The Ninth International Conference on Mobile Data Management

Efficient Cost-Based Tracking of Scheduled Vehicle Journeys

Dalia Tiešytė Christian S. Jensen
Department of Computer Science, Aalborg University, Denmark
Email: {dalia,csj}@cs.aau.dk

Abstract

Applications in areas such as logistics, cargo delivery, and collective transport involve the management of fleets of vehicles that are expected to travel along known routes according to schedules. There is a fundamental need by the infrastructure surrounding the vehicles to know the actual status of the vehicles. Since the vehicles deviate from their schedules due to road construction, accidents, and other unexpected conditions, it is necessary for the vehicles to communicate with the infrastructure. Frequent updates introduce high communication costs, and server-side updates easily become a bottleneck. This paper presents techniques that enable the tracking of vehicle positions and arrival times at scheduled stops with little communication, while still offering the desired accuracy to the infrastructure of the status of the vehicles. Experimental results with real GPS data from buses show that the proposed techniques are capable of reducing the number of updates significantly compared to a state-of-the art approach where vehicles issue updates at pre-defined positions along their routes.

1. Introduction

Geographical positioning and wireless communication technologies enable the tracking of the status of moving vehicles by the surrounding infrastructure. Applications exist in areas such as logistics, transit services, cargo delivery, and collective transport that involve the management of fleets of vehicles that are expected to travel along known routes according to schedules. In general, real-time vehicle status information is useful for the managers and the users of the fleets. This paper focuses on the tracking of the current locations of vehicles and their expected arrival times at scheduled stops. For example, such information is very helpful to the users of public buses.

Systems already exist that enable this type of monitoring. For example, some existing public transportation systems [8, 11] employ PCs on-board the vehicles, GPRS and WiFi for data communication, and GPS and tag readers for positioning. The systems use variable displays at bus stops for informing the users of expected bus arrival times. The key challenge is to accomplish the monitoring accurately and efficiently.

We consider scenarios where the vehicles follow their routes and time schedules with the best effort. A route is defined by a sequence of road segments in a digital road network, and a schedule consists of stops along a route, where each point has an associated (scheduled) arrival and departure time. These points are termed timing points.

When a vehicle travels along public roads, it is inherently difficult to predict the progress of the vehicle, as its progress is affected by external factors such as waiting times at traffic lights, congestion, road construction, weather conditions, and accidents. The infrastructure that surrounds a vehicle must therefore be informed on a continual basis of the status of the vehicle in order to ensure an appropriate degree of consistency between the actual status of the vehicle and the knowledge of this in the infrastructure. However, frequent updates introduce high communication costs, and server-side updates easily become a bottleneck. Efficient tracking techniques are then needed that reduce the numbers of updates sent from the vehicles to the server that represents the infrastructure, and from the server to the vehicles, while maintaining sufficiently accurate information in the infrastructure on the status of the vehicles.

In an existing approach to tracking, the server predicts a vehicle's near-future position and shares this prediction with the vehicle [14, 15, 16, 17, 18]. The vehicle issues an update to the server with its current position-related status when its actual position deviates from the predicted position by more than an agreed-upon threshold. A new prediction is then formed, and the procedure repeats itself. Zhou et al. [19] propose to optimize the accuracy threshold so that it would reduce the total update and querying costs (the objects are inside the region of interest are queried to specify their location). The paper assumes random walk movement inside the region of the most recent updated location.

We extend this prediction sharing scheme. In particular, we extend the scheme to allow for complex prediction functions and cost-driven decisions. We enable position prediction functions that are simple linear functions of time as well as complex functions that utilize a number of influencing factors. The challenge is to keep the communication costs low even with less accurate predictions. We allow the server to update its prediction when new information, e.g., traffic reports, arrives, and to send this new prediction to the vehicle either when required to offer accuracy guarantees or when this is expected to reduce the later communication costs.

The contributions of this paper are the following. The paper offers a detailed design of a setting for the tracking of the locations and arrivals at timing points of scheduled vehicles such as public buses. Algorithms are proposed that enable tracking with accuracy guarantees in this setting. An advanced cost function is used as a means of controlling when to issue updates. This function exploits the probability distribution of the prediction error to evaluates the costs of future updates. An experimental study based on GPS data from public buses shows that the proposed techniques are capable of reducing the amounts of required updates significantly in comparison to the current state-of-the-art in logistics and public transportation.

The paper is outlined as follows. Section 2 presents the tracking system architecture, including its approach to communication-efficient vehicle tracking. Section 3 defines a cost function that is expected to reduce the total communication costs. Section 4 reports on an experimental study, and Section 5 concludes.

2. Tracking System Framework

An information system for transportation may be viewed as a dynamic, distributed system. The key objective of a tracking system is to maintain the information system in a consistent state, meaning that some degree of consistency must be maintained between the actual status of a fleet of vehicles and the record of this status in the surrounding infrastructure.

Shared prediction-based tracking is an approach to tracking that aims at maintaining this consistency in a cost-efficient manner, by enabling the vehicles to only issue updates to the server representing the infrastructure when actually needed in order to maintain the consistency.

In the ideal case, when the movement of a vehicle "matches" the prediction that represents the server's belief, no updates are needed. In reality, deviations do happen, and the challenge is then to keep the communication and update costs low.

2.1. Accuracy Guarantees

In this paper, we propose techniques for maintaining consistency with respect to a vehicle's current location and its arrival time at the next scheduled stop (called a timing point) along a journey, meaning that we consider position- and time-based tracking. The current value (measured, in the case of location; predicted, in the case of arrival time) for these variables of a vehicle are not allowed to deviate by more than a given threshold thr from the value that is known by the server and to the outside world. The accuracy threshold is chosen according to the requirements of the application domain. An optimal trade-off between accuracy and update costs can be achieved by tuning the system parameters.

We consider two scenarios. In the first, a vehicle and the server share the same prediction function and arguments for this function. The vehicle is also capable of measuring its current position. Thus, the vehicle is also better at forming predictions of its variables than is the server. The vehicle issues an update to the server when needed so that its actual location or expected arrival time never deviate from the server's prediction by more than thr. For arrival times, it is meaningful to only require updates for arrivals later than thr. (For early arrivals, the vehicle synchronizes by waiting at the timing point.)

In the second scenario, which generalizes the first, the server has access to additional real-time data (e.g., traffic information). Thus, the server may form better predictions of a vehicle's variables than the vehicle itself. Therefore, the server is allowed to update a vehicle with a new prediction. Threshold thr is then shared between the vehicle and the server, and the vehicle and server use a threshold thr_{vt} and thr_{ct}, respectively, so that $thr_{\mathrm{vt}} + thr_{\mathrm{ct}} = thr$. The server now also has to update the vehicle when its prediction deviates from the prediction assumed by the vehicle by thr_{ct} (which is know to the server). The thresholds can be set so that overall system performance is optimized.

In this scenario, the server can update the vehicle "early," which is reasonable if this may reduce the number of subsequent updates from the vehicle, thereby reducing the overall update cost. We employ a cost function for estimating the expected costs of updates during the remaining part of a vehicles journey. This function is discussed in Section 3.

We propose tracking algorithms that offer accuracy guarantees (assuming an error-free system, e.g., without loss of communication). The algorithms permit any kind of prediction function to be used. Functions exist that exploit neural networks (e.g., [4, 7, 9, 10, 12]), Support Vector Machines [1], and Kalman filtering (e.g., [2, 3, 6, 13]). Domain specific prediction functions such as deviations from schedules are also possible.

2.2. System Architecture

Figure 1 describes the communication between the server and a vehicle. The *Vehicle Tracker* (VT) module of a vehicle tracks the current values of the vehicle's location and arrival time, as is known by the vehicle. It issues updates to the server as needed to maintain consistency. Each vehicle is equipped with a GPS receiver that passes position

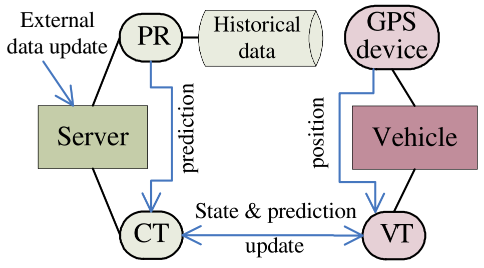


Figure 1. The Communicating Entities

updates to the VT.

Next, the server's *Central Tracker* (CT) module tracks the value of a continuous variable, which can be either the current position of the vehicle or the predicted arrival time at the next timing point (e.g., bus stop) and controls the communication between the vehicle and the server.

Finally, the *Predictor* (PR) module is responsible for forming predictions of current positions and arrival times for vehicles on the server side. It obtains current vehicle positions from the CT, and it also has an access to a database of external historical and real-time data (e.g., historical trajectories and current traffic information).

The VT module For each of the two variables being tracked, the VT shares a prediction function f_{sh} with the CT, and it has a function f_{vt} that captures its own prediction of the variables. using these, the VT implements the function $updateCT$ that decides when the CT has to be updated. More specifically, the VT uses the following entities:

- Function $f_{\mathrm{sh}} : T \rightarrow VAR$ predicts variable $var \in VAR$ at time $t \in T$ and is shared with the CT. It can differ by at most thr_{ct} from the server's own prediction.
- Function $f_{\mathrm{act}} : T \rightarrow P$ captures the vehicle's measured position $p \in P$ at time $t \in T$. This function represents the GPS receiver.
- Function $f_{\mathrm{vt}} : T \rightarrow VAR$ predicts variable $var \in VAR$ at time $t \in T$ using the current position $f_{\mathrm{act}}(t)$.
- The vehicle-side tracking threshold thr_{vt}.
- Additional data that may be used by the prediction algorithms (e.g., historical data, the vehicle's route and schedule).

The general VT algorithm is given in Algorithm 1. It waits for an update from either the server or the GPS receiver. In the first case (lines 3–5), the shared prediction is set to the prediction received from the server (line 4). In the second case (lines 6–12), the vehicle modifies its own prediction f_{vt} (line 7). If the threshold thr_{vt} is reached (lines 8–11,) the server is updated with the status of the vehicle, and the shared prediction is modified according to this status. The algorithm terminates at the end of the vehicle's journey.

Algorithm 1: General Algorithm for VT

```
1:  repeat
2:    wait for event $E$
3:    if $E$ is an update $f_{\mathrm{pr}}$ from the server then
4:      $f_{\mathrm{sh}} \leftarrow f_{\mathrm{pr}}$
5:    end if
6:    if $E$ is a GPS update $up$ then
7:      $f_{\mathrm{vt}} \leftarrow modifyVT(up)$
8:      if $|f_{\mathrm{sh}}(t) - f_{\mathrm{vt}}(t)| \geq thr_{\mathrm{vt}}$ then
9:        CT.$update(up)$
10:       $f_{\mathrm{sh}} \leftarrow modifyCT(up)$
11:     end if
12:   end if
13: until end of journey
```

The CT module The CT implements the tracking on the server side. In doing so, it receives and exploits external-data updates. The CT thus calls the PR module when it receives a position update from a vehicle or when the prediction may change due to external reasons. The CT module implements the function $updateVT$ that decides when the vehicle needs to be updated in order to maintain consistency. The module uses the following entities:

- Function $f_{\mathrm{sh}} : T \rightarrow VAR$, shared with the VT, predicts the variable $var \in VAR$ at time $t \in T$.
- Function $f_{\mathrm{pr}} : T \rightarrow VAR$ is the server's own prediction of variable $var \in VAR$ at time $t \in T$.
- The server-side tracking threshold thr_{ct}.

The general algorithm of the server-side CT module is given in Algorithm 2. The algorithm is called for each vehicle when the journey of the starts and the vehicle's VT is being initialized. During initialization, the PR calculates the vehicle's trajectory and the VT is updated with the prediction (if needed) (line 1). The algorithm then loops until the end of the vehicle's journey (lines 3–16). The vehicle is either updated when the threshold is reached (lines 13–15) or when f_{pr} changes and it is decided by a cost function that an update is beneficial (lines 7–8). The shared prediction is modified with every update from the vehicle (lines 9–11).

The PR module This module implements the function $f_{\mathrm{pr}} : T \rightarrow VAR$ that predicts the variable $var \in VAR$ at time $t \in T$. It is called by the CT module when predictions need to be updated.

2.3. Update Policies

We proceed to cover three update policies: timing point-based tracking, position-based tracking, and time-based tracking. The latter two are based on the prediction algorithms descried above.

Timing Point-Based Tracking The tracking algorithms currently employed in public transportation (e.g., [8]) and

Algorithm 2: General Algorithm for CT

```
1:  $f_{\mathrm{pr}} \leftarrow \mathrm{PR}.\mathit{getPrediction}()$
2:  $f_{\mathrm{sh}} \leftarrow \mathrm{PR}.f_{\mathrm{pr}}$; $\mathrm{VT}.\mathit{update}(f_{\mathrm{pr}})$
3:  **repeat**
4:      wait for event $E$
5:      **if** $E$ is an update $up$ from VT or an external source **then**
6:          $\mathrm{PR}.\mathit{update}(up)$
7:          **if** update of VT is needed **then**
8:              $\mathrm{VT}.\mathit{update}(f_{\mathrm{pr}})$; $f_{\mathrm{sh}} \leftarrow \mathrm{PR}.\mathit{getPrediction}()$
9:          **else if** $E$ is an update $up$ from VT **then**
10:             $f_{\mathrm{sh}} \leftarrow \mathit{modifyCT}(up)$
11:         **end if**
12:     **end if**
13:     **if** $E\ |f_{\mathrm{sh}}(t) - f_{\mathrm{pr}}(t)| \geq \mathit{thr}_{\mathrm{ct}}$ **then**
14:         $\mathrm{VT}.\mathit{update}(f_{\mathrm{pr}})$; $f_{\mathrm{sh}} \leftarrow \mathrm{PR}.\mathit{getPrediction}()$
15:     **end if**
16: **until** end of journey
```

logistics systems often rely on timing points. When a vehicle arrives at a timing point, it updates the server with its current status. The number of updates is fixed—it is equal to the number of timing points.

Such tracking has several drawbacks. First, the server is unaware of the deviations of the actual travel pattern from the expected travel pattern when the vehicle is traveling in-between two timing points. This renders it difficult to accurately predict near-future travel times. Second, only poor position accuracy guarantees may be given in-between the timing points. Third, unnecessary costs are incurred when the vehicle travels as expected.

This policy can be used if it is crucial to know exact arrival and departure times at timing points, but is less important to reduce the tracking costs and to provide position accuracy guarantees.

Position-Based Tracking The general framework described in Section 2 is assumed. With position-based tracking, the variable that is being tracked is the current position of a vehicle $p \in P$ (i.e., VAR becomes P). The vehicle is aware of the shared position prediction function f_{sh}, and it compares its current position (using f_{vt}) with the server's assumed position. When the predicted position deviates from the actual position by threshold value $\mathit{thr}_{\mathrm{vt}}$, the vehicle issues an update to the server.

The server updates the vehicle if the difference between its actual predicted position $f_{\mathrm{pr}}(t)$ and the shared prediction $f_{\mathrm{sh}}(t)$ reaches $\mathit{thr}_{\mathrm{ct}}$, or if the server's prediction function changes sufficiently for the server's cost function to trigger an update.

Time-Based Tracking We again assume the general framework from Section 2. The variable being tracked is the arrival time $t \in T$ (i.e., VAR now becomes T) at the next timing point. Similarly to position-threshold based tracking, the vehicle is aware of the server's (approximate) arrival time prediction, and it compares its own current prediction with the server's prediction. The vehicle updates the server as needed in order to stay within the time deviation threshold $\mathit{thr}_{\mathrm{vt}}$. The server measures the deviation between the arrival time predictions given by f_{sh} and f_{pr}, and it issues an update to the vehicle when either threshold $\mathit{thr}_{\mathrm{ct}}$ is reached or the cost function triggers an update.

Using position tracking is not optimal when the objective is to predict arrival times. With position tracking, if a vehicle stops within distance $\mathit{thr}_{\mathrm{vt}}$ of a timing point, it is possible for the server to believe that the vehicle is at the timing point, which may yield incorrect information in the infrastructure. This does not happen with time-based tracking. Also, updates are not required where the vehicle's actual position deviates considerably from the server's prediction if this is expected to not delay the arrival.

It should also be noted that actual arrival time accuracy guarantees cannot be provided, as the arrival time believed by a vehicle is a prediction that may not hold; however, position accuracy guarantees may be derived given all prediction functions.

3. Early-Update Cost Modeling

In cases where the prediction functions f_{pr} employed by the server are based on previous traversals of the route by other vehicles or where the prediction functions are sensitive to constantly changing traffic conditions, these functions are subject to frequent change.

The server is required to update the shared prediction f_{sh} when needed to comply with threshold $\mathit{thr}_{\mathrm{ct}}$. This entails the sending of an update to the vehicle. In addition, the server is allowed to update f_{sh} at other times, which may be exploited to improve the tracking efficiency, as discussed in Section 2.1. For this case, we use a cost function for determining whether, when the server forms a new prediction f_{pr}, the server should update the vehicle with this new prediction. This function estimates the cost of the tracking for the remainder of the journey.

The actual communication cost throughout the journey is a weighted sum of the number of updates from the vehicle to the server $\mathit{up}_{\mathrm{vt}}$ and from the server to the vehicle $\mathit{up}_{\mathrm{ct}}$: $\mathit{cost} = \mathit{up}_{\mathrm{vt}} c_{\mathrm{vt}} + \mathit{up}_{\mathrm{ct}} c_{\mathrm{ct}}$, where c_{vt} is the cost of a single update sent from the vehicle to the server and c_{ct} is the cost of a single update sent from the server to the vehicle. The numbers of updates depend on the update decisions made by the server.

The above cost is known only after the journey is completed; therefore, estimates of the cost are needed for determining whether or not to issue an update to the vehicle. The following generic cost function estimates the total cost of updates issued by the server and vehicle from the current time and until the end of the ongoing journey, given the current server state s_{cur} that captures the information currently available to the server:

$$cost(s_{\text{cur}}) = c_{\text{vt}} \sum_j P(up_{\text{vt}} = j | s_{\text{cur}}) j + c_{\text{ct}} \sum_j P(up_{\text{ct}} = j | s_{\text{cur}}) j \quad (1)$$

The function sums up the products of the costs of one update, the probabilities that the vehicle and server issue j updates, and j, where j varies from 1 to the maximum possible number of updates. Later, to specify the probabilities, we will assume that:

- More recent predictions are better than earlier ones.
- Server-side predictions are at least as good as vehicle-side predictions after the vehicle issues an update.
- The system is error-free: the GPS measurements are accurate, and no server-vehicle communication delays occur.

The cost until the end of the journey can be difficult to estimate; it is more feasible to estimate the cost of a part of the journey until some time t_k, assuming that at that time, the server is in a state s_k that is independent of the current decision as to whether or not to update the vehicle. Denote the state of server s_- if the decision d_- to not update is taken, and denote it s_+ if the decision d_+ to update is taken and the update is issued. Then the server should issue an update if $cost(s_-, t_k) > cost(s_+, t_k) + c_{\text{ct}}$.

The sums of probabilities are defined as in Equation 1, except that the time interval extends only until the next server update, and only the vehicle's updates are considered:

$$cost(s_{\text{cur}}, t_k) = c_{\text{vt}} \sum_j P(up_{\text{vt}} = j | t_k, s_{\text{cur}}) j \quad (2)$$

Example Before we proceed, we consider a simple "toy" example that clarifies the need for a cost function. Assume that $thr_{\text{ct}} = thr_{\text{vt}} = 100\,\text{m}$ and that position-based update is used. Initially, all functions are equal: $f_{\text{act}}(t) = f_{\text{sh}}(t) = f_{\text{pr}}(t) = 20\,\text{m/s} \times t$. After a while, the vehicle enters congestion, and its actual speed becomes 10 m/s. The server is informed about the traffic situation, but its newly predicted speed, 11 m/s, still exceeds the actual speed.

Before the difference $|f_{\text{sh}}(t) - f_{\text{pr}}(t)|$ reaches thr_{ct}, the vehicle's actual position deviates from the shared prediction by 100 m, and the vehicle issues an update. Without the cost function, the vehicle will keep on issuing an update every 10 s. In case the server updates the vehicle with its new prediction, updates are issued only every 100 s.

3.1. Position-Based Tracking

We estimate the mean of the costs of updates sent from the vehicle to the server during time interval $[t_{\text{cur}}, t_{\text{k}}]$. Therefore, the problem can be split into two parts: (1) Estimate the time t_{up} of the next server-vehicle update. (2) Estimate the number of vehicle updates until time t_{up}.

Estimating the Number of Vehicle Updates The vehicle updates the server when its position f_{act} deviates from the shared prediction f_{sh} by thr_{vt}. The probability that at least one update is issued by time t is then equal to the probability of the vehicle's position deviating from the shared prediction by at least thr_{vt} (assuming no server updates are issued):

$$P(up \geq 1) = P(thr_{\text{vt}} \leq |f_{\text{act}}(t) - f_{\text{sh}}(t)|) \quad (3)$$

Further, assuming that $|f_{\text{act}}(t) - f_{\text{sh}}(t)|$ is monotonically increasing and that the vehicle's update can only reduce the difference $|f_{\text{act}}(t) - f_{\text{sh}}(t)|$ at some future time t by thr_{vt}, the probability that i updates are issued until time t is equal to the probability that the deviation of the actual position from the predicted position is between $thr_{\text{vt}} i$ and $thr_{\text{vt}}(i+1)$:

$$P(up_{\text{v}} = i) = P(thr_{\text{vt}} i \leq |f_{\text{act}}(t) - f_{\text{sh}}(t)| < thr_{\text{vt}}(i+1)) \quad (4)$$

Substituting t_{up} for t, we obtain the probability that the number of vehicle updates until the next server update equals i.

The probability of deviation depends on the reliability of the predictions. We fix the time at which a prediction is made to be current time t_{cur}. At time $t_{\text{up}} = t_{\text{cur}} + \Delta t$, the prediction error $e(\Delta t)$ equals $|f_{\text{act}}(t) - f_{\text{pr}}(t)|$. The error variance, $\sigma_e(\Delta t)$, can be evaluated statistically using historical data and simulations. The error mean is assumed to be 0, and its distribution is assumed to be Gaussian: $e(\Delta t) \sim N(0, \sigma^2_{e(\Delta t)})$. These are standard assumptions about the error of a prediction function. The probability distribution function of the error $e(\Delta t)$ is expressed as:

$$F(\Delta t, z) = \frac{1}{\sigma_{e(\Delta t)}\sqrt{2\pi}} \int_{-\infty}^{z} e^{-x^2/2\sigma^2_{e(\Delta t)}} dx \quad (5)$$

Denote the difference between the server's and the shared prediction functions as $err(t) = f_{\text{pr}}(t) - f_{\text{sh}}(t)$. The probability that the actual arrival time deviation from $f_{\text{sh}}(t)$ is in-between thr_{vt} and $thr_{\text{vt}}(i+1)$ can be defined as: $P(thr_{\text{vt}} i \leq |f_{\text{act}}(t) - f_{\text{sh}}(t)| < thr_{\text{vt}}(i+1) \mid s_{\text{cur}}) = P(thr_{\text{vt}} i + err(t) \leq e(\Delta t) < thr_{\text{vt}}(i+1) + err(t) \mid s_{\text{cur}})$.

The probability that a variable distributed according to Equation 5 belongs to an interval (z_1, z_2), is $F(\Delta t, z_2) - F(\Delta t, z_1)$. The probability that the number of updates equals i is: $P(up = i) = P(thr_{\text{vt}} i \leq e(\Delta t) < thr_{\text{vt}}(i+1) \mid s_{\text{cur}}) =$
$F(\Delta t, err(t) + thr_{\text{vt}}(i+1)) - F(\Delta t, thr_{\text{vt}} i + err(t))$.

Equation 2 has to be evaluated with both decisions d_- and d_+, inserting $P(up = i)$ from the equation above. In the case of d_+, $f_{\mathrm{sh}}(t) = f_{\mathrm{pr}}(t)$, and the cost c_{ct} has to be added to the journey cost.

Estimating the Time of the Next Server Update An update from the server can be issued in three cases: (1) when $|f_{\mathrm{sh}}(t) - f_{\mathrm{pr}}(t)|$ reaches the threshold thr_{ct}, (2) when the server changes its prediction due to changes in external data, or (3) when the update would reduce further communication costs.

In the first case, the next update is issued at time $\min_t\{|f_{\mathrm{sh}}(t) - f_{\mathrm{pr}}(t)| \geq thr_{\mathrm{ct}}\}$, if the prediction functions do not change. Without this simplifying assumption, the actual deviation time may be later.

In the second case, the average interval in-between external updates Δup_{ext} can be estimated using historical data; however, it is not always necessary to update the vehicle when a new prediction is formed. This again depends on the change of prediction function, and the estimation of the cost function. To eliminate this complexity, we ignore updates due to the cost function. The interval Δup that introduces the deviation $|f_{\mathrm{sh}}(t) - f_{\mathrm{pr}}(t)| \geq thr_{\mathrm{ct}}$ is estimated as:

$$\Delta up = \frac{\Delta up_{\mathrm{ext}}}{P(|f_{\mathrm{sh}}(t_{\mathrm{cur}} + \Delta t) - f_{\mathrm{pr}}(t_{\mathrm{cur}} + \Delta t)| \geq thr_{\mathrm{ct}})}$$

Here Δt is the average time interval until the next external-data update (with respect to the most recent update). The probability that the new prediction will differ by at least thr_{ct} from the shared prediction can be evaluated statistically.

In the third case, an update is issued when this is expected to reduce the remaining update cost. This leads to a recursive estimation of the cost function, and therefore we eliminate this case.

Finally, the estimate of the server's update time is $t_{\mathrm{up}} = \min(\min_t\{|f_{\mathrm{sh}}(t) - f_{\mathrm{pr}}(t)| \geq thr_{\mathrm{ct}}\}, t_{\mathrm{cur}} + \Delta up)$.

3.2. Time-Based Tracking

The cost function is evaluated similarly as for position-based tracking. In time-based tracking, the prediction functions $f_{\mathrm{vt}}(t)$, $f_{\mathrm{sh}}(t)$, and $f_{\mathrm{pr}}(t)$ predict arrival times instead of positions; therefore, both the domains and ranges of these functions are time (as described in Section 2.3). The threshold distances thr_{vt} and thr_{ct} are measured between the predicted arrival times at the nearest timing point rather than between the predicted and the measured positions.

4. Experimental Evaluation

In this section, we report on an empirical evaluation of the proposed techniques. In particular, we evaluate the tracking techniques and the cost function. Real GPS data collected from the city buses in the Copenhagen area [8] were used for the evaluation.

4.1. Experimental Setup

A real bus route with 49 bus stops was used in the evaluation. The actual duration of a journey on this route is 1 hour and 43 minutes, during which a distance of nearly 33 kilometers is covered. GPS positions were measured once per second and were map-matched to the polylines of a digital map that correspond to the bus route.

Since we would like to evaluate the tracking techniques based on data obtained when using different prediction functions, the prediction function was simulated according to the following framework:

- The predicted travel times in-between the bus stops and the waiting times at the stops are in the range of 0.5 to 2 times the actual times. The prediction error is distributed normally within a given range. In-between the timing points, the vehicle's speed is predicted to be constant.
- The vehicle and server update the shared prediction function by calculating a time delay and adding this to the initial prediction.
- An external update to the server (in reality, this could be, e.g., new traffic information) is generated for each timing point. A new prediction for the next timing point is generated; the default error variance is 50 s, and the new predicted time cannot be further from the actual arrival time than the old prediction.

Timing-point based update would issue an update per timing point, i.e., 49 updates for our setting. The experimental results show that in most cases, this amount can be reduced significantly while also providing pre-defined accuracy guarantees. For position-based tracking (PBT) and time-based tracking (TBT), the total numbers of updates are the sums of vehicle and server updates that occur during the journey. We consider three update polices that the server may use: update when (1) the prediction function changes, (2) the server's threshold is reached, or (3) the cost function estimates that the future updates would be reduced.

4.2. Default Parameters

Table 1 lists the default parameters used in the experiments. Most values are the same for both PBT and TBT. Thresholds for TBT are given in parentheses. The prediction variance is defined as the deviation variance of the predicted travel time in-between two timing points or the waiting time deviation at a timing point. Default prediction variances are chosen to be large in order to check how the algorithms behave when prediction functions are inaccurate. Server predictions are set to change quite frequently. Additional delay allows the arrival time to be later than expected.

Table 1. Default Parameters

Parameter	Value
Initial prediction variance	100 s
Server prediction variance	50 s
Average delay detection time	300 s
Server and vehicle thresholds thr_{ct}, thr_{vt}	400 m (100 s)
Delay	1000 s
Single update costs c_{vt}, c_{ct}	$c_{\mathrm{vt}} = c_{\mathrm{ct}} = 1$
Number of timing points	49

4.3. Experimental Results

The experimental study evaluates the tracking techniques under varying circumstances. We focus here on the influence of varying accuracies and thresholds, as these two are arguably the most interesting parameters.

Results obtained when varying other parameters yield similar results. For example, adding more delay to GPS points has an effect similar to increasing the initial variance. Only extreme delays require significantly more updates. Increasing the number of timing points increases the prediction accuracy and therefore decreases the required number of updates. Timing point-based tracking requires one update per timing point; however, the corresponding accuracy guarantees are (bounded by) the distances or travel times in-between two timing points.

Prediction Accuracy The update frequency is expected to be affected by the accuracy of both the initial prediction and updates to the server's prediction: with more accurate prediction, less updates are expected. Three kinds of experiments were performed: (1) the initial prediction accuracy is fixed, while the updated prediction's accuracy decreases, (2) the updated prediction's accuracy decreases, while the initial prediction accuracy is fixed, and (3) the accuracy of both the initial and updated predictions decreases.

The experimental results (Figures 2–4) show that larger updated prediction variances (and therefore lower accuracies) increase the amount of required updates. The initial accuracy is less significant, as it is improved as soon as a server update takes place. Using the cost function ("Cost") requires fewer updates than updating with each server prediction change ("All") or updating only when the server threshold is reached ("None").

Compared with timing-point based tracking (49 updates), only updating the bus every time the prediction changes is more costly. The other techniques work well even with less accurate prediction functions. TBT shows to be very efficient if used with a cost function.

Thresholds The higher the tracking thresholds, the lower the amounts of updates are expected to be. Two kinds of experiments were performed: (1) The server's and the vehicle's thresholds are equal ($thr_{\mathrm{ct}} = thr_{\mathrm{vt}}$) and (2) they vary from 100 m to 1000 m (PBT) or from 30 s to 300 s (TBT).

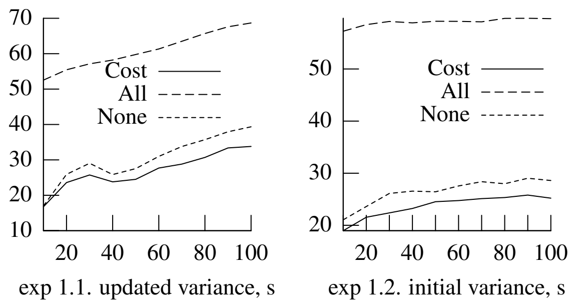


Figure 2. Varying Prediction Accuracy (PBT)

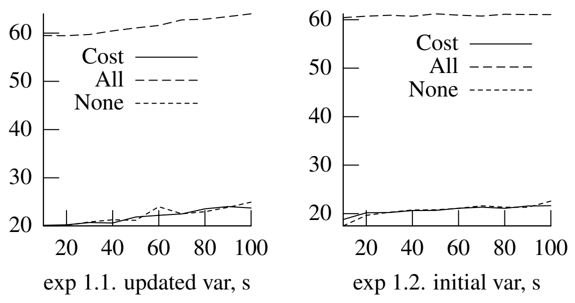
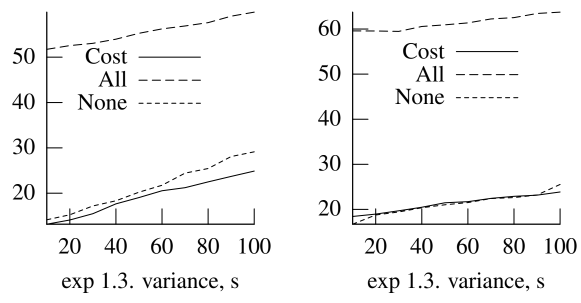


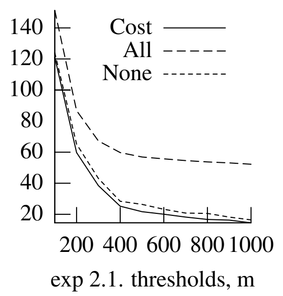
Figure 3. Varying Prediction Accuracy (TBT)



(a) Position-based tracking (b) Time-based tracking

Figure 4. Varying Initial and Updated Accuracies Simultaneously

The experimental results (Figures 5 and 6) show that smaller vehicle and server thresholds increase the amounts of required updates. The ratio between the vehicle's and server's thresholds is an important factor. The studies show that it is infeasible to set $thr_{\mathrm{vt}} > thr_{\mathrm{ct}}$. When the vehicle and server thresholds become equal ($thr_{\mathrm{vt}} = thr_{\mathrm{ct}} = 500$,) further increasing vehicle threshold and decreasing server threshold does not have significant influence on communication costs. Using the cost function requires less updates than updating with each change of prediction or updating only when the server threshold is reached.



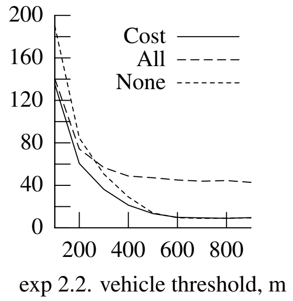
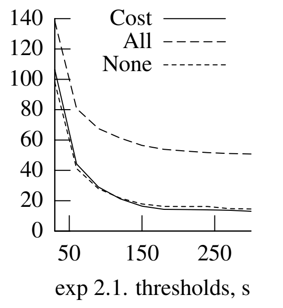


Figure 5. Varying Thresholds (PBT)



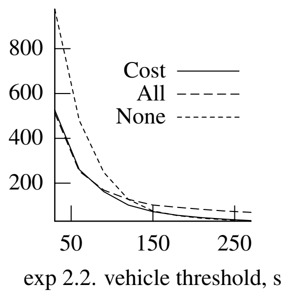


Figure 6. Varying Thresholds (TBT)

5. Conclusion and Research Directions

A number of applications in the areas of logistics, transit services, cargo delivery, and collective transport benefit from maintaining real-time information on vehicle status, most notably the vehicles' positions on their routes, in the infrastructure surrounding the vehicles.

This paper proposes cost-efficient tracking techniques with accuracy guarantees for vehicles that travel along known routes according to schedules. A vehicle and the surrounding infrastructure, represented by a server, communicate only this is required to preserve pre-defined accuracy guarantees, or when an update is expected to reduce the subsequent communication costs. Experimental results show that the communication costs can be reduced several times in comparison to a currently used timing point-based tracking policy.

Several research directions exist. A cost model is desirable that makes it possible to accurately compute the minimum communication costs that are required to preserve accuracy guarantees. Such a model would expose the limits for improvement of the tracking policies. Furthermore, it is of interest to develop accurate prediction functions that reduce the required amounts of updates. Experimental results have shown that with accurate predictions of arrival times, communication costs are very low.

Acknowledgments This research was conducted within the project GPS Data Management with Applications in Collective Transport, funded by the Danish Research Agency, contract number 2106-05-0027.

References

[1] Y. Bin, Y. Zhongzhen, and Y. Baozhen. Bus arrival time prediction using support vector machines. *Int. Transp. Syst.*, 10: 151–158.

[2] F. Cathey and D. Dailey. A prescription for transit arrival/departure prediction using automatic vehicle location data. *Transp. Res. Part C*, pp. 241–264, 2003.

[3] M. Chen, X. Liu, and J. XiaDOI Dynamic prediction method with schedule recovery impact for bus arrival time. *Transp. Res. Rec*, 1932(1): 208–217, 2005.

[4] S. I.-J. Chien, Y. Ding, and C. Wei. Dynamic bus arrival time prediction with artificial neural networks. *Transp. Engrg.*, 128: 429–438, 2002.

[5] D. Crout. Accuracy and Precision of the Transit Tracker System. *Transp. Res. Rec*, 1992(-1): 93–100, 2007.

[6] D. Dailey, S. Maclean, F. Cathey, and Z. Wall. Transit vehicle arrival prediction: an algorithm and a large scale implementation. *Transp. Res. Rec., Transportation Network Modeling*, pp. 46–51, 2001.

[7] J. R. Hee and L. R. Rilett. Bus arrival time prediction using artificial neural network model. In *IEEE ITSC*, pp. 988–993, 2004.

[8] Hovedstatens Udviklinksraad (HUR). *http://www.hur.dk/.*

[9] R. H. Jeong. *The prediction of bus arrival time using automatic vehicle location systems data.* Doctoral dissertation, Texas A&M University, 2004.

[10] R. Kalaputapu and M. J. Demetsky. Modeling schedule deviations of buses using automatic vehicle location data and artificial neural networks. *Transp. Res. Rec.*, 1497: 44–52, 1995.

[11] Nordjyllands Trafikselskab (NT). *http://www.nordjyllandstrafikselskab.dk/.*

[12] T. Park, S. Lee, and Y.-J. Moon. Real time estimation of bus arrival time under mobile environment. In *ICCSA*, 3043: 1088–1096, 2004.

[13] A. Shalaby and A. Farhan. Prediction model of bus arrival and departure times using avl and apc data. *Journal of Pub. Transp.*, 7: 41–61, 2004.

[14] D. Tiešytė and C. S. Jensen. Challenges in the tracking and prediction of scheduled-vehicle journeys. In *IEEE PerCom Workshops*, pp. 407–412, March 2007.

[15] A. Čivilis, C. S. Jensen, and S. Pakalnis. Techniques for efficient road-network-based tracking of moving objects. In *IEEE TKDE*, 17(5): 698–712, 2005.

[16] O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, and G. Mendez. Cost and imprecision in modeling the position of moving objects. In *ICDE*, pp. 588–596, 1998.

[17] O. Wolfson, A. P. Sistla, S. Chamberlain, and Y. Yesha. Updating and querying databases that track mobile units. *Distrib. and Par. Databases*, 7: 257–387, 1999.

[18] O. Wolfson and H. Yin. Accuracy and resource consumption in tracking moving objects. In *SSTD*, pp. 325–343, 2003.

[19] J. Zhou, H. Leong, Q. Lu, and K. Lee. Generic adaptive moving object tracking algorithms. *ICPP*, pp. 93–100, 2006.