によってアプリケーション供給者にデータを配達す

るシステムである。車に搭載されているセンサは約 120 種類ほどあり、これらの情報を収集、加工することで社会的に有用な情報を生み出そうとしている。

プローブ情報システムを利用することで現在利用されていなかった車両情報が今後有効に活用されることになる。現在は各車両にて各センサの情報を持つが、車内でのみ利用されており車両外で利用されてはいない。そのため、各車両がどういった動きをしているのかを車両外から知る方法は限られている。一方、プローブ情報システムは各車両から情報を複数の車両から収集するため、車両情報の動きがより正確に、かつ幅広く取得できる。各車両から情報を取得できることで、ある地点で起こった事故や渋滞についての情報が取得できるようになる。

プローブ情報システムを利用することで、各車両の情報をリアルタイムに取得できるため、車両ごとの速度と位置情報を加工し道路状況が把握することができる。また、車両情報を常に監視することで、駐車場の停車位置を管理したり、盗難防止など安全面でも活用できるサービスも考えられる。

プローブ情報システムは次のような３つの工程となっている。まず、センタ側は各車両の車両情報の収集を行う。インターネットを通信基盤として用い、多数の車両から情報を収集することを可能とする。次に、センタ側は収集した車両情報を必要に応じた精度で加工・分析を行う。最後に、加工・分析した車両情報を利用者に対して提供を行う。各機能を分散させることでより利用者に適したサービスを行うことが可能となる。

データ辞書モデル

データ辞書とは、自動車の保持する情報の全てを名前と値の１対１のエントリで示すモデルである。図 3.1 ではライトの On/Off、外気温、車速などの情報がデータ辞書に格納されている状態を示している。

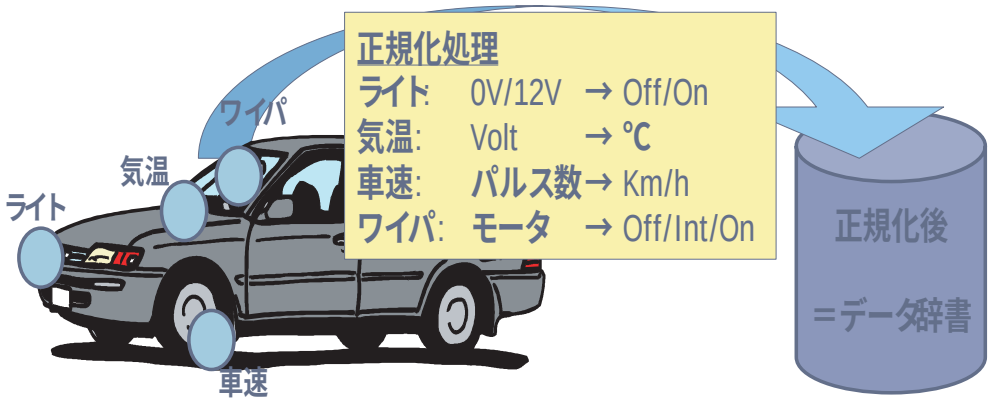

図 3.1: データ辞書モデル

車両情報を扱う際の問題点として、自動車毎に保持しているセンサの単位や精度にばらつきがある。プローブ情報システムなど車両情報を扱うサービスを構築する際、各車両から取得できる情報の単位や精度が異なると情報の共有が困難である。そこで車両やシステムに依存することなく車両情報を収集可能とするため、データ辞書モデルでは単位、精度を統一し正規化を行っている。

現在、データ辞書モデルは ISO/TC204/WG16/SW16.3[19] に提案されており、現在検

討が進んでいる。データ辞書モデルでは、Name(センサ名) や Type(タイプ)、Valid value rule(データエレメントの定義) などが決められており、センサは同じデータ形式で取得できるようになっている。センサの種類については、各車両によって搭載されているセンサの種類が異なると考えられるため、センサの搭載が必須となるコアデータ定義は、time stamp(時間) と location stamp(緯度・経度・高度) となっている。またコアデータ以外のデータに関しては適宜追加できるようになる。そのため、新たなセンサやセンサの利用法が出てきた場合にはデータ辞書の定義を行うことで、各車両共通のデータフォーマットで利用することが可能となる。

データ辞書モデルが今後標準化されることによって、車両の種類やシステムの違いに捕らわれず、センサの情報がひとつのデータ定義で取得可能となる。また、センタ側でそれらの情報が収集された場合、データ形式が決まっているため、データの交換が容易となる。

3.4 まとめ

現状の車両情報を利用したサービスを、官、民間、研究の三つに分類し問題点の整理を行う。表 3.1 にそれを示す。

官主導の VICS や ETC、AHS では情報の利用コストはかからないものの、専用の機器を購入するためのコストがかかる。また、サービスに依存したシステムとなっており、車両情報は利用されていない。

民主導で行っているサービスでは官主導のサービスと同様、各種サービスを利用するためには専用の機器を購入する必要があり、それらを機器を利用し他のサービスを利用できない。また、各種サービスのコンテンツは自動車メーカやサービス提供者に限られ、オープンにサービスの提供を行える環境となっていない。また、通信コストが高いためサービスを行う上での負担となっている。車両情報の利用は各システムに依存した収集を行っているため、ほかのサービスが車両情報を利用できる環境となっていない。

研究が進むプローブ情報システムでは既存のインターネット基盤を利用するため独自の機器を購入する必要がない。また、データ辞書モデルによって車両情報が統一した定義で正規化され、車両情報を扱うサービスの構築が容易となる。しかし、通信費用のコストが大きく負担となる。また、車両情報を収集するためのサービスに共通したインターフェースがない。現段階では各サービスに依存したインターフェースの構築に限られている。

表 3.1: 車両情報を利用したサービスの現状

	機器	通信コスト	サービス参入コスト	車両情報の利用
官	独自	低	参入不可	なし
民間	独自	高	高	各サービスに特化した収集を行っている
研究	共通	高	低	各種車両情報を収集可能

第4章　車両情報管理モデル

本研究ではサービスが必要とする車両情報を収集し利用するため、車両情報管理の方法および利用に関するモデルの提案を行った。本章では現在の車両情報を収集するモデルの問題点を挙げた上で、モデルの検討を行う。次に検討結果をもとに本研究で提案する車両情報管理モデルについて述べる

4.1　現在の車両情報利用モデル

プローブ情報システムの実験で見られるように、車両情報を定期的にセンタに収集するモデルとなっている。イメージ図を図 4.1 に示す。図では横軸に時間をとり、緯度、経度、高度で構成される空間の推移を示している。各時間に走行している車両を円で示しており、それが時間が立つにつれ移動している様子を示している。センタから定期的に各車両に対し情報収集を行っている。

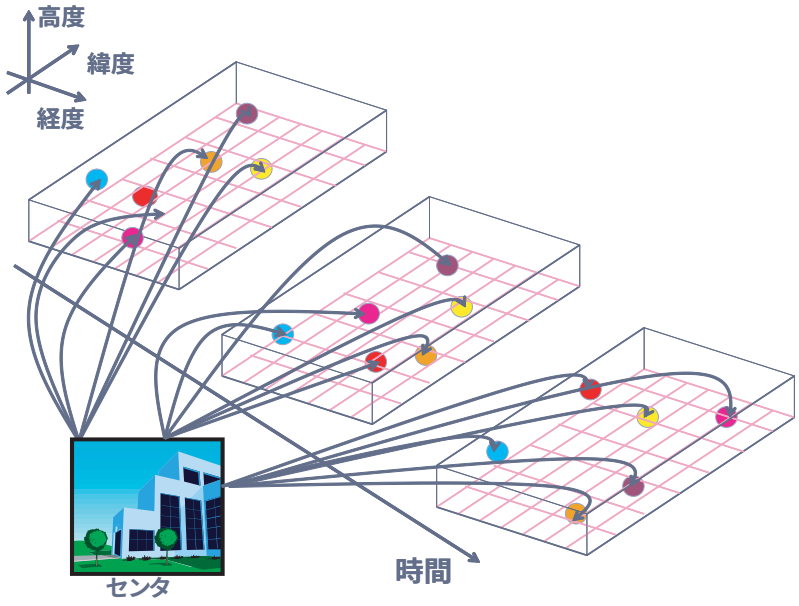

図 4.1: 定期的に車両情報をセンタに収集するモデル

車両情報を定期的に収集することで、リアルタイムに車両の状態が把握できる。しかし、常に情報を車両-センタ間で通信するため通信コストが高い。情報を取得する車両が増加すると、それだけ通信コストが増加することになる。また収集した情報には、生成された車両情報全てが含まれるため情報量が多くなり、情報の加工、提供のコストがかかる。またそれら多くの情報に対応するためのセンタ設備が必要となる。

表 4.1 にセンサ数、取得間隔によって一ヶ月間 (30 日間) にどの程度の通信量が発生するかを示す。センサは全て int 型で取得し、毎日 2 時間走行することとする。また、画像についてカラー 24bit で 320 × 240 のサイズは 15KB、160 × 120 のサイズは 0.4 キロバイトとする。情報量の単位はメガバイトとする。

表 4.1: 車両情報量

収集間隔	10 個	50 個	120 個	画像 (320 × 240)	画像 (160 × 120)
1 秒	8.64	43.20	103.68	2592	69.12
10 秒	0.86	4.32	10.36	259.2	6.91
1 分	0.14	0.72	1.72	43.2	1.15
10 分	0.01	0.07	0.17	4.32	0.11
30 分	0	0.02	0.05	1.44	0.03
60 分	0	0.01	0.02	0.72	0.01

表 4.1 で見られるように 1 秒間隔で車両情報をセンタに収集するには月 100 メガバイト程度通信する必要がある。画像を含めるとより高い通信コストがかかる。表は一台あたりの情報量であるため、情報を取得する車両が増加すればそれだけ、通信量、データ量がセンタ側の設備に大きな負担がかかることがわかる。

このように現在の車両情報を定期的に収集するモデルでは通信コストの負担が大きく、システムの運用にも多大なコストがかかる。そのため車両情報を利用したサービスへの参入の大きな障害となる。

次に車両情報の取得方法について述べる。現在車両から情報を取得するためのインターフェースは各サービスに依存した方法で構築しており、各種サービス共通した車両情報を収集するためのインターフェースが定まっていない。車両情報を利用したサービスを構築するためにインターフェースを独自に構築するには時間、コストがかかる、また、一度インターフェースを構築すると変更が聞きづらくなるためサービス内容の変更が困難で、ニーズにあったサービスの提供が困難となる。

4.2 車両情報利用モデルの検討

第 3 章でサービスの分類結果のように、車両情報の様々な利用方法が今後考えられる。これらサービスを容易に構築できるよう、車両情報を利用するためのモデルの検討が必要となる。

前節で述べた問題点を元に、モデルを構築するために検討する必要がある項目は次の点である。

- 車両情報をセンタへ収集するためのコスト削減
- 車両情報を取得するための共通したインターフェースの構築

4.2.1 センタへ収集するためのコスト削減

車両情報を利用したサービスを行う際の障壁として通信コストとセンタのコストが挙げられる。車両からセンタへ収集する際コストが発生し、収集回数、車両台数が増加するごとにコストも増

加していく。また、収集する情報量が大きいためセンタ側に規模の大きい設備が必要となる。

これら問題を解決するためには、収集する情報量を削減する必要がある。そのためにはサービスが必要とする情報のみを収集することで情報量を減らすことができる。また必要な情報を収集することで情報の質を落とすことなくサービスを提供することができる。

サービスが必要とする情報を取得する方法としていくつか考えられる。ひとつには車両情報センタを一つに集約し、そこからサービスに必要な情報を収集する方法がある。しかし、この方法では車両情報センタの管理者を誰にするのかといった問題や国もしくは世界の車両情報を一括して管理するシステムの構築コストを誰が負担するのかという運用の問題がある。次の方法として車両内に車両情報を蓄積する方法がある。各車両内に情報を蓄積しサービスが必要な情報を収集するモデルである。この方法では各車両から収集する情報量が削減でき、かつサービスごとに必要な車両情報を取得できる。図 4.2 に示すように車両内とセンタの双方を利用して車両情報を利用できる。例えばリアルタイムの交通情報提供サービスを行う際には車両情報をセンタに定期的に収集する一方、メンテナンスサービスを行う際には車両情報を車両内に蓄積しておき必要な時期に収集するといった利用方法がある。

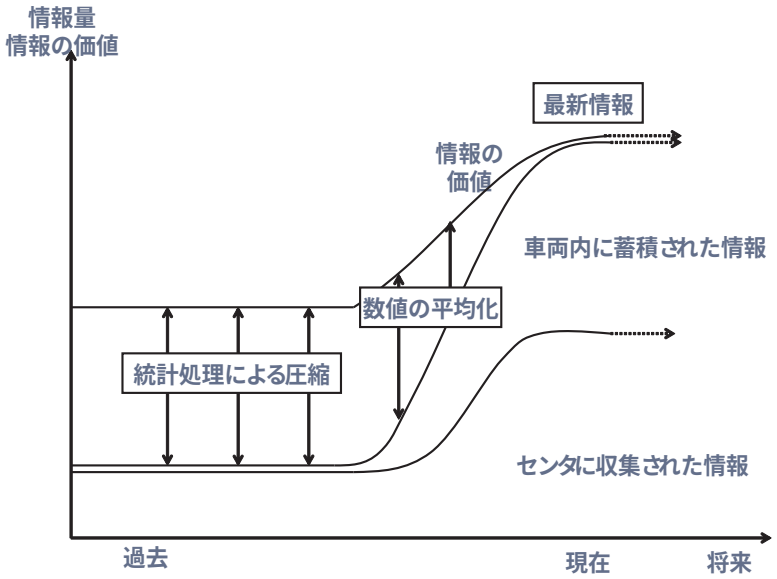

図 4.2: センタ、車両内を利用した車両情報

以上より車両情報を利用するための方法として車両内に情報を蓄積する手法を取り入れ詳細な検討を行っていく。

車両情報の蓄積方法

車両情報を車両内に蓄積する方法として次の方法が考えられる。車両に情報を蓄積する方法としてどれが効率よく蓄積し、サービス側からアクセスできるかについて検討する。

1. 車両情報を常に全ての情報を蓄積していく
2. センサごとに蓄積間隔を定義し、これに従って蓄積する

3. サービス側が指定したセンサ情報を蓄積する

１つ目の方法として車両情報を常に車両に蓄積していく方法がある。現在のハードディスクドライブの大きさから考えると全ての情報を蓄積するのは可能である。表 4.1 で見られるように一ヶ月あたりセンサの情報のみ蓄積していくと車両の平均的な寿命である 10 年間蓄積し続けても 12GByte となり、蓄積できる容量である。しかし、全ての情報を蓄積するため情報量が非常に多くなり、サービスの必要な情報を検索するための時間が増加する。また、長期の蓄積情報を持つことは、短期、現在時間での利用の妨げとなる。記憶装置、検索処理能力を考慮して蓄積する期間を決定する必要がある。

次に２つ目の方法として、センサごとに静的に収集間隔を定義し、これに従って蓄積する方法がある。各センサごとに車両を取得する間隔が異なるため、情報量が減ることが考えられる。しかし、蓄積間隔をセンサごとに決めることは、車両情報を利用したサービスが間隔を設定することができない。そのため、サービスによってはより細かい取得間隔でデータを利用したい場合や、もっと大まかな情報のみでよい場合に対応できない。センサごとに定義された間隔ではサービス側の要求に柔軟に対応できない。

最後に３つ目の方法について述べる。センサ情報をサービス側の指定に従って蓄積することで、サービスに特化した車両情報の利用が行える。図 4.3 のようにあるセンサは１分ごとに、ある別のセンサは１秒ごとに蓄積する、などサービスに応じた蓄積が可能となる。センタへ蓄積された情報から必要な情報を取得する際、全て車両情報を蓄積する場合と比べ検索時間が少なくすむ。また、不必要な情報を蓄積しないためサービスにあった情報利用が可能となり、サービス、車両所有者双方の利益となる。

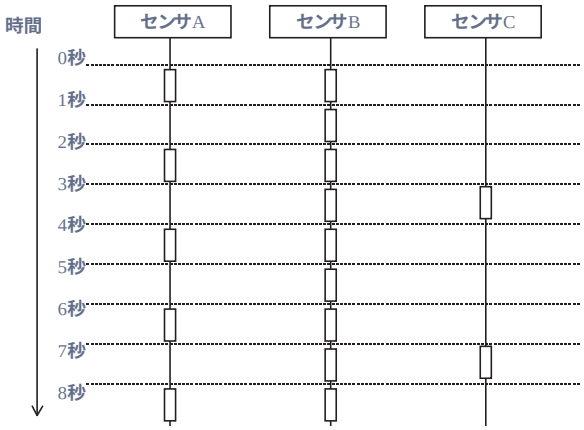

図 4.3: センサ情報を蓄積するタイミング

一つ目や二つ目で述べたように、定期的に車両情報を蓄積する方法は蓄積される情報量が多くなり検索の時間が多くかかる。一方、三つ目の方法のようにサービスに必要な情報のみを蓄積することで蓄積する情報量をサービス利用に限ることができ検索時間の減少にもつながる。以上のことから、３つ目の方法がより車両の所有者、サービスからより有効に車両情報を蓄積、利用できる方法であるといえる。

蓄積する車両情報のデータ構造

車両情報を車両内に蓄積するにあたり、蓄積する車両情報のデータ構造を定義する。データ構造を定義することで蓄積された車両情報をアクセスする際、サービスが必要な情報を抜き出して収集することが可能となる。

蓄積する車両情報の構造は、時刻、位置 (緯度、経度、高度) および一つ以上のセンサ情報を 1 セットとし車両に蓄積することとする。表 4.2.1 にデータセットを示す。時刻、および位置はデータ辞書でもコアデータとして提案されており必ず取得できる情報である。第 3 章のサービス分類で示したように、時刻や位置を軸とした車両情報が様々に利用できる。これら時刻、位置情報とセンサの情報をセットとして車両内に保持することで、時刻や位置を軸としてサービスに必要となる車両情報を抽出することができる。

表 4.2: センサデータセット

時刻	緯度	経度	高度	センサ情報	(センサ情報 …)

4.2.2 車両情報取得のためのインターフェースの検討

従来車両情報を収集するためのインターフェースはサービス依存で独自に構築されていた。そのため他のサービスを行うために利用しづらいインターフェースにもなっていた。そこで各サービスで利用できる共通したインターフェースを定義することで、各種サービスを構築する際独自のインターフェースを設ける必要がなくなる。

インターフェースを定義するにあたり、第 3 章で記述したサービスの分類を参考に検討を行った。各種サービスを構築するために必要な情報を取得するインターフェースを容易することでサービスの構築が容易となる。

インターフェースの定義ではサービス分類と同様、空間および時間を軸としてインターフェースを定義する。空間を軸とするインターフェースとして、緯度、経度、高度を引数として各車両情報を取得する。空間を軸とすることで、ある地点や道路の情報、地域の情報をサービスが収集できる。もう一つの軸として時間を扱うインターフェースを持つこととする。車両には過去から現在までの車両情報が含まれ、それら時間を指定することで車両の挙動を把握するサービスが構築できる。図 4.4 にその様子を示す。図 4.1 と同様、横軸に時間をとり、空間の推移をあらわしている。空間内にある丸で示した車両が時間の経過とともに移動していく様子を示している。各空間を推移していく車両情報を収集することで車両の状態変化が把握できる。

もう一つのインターフェースの機能として最新の車両情報にアクセスできる機能が必要となる。

以上、各車両情報の特徴を把握した車両情報収集をするため、以下の 3 つの収集方法を元にインターフェースの定義を行う。

- 空間を軸とした収集方法
- 時間を軸とした収集方法
- 最新情報の収集方法

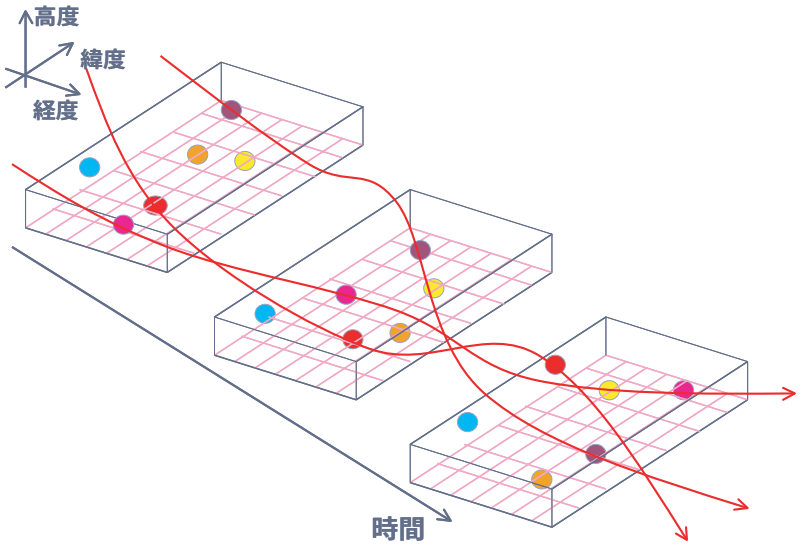

図 4.4: 移動体を軸として見る車両情報

4.3 車両情報管理モデル

検討結果を元に、車両情報の管理および利用に関するモデルの提案を行う。本モデルの概要図は次の図 4.5 に示す。

車両情報管理モデルでは各種センサ (sensor) をデータ辞書 (Data Dictionlary) を用い正規化した車両情報を蓄積するための容量 (以下、記憶容量)(図の網掛け部分の中の Storage)、蓄積された車両情報を収集するためのインターフェース (I/F) からなる。

各車両に記憶容量を持つことでサービスに必要な情報を蓄え、そこからサービスが必要な時期に情報を収集することできる。

インターフェースを定義することで蓄積された車両内の情報から必要な分だけサービスがインターネットを経由して収集することができ、サービスごとで必要となる車両情報を設定された間隔で取得が可能となる。また、最新の車両情報を収集する機能もインターフェースの機能として持つことで、現在、過去の情報をうまく使い分けることができる。

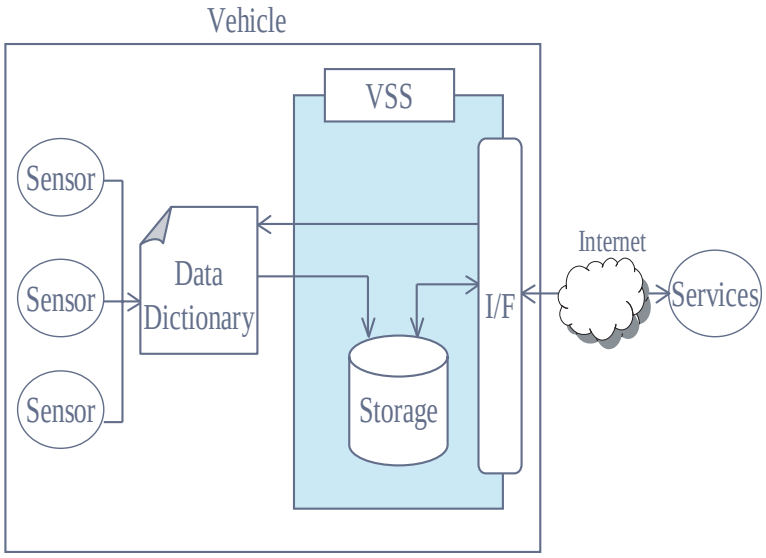

図 4.5: 車両情報管理モデル概要図

第5章　設計

本章では車両情報管理モデルに基づき車両情報蓄積および利用のシステムの設計を行った。まず車両情報を蓄積するためのシステムについて述べ、次に車両情報にアクセスするためのインターフェースの詳細な設計について述べる。最後にシステムの流れについて述べる。

5.1　車両情報の蓄積システム

車両情報をサービスの要求に従って蓄積するようモデルの提案を行った。このモデルを具現化するためシステムの設計を行った。まずシステム概要を述べ、次にシステムで利用する Storage Request Message(以下、**SRM**) の詳細について述べる。

5.1.1　車両情報蓄積システム概要

システムの概要図について図 5.1 に示す。車両情報蓄積システムでは、あらかじめサービスが蓄積する車両情報について指定を行うことでサービスに特化した車両情報の蓄積が行える必要がある。そこでサービスが必要とする情報を車両内に蓄積するための機能として SRM を用いたシステム設計とした。SRM とはサービスが必要とする車両情報を記述するメッセージで、そのメッセージを車両に送信することで車両内にサービスが必要とする情報の蓄積を開始する。SRM の詳細は後述する。図であるように、サービスは蓄積する必要がある車両情報に関する記述を SRM に行う。この記述された SRM を車両に対して送信する。受け取った車両は SRM の指示に従って車両情報の蓄積を開始する。

5.1.2　SRM

SRM とはサービスが必要とする車両情報を蓄積するよう、車両へ指示するためのメッセージである。サービスが必要とする車両情報を蓄積するためには、サービス側が必要とするセンサやその利用方法、車両についての設定等を詳細に指定でき、それを車両に指定できるメッセージが必要となる。

車両情報の蓄積に際してサービスに利用するセンサの指定および蓄積間隔を指定する必要がある。指定した間隔でセンサ情報を車両内に蓄積することで、サービスに応じたセンサの利用ができる。センサ情報を蓄積する際、はずれ値を削除する要求もある。また、センサ情報を蓄積するだけではなく、車両から定期的にセンサ情報を送信する利用方法も考えられる。　また、通信環境や通信メディアに関する要求もある。無線 LAN のように帯域が広い場合のみ車両情報の送受信を行うといった場合や、特定のメディア利用時のみ車両情報の通信を許可する場合もある。車両情報の取得方法によっては一度に大量の車両情報を送信する可能性もあるので、必要に応じた通信環境で利用できるようにしていく必要がある。以上の要求を表 5.1 に利用方法および記述す

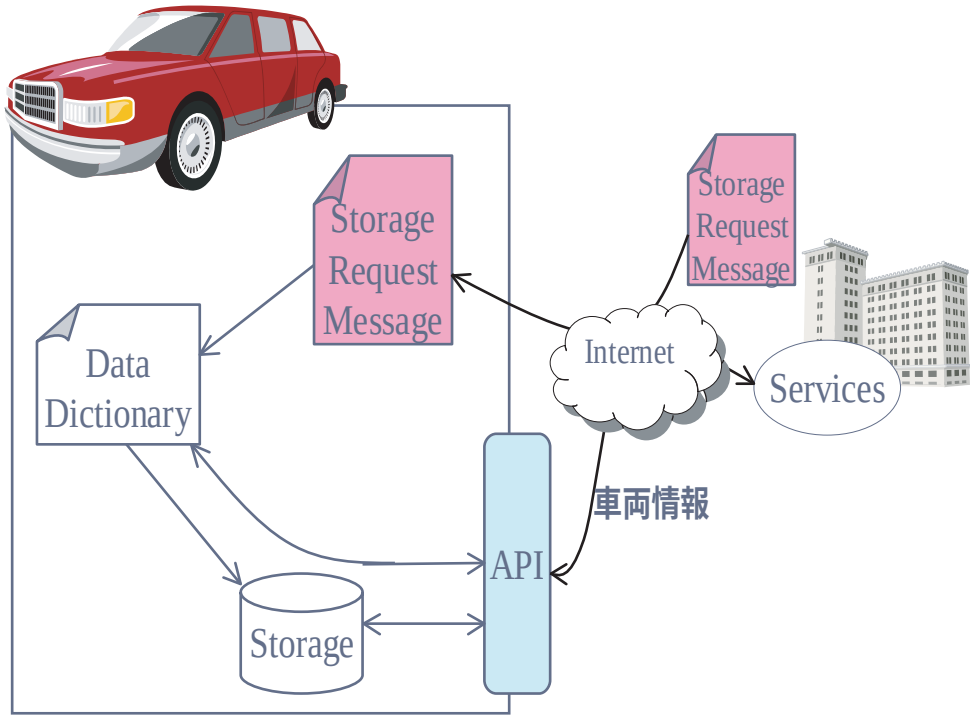

図 5.1: システム概要図

る要素名をまとめた。各利用方法に必要な要素名を挙げ、サービスの様々な車両蓄積方法の要求を満たすよう SRM の定義を行った。

SRM は、Extensible Markup Language(XML)[6] で記述するものとした。特定のプラットフォームに依存することなく様々なシステムから利用できる XML を今回の設計では利用した。RELAX NG[7] を用い、SRM の定義を行った。SRM の定義については付録に記する。サンプル例として、温度、湿度を利用した記述例を示す (図 5.2)。

SRM は必要なセンサについて記述を行う。センサを蓄積する間隔や車両側の記憶容量があふれたときの対策、車両側から動的にセンタに向かって情報を送信する設定などを指定することができる。図の tempreture(温度) の場合には、温度情報を 10,000 ミリ秒間隔で蓄積を行う。また、車両側の蓄積領域があふれた場合には蓄積されているすべての情報をセンタ側に送信し、随時 600,000 ミリ秒ごとに車両から動的にセンターへ情報を送信するように設定を行っている。この例のようにサービスに必要となる車両情報の扱いに関して SRM に記述し、車両に SRM を送信することで指定した情報の蓄積を行う。

5.2 車両情報へアクセスするためのインターフェース

車両情報へアクセスするためのインターフェースとして、前章のモデルに基づき時間や空間を軸として車両情報が収集できるよう API(Application Program Interface) の設計を行った。サービスに必要な車両情報の取得方法を網羅するため、時間、空間を軸とした車両情報の取得方法を元に API として定義した。第 3 章のサービス分類で述べたように、時間、空間を軸とすることで様々なサービスに対応できる。

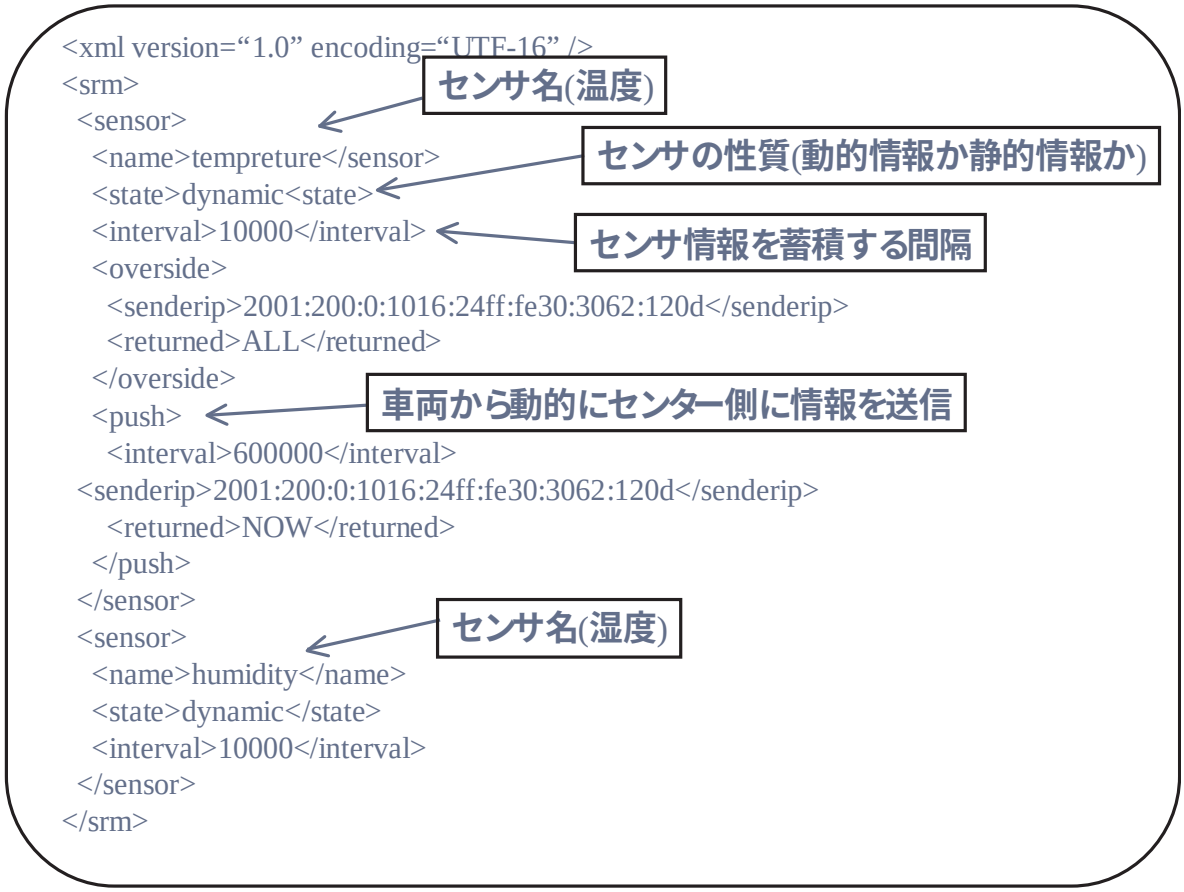

図 5.2: 温度・湿度センサに関する SRM 記述例

表 5.1: Storage Request Message 要素一覧

利用方法	要素名
取得したいセンサ一覧を指定	・センサ名
車両情報を蓄積する間隔の指定	・センサの蓄積間隔
蓄積するセンサ情報の制御 (はずれ値を蓄積しない、など)	・センサ値を制御するための条件文
センサ情報を車両側から定期的にアップ	・アップする間隔 ・アップする情報 (平均、最大値、最小値など) ・アップするあて先 ・開始・終了時間 ・送信できなかった際の対処 (リクエスト数)
車両情報の蓄積容量を超えたときの対処	・対処方法 (削除、蓄積情報の送信など) ・送信する値 (容量を越えた情報を送信する場合)
通信状態による制御	・帯域条件 ・送信するセンサ名・センサ情報
メディアによる制御	・メディア名 ・送信するセンサ名・センサ情報

必要な車両情報へアクセスするための API 一覧を次の表 5.2 に示す。この表では、サービス側からの指示 (引数) に対してどういった車両情報を返すか (戻り値) について述べている。API の定義にあたり軸とした時間、空間を観測という基準でそれぞれ述べる。

定点観測

ある地点の渋滞情報を収集するといった定点観測を利用したサービスを構築するための API の定義を行った。この API では位置を基準として時刻の指定や車両の指定などを行い、必要な情報を取得していく。例えば表の最上段の getAllSensorData() では、車両 ID および位置を指定することで、その場所にいる車両の全てのセンサ情報を取得する。

経路観測

経路観測ではある道路の動きを注視する。道路を移動した車両のリストやセンサ情報を取得することで車両の動きを見ることができ、また経路調査など交通の流れについて把握できる。この経路観測に対応する API の定義を行った。2 地点を指定することで、その道路を走行中の車両から情報を収集する。

表 5.2: API リスト

		引数					戻り値			
観測名	API名	車両ID	位置	時刻	時間帯	センサ名	センサの値	位置	時刻	車両ID
定点観測	getAllSensorData()	○	○				○	○	○	
			○				○	○	○	○
	getSensorData()	○	○			○	○	○	○	
		○	○	○		○	○	○	○	
		○	○		○	○	○	○	○	
		リスト	○			○	○	○	○	○
		リスト	○	○		○	○	○	○	○
		リスト	○		○		○	○	○	○
	getVehicleList	リスト	○					○	○	○
		リスト	○	○				○	○	○
		リスト	○		○			○	○	○
	getAverageData()	リスト	○		○	○	平均値			○
	getMaxData()	リスト	○		○	○	最大値			○
	getMinData()	リスト	○		○	○	最小値			○
経路観測	getAllSensorData()		リスト				○	○	○	
		リスト	リスト				○	○	○	○
	getSensorData()	○	リスト			○	○	○	○	
		○	リスト	○		○	○	○	○	
		○	リスト		○	○	○	○	○	
		リスト	リスト			○	○	○	○	○
		リスト	リスト	○		○	○	○	○	○
		リスト	リスト		○		○	○	○	○
	getVehicleList	リスト	リスト					○	○	○
		リスト	リスト	○				○	○	○
		リスト	リスト		○			○	○	○
	getAverageData()	リスト	リスト		○	○	平均値			○
	getMaxData()	リスト	リスト		○	○	最大値			○
	getMinData()	リスト	リスト		○	○	最小値			○
衛星観測	getAllSensorData()	○	範囲				○	○	○	
		リスト	範囲				○	○	○	○
	getSensorData()	○	範囲			○	○	○	○	
		○	範囲	○		○	○	○	○	
		○	範囲		○	○	○	○	○	
		リスト	範囲			○	○	○	○	○
		リスト	範囲	○		○	○	○	○	○
		リスト	範囲		○		○	○	○	○
	getVehicleList	リスト	範囲					○	○	○
		リスト	範囲	○				○	○	○
		リスト	範囲		○			○	○	○
	getAverageData()	リスト	範囲		○	○	平均値			○
	getMaxData()	リスト	範囲		○	○	最大値			○
	getMinData()	リスト	範囲		○	○	最小値			○
移動観測	getAllSensorData()	○	○				○	○	○	
		○	範囲				○	○	○	
		○	○		○		○	○	○	
		○	範囲		○		○	○	○	
		○			○		○	○	○	
	getSensorData()	○	○			○	○	○	○	
		○	範囲			○	○	○	○	
		○	○		○	○	○	○	○	
		○	範囲		○	○	○	○	○	
		○			○	○	○	○	○	
	getAverageData()	○			○	○	平均値			
	getMaxData()	○			○	○	最大値			
	getMinData()	○			○	○	最小値			
	getCurrentPosition()	○						○(現在)		
	getCurrentSensorData()	○				○	○	○	○	
	getCurrentAllSensorData()	○					○	○	○	

衛星観測

衛星観測では一定の位置範囲に絞って車両の動向を見る。衛星観測に対応した API では位置範囲を指定すると供に時刻・センサ名などを併せて指定することによりある地域にいる車両の情報を取得する。例えば、表の衛星観測の欄の最後の getMinData() では車両一覧と地域、時間帯、センサ名を指定することで、指定された地域内、時間帯に走行していた車両から指定したセンサの情報を取得する。

移動観測

移動観測では時間を基準として、指定された時間帯の車両情報を取得する。また指定した位置内にいる場合の車両情報を取得する。そうすることで、各車両がどういった状態変異をしたかを見ることができる。表の移動観測の欄の一番上の getAllSensorData() では、車両 ID および位置を指定することで、指定した車両がその位置を通過した履歴情報を取得する。

5.3 システムの動作概要

車両情報蓄積システムおよびインターフェースを用いた全体のシステムの動作概要に図 5.3 ついて述べる。

1. 蓄積する必要のある車両情報に関して SRM を用いて記述
2. 記述された SRM を車両へ送信し、指定した車両情報の蓄積を車両内で開始
3. 蓄積されている車両情報は各サービスに応じて利用しやすい API のメソッドを利用し車両情報を収集
4. サービス終了後、蓄積された車両情報の削除

サービス提供を例にシステムの動作概要を見ていく。まず一番目として、サービスが必要とする車両情報に関して SRM に記述する。SRM には必要となるセンサ名や蓄積間隔を記述する。次に２番目として記述した SRM をサービスが車両に送信し、車両は SRM に従い指定された車両情報の蓄積を開始する。３番目として、蓄積された車両情報へアクセスする API のメソッドを用い、必要な情報をサービスが収集する。最後にサービス終了後、蓄積された車両情報の削除を行う。

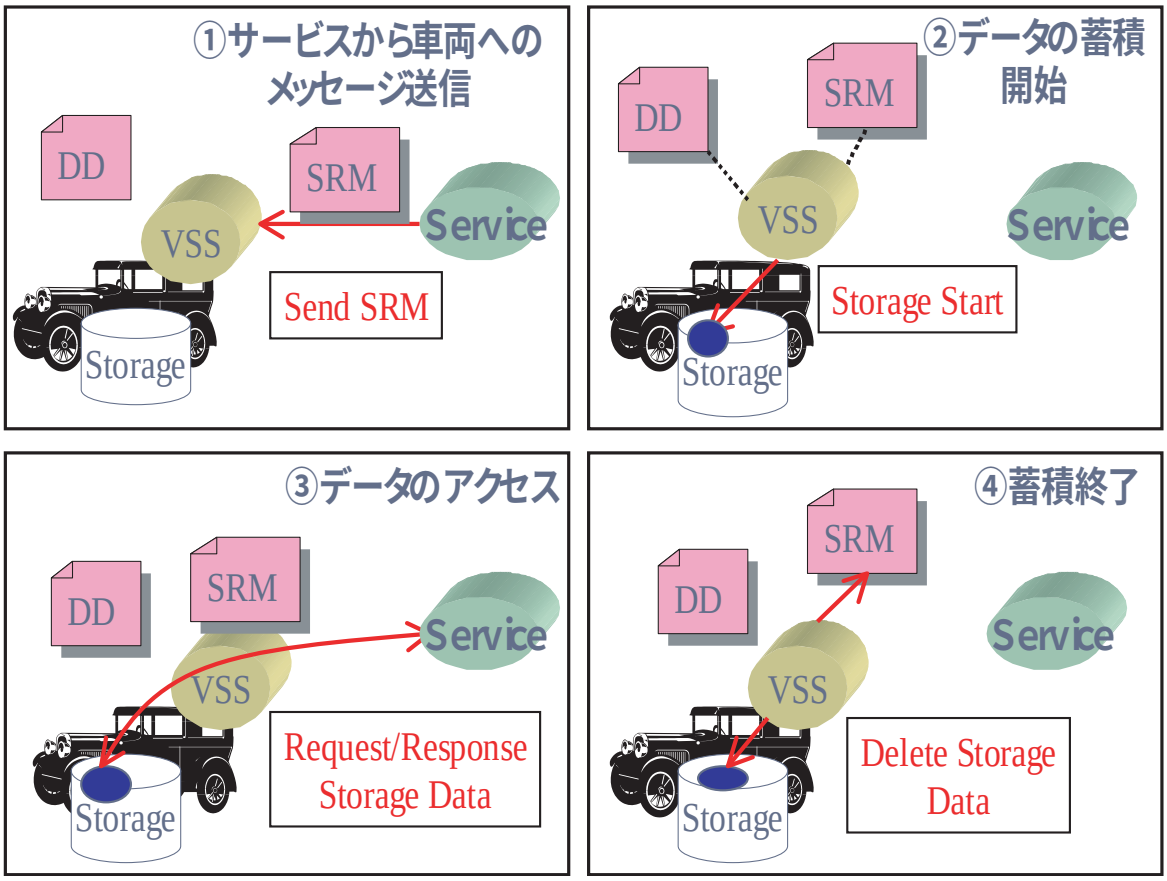

図 5.3: システム動作概要図

第6章　実装

設計に基づきプロトタイプシステムの実装を行った。本プロトタイプシステムでは第５章で行った設計のうち一部の機能を実現しており、以下の二つの機能で構成される。

- サービスから車両へ、センサ情報の蓄積を要求する機能
- 車両に蓄積された情報へアクセスするためのインターフェース

一つ目はSRMを利用したシステムで、サービスが必要とする車両情報の蓄積を車両に対して指示するためのシステム (以下、SRMシステム) である。二つ目としてサービスが指定して蓄積された車両情報にアクセスするためのシステム (Vehicle Storage Reuqest(以下、VSRシステム)) を構築した。

本システムの開発は次の表6.1の環境で実装を行った。両システムともWeb Services[27]を用い構築し、車両情報を蓄積するためにRDBのPostgreSQLを利用し車両情報を蓄積した。

表 6.1: 開発環境

開発言語	J2SDK 1.4.1
蓄積容量	PostgreSQL 7.4
HTTP Server	Apache 2.0.43
Servlet Engine	Apache Tomcat 4.1.29
SOAP Engine	Axis 1.1
DataBase Connector	JDBC 1.2

6.1　SRMシステム

本節では項目の一つ目であるSRMを利用した車両情報蓄積の指定を行うためのシステムについて述べる。

本システムではサービスは車両内に蓄積する必要のある情報に関してSRMに記述を行い車両にSRMをSOAP/HTTP[8]で車両に送信する。受けとった車両は指定された線さ情報をRDBに蓄積する。

以下にSRMシステムで構築したWeb Service Definition Language(WSDL)[9]を一部抜粋して図6.1に示す。

上記で示したように、サービスがSRMを車両へ引数として投げる。車両はSRMを受け取り指定された記述に従い車両情報の蓄積を開始する。本システムではSRMの指定されたセンサ名をデータベースのテーブルに追加をすると供にセンサ情報を作成されたテーブルに挿入する。以下にテーブルの例を示す。初期設定としてtime(時刻)、lat(緯度)、lon(経度)、alt(高度)がありSRM

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions>
            :
<complexType name="Push">
<sequence>
<element name="interval" type="xsd:string" />
<element name="returned" type="xsd:string" />
<element name="senderip" type="impl:ArrayOf_xsd_byte" />
<element name="start" type="xsd:dateTime" />
<element name="stop" type="xsd:dateTime" />
</sequence>
</complexType>
<complexType name="Correct">
<sequence>
<element name="attri" type="xsd:string" />
<element name="value" type="xsd:string" />
</sequence>
</complexType>
<complexType name="Oversides">
<sequence>
<element name="senderip" type="impl:ArrayOf_xsd_byte" />
<element name="returned" type="xsd:string" />
</sequence>
</complexType>
<complexType name="Sensor">
<sequence>
<element name="name" type="xsd:string" />
<element name="interval" type="xsd:int" />
<element name="push" nillable="true"
type="impl:ArrayOf_impl_Push" />
<element name="oversides" nillable="true" type="tns1:Oversides" />
<element name="correct" nillable="true"
type="impl:ArrayOf_impl_Correct" />
</sequence>
</complexType>
</schema>
            :
</wsdl:definitions>
```

図 6.1: SRM システムで利用する WSDL(一部抜粋)

の要求に応じて必要なコラムを作成し車両情報を蓄積する。図 6.2 の場合、templeture(温度) および humidity(湿度) を加えてある。

```
simulator=> SELECT * from sensors ;
        time                  — lat     — lon     — alt  — tempreture  — hu-
midity
——————————————————————+———+———+——-+————+———-
2003-12-14 09:44:07.126021 — 3520000 — 12800000— 0 — 15 — 40
2003-12-14 12:04:55.987959 — 3510000 — 12810000— 0 — 18 — 40
2003-12-14 12:09:24.932292 — 3511000 — 12811000— 0 — 20 — 50
```

図 6.2: テーブル例

6.2 VSR システム

VSR システムでは前小節で指定し蓄積された車両情報を必要に応じてサービスが取得する機能を持つ。本システムの一つの API を例にあげる。

例で述べる API ではサービスが、車両情報を取得したい車両 ID、センサ名、位置、時刻を引数として、車両から蓄積されたセンサ情報を取得する。次に図 6.3 挙げるのがそのメソッドとなる。　このメソッドでは引数を元に車両情報のサービスに接続を行い必要となる車両情報の一覧

```
public Vector getSensorData( String ID, String sensorname, Pisition position,
      Calendar time){
   Vector vector = new Vector();
      :
   //サービスへ接続
   Vsr vsr = service.getvsr( url );
      :
   vector = vsr.getVssData( vec );
   return vector;
}
```

図 6.3: API のメソッド例

を Vector 形式で取得する。車両とサービスの間では次のようなデータのやり取りを行う。WSDL の一部抜粋したものを図 6.4 に示す。　この例では引数及び変数を Vector 形式でデータのやり取りを行った。サービスからのリクエストを受け取った車両のシステムでは引数を元に車両情報を

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions>
            :
<wsdl:types>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://xml.apache.org/xml-soap">
<import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
<complexType name="Vector">
<sequence>
<element name="item" minOccurs="0" maxOccurs="unbounded"
type="xsd:anyType" />
</sequence>
</complexType>
</schema>
            :
<wsdl:message name="getVssDataRequest">
<wsdl:part name="vssElements" type="apachesoap:Vector" />
</wsdl:message>
<wsdl:message name="getVssDataResponse">
<wsdl:part name="vssData" type="apachesoap:Vector" />
</wsdl:message>
            :
</wsdl:definitions>
```

図 6.4: VSR で利用する WSDL(一部抜粋)

SQL を用い RDB から取得して車両へ結果を返す。

第7章　実証実験

本章では設計に基づき構築したシステムを用い実証実験をおこなった。実験の検証項目として以下の二つの項目の検証を行った。

1. 車両情報管理モデルのコスト性
2. モデルの規模性

7.1　車両情報管理モデルのコスト性

車両情報管理モデルでは、サービスが必要とする情報を車内に蓄積することで通信量、データ量の削減を行い、車両情報を利用したサービスの開発、運用のコストを抑え、よりサービスへ参入しやすい環境を構築することを目的としている。

そこで実証実験では、本研究で提案した車両情報管理モデルのコスト性の検証を行う。定期的に車両情報をセンタへ収集するモデル (以下、定期収集モデル) と比較し、車両情報管理モデルのコストの削減率を検証する。

7.1.1　実験項目

実験では次の３つのサービスを両モデルを利用して構築する。

- リアルタイムスピード情報
- 交通量サービス
- 車両動態管理

リアルタイムスピードサービスでは現在のスピード情報を一秒間隔で収集するサービスとする。次に交通量サービスでは最近一ヶ月の交通量を過去にさかのぼって調査するサービスとする。最後の車両動態管理ではある一台の車両に注目し一週間ごとの平均を元に、一年間の動態推移を見るサービスとする。これら３種類のサービスを両モデルを用いて構築し、コストの比較検討を行う。この３つのサービスを構築することで、第３章で述べた車両情報の各分類それぞれの特徴を利用することができる。コストを比較する項目としては次の３項目とする。

- 通信量
- データ量
- サービス数

一つ目の項目として通信量を比較する。車両情報を利用したサービスを行うためにはセンタ-車両間で通信する必要がある。通信費用は多数の車両へサービスを行うサービスにとってもサービスの利用者にとっても大きな負担となる。そのため本モデルによって通信量がどの程度軽減されたかについての検証を行う。

二つ目の項目としてセンタに収集されるデータ量を比較する。各種サービスを構築するためにセンタに収集する必要のあるデータ量を比較することでサービスを行う際のセンタの規模について検討を行う。車両から収集するデータ量が抑えられれば、運用に関するコストの低減ができる。

最後の項目としてサービス数を比較する。一台の車両が利用するサービス数が増える場合、車両への通信量が増加する。そのため一台の車両が利用するサービス数が増加することで、通信、データの増加量を検証する。

これらの項目を両モデルで比較し、車両情報管理モデルのコスト性についての検証を行う。

実験では実装したシステムを実車に乗せ、実際の車両情報取得時間、データ量を計測する。その結果を元に各種サービスを構築し、両モデルの比較、検討を行う。

7.1.2 実験環境

以下に実験環境 7.1 及び実験の構成図 7.1 について示す。車両側のシステムを載せるために soekris を用い、モバイルルータを経由しサービスからアクセスできるようにした。また、本実験では IPSensor を用い SNMP を利用しセンサ情報を蓄積した。蓄積された車両情報を実装したシステムを用いて車両情報の取得を行う。通信機器には b-mobile を利用した。

表 7.1: 実車実験環境

	車両側 (soekris)	サービス側 (DELL Dimension4300)
CPU	Geode 266 MHz	Pentium4 1.5 GHz
OS	FreeBSD-4.9R	Vine Linux 2.6r1
MEMORY	128M	512M
通信機器	b-mobile	

7.2 モデルの規模性に関する実験

車両情報を利用したサービスは多数の車両に対してサービスを提供することが多い。そのため、車両情報管理モデルの規模性を検証することで、多数の車両に対して行うサービスへの実現可能性を検証する。

まず実験で利用した HAKONIWA(シミュレータ) について述べる。次にモデルの規模性の検証方法について述べる。

7.2.1 HAKONIWA

HAKONIWA(箱庭)[31] とは都市内の交通状況を計算機上で模擬し、実車のプローブ情報と同等の模擬情報をシミュレーションする環境である。実際に車両を利用した実証実験を行うには費

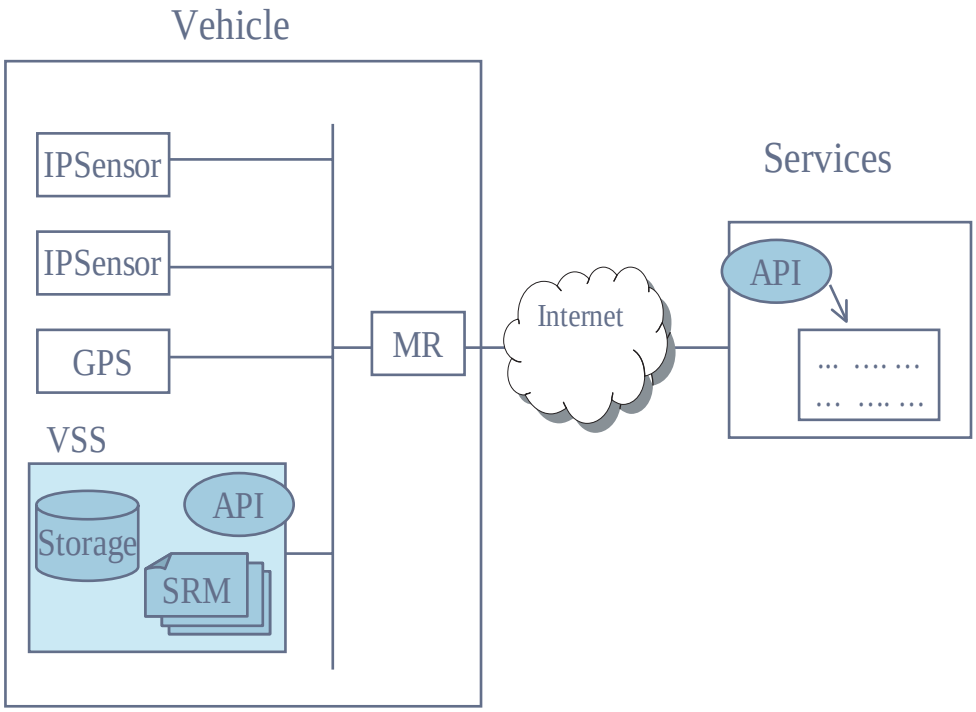

図 7.1: 実車実験環境図

用や時間の問題から実施できる規模が制限される。そのため、計算機上でシミュレーションを行うことにより、費用を抑え大規模な実験が行うことができる。また、このシミュレーションに交通流シミュレータや通信シミュレータとを組み合わせて利用することにより、より現実に近い環境でのインターネット ITS に関するアプリケーションの実験を行うことができる。通信環境や気象環境などの情報を組み込んだ上で各車両の情報を生成する。この生成された各車両の情報を利用することで、複数の車の車両情報を扱いアプリケーションの検討が可能である。

7.2.2 実験概要

前節で述べた HAKONIWA を利用して実験を行った。実験では車両情報管理モデルを用いてサービスを構築し、サービスに必要となる台数を求めることでサービスの実現性を検証する。

この検証では定点観測に着目し、全体の走行車両から車両情報を取得した際、情報を収集できる地点数を測る。それを元に全体の車両に対するプローブ車両の台数を変化させ車両情報取得可能な地点の増減を見る。この実験を行うことで、 サービスに必要となる情報に対する車両数を図ることができる。

本実験では表 7.2 の実験環境を用いて実験を行った。 実験構成図を図 7.2 に示す。HAKONIWA が生成した各車両の情報の蓄積を行い、本モデルの検証を行った。図のようにシミュレーションが生成した各車両の情報をそれぞれデータベースに蓄積し、蓄積された情報をもとに検証を行った。

表 7.2: 規模性実験環境

	各車両マシン (DELL Dimension4300)	シミュレータ
CPU	Pentium4 1.5 GHz	Xeon 2.8GHz × 2
OS	Vine Linux 2.6r1	WindowsXP Professinal
MEMORY	512M	512M

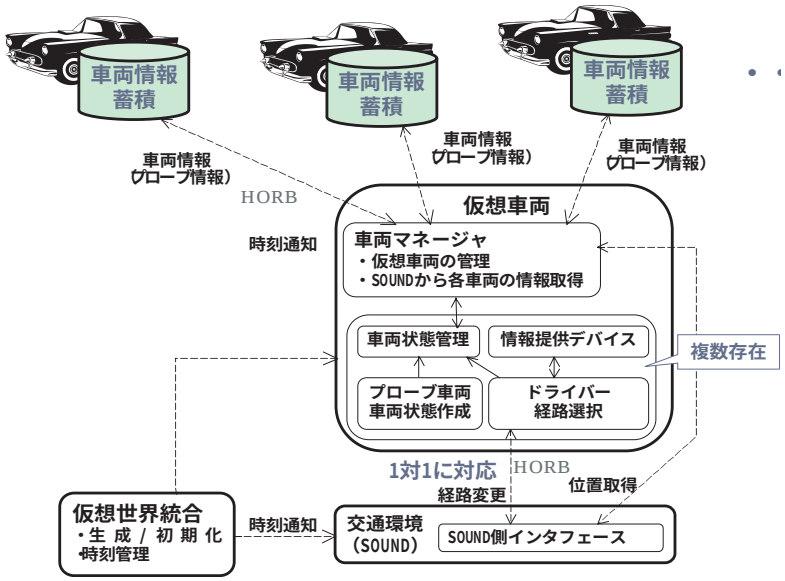

図 7.2: 規模性実験環境

7.3 実験結果

コスト性の検証を行うため行った実験結果を述べる。

車両内に蓄積されている情報全てを取得するのにかかった取得時間の結果を示す。この実験では車両走行中の１時間サービスから車両に対して全ての車両情報を取得するリクエストを出した。次の図 7.3 にその経緯を示す。

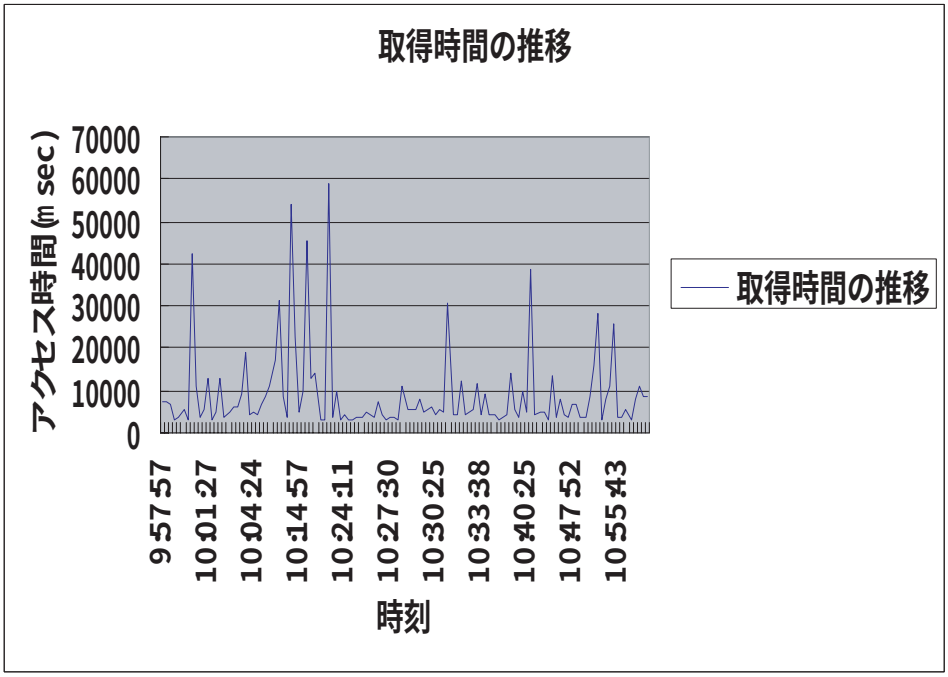

図 7.3: 情報取得時間の推移

この図で見られるように、車両情報を取得する時間は大きく変化している。車両情報を取得するための最小取得時間は 2,776 ミリ秒、最大取得時間は 58,724 ミリ秒と一分近く取得するための時間がかかる。また、平均取得時間は 9,015 ミリ秒だった。

図 7.3 のグラフを統計処理したものを図 7.4 に示す。車両情報の取得時間は 8,000 ミリ秒から 16000 ミリ秒に集中している。また、16,000 ミリ秒以内に 80 %ほどの情報が取得できることがわかる。

次に蓄積車両情報全てをサービス側から取得した時間と通信状況を比較するため、Ping によるネットワークの状態、データベース検索時間の比較結果を示す。図 7.4 に Ping の結果、車両取得時間、データベース検索時間の比較グラフを示す。このグラフは比較しやすいように平均値を合わせて描写している。図で分かるように Ping にかかった時間と同様に車両情報の取得時間もかかった。

車両に対するサービスの増加による車両情報の取得時間、データベースの検索時間の変化の結果を示す。図 7.6 ではサービス数すなわち車両に対して車両情報へのリクエストを増加させた結果、サービス数が一つ増加することによって蓄積車両情報への取得時間がおおよそ 1.5 倍かかった。また、車両情報の取得時間ほど増加は見せなかったものの、データベースの検索時間も増加している。

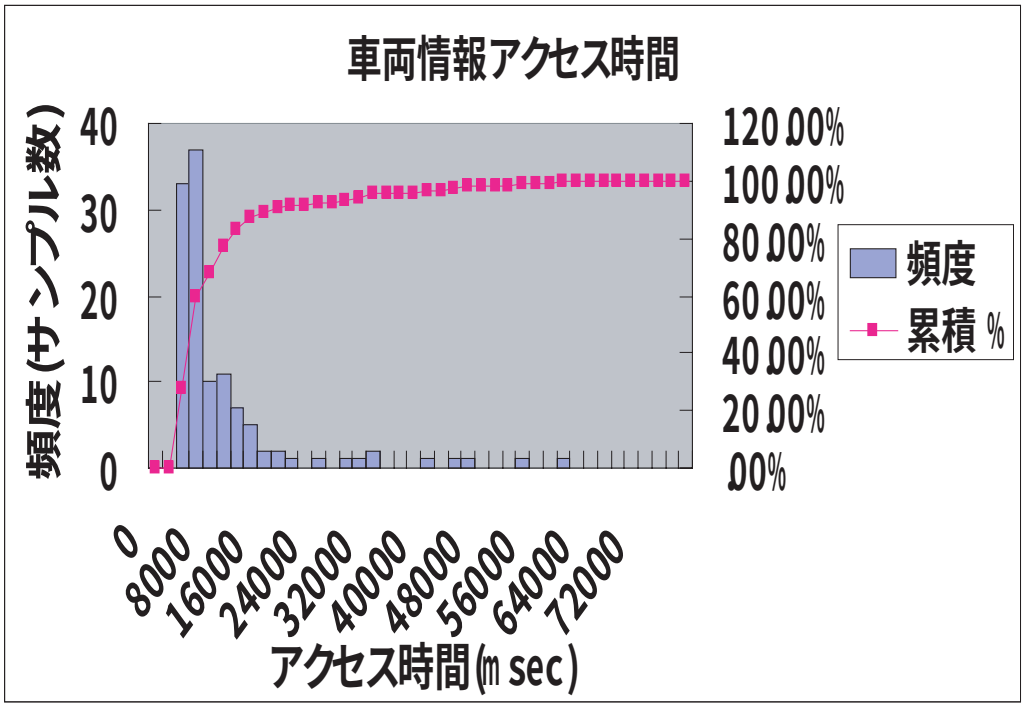

図 7.4: 情報取得時間と通信環境との比較

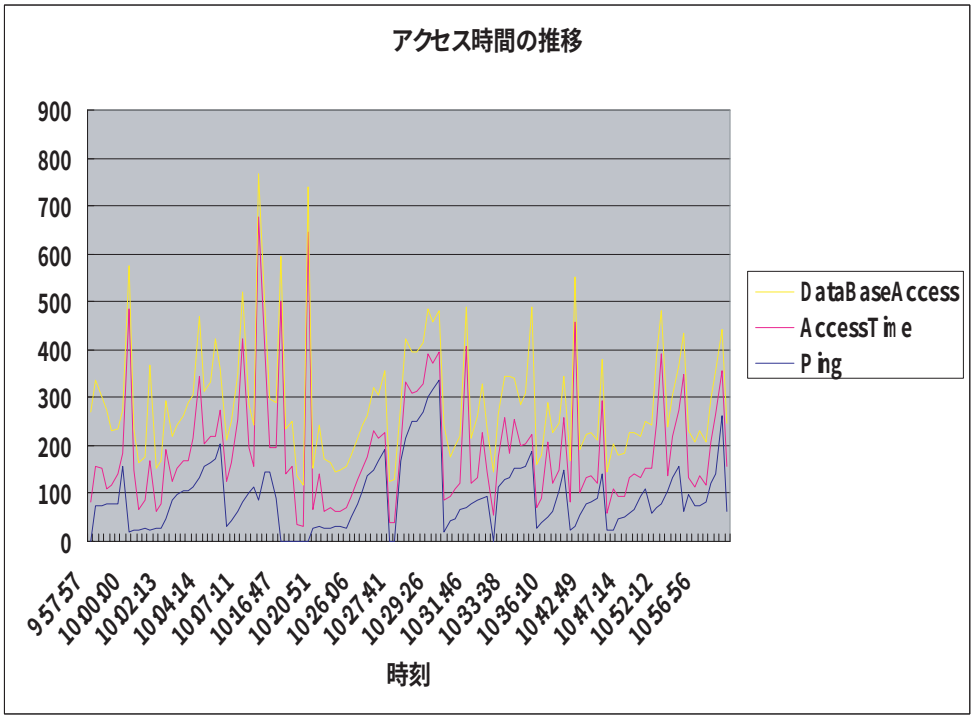

図 7.5: 情報取得時間と通信環境との比較

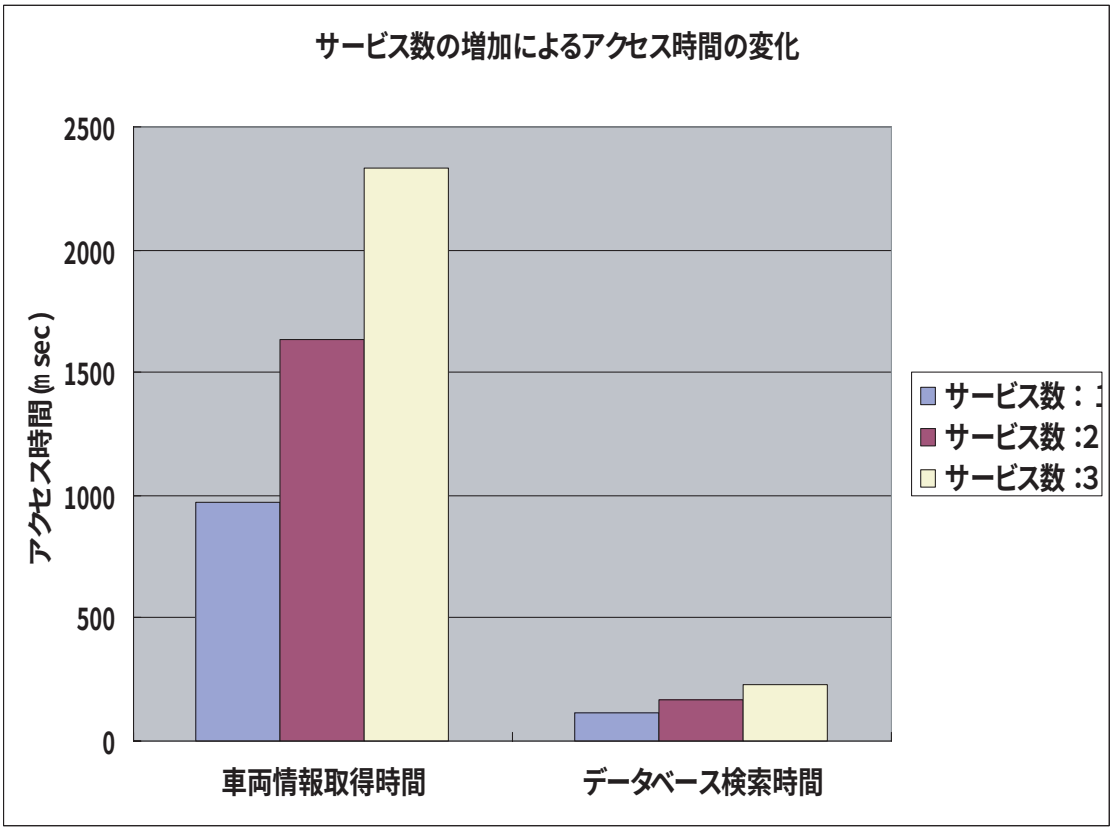

図 7.6: サービス増加による取得時間の変化

第8章　評価

本章では、まず本研究で提案した車両情報管理モデルの定性評価を行った。次に定性評価として前章でおこなった実証実験の結果を受けて車両情報管理モデルのコスト性、モデルの有用性の評価を行う。

8.1　定性評価

本研究では、サービス、車両にコストをかけない形で車両情報を利用するため車両情報管理モデルの提案を行った。モデルではサービスが必要な車両情報を車両内に蓄積することで、サービスはセンタへ必要な情報のみを収集することができる。また、サービスに共通した API を定義することで、サービス形態にあった車両情報を取得できる。以上のことを表 8.1 でまとめる。

表で示すように車両情報を利用したサービスを構築するための要件として大きく二つを上げた。コストの削減とサービスに共通したインターフェースの定義の必要性である。コストの削減では通信量やセンタに収集するデータ量を削減することで、サービス参入の障壁を低くする。現在は定期的な車両収集方法は通信量やデータ量が多くなる。しかし、車両情報管理モデルでは車両内に情報を蓄積するため、サービスに必要な情報のみを通信するため通信量の削減ができる。必要な情報のみをセンタに収集するためデータ量の削減ができる。またサービスごとに車両情報を蓄積するため車両情報を複数のサービスから利用できる。

車両情報を取得するためのサービスに共通したインターフェースを本研究で提案した。現在のインターフェースはサービスに特化したもので、他のサービスからは利用しにくいが、提案したインターフェースでは各種サービスを分類し、それらを網羅するようインターフェースの定義を行った。

8.2　定量評価

本節では各実証実験の結果を元にコスト性評価、モデルの規模性評価を行う。

8.2.1　車両情報管理モデルのコスト性評価

第 7 章で行った実験結果を受けて、サービス増加による車両情報取得時間の推移を図 8.1 に示す。通信環境がよい場合にはサービスが 10 ほどあっても車両情報が取得可能であるが、通信状態が悪化している状況ではタイムアウトしてしまう可能性が高い事がわかる。そのため現在のシステムでは車両は多数のサービスの利用ができない。

次に車両情報管理モデルと定期収集モデルとの比較を行う。

まず、両モデルでは次のような前提 8.2 の下にサービスの構築を行うこととする。現在のモデルではセンタ側に一秒に一回車両情報を送信する。一方、本研究で提案するモデルではサービス

表 8.1: 定性評価

要件	現状	コメント	結果	コメント
コストの削減				
車両情報をサービスの要求に従い車内に蓄積できること	×	サービスの要求に従った車両情報の蓄積ができない	○	SRM を利用したシステムを構築し、サービス側の指定した車両情報を蓄積できた
サービスを行う上でセンタ-車両間の通信量を削減すること	×	定期的に車両情報を収集するなど通信量が多い	○	定期収集モデルと比較した実験を行い、サービスに必要な情報のみを収集することで通信量を大幅に削減できた
サービスを行う上でセンタに収集するデータ量を削減すること	×	定期的にセンタに収集されるためデータ量が多い	○	センタに収集する情報をサービスが必要な分のみとすることで、センタに収集されるデータ量の削減ができた
車両情報を複数のサービスから利用できること	×	各サービスに依存するため、他のサービスから利用できない	△	少数のサービスに限り利用可能
サービスに共有したインターフェースの定義				
サービスが必要とする車両情報のみを収集できること	×	実験やサービスに特化した方法でした収集できない	○	センサ情報を空間、時間軸で分類しそれぞれサービスにあったインターフェースを利用できるよう定義した
サービスが複数の車両に対してサービスを提供できること	△	複数のサービスに対して提供できるが、大規模なサービスは困難	△	複数の車両に対してはサービスできるものの多数の車両に対するサービス提供実験が行えなかった。

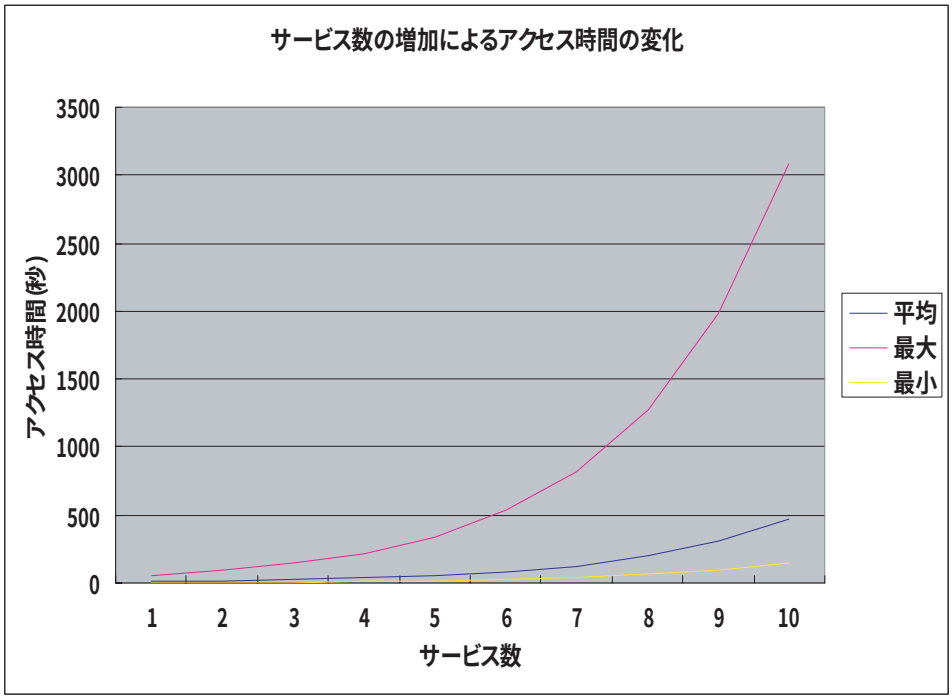

図 8.1: サービス増加による取得時間の変化の推移

に応じた車両情報を適宜取得する。取得するデータ及び取得した車両情報をセンタへ蓄積する方法は同じとする。また、一秒に一回センサから生成された情報をデータ１セットとする。通信量は実験で行った際に利用した通信量を元に比較を行う。

表 8.2: モデルの前提

	定期収集モデル	車両情報管理モデル
データの取得間隔	1 秒に一回	任意 (サービスごとに指定)
車内へのデータの蓄積間隔	行わない	一秒に一回
取得方法	Web Services(XML)	
取得するデータ	時刻、位置及び全てのセンサ情報	
センタ側のデータ蓄積方法	同様の形式で取得 (データ１セット：28 バイト)	
通信量	2130 バイト＋取得データ数× 250 バイト	

次に各サービスに利用する車両情報の要件を次の表 8.3 に定める。車両は一年間、毎日二時間走行することとする。

この前提を元に両モデルの比較結果を次の表 8.4 に示す。各サービスを両モデルを用いて構築し、その通信量、データ量の結果を示す。リアルタイムの車両情報取得サービスでは定期収集モデルが車両情報管理モデルと同じ通信量で納まっている。

次に交通量サービス、動態管理サービスの結果を見る。車両情報管理モデルのほうが定期収集モデルにくらべ通信量、データ量ともに非常に低く抑えられている。定期収集モデルでは常に情

表 8.3: サービス内容

	リアルタイムスピード情報	交通量サービス	車両動態管理
サービス期間	一年間		
車両情報利用時間	毎秒	一ヶ月 (毎分)	毎月の平均値
利用するセンサ	スピード、方位	スピード、方位	燃料、エンジン回転数

報をセンタに収集しているため、リアルタイムサービスと通信量、データ量が変わらない。しかし、車両情報管理モデルでは車内に情報を蓄積することで必要な情報のみをセンタに収集でき、通信量、データ量ともに大きく削減できた。

以上のことから車両情報管理モデルを利用することで、サービスが必要な情報のみを収集することで通信量、データ量を削減でき、コストを抑えることが可能となった。課題として、最新の車両情報を収集するために軽量なプロトコルを利用することで通信量を抑える必要がある。

表 8.4: モデルの比較結果

		定期収集モデル			車両情報管理モデル		
	サービス数	1	5	10	1	5	10
リアルタイムスピード情報	通信量	6254.64	31273.2	62546.4	6254.64	31273.2	62546.4
	データ量	73.584	367.92	735.84	73.584	367.92	735.84
交通量サービス	通信量	6254.64	31273.2	62546.4	0.902	4.511	9.021
	データ量	73.584	367.92	735.84	0.101	0.504	1.008
車両動態管理	通信量	6254.64	31273.2	62546.4	0.005	0.026	0.051
	データ量	73.584	367.92	735.84	0	0.002	0.003

8.2.2 モデルの実現に必要となる台数

この実験では全体の車両に対するプローブ車両の台数を変化させることで、定点観測を行うためにどれだけ車両が必要かを見ていく。次に全車両からデータを取得できた場合の様子を図 8.2 に示す。

今回の実験では交差点を一地点として数え、一地点に 3 台以上通過していることが確認された場合カウントを行った。全車両から情報を取得する場合、49 地点からの情報が取得可能となる。次の表 8.5 にプローブ車両の推移及び観測可能な地点数の推移を記述する。表のように車両台数と比較して、地点数の割合が総じて高いことがわかる。これは交差点では曲がることで速度が下がったり、信号待ちをすることで時間がかかっていることが考えられる。そのため、必要となる地点数は全体の車両台数と比較して低い割合で情報を取得することができる。本実験では走行中の車両から情報を取得したためこうした割合になっているが、サービスを行う際には実際に走行していない車両が多いと考えられるため、それらを考慮してサービス展開を図る必要がある。

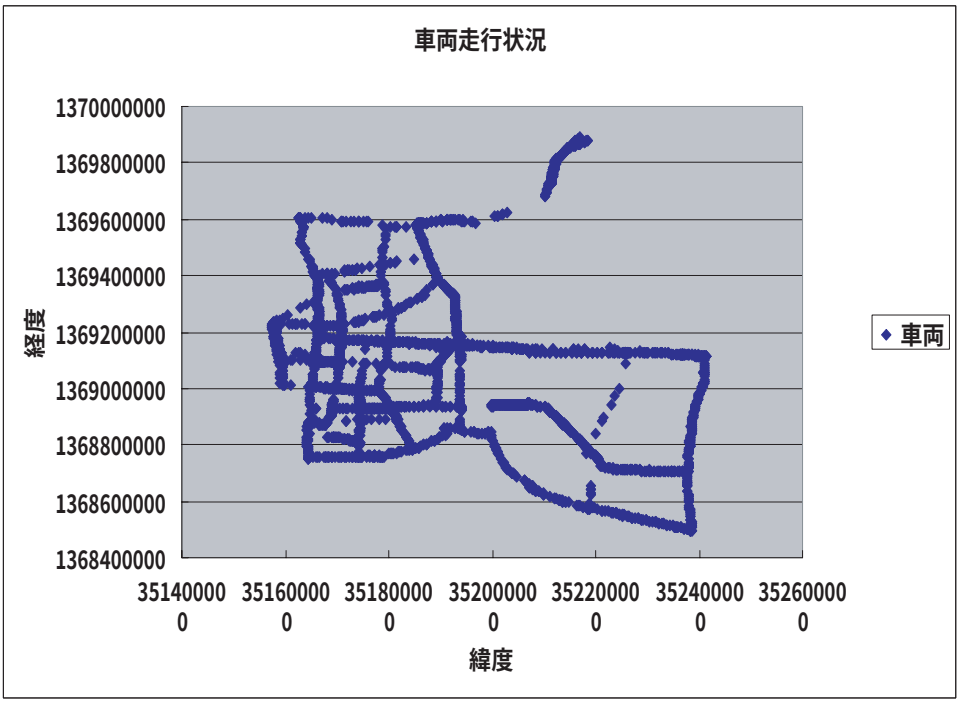

図 8.2: シミュレーションによる車両走行の様子

表 8.5: 観測地点の割合

車両台数の割合	地点数	地点数の割合
100 %	49	100 %
90 %	48	97 %
70 %	45	91 %
50 %	42	85 %
30 %	35	71 %
10 %	13	26 %
5 %	5	10 %

8.2.3 定量評価のまとめ

実際の通信環境を利用した実験及び規模性実験の結果により次のことが言える。

まずはじめに、通信環境を利用した実験ではサービス側が指定した車両情報を蓄積し、必要となる車両情報を取得することができた。サービス側はサービス内容に応じた API を利用することで車両情報を任意に取得可能となった。車両情報を取得する時間は通信環境に大きく依存し、8 割強は数秒程度で取得できるが、通信状況によってはタイムアウトしたり非常に時間かかることが確認された。

上記の実験を元に現在のモデルと本研究で提案したモデルをいくつかサービス例を構築することで、通信量、データ量の比較を行った。過去にさかのぼったり統計処理するサービスでは、サービスに利用する情報のみを取得することで大きく通信量、データ量の削減できた。

次にモデルの実現性についての検証では、交通状況にも影響されるものの、サービスによっては定点観測で見られるように少ない車両だけでも必要な情報が取得できることが確認された。

第9章　結論

9.1　まとめ

本研究では車両情報の管理及び利用方法の提案を行った。サービス側から必要となるセンサを指定することでそれら情報の蓄積を可能とすると供に、蓄積された車両情報をアクセスするためのインターフェースの提案を行った。また、本モデルに従いシステムの構築、実証実験を行い車両情報の有効な利用が可能となった。

本研究によりサービス側が必要となる車両情報を容易に利用が可能となる環境を構築した。これにより車両情報を利用したサービスの普及、推進できる。

9.2　今後の課題

今後の課題として規模性、通信環境への対応が挙げられる。

規模性の問題点として、本システムではサービス側が多数の車両に対して車両情報を取得するには多くのアクセス時間がかかる。車両情報を取得するインターフェースに並列処理を施すなど効果的に多数の車両から情報を取得できるシステムが必要となる。また、車両情報を取得するためのインターフェースとして本研究では Web Services を利用したが、インターフェースを統一し規格化するにあたりどういったインターフェースが最適化を検討する必要がある。

通信状況によって車両情報の取得時間が変化し、最悪の場合には車両が走行時にもタイムアウトを起こす可能性がある。通信機器の切り替えへの対応やメッセージングを利用することで通信環境がよい場合に車両情報が取得できる必要がある。

謝辞

本研究を進めるにあたり、ご指導いただきました、慶應義塾大学環境情報学部教授 村井純博士、並びに同学部教授 徳田英幸博士、同学部助教授 楠本博之博士、同学部教授 中村修博士、同学部専任講師 南政樹氏に感謝いたします。

また、常日頃からご指導頂きました慶應義塾大学院政策・メディア研究科 特別研究専任講師 植原 啓介博士に深く御礼申し上げます。研究を進めるにあたりご指導、ご助言いただきました慶應義塾大学大学院 政策・メディア研究科 特別研究助手 佐藤雅明氏、同大学大学院 SFC 研究所 研究員 渡辺恭人博士、同大学大学院修士課程 日野哲志氏、西原 サヤ子氏に感謝いたします。

本研究を進めていくにあたりご支援くださった慶應義塾大学環境情報学部・総合政策学部の徳田村井 楠本 中村 南研究室ならびに同研究室 NACM 研究グループの皆様に感謝の念を表します。

参考文献

[1] 渡辺恭人, 竹内奏吾, 栗栖俊治, 寺岡文男, 村井純:”プライバシー保護を考慮したちり位置情報システムの実装と評価”, 電子情報通信学会論文誌 B Vol.J86-B No.8 pp.1434-1444 2003 年 8 月

[2] 前川誠:”ITS の動向と NEC の取り組み”, NEC 技報 第 54 巻第 7 号 (通巻 389 号) 2001 年 7 月

[3] 財団法人日本自動車研究所 (http://www.jari.or.jp)

[4] 博士論文 植原啓介:”移動体インターネットと自動車社会の協調システムアーキテクチャに関する研究”, 2002

[5] Simple Network Management Protocol(http://net-snmp.sourceforge.net/)

[6] Extensible Markup Language(http://www.w3.org/XML/)

[7] RELAX NG(http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=relax-ng)

[8] Simple Object Access Protocol(http://www.w3.org/TR/SOAP/)

[9] Web Service Definition Language(http://www.w3.org/TR/wsdl)

[10] 「平成 14 年度情報経済基盤整備事業 インターネット ITS に係る基盤要素技術開発報告書」 慶應義塾大学 SFC 研究所, 2003

[11] Advanced Traffic Information Services(http://www.atis.co.jp)

[12] Intelligent Community Vehicle System(http://www.atis.co.jp/)

[13] internavi(http://www.internavi.ne.jp/)

[14] CARWINGS(http://www.nissan-carwings.com/)

[15] G-BOOK(http://g-book.com/pc/)

[16] OnStar(http://www.onstar.com/)

[17] NetworkCar(http://www.networkcar.com/)

[18] インターネット ITS 協議会 (http://www.internetits.org)

[19] ISO(http://www.iso.ch)

[20] 「インターネット ITS 研究開発報告書」 慶應義塾大学インターネット ITS 共同研究グループ, 2002

[21] 田島隆之, 若山公威, 佐藤龍哉, 時津直樹, 岩田彰:”インターネット ITS におけるプローブデータ通信量の削減”, 高度交通システム 12-10, 2003.3.7

[22] 「プローブ情報収集のための統合型車載システムの開発に関するフィージビリティスタディ報告書」 財団法人 機械システム振興協会, 委託先 (財) 自動車走行電子技術協会, 2003

[23] 目黒浩一郎, 平岡規之, 杉浦孝明, 清水新太郎:”輸送事業分野におけるプローブ情報システムの活用と普及に関する研究”, 研究論文 三菱総合研究所/所報 No.39, 2001

[24] Vehicle Information and Communication System(http://www.vics.or.jp/)

[25] Electronic Toll Collection(http://www.jhnet.go.jp/etc/)

[26] 町田潤郎:"ETC(自動料金支払) システムの開発", NEC 技報 第 54 巻第 7 号 (通巻 389 号) 2001 年 7 月

[27] Web Services(http://www.w3.org/2002/ws/)

[28] Advanced Cruise-Assist Hightway System(http://www.its.go.jp/ITS/j-html/whatsITS/safety.html)

[29] Research and Development Turner-FairBank Highway Research Center:"Design of an ITS-Level Advanced Traffic Management System", Publication No.FHWA-RD-95-181, July 1996

[30] 「基準認証研究開発事業プローブ情報システムにおける個人情報保護に関する標準化」 慶應義塾大学 SFC 研究会, 2003

[31] 春田仁, 渋谷裕久, 堀口良太, 赤羽弘和, 森川高行, 中村秀樹, 植原啓介:"インターネット ITS アプリケーション検証シミュレータ", 第 27 回土木計画学研究発表会講演集，No.27，CD-ROM，2003

付 録A　Storage Request Message

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
<define name="srm">
<element name="srm">
<ref name="attlist.srm"/>
<ref name="servicename"/>
<zeroOrMore>
<ref name="sensor"/>
</zeroOrMore>
</element>
</define>
<define name="attlist.srm" combine="interleave">
<empty/>
</define>
<define name="servicename">
<element name="servicename">
<ref name="attlist.servicename"/>
<text/>
</element>
</define>
<define name="attlist.servicename" combine="interleave">
<empty/>
</define>
<!– oversides: none: DELETE, exist: RETURN –>
<define name="sensor">
<element name="sensor">
<ref name="attlist.sensor"/>
<ref name="name"/>
<ref name="state"/>
<ref name="interval"/>
<optional>
<ref name="corrent"/>
</optional>
<optional>
<ref name="oversides"/>
</optional>
```

<optional>
<ref name="push"/>
</optional>
</element>
</define>
<define name="attlist.sensor" combine="interleave">
<empty/>
</define>
<!– name: sensor name –>
<define name="name">
<element name="name">
<ref name="attlist.name"/>
<text/>
</element>
</define>
<define name="attlist.name" combine="interleave">
<empty/>
</define>
<!– state: Static or Dynamic –>
<define name="state">
<element name="state">
<ref name="attlist.state"/>
<text/>
</element>
</define>
<define name="attlist.state" combine="interleave">
<empty/>
</define>
<!– interval, unit: Milli-Sec –>
<define name="interval">
<element name="interval">
<ref name="attlist.interval"/>
<text/>
</element>
</define>
<define name="attlist.interval" combine="interleave">
<empty/>
</define>
<!– corrent sensor data –>
<define name="corrent">
<element name="corrent">
<ref name="attlist.corrent"/>
<ref name="attri"/>

<ref name="value"/>
</element>
</define>
<define name="attlist.corrent" combine="interleave">
<empty/>
</define>
<!– attri: (OVER—UNDER—VALUE—BETWEEN) –>
<define name="attri">
<element name="attri">
<ref name="attlist.attri"/>
<text/>
</element>
</define>
<define name="attlist.attri" combine="interleave">
<empty/>
</define>
<!– if BETWEEN , sample 2004/3000 –>
<define name="value">
<element name="value">
<ref name="attlist.value"/>
<text/>
</element>
</define>
<define name="attlist.value" combine="interleave">
<empty/>
</define>
<!– oversides: AVERAGE, ALL, new XXhours defalut:DELETE ALL –>
<define name="oversides">
<element name="oversides">
<ref name="attlist.oversides"/>
<ref name="senderip"/>
<ref name="returned"/>
</element>
</define>
<define name="attlist.oversides" combine="interleave">
<empty/>
</define>
<!– IPAddress –>
<define name="senderip">
<element name="senderip">
<ref name="attlist.senderip"/>
<text/>
</element>

</define>
<define name="attlist.senderip" combine="interleave">
<empty/>
</define>
<!– RETURN THINGS: (AVERAGE, ALL) –>
<define name="returned">
<element name="returned">
<ref name="attlist.returned"/>
<text/>
</element>
</define>
<define name="attlist.returned" combine="interleave">
<empty/>
</define>
<!– push data from vehicle to service-server –>
<define name="push">
<element name="push">
<ref name="attlist.push"/>
<ref name="interval"/>
<ref name="returned"/>
<ref name="senderip"/>
<optional>
<ref name="start"/>
</optional>
<optional>
<ref name="end"/>
</optional>
<optional>
<ref name="bandwidth"/>
</optional>
<zeroOrMore>
<ref name="media"/>
</zeroOrMore>
</element>
</define>
<define name="attlist.push" combine="interleave">
<empty/>
</define>
<!– Start Time : –>
<define name="start">
<element name="start">
<ref name="attlist.start"/>
<text/>

```
</element>
</define>
<define name="attlist.start" combine="interleave">
<empty/>
</define>
<!– End Time : –>
<define name="end">
<element name="end">
<ref name="attlist.end"/>
<text/>
</element>
</define>
<define name="attlist.end" combine="interleave">
<empty/>
</define>
<!– BandWidth –>
<define name="bandwidth">
<element name="bandwidth">
<ref name="attlist.bandwidth"/>
<text/>
</element>
</define>
<define name="attlist.bandwidth" combine="interleave">
<empty/>
</define>
<!– media name –>
<define name="media">
<element name="media">
<ref name="attlist.media"/>
<text/>
</element>
</define>
<define name="attlist.media" combine="interleave">
<empty/>
</define>
<start>
<choice>
<ref name="srm"/>
</choice>
</start>
</grammar>
```