Thrifty Tracking: Online GPS Tracking with Low Data Uplink Usage

James Biagioni jbiagi1@uic.edu
A.B.M. Musa amusa2@uic.edu
Jakob Eriksson* jakob@uic.edu

Department of Computer Science
University of Illinois at Chicago

ABSTRACT

A typical *online* GPS tracking system uses a cellular uplink to report the location of a device to a central server, and in a study based on 1.6 billion location updates we find at least 90% are sent with a fixed 1–300 second period. Through experiments with the cost of cellular data transmission we also find that every packet sent incurs significant overhead.

With these observations in mind, we describe a *thrifty tracking* system that allows the specification of a target error or budget-bound, while it optimizes the other. In our experiments, *thrifty tracking* outperforms the status quo by up to 20× while providing improved guarantees and flexibility.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; H.2.8 [**Database Management**]: Database Applications—*spatial databases, GIS*

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

GPS tracking, sampling, extrapolation, data usage

1. INTRODUCTION

Tracking assets and people using the global positioning system is becoming increasingly popular. Applications are widespread, including anti-theft devices, freight logistics, public transit arrival-time predictions, and crowd-sourced traffic maps. As a result, reducing the energy consumption of GPS devices has been the subject of intense scrutiny in the past several years [4, 8, 2, 6, 10, 9]. In tracking applications, however, energy is arguably a secondary concern: it is either abundant, as in most vehicular applications, or already expended, as in the crowd-sourced traffic maps example above. The focus of this paper is on the uplink.

The data usage requirements of a GPS tracking system are modest: a differential update could be represented using just a few bytes. At the typical 1 Hz update frequency of an off-the-shelf GPS receiver, this seems a negligible amount. However, during testing, a simple GPS tracking application came close to exceeding our 250 MB/month limit. Below, we examine how to reduce this data usage while preserving tracking performance.

The primary contributions of this paper are: (a) a unified extrapolator that predicts future movements based on current conditions, (b) a unified sampler that allows the user to specify a performance target for *error* or *budget* (along with an adjustable *delay* parameter), while it optimizes the other, (c) a characterization of current tracking behavior, based on large-scale GPS probe data, and (d) an end-to-end evaluation of the above methods on real-world GPS traces.

2. STATE OF THE PRACTICE

To gain an understanding of typical online GPS tracking behavior we studied a dataset consisting of 1.6 billion GPS points, collected by 25 different data providers from 2010–2012. Our analysis showed a clear pattern of periodic reporting across probes, with fixed 1–300 second periods. After removing periodic samples, 11.4% remained. We identified the majority of non-periodic transmissions as likely ignition on/off events or transmission delays and losses, with no evidence of spatial periodic sampling, or even policies as simple as not transmitting when stationary. Therefore, both anecdotally and quantitatively speaking, we believe there is ample room for improvement to the status quo in online tracking.

3. THRIFTY TRACKING OVERVIEW

Figure 1 illustrates our general *thrifty tracking* architecture which accommodates all known, and a variety of new tracking methods. Starting in the top-left of the figure, a GPS receiver samples the (continuous) device location. If a GPS energy conservation mechanism is in use, this is done before the thrifty tracking system receives the GPS samples.

The incoming *raw* GPS trace (1) is first passed through a noise filter to remove any obvious outliers (based on impossible velocities), and then decorated by an *annotator* with ad-

* This material is based upon work supported by the U.S. National Science Foundation under Grants CNS-1017877, CNS-1149989 and DGE-0549489.

SIGSPATIAL'13 Nov 05-08 2013, Orlando, FL, USA.
ACM 978-1-4503-2521-9/13/11.
http://dx.doi.org/10.1145/2525314.2525469.

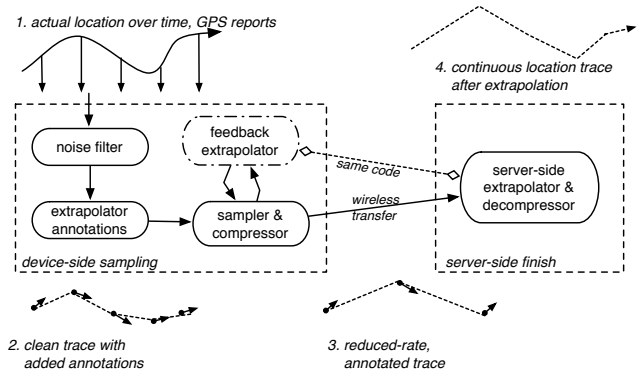

Figure 1: Thrifty tracking architectural diagram.

ditional information not provided by the GPS receiver (e.g. heading and acceleration). The decorated GPS trace (2) is then passed to a sampler that unilaterally decides whether to forward a given trace point, with any necessary annotations, over the wide-area link based on the error or budget target. The resulting sampled trace (3) is then fed to two identical extrapolators: one running on the remote server, and one running on the mobile device. On the server side, the extrapolator output (4) is made available for use by the trace consumer. On the client side, an identical extrapolator produces a continuous location estimate for local use. By comparing the output of the local extrapolator against the incoming raw GPS location, an error-aware sampler can then make its forwarding decision based on the difference between the current estimate and the measured location.

Our evaluation of these techniques is based on three distinct datasets, collected by Microsoft [3] (500K GPS points, 1490 hours) and OpenStreetMap (700K pts, 195 hrs) volunteers, as well the UIC campus shuttle (1.9M pts, 530 hrs).

4. GPS TRACE EXTRAPOLATION

By predicting the future location of a device, or *extrapolating* its location trace, improvements can be made to both the timeliness and accuracy of tracking. The most basic extrapolation method ("Constant Location" (CL)) predicts that the future location, for all times in the future, will be the same as the most recently reported location. By applying this extrapolation method, we improve the timeliness of our tracking: we are now able to provide an immediate estimate, at any point in time. However, this gain in timeliness is matched by a loss in accuracy: for a moving device, predictions made by this extrapolator grow increasingly inaccurate with time since the last update.

In this section, we explore how more sophisticated extrapolation methods can be used to improve accuracy. Specifically, we evaluate the following two basic techniques: Constant Velocity (CV), which produces a straight-line path from the most recent location, at the velocity from the most recent report, and Constant Acceleration (CA), which is based on CV, but also includes acceleration. Additionally, if movements are restricted to a known map, we evaluate the use of an advanced Map-Based extrapolator.

Our map-based extrapolator operates by traveling along the current road until an intersection is reached. Once at the intersection, it may (a) stop there, (b) continue straight through the intersection, if possible, or (c) decide how and whether to turn based on past movement history. We investigated several turn-prediction techniques, settling on an n^{th} order Markov model (NMM) [5] as the best performer.

4.1 Extrapolator Performance

To evaluate the extrapolators described above, we compare their predicted locations to the measured locations throughout the three datasets described in §3. Specifically, for all traces, we compute the mean duration $\overline{D^{max}}$ for which the extrapolation error is below a given max error threshold. Intuitively, this indicates how long each extrapolator "lasts".

Figure 2 shows the value of $\overline{D^{max}}$ for the three best performing extrapolators (CL, CV and NMM) on our Microsoft dataset, for varying values of max. Overall, we observe that the basic extrapolation methods outperform their map-based counterpart when the value of max is low. However, as we increase max the map-based method eventually reaches performance parity (between 25 and 50m), and then outperforms the basic methods by an increasingly large margin. We also found that the best method for extrapolation can often vary between samples based on the current position of the vehicle and the short-term history of previous travel. Similar behavior was exhibited on our UIC data.

Since the OSM traces are drawn from locations anywhere in the world, we were unable to explore map-based extrapolation techniques due to a lack of a feasible planet-wide map-matcher. However, of the basic extrapolation methods, we found that CV performed best across all values of max due to an abundance of driving at constant speed along long, straight roads in the traces we obtained.

While our finding that advanced map-based extrapolation works poorly at low error-thresholds is seemingly counterintuitive, there are at least two factors that limit its accuracy in situations where strict error tolerance is desired: *road position* and *GPS error*. Since digital road maps often position the road centerline on the dividing line between opposite directions of travel, matching a GPS location to a map for the purpose of extrapolation often introduces 1 or 2 lane-widths of error. Moreover, even if we have perfectly-aligned maps, locations extrapolated along the road will often lie 5–10 meters away from the reported position simply due to GPS error.

4.2 Unified Extrapolation

Based on our observations in §4.1 we conclude that while certain extrapolators work well under certain conditions, no single extrapolator offers the best performance in all circumstances. To create a better extrapolator, we propose a unified extrapolator which automatically chooses the best extrapolation method for the current circumstances.

Since our unified extrapolator cannot know what will happen in the future, this is a challenging task with no guaranteed results. We cast this as a classification problem, relying on past experience to train a classifier: supervised learning applies here as the recorded history reveals exactly which extrapolator works best. The classifier we used was a decision tree implementation provided by the *Scikit-learn* [1] machine learning library, trained using a set of features drawn from a 60-second sliding window of location samples immediately preceding the current position.

To determine an upper bound on the performance of our unified extrapolator, we produce an *Oracle* extrapolator that looks into the future to select the extrapolation method that

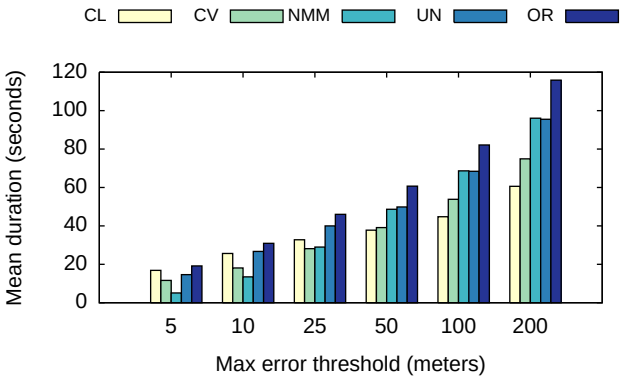

Figure 2: $\overline{D^{max}}$ for the three best extrapolators and the Unified (UN) and Oracle (OR) extrapolator across our Microsoft dataset.

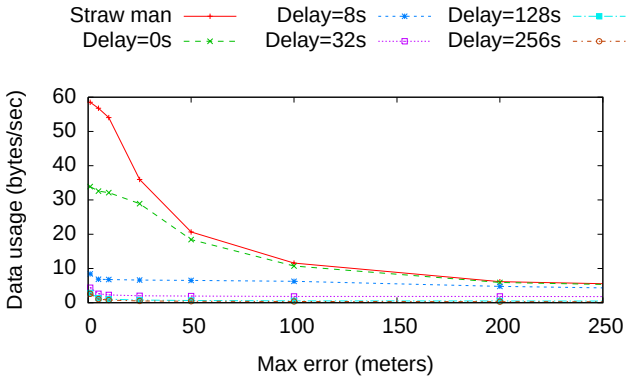

Figure 3: Budget usage with increasing max error bound for various delay bounds.

works best among all possible alternatives. The performance of our Oracle extrapolator is shown as the column labeled *OR* in Figure 2. In the worst case we find that the Oracle's performance matches that of the single-best extrapolation method, and in the best case far exceeds the performance of any alternative. In trying to replicate the Oracle's behavior, we also see that our unified extrapolator *(UN)* was able to closely match the performance of the best individual extrapolator for any given value of max, and in some ranges exceed the performance of all individual extrapolators by adaptively choosing the best method at a particular moment in time.

The remaining disparity between the *UN* and *OR* columns in Figure 2 suggests there is still room for improvement in adaptively selecting the best extrapolation method, however this is a challenge we leave for future work.

5. ADAPTIVE SAMPLING

Online tracking systems trade off two performance attributes: accuracy and cost. While periodic sampling provides highly predictable cost, it also provides very loose guarantees on accuracy. Alternatively, sampling at a fixed distance interval provides a strict error bound, but very loose constraints on cost. As shown in §2, uniform periodic sampling is the policy of choice in today's tracking systems. It gives the user direct control over cost—a likely explanation for its current popularity, but sacrifices timeliness and accuracy. Introducing the extrapolation methods from §4 can significantly improve accuracy and timeliness, as they provide an instantaneous location estimate at any time. However, further improvements can be achieved by replacing uniform periodic sampling with adaptive sampling techniques.

5.1 Feedback, Delay, and GPS Compression

Replicating the extrapolation process performed at the receiver by the sender enables it to directly observe any incurred error. This allows the sender to choose the samples it transmits to maximize the accuracy gained from each transmission. For maximum timeliness, a sampler must decide whether or not to transmit each sample as soon as it arrives, allowing relatively little room for maximizing sampling efficiency. However, if the user is willing to tolerate a fixed delay in the reporting, significant gains can be made by choosing when to transmit a sample. Adding a delay window also provides an opportunity for GPS trace compression [7], which can yield further data usage savings.

Two types of GPS compression are required for our samplers: error-bound and size-bound. For error-bound compression we use the algorithm proposed by [7]. Our size-bound GPS compressor uses the same approach, but stops when a maximum sample count has been reached.

5.2 Sampling with configured error and delay

With an error-bound and fixed delay configured by the user, the task of the sampler is to minimize cost while enforcing the error bound. For each incoming location from the GPS receiver, the sampler measures the distance between the extrapolated trace and the current location. If the distance exceeds the error bound, this sample must be transmitted. If zero delay is configured, the sample is transmitted immediately, updating the server and restarting the extrapolation.

With a non-zero delay of T seconds, the sample (and the surrounding window of samples) may be transmitted at any time between the present and T seconds into the future. The optimal time to transmit is when the resulting error is minimized. Since future errors are unknown, we need to estimate the future extrapolation error. For this, we maintain statistics on the extrapolator's past performance: its expected error over a given time interval, and the expected duration it stays below some maximum error. Then, at a given time-step we can decide whether to transmit the current window containing the oldest sample or defer transmitting in the hopes of finding larger errors (to be corrected) in the future. Therefore, the current window is only sent if it has a greater mean error than the expected mean error of all other candidate transmission windows.

Figure 3 shows budget-usage as the error threshold is varied, for different delays. These results are based on a fixed-location extrapolator, to highlight the behavior of the sampler in isolation (see §6 for our combined unified sampler/extrapolator evaluation). Here, we compare our sampler with a basic error-bound straw man solution. This solution transmits a single sample with a fixed distance interval equal to the error threshold, thus guaranteeing the error never goes above threshold. While the straw man provides a guaranteed error bound, it does so at high cost. Our sampler outperforms the straw man by a considerable margin, even with zero delay configured, demonstrating that while compression is important, adaptive sampling offers a substantial advantage on its own. Furthermore, as one might expect, as error-tolerance increases the cost decreases.

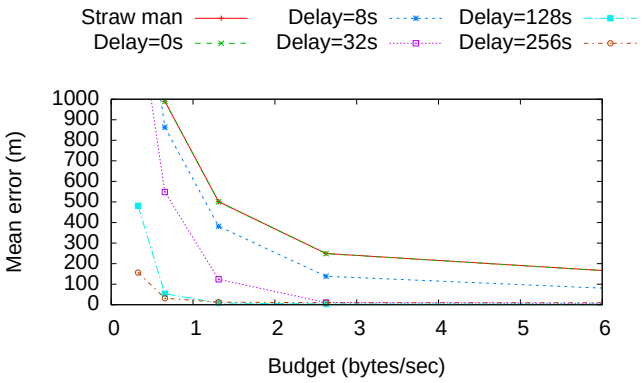

Figure 4: Mean error with increasing budget for various delay bounds.

5.3 Sampling with a given budget and delay

With a target budget and delay configured by the user, the sampler minimizes error while meeting the budget and delay targets. Intuitively, the larger the budget and the larger the delay, the smaller the error. When sampling on a budget we must first have enough savings to make a minimum-size transmission. Then, to decide when to transmit, we compare the errors produced by extrapolation and the errors produced by compressed transmission using our current savings, and transmit if the improvement exceeds the expected GPS error. Intuitively, if the improvement does not exceed the expected GPS error, it is not worth spending our hard-earned bytes transmitting this window.

Figure 4 shows the mean error incurred vs. specified budget, for several specified delays. Here, the straw man solution transmits a single sample whenever it has the savings. This figure shows that our sampler reduces error quickly when given more budget, validating the effectiveness of our technique. Moreover, as we increase the delay, mean error is reduced as our sampler is able to select more appropriate windows to transmit. Interestingly, the straw man matches our sampler's performance when we have zero delay. This is because compressing a single sample always yields a perfect result, causing the sampler to transmit whenever the extrapolator error exceeds GPS error. As a result of the small budgets shown, and the movement patterns of vehicles in our datasets, this is virtually always the case.

6. END-TO-END EVALUATION

In this section, we look at the performance of our system with the unified extrapolator and adaptive sampler combined into an end-to-end system. In our study of typical GPS tracking behavior discussed in §2 we identified a pattern of periodic reporting, most prominently at 15 and 90 seconds. Here we use these periods to guide our analysis by characterizing them as two *types* of system operator: one whose chief concern is *accuracy*, and the other whose chief concern is *cost*.

For our accuracy-concerned operator we adopt the 15 second sampling interval (i.e., 5760 samples/day/tracked individual). According to our experiments with the AT&T wireless network, each sample will cost 84 bytes to transmit (using UDP), resulting in a budget of 5.6 bytes/second. With this budget, our operator can expect mean tracking error of 110 meters using *fixed* distance-interval sampling. As shown in Table 1, using our system this operator can reduce

Opt. Criteria	fixed	w/o delay	w/delay
Data usage (bytes/s)	5.6	0.85	0.75
Mean error (m)	750	175	35

Table 1: End-to-end evaluation results.

their data usage to 0.85 bytes/second while maintaining the same mean error (*w/o delay* column). Moreover, if they are willing to accept a 7.5 second delay in transmissions (1/2 the sampling interval), they can further reduce their data usage to 0.75 bytes/second (*w/delay* column). A delay of 1/2 the sampling interval is a reasonable value, as this is equal to the mean delay of fixed distance-interval sampling.

For our cost-concerned system operator we adopt the 90 second sampling interval (i.e., 960 samples/day/tracked individual). This will result in a budget of 0.93 bytes/second (using UDP), and mean tracking error of 750 meters using *fixed* time-interval sampling. As shown in Table 1, using our system this operator can reduce their mean error to 175 meters while maintaining the same budget (*w/o delay*), or 35 meters (*w/delay*) if they are willing to accept a 45-second delay in transmissions (1/2 the sampling interval).

Overall, we see that our system is capable of substantial reductions in data usage (85% w/o delay or 87% w/delay) and error (77% w/o delay or 95% w/delay) based on the needs of the provider. Crucially, this benefit is afforded by simply providing a target accuracy or budget-bound, and the system is able to adaptively reduce cost with no further intervention.

7. CONCLUSION

Given the rising popularity of GPS tracking applications, reducing their cost requirements is a pressing need. To this end, we designed a unified *thrifty tracking* system that provides predictable performance, and savings in terms of data usage, making it an attractive solution for today's GPS tracking systems.

8. REFERENCES

[1] Scikit-learn. `http://scikit-learn.org/`.
[2] D. Ashbrook and T. Starner. Using gps to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5):275–286, 2003.
[3] J. Biagioni and J. Krumm. Days of our lives: Assessing day similarity from location traces. In *UMAP '13*, volume 7899 of *LNCS*. Springer, 2013.
[4] D. H. Kim, Y. Kim, D. Estrin, and M. B. Srivastava. Sensloc: sensing everyday places and paths using less energy. In *SenSys '10*. ACM, 2010.
[5] J. Krumm. A markov model for driver turn prediction. *SAE SP*, 2193(1), 2008.
[6] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao. Energy-accuracy aware localization for mobile devices. In *MobiSys '10*. ACM, 2010.
[7] N. Meratnia and A. Rolf. Spatiotemporal compression techniques for moving point objects. In *EDBT '04*, pages 765–782. Springer, 2004.
[8] J. Paek, J. Kim, and R. Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones. In *MobiSys '10*. ACM, 2010.
[9] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *SenSys '09*. ACM, 2009.
[10] Z. Zhuang, K.-H. Kim, and J. P. Singh. Improving energy efficiency of location sensing on smartphones. In *MobiSys '10*. ACM, 2010.