Energy Efficient GPS Sensing with Cloud Offloading

Jie Liu, Bodhi Priyantha, Ted Hart
Microsoft Research
Redmond, WA 98039, USA
{liuj,bodhip,tedhar}@microsoft.com

Heitor S. Ramos, Antonio A.F. Loureiro
Federal University of Minas Gerais
Belo Horizonte, MG,
{hramos,loureiro}@dcc.ufmg.br

Qiang Wang
Harbin Institute of Technology
Harbin, China
wangqiang@hit.edu.cn

Abstract

Location is a fundamental service for mobile computing. Typical GPS receivers, although widely available, consume too much energy to be useful for many applications. Observing that in many sensing scenarios, the location information can be post-processed when the data is uploaded to a server, we design a Cloud-Offloaded GPS (CO-GPS) solution that allows a sensing device to aggressively duty-cycle its GPS receiver and log just enough raw GPS signal for post-processing. Leveraging publicly available information such as GNSS satellite ephemeris and an Earth elevation database, a cloud service can derive good quality GPS locations from a few milliseconds of raw data. Using our design of a portable sensing device platform called CLEO, we evaluate the accuracy and efficiency of the solution. Compared to more than 30 seconds of heavy signal processing on standalone GPS receivers, we can achieve three orders of magnitude lower energy consumption per location tagging.

Categories and Subject Descriptors

C.3 [**SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS**]: [Real-time and embedded systems]

General Terms

Design

Keywords

location, assisted GPS, cloud-offloading, coarse-time navigation

SenSys'12, November 6–9, 2012, Toronto, ON, Canada.

1 Introduction

Location is a fundamental service in mobile sensing. In outdoor applications such as wildlife tracking [28, 26], participatory environmental sensing [20], and personal health and wellness applications, GPS is the most common modality for tagging data samples with their locations. GPS receiving, although becoming increasingly ubiquitous and lower in cost, is processing-intensive and energy-consuming.

Take ZebraNet sensor nodes [28] as an example. On average, one GPS location fix requires turning on the GPS chip for 25 seconds at 462mW power consumption, which dominates its energy budget. As a result, the unit is equipped with a 540-gram (1.2 pound) solar cell array and a 287-gram (0.6 pound) 2A-h lithium-ion battery in order to support one GPS position reading every 3 minutes. Power generation and storage accounts for over 70% of the sensor unit's total weight of 1151 grams (2.5 pounds).

Recent mobile sensing applications, especially those leveraging participatory sensing paradigms, typically use smart phones as sensors. While smart phones have built-in GPS and cellular-based communication capabilities, their battery life is rarely longer than a few days. A typical smart phone will completely drain its battery in about 6 hours if the GPS is running continuously [14, 21]. This prevents them from being used in unattended deployments for long periods of time.

As we will elaborate in section 2, there are two reasons behind the high energy consumption of GPS receivers: 1) the time and satellite trajectory information (called *Ephemeris*) are sent from the satellites at a data rate as low as 50bps. A standalone GPS receiver has to be turned on for up to 30 seconds to receive the full data packets from the satellites for computing its location. 2) The amount of signal processing required to acquire and track satellites is substantial due to weak signal strengths and Doppler frequency shifts. As a result, a GPS chip cannot easily be duty-cycled for energy saving. In addition, it requires a powerful CPU for post-processing and least-square calculation.

In this paper, we address the problem of energy consump-

tion in GPS receiving by splitting the GPS location sensing into a device part and a cloud part. We take advantage of several key observations.

- Many mobile sensing applications are delay-tolerant. Instead of determining the location at the time that each data sample is obtained, we can compute the locations off-line after the data is uploaded to a server. This is quite different from the design required by turn-by-turn navigation in standalone GPS devices.
- Much of the information necessary to compute the location of a GPS receiver is available through other channels. For example, NASA publishes satellite ephemeris through its web services, so the device does not have to stay on long enough to decode them locally from satellite signals. The only information that the device needs to provide is a rough notion of time, the set of visible satellites, and the "code phase" information from each visible satellite, as we will explain in section 2. Furthermore, code phases can be derived from every millisecond of baseband signals. Thus, there is significant opportunity to duty-cycle the receivers if the location computation is moved to the cloud.
- In comparison to the constraints on processing power and energy consumption, storage is relatively cheap to put on sensor devices, so we can liberally store raw GPS baseband signals together with sensor data.

Due to the split between local and cloud processing, the device only needs to run for a few milliseconds at a time to collect enough GPS baseband signals and tag them with a rough time stamp. A cloud service can then process the signals off-line, leveraging its much greater available processing power, online ephemeris, and geographical information to disambiguate the signals and to determine the location of the receiver. We call this approach *Cloud-Offloaded GPS* (CO-GPS).

The CO-GPS idea also bears several challenges. For example, for an energy and processing cycle constrained embedded sensor, we cannot maintain a real-time clock that is synchronized to the satellites. Secondly, it is hard to get a nearby reference location (whose importance will be explained in section 2) as proposed in [23]. Finally, because of aggressive duty cycling, the signal quality may suffer from temporary degradation. In this paper, we evaluate the accuracy of CO-GPS using real GPS traces and show that the time synchronization requirement is not very high and the amount of data needed for off-line processing is only a few kilobytes. We built a sensor node, called CLEO, based on the CO-GPS principle using a GPS receiving front end chip MAX2769 and a WWVB-based time synchronization mechanism [19]. Through measurements, we show that it takes less than 0.5mJ to collect a GPS data point, in comparison to the order of 1J for GPS sensing on mobile phones – or more than three orders of magnitude energy efficiency gain on devices. In other words, with a pair of AA batteries (2Ah), CLEO can theoretically sustain continuous GPS sensing (at 1s/sample granularity) for 1.5 years.

The rest of the paper is organized as follows. In section 2, we first give an overview of how a typical GPS receiver processes satellite signals, in order to motivate our solution. Section 3 describes the principle of CO-GPS. We evaluate performance of CO-GPS using real GPS traces in section 4. Finally, section 5 presents the design of the CLEO sensor node and evaluates its energy consumption in GPS receiving.

2 GPS Receiving Overview

A GPS receiver computes its location by measuring the distance from the receiver to multiple GNSS satellites (also called *space vehicles*, or SVs for short). Ultimately, it needs to infer three pieces of information:

- A precise time T;
- A set of visible SVs and their locations at time T;
- The distances from the receiver to each SV at time T, often called the *pseudoranges*.

Typically, these are obtained from processing the signals and data packets sent from the satellites. With them, a receiver can use least-square (LS) minimization to estimate its location.

To make this paper self-contained and to motivate our solution, we give a brief (and much simplified) description of the GPS receiving process. We start with standalone GPS receiving and then discuss Assisted GPS (A-GPS) and in particular a technique called *coarse-time navigation*. For a more formal treatment of the principles of GPS and A-GPS, please refer to [12, 27].

2.1 GPS Signals

There are 31 (plus one for redundancy) GNSS satellites in the sky, each orbiting the Earth about two cycles a day. A set of ground stations monitor the satellites' trajectory and health, and send the satellite parameters to the satellites. These parameters include two kinds of trajectory information: the *almanac*, which contains the coarse orbit and status information, and the *ephemeris*, which contains the precise values of the satellite's trajectory. All satellites are time-synchronized to within a few microseconds, and after clock correction, their time stamps can be synchronized within a few nanoseconds.

The satellites simultaneously and continuously broadcast time and orbit information through CDMA signals at L_1 =1.575 GHz towards the Earth. The bit-rate of data packets is a mere 50 bps, but the bits are modulated with a higher frequency (1MHz) signal for detecting propagation delays. A full data packet from a satellite broadcast is 30 seconds long, containing 5 six-second-long frames, as shown in Figure 1. A frame always starts with a preamble (called *Telemetry Word*, or TLM) and a time stamp (called *Handover Word*, or HOW). Each data packet contains the ephemeris of the transmitting satellite and the almanac of all satellites. In other words, a precise time stamp can be decoded every 6 seconds, and the high accuracy satellite trajectory can be decoded every 30 seconds. This low data rate explains why a standalone GPS, as seen in vehicles, can take up to a minute to acquire its first location, a metric called time to first fix (TTFF). The ephemeris information is constantly updated by the ground stations. In theory, the ephemeris data included in the SV broadcast is only valid for 30 minutes.

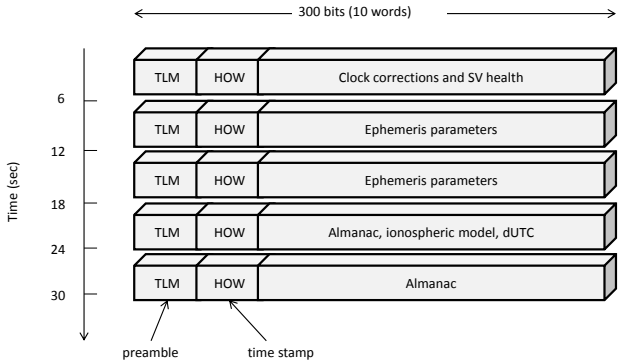


Figure 1. The frame content of a GPS packet of length 1500 bits

While the precise time and satellite locations are decoded from the packets, the pseudorange from each SV to the receiver is obtained using much lower-level signal processing techniques. For that, we need to understand GPS signal modulation.

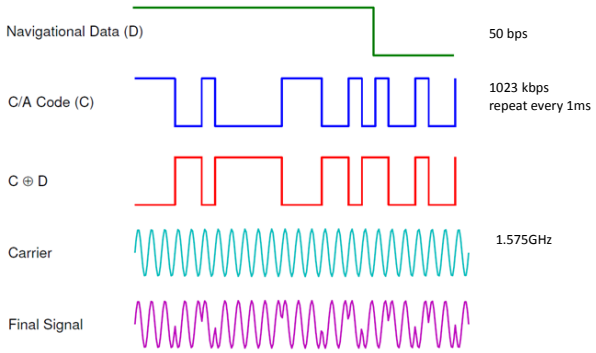


Figure 2. An illustration of GPS signal modulation scheme.

As illustrated in Figure 2, each satellite encodes its signal (CDMA encoded) using a satellite-specific coarse/acquisition (C/A) code of length 1023 chips at 1023 kbps. Thus, the C/A code repeats every millisecond, resulting in 20 repetitions of the C/A code for each data bit sent. The purpose of the C/A code is to allow a receiver to identify the sending satellite and to estimate how long it took the signal to propagate from that satellite to the receiver. Typically, the GPS signals take from 64 to 89 milliseconds to travel from a satellite to the Earth's surface. Since light travels at 300 km/ms, in order to obtain an accurate distance measurement, the receiver must estimate the signal propagation delay to the microsecond level. The millisecond (NMS) and sub-millisecond (subMS) parts of the propagation time are derived very differently: the NMS is decoded from the packet frames, while the subMS propagation time is detected at the C/A code level using correlations.

2.2 Acquisition

When a GPS receiver first starts up, it needs to detect what satellites are in view. This is done by detecting the presence of the corresponding C/A codes in the received signal, typically by correlating the signal with each known C/A code template. Since the C/A codes are designed to be orthogonal to each other, a visible satellite will show a spike in the correlation results, and an invisible satellite will not cause any detectable spike.

A challenge in acquiring satellites is the Doppler frequency shift caused by the motion of the satellite and by any movement of the receiver on the ground. For example, a rising GPS satellite can move at up to 800m/s towards a receiver, causing a frequency shift of $L_1 \cdot 800/c = 4.2$kHz, where c is the speed of light. A shift of the same magnitude occurs in the opposite direction for a setting satellite. To reliably compute a correction under this shift, the receiver must generate the C/A code within 500Hz of the shifted frequency. Therefore, in the frequency dimension, the receiver needs to search up to 18 bins. Most GPS receivers use 25 to 40 frequency bins to accommodate local receiver motion and to provide better receiver sensitivity.

After compensating for the Doppler frequency shifts, the receiver must determine code phase delays. Because the receiver does not have a clock synchronized with the satellite, and because the signal propagation delay can be affected by atmospheric conditions, the receiver must search over the delay dimension. The receiver usually over samples the 1023 bps C/A code. Assuming that the receiver samples the baseband signal at 8 MHz, in a brute force way, the receiver will search 8184 code phase positions to find the best correlation peak. Figure 3 shows an example of such correlation result in the frequency and code phase search space that indicates a successful satellite acquisition.

Code phases change over time as the satellites and the device on the ground move. In continuous operation, GPS receivers use a tracking mode to adjust previously acquired Doppler frequency shifts and code phases to the new ones. This is a relatively inexpensive process, using feedback loops. So, once a GPS produces its first location fix, subsequent location estimates become fast. However, once the GPS receiver stops tracking, the utility of previously known Doppler shifts and code phases diminishes quickly. Typically, after 30 seconds of non-tracking, the GPS receiver has to start all over again.

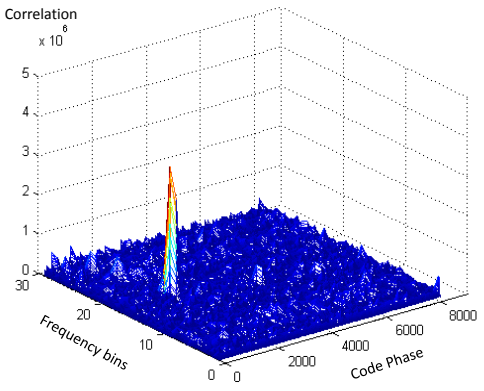


Figure 3. An example of acquisition result.

Acquisition is an expensive operation as it must search through 30+ frequency bins times 8,000+ code phase possibilities for every single satellite. If the receiver has some prior information about the satellites, it may be able to do less work. Examples are:

- **Cold Start**. When the receiver has no prior knowledge of the satellites and its own location, it has to search the entire space. Usually, GPS receivers do not buffer the raw data to perform the search, rather they perform one code phase search per millisecond as the signal streams in. Although there are hardware correlators that perform acquisition in parallel, it still takes a few seconds to acquire one satellite. This is one of the main reasons for the slow initial position fix and high energy consumption for standalone GPS devices.
- **Warm Start**. When the receiver has a previous lock to the satellites, it can start from the previous Doppler shift and code phases and search around them. In general, if the previous lock is less than 30 seconds old, a warm start can quickly find the new lock. Otherwise, the receiver has to revert to the cold start process.
- **Hot Start**. When the previous satellite locks are within a second, the receiver can skip the acquisition process and start directly from tracking to refine the Doppler and code phases. In this mode, all information that the receiver needs is already in place. This is the reason that in mobile phones, "continuous GPS sampling" is defined as one location sample per second.
- **A-GPS**. There are multiple ways that an infrastructure can help the GPS receiver start up faster. In particular, in the Mobile-Station Based A-GPS (or AGPS-B) mode, the infrastructure provides the up-to-date ephemeris data so that the GPS receiver does not have to decode them from the satellite signals. The first successful decoding of HOW is enough to provide a location fix. In this case, the TTFF is usually around 6 seconds. In the Mobile-Station Assisted A-GPS (or AGPS-A) mode, the infrastructure is given the estimated location, so it can provide initial values for Doppler and code phase searches. This allows the receiver to jump directly into the warm start process.

2.3 Location Calculation

An important output of satellite acquisition and tracking processes is the code phase produced by the correlation peaks. It gives the sub-millisecond level propagation delay. If the receiver has decoded the satellite time stamps (HOW), it knows the time that the signals have left the satellites. Then, it can add these sub-millisecond delays to obtain the whole propagation delay and thus the pseudoranges.

With correct tracking, the receiver can *decode* the packets sent by the SVs. In general, without assistance information, the receiver needs to decode SV ephemeris every 30 minutes (its valid time span) and time stamps every 6 seconds. Decoding is energy consuming since it has to run tracking continuously for the packet duration in order to receive all the bits. With A-GPS, the receiver is not required to decode ephemeris, but it must still decode HOW.

Next, given ephemeris, the propagation delays obtained from code phases, and HOW, the GPS receiver performs *position calculation* using constraint optimization techniques such as Least Squares minimization. Usually, the local clock at the receiver does not know the precise satellite time, so it is treated as one variable in the minimization solver. With the receiver's latitude, longitude, altitude, and the precise satellite time as optimization variables, typical receivers must have at least 4 SVs in view.

Notice that once the satellites are acquired, distance measurements, and thus the location of the receiver, can be estimated every millisecond. A typical GPS receiver will average over multiple Least Square solutions to further reduce noise and improve location accuracy.

Finally, we discuss how much data is necessary for satellite acquisition. Since the baseband C/A code repeats every millisecond, in the ideal case, 1ms of data is enough for satellite acquisition. Assume a 8 MHz baseband sampling frequency, the minimum amount of data needed for acquisition is 8*1023 = 8184 samples. For most GPS receiver chips, each sample is two bits (one bit sign and one bit magnitude), thus the storage requirement for 1ms of baseband data is 2046 bytes. One corner case that deserves special attention is the bit transition in the middle of the 1ms signal. Since the C/A code is used to modulate the data packets at 50bps, for every 20ms, there is a possibility of a bit transition. If the transition is in the middle of the 1ms sample, then the acquisition of the corresponding satellite will fail. So, in practice, 2ms is more reliable for satellite acquisition.

2.4 Coarse-Time Navigation

The above discussion assumes that the receiver is standalone with no assistance information. A-GPS receivers receive assistance information from servers to improve their TTFF. Typically, the assistance information includes the ephemeris data so the receivers do not have to decode them from the satellite signals. Some A-GPS approaches also provide Doppler shift and code phase guesses to the receiver so their acquisition searches do not start blindly.

One A-GPS mechanism called *Coarse-Time Navigation* (CTN) is particularly relevant to this paper. With this method, the receiver does not require the timestamp (HOW) decoded from the satellite. Instead, it only needs a coarse time reference and treats common clock bias (i.e. the difference between the receiver clock and the ideal satellite clock) as a variable in Least Square minimization. This method is first described in [27] and further exploited in [23] to reduce mobile phone GPS energy consumption.

Without decoding the HOW, the receiver cannot synchronize to the satellites' transmission times. So, a key idea in using CTN is to leverage a nearby landmark to estimate NMS. Since light travels at 300 km/ms, two locations within 150km of each other will have the same millisecond part of the propagation delay, rounded to the nearest integer. For mobile phones, it is natural and convenient to use cell tower locations as the landmarks [23]. A cell tower usually covers a radius less than 10km.

Although the millisecond part of the pseudorange can be estimated by using the landmark location, the lack of a synchronized clock presents an additional challenge. Since the

satellites move in the sky, a wrong estimate of time results in using a wrong value for the satellite location and thus the calculations yield an erroneous receiver location. Integer rollover is the main source of such errors in CTN. Van Diggelen [27] proposes the following method to reconstruct the full pseudoranges while avoiding the integer rollover problem. The signal travel time can be written as

$$NMS^{(k)} + \varphi^{(k)} = \tau^{(k)} - \delta_t^{(k)} + b + \varepsilon^{(k)} \quad (1)$$

$$= \hat{\tau}^{(k)} - d^{(k)} - \delta_t^{(k)} + b + \varepsilon^{(k)} \quad (2)$$

where $NMS^{(k)}$ and $\varphi^{(k)}$ are the millisecond and sub-millisecond components, respectively, of the transmission delay from SV k, $\tau^{(k)}$ is the actual travel time of the signal leaving SV k, $\delta_t^{(k)}$ is the satellite clock errors obtained from the ephemeris at the a priori coarse time for the satellite k, and $\varepsilon^{(k)}$ represents some unknown errors (tropospheric model error and other stochastic errors). The common clock bias b is the unknown to be eliminated.

As $\tau^{(k)}$ is unknown, it can be written as $\tau^{(k)} = \hat{\tau}^{(k)} - d^{(k)}$, where $\hat{\tau}^{(k)}$ is the estimated travel time from the a priori position at the coarse time of transmission, and $d^{(k)}$ is the error in $\hat{\tau}^{(k)}$. The method involves the selection of a reference satellite, $k = 0$, where $NMS^{(0)} = round(\hat{\tau}^{(0)} - \varphi^{(0)})$ is the millisecond part of the pseudorange of the reference SV[1]. This value is used to reconstruct the millisecond travel time of all other satellites relative to the reference satellite. Thus, if we subtract the Eq. (2) from the reference satellite full travel time we get

$$\begin{aligned} NMS^{(k)} = {} & NMS^{(0)} + \varphi^{(0)} - \varphi^{(k)} \\ & + \left(\hat{\tau}^{(k)} - d^{(k)} - \delta_t^{(k)} + b + \varepsilon^{(k)}\right) \\ & - \left(\hat{\tau}^{(0)} - d^{(0)} - \delta_t^{(0)} + b + \varepsilon^{(0)}\right) \end{aligned} \quad (3)$$

We still do not know the values of $d^{(0)}$ and $d^{(k)}$, but considering that we have an initial position and coarse time close to the correct values (about 100 km and 1 min), the order of magnitude of $(-d^{(k)} + \varepsilon^{(k)} + d^{(0)} - \varepsilon^{(0)})$ is far less than $1/2$ second, thus, we can correctly estimate $NMS^{(k)}$ by

$$\begin{aligned} NMS^{(k)} = {} & round\left(NMS^{(0)} + \varphi^{(0)} - \varphi^{(k)}\right. \\ & \left. + (\hat{\tau}^{(k)} - \delta_t^{(k)}) - (\hat{\tau}^{(0)} - \delta_t^{(0)})\right). \end{aligned} \quad (4)$$

Now, the full pseudorange is estimated by using NMS as the millisecond part, and the sub-millisecond is obtained by the code phase estimate. So, with CTN, the only device-dependent data are the acquisition results, which can be computed with as little as 1ms of raw signal. This is the key observation to move the location computation completely to the cloud.

3 CO-GPS Design

The design of Cloud-Offloaded GPS (CO-GPS) leverages the CTN principle but removes the dependency on nearby landmarks. For embedded sensors without cellular connections that are expected to have high mobility over their lifetimes, it is not always possible to provide nearby landmarks. Our key idea is to leverage the computing resources in the cloud to generate a number of candidate landmarks and then use other geographical constraints to filter out the wrong solutions.

3.1 Shadow Locations

The CO-GPS solution assumes a simple flow of information. In addition to application-specific functionality such as sensing, the device has three main components for location sensing — a GPS receiving module, a time synchronization module, and a data storage space.

In this section, we assume that the device is reasonably synchronized with a global clock, which we will elaborate upon further in section 5.2. When the device needs to sense its location, it simply turns on the GPS receiving front end and records a few milliseconds of GPS baseband signal. As we discussed in the previous section, we require at least 2ms worth of data to avoid possible bit boundaries.

The challenge of deriving receiver location with no reference landmark is the possible outliers, which we call *shadow locations*. Figure 3.1 illustrate the reason behind it using two satellites. Here, we model the pseudoranges from each SV as a set of waves, each 1 light-ms apart. Clearly, these waves intersect at multiple locations. Since we do not know the exact millisecond part of the propagation delay, all intersections, $A, B, C, D, \ldots$ are feasible solutions, even though only one of them is real. When more satellites are visible, more constraints are added to the triangulation, which helps resolve the ambiguity. However, the number of satellites alone is not enough.

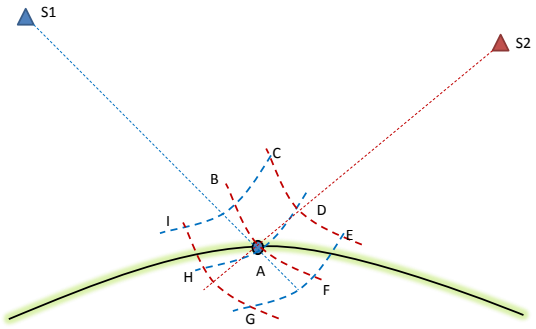


Figure 4. An illustration of multiple feasible solutions under NMS ambiguity.

To illustrate this effect, we take a 1ms real GPS trace and apply CTN with an array of landmarks across the globe. There are 6 satellites in view. The landmarks are generated by dividing the latitude and longitude with a 1^o resolution around the globe. In other words, we picked $180 \times 360 = 64800$ landmarks with adjacent distance up to 111km on the equator . Figure 5 shows the total of 166 converged points.

[1] [27] recommends the use of the highest satellite in view as reference, and provides a good reason for that.

Figure 5. All converged solutions for an example data trace, without landmark knowledge.

3.2 Guessing Reference Locations

The first step in eliminating shadow locations is to reduce the number of possible landmark guesses. Of course, if we know the past location of the sensor and can assume that it has not moved more than 150km between the samples, then we can use the past location as the landmark. However, in the bootstrap process, or when the time difference between readings is large enough to allow movement greater than 150km, we have to assume no prior knowledge of the location of the sensor.

Following the acquisition process, we can obtain the set of visible satellites and the frequency bin (identifying the Doppler shift) for each satellite. Knowing the signals' transmission frequency, the Doppler shifts tell us the relative velocity between the satellites and the receiver. If we know the absolute velocities of the satellites, which can be obtained from the ephemeris, then we can derive each angle between the satellite and the receiver, which defines a set of intersecting cones, as illustrated in Figure 3.2.

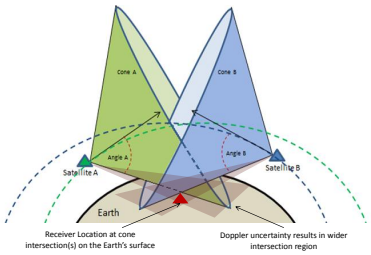


Figure 6. Two satellites A and B at the time of signal reception, the tangents to their orbits, and the angles calculated from the Doppler shifts at the receiver.

More specifically, as illustrated in Figure 3.2, let $S_k = (x_k, y_k, z_k)$ be the position of satellite k, and $V_k = (V_k^x, V_k^y, V_k^z)$ be its (absolute) velocity vectors in the Earth coordinate, the center of the Earth being the origin. Let $R = (x, y, z)$ be the location of the receiver. Due to the rotation of the Earth, the receiver inherits velocity $V_R = (V_x, V_y, V_z)$, uniquely determined by R. Relative to the earth rotation and satellite speeds, the receiver's motion is negligible. Let $\bullet$ be the operator for inner product and $||x||$ be the operator of 2-norm. Then,

$$\phi_k = cos^{-1}\left(\frac{V_k \bullet V_R}{||V_k|| \cdot ||V_R||}\right) \tag{5}$$

is the angle between the two velocity vectors. So the relative velocity of V_k in the direction of V_R is $V_R + V_k \cdot cos(\phi_k)$. So, the k-th Doppler shift can be represented as

$$D_k = \frac{||V_R + V_k \cdot cos(\phi_k)|| \cdot L_1}{c}, \tag{6}$$

where L_1 is the GPS signal frequency and c is the speed of light. In this set of equations, (x, y, z) are unknowns, so we need three satellites to uniquely identify the intersection. In reality, we may have more satellites in view and the problem can be formulated as a constrained optimization problem to minimize the sum of squared angle errors, such as in angle of arrival (AOA) localization solutions [1].

There are several uncertainties in our setting that make the problem more challenging. First of all, we do not have precise time stamps, so the satellite locations and their velocities may contain errors. Furthermore, the Doppler shifts we calculated during the acquisition process fall into bins that can be as wide as 500Hz, which in the worst case can cause 95m/s error in speed, or 12^o error in latitude. In implementation, we start with the solution assuming the precise Doppler shifts and time stamps, and search the nearby grid point landmarks to seek for converged solutions.

3.3 Solution Pruning

Due to the landmark guessing errors, the landmarks alone cannot rule out all shadow locations. Because a light-ms is 300km, the elevation of a shadow location is likely to be far away from the Earth's surface. For example, Figure 7 shows the number of possible solutions when we limit the elevation to be within the [-500, 8000] meter range.

Figure 7. All converged solutions for an example data trace, limited to the [-500, 8000] meter elevation range.

Obviously, absolute elevation by itself does not yield a unique solution. However, the true elevation of the Earth's

surface is known, and is available through web services from map providers. For example, the United States Geological Survey (USGS) maintains a service that returns the elevation of the Earth's surface at any given latitude/longitude coordinate. To see how this can be useful, we return to Figure 3.1. Location D is computed when the *NMS* we use for S_2 is one millisecond less than the true *NMS*. Assume the angle between S_2 and the tangent to the surface of the Earth is α, then the elevation difference between A and D is $\|z_D - z_A\| = 300 \cdot sin(\alpha)$. Taking $\alpha = 15^o$, which is the minimum elevation angle that GPS receivers consider a good view of a satellite, then $\|z_D - z_A\| = 77km$. Adding more satellites will make this difference even larger. It is almost impossible to have two nearby locations (hundreds km apart) such that both elevations are correct. We validate empirically that using an elevation map can always yield the correct solution.

3.4 Accuracy Considerations

A key design consideration of CO-GPS is the tradeoff between accuracy and energy expense. GPS signals are very weak when they reach the Earth's surface, and they suffer from multipath errors and obstruction by objects. Typical GPS receivers use long signal durations and tracking loops to overcome the low signal quality and to improve location accuracy progressively. Notice that the longer the signal is, the more robust the correlation spikes. This is the right thing to do for standalone GPS, since they need to subsequently decode the packet content, which requires good signal quality. However, sampling and storing large quantities of raw data brings energy and storage challenges to embedded sensor devices.

In CO-GPS, the only things we acquire from the signal are the code phases and Doppler shifts. The timing and ephemeris data are all derived off line. Because we do not decode signals, we can use much shorter signal lengths. An additional advantage of this is that it increases the likelihood of detecting satellites that are only intermittently visible, and whose signals fade out over the course of a longer signal sample.

The accuracy of time stamps is another concern. CTN can tolerate a certain amount of time stamp error, since it treats common time bias as another optimization variable. However, when this time error is too big, the least-squares process may not converge, or may converge to a wrong value. From the energy-efficiency perspective, maintaining a highly accurate global clock prevents the device from sleeping for long periods of time. To address the clock synchronization challenge, we can rely on clock radio [19] in different parts of the world. Although technical details vary, the mechanism is the same: a low power radio receiver can tune in to a low frequency band (60kHz in US) to receive atomic clock synchronization signals. The signals are not always available, depending on the receiver location, but is guaranteed to be present for a few hours every day in the entire continental US. Real-time clocks can be used to keep clock drift under a desired threshold between synchronizations. Obviously, the more accurate a real-time clock is, the longer it can sustain correct values between synchronization, and the less energy the device needs to spend on receiving the WWVB signals.

In Section 4, we will evaluate the effects of signal length, sampling rate, idling periods, and time accuracy in more detail.

3.5 Web Services

The cloud portion of CO-GPS has two main responsibilities: to update and maintain the ephemeris database, and to compute receiver locations given GPS raw data. We implemented these services on a cloud computing platform, Windows Azure, to achieve high availability and scalability.

3.5.1 Ephemeris Service

There are at least three sources of GNSS satellite ephemeris data sources on the web:

- NGS: The National Geodetic Survey of National Oceanic and Atmospheric Administration (NOAA)[2] publishes GPS satellite orbits in three types:
  - final (igs): The final orbits take into account all possible sources that may affect satellite trajectory. It usually takes NGS a few weeks to process and retrospect all inputs and to make igs available online.
  - rapid (igr): The rapid orbits are at least one day behind the current time. Most factors that affect satellite trajectories are taken into account, but not all.
  - Ultra-rapid (igu): The ultra-rapid orbits are predicted from known satellite trajectories into the near future. NGS's ultra-rapid orbits are published four times a day.

  Ephemeris published from NGS are in 15 minute intervals, and contains the location and clock correction for each GPS satellite.
- NGA: The National Geospatial-Intelligence Agency also publishes an independent GPS orbit data[3]. This data source, in addition to the satellite positions and clock correction at 5-minute epochs, also contains the velocity vector and the clock drift rate for each satellite. While it may take GNS 12-14 days to produce the final ephemeris, the NGA final ephemeris (called *Precise*) is usually available with a 2-day latency. In addition, NGA provides up to 7 days of predicted ephemeris, which contains the current ephemeris, but is less accurate.
- JPL: While NGS and NGA services are free, NASA JPL provides a paid service called Global Differential GPS (GDGPS) system[4]. It contains real-time ephemeris (position and clock correction only) updated every minute.

Our current implementation uses a combination of NGA and NGS data in the following order. We use NGA Precise as much as possible for historical dates. When NGA Precise is not available, we use NGS Rapids to the most recent date. After that, we use NGS Ultra-Rapids for real-time and near real-time location queries. We implemented a "monitor role" in Azure that periodically fetches data files from NGA and

[2] http://www.ngs.noaa.gov/orbits/

[3] http://earth-info.nga.mil/GandG/sathtml/ephemeris.html

[4] http://www.gdgps.net/

NGS. Then we perform a polynomial interpolation among the 5 or 15 minute sampling points to obtain a set of parameters [10]. Thus, we have a function that given a satellite and a time stamp returns its location and clock correction at that time. We further differentiate the polynomial to obtain satellite velocity at an arbitrary time.

3.5.2 Location Service

Putting everything together, the CO-GPS back-end web service must perform the following steps, as shown in Figure 8.

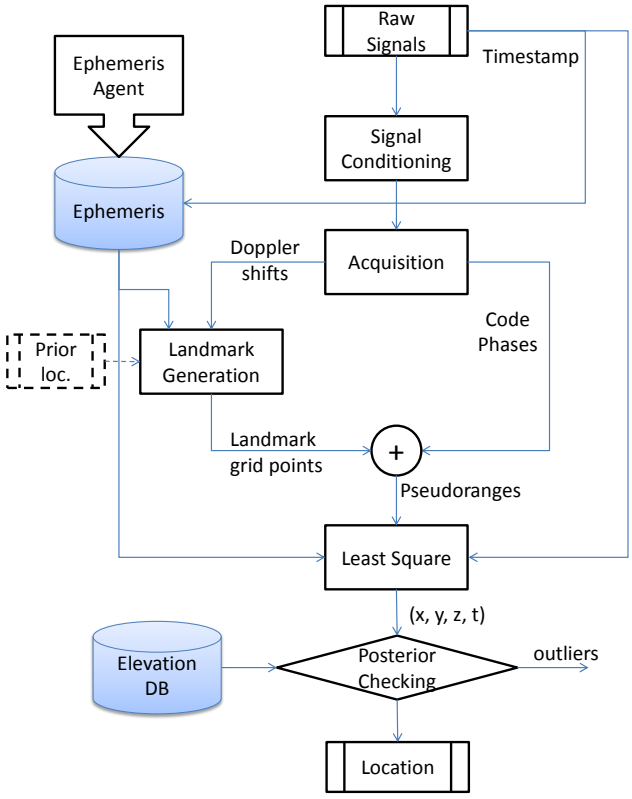


Figure 8. The flow of CO-GPS back-end web service.

We implement GPS satellite acquisition, course-time navigation, and least square procedures on .NET using C# and sho[5], and deploy the service on the Windows Azure cloud platform. A web role enqueues the location requests with GPS signal data and sensor configurations, a worker role dequeues and processes the requests, and a monitor role manages ephemeris updates. We aggressively use multi-threaded parallelism to achieve better scalability.

The Doppler shifts and code phases for each satellite are determined as part of the process of acquisition. The Doppler shifts are used with satellite velocities to infer the initial landmarks, as described in section 3.2. Or, if the location of the device is known in the recent past, then that history is used as a landmark. Once the set of landmarks is obtained, we can use equation (3) to calculate the NMS and apply the CTN method in parallel to process each landmark and solve for its location. Finally, absolute elevation constraints and the USGS elevation database are checked, and any result that is outside the error bound is discarded.

[5] http://research.microsoft.com/en-us/projects/sho/

4 Evaluation

We evaluate the quality and limitations of the CO-GPS approach using raw GPS samples.

This evaluation uses about 100 sets of raw GPS data taken from six different locations in both the northern and southern hemispheres of Earth. We used a SiGe GN3S v3 sampler dongle[6], which gives us the flexibility of varying the sample length for evaluation. Our data set contains the baseband GPS signal sampled between 10 to 60 seconds.

To measure the server-side performance, we use a quad-core PC with Intel Xeon 3520 @ 2.66GHz and 6GB memory. The acquisition process takes about 2.6 seconds to finish. Variations in signal length have a linear effects on the execution time mainly due to computing correlations in the frequency domain. Once the code phases are obtained, CTN and least square processes take less than 300ms to calculate the actual location. So if in a bootstrapping process we need to test 10 landmark hypotheses, the total execution time on the server side is less than 6 seconds.

For fair comparison, as the ground truth for these samples, we use a software-defined GPS package, called Soft-GNSS [4], to calculate the receiver's positions from the traces. In all data traces, Soft-GNSS achieves Geometric Dilution of Precision (GDOP) values below 6, which is considered at least *good*. The results produced by Soft-GNSS have expected errors less than 20 m.

When considering the values presented here it is important to note just how different the CO-GPS approach is from a standalone GPS implementation when the same signal trace is used. In addition to regular GPS error sources, CO-GPS adds the following possible sources of error: (i) the position is calculated by using code phase from samples that are closer to each other than in regular GPS; therefore they are more likely to suffer from transient noise that spans over multiple samples, (ii) CO-GPS does not use the lock loops (PLL and DLL) that are implemented in the tracking steps in regular GPS; only the code phase and Doppler frequency estimated in the acquisition step are used, which may contribute to less accurate results, and (iii) the use of CTN technique adds additional potential error, especially when the number of satellites is low.

4.1 Acquisition Quality

Since the goal of CO-GPS is to achieve the best possible energy efficiency in GPS sensing, we first evaluated how much data to use and the best duty-cycle strategy to employ to determine an appropriate trade-off between accuracy and low energy use. One of the key parameters that improves single location calculation is the number of satellites that can be acquired. So, we first vary the parameters to check the acquisition quality.

A potential method to improve accuracy is to use multiple chunks within each GPS location fix. That is, the receiver wakes up multiple times within a short period and collects

[6] available from Sparkfun Electronics http://www.sparkfun.com/products/10981

chunks of raw samples. Then the backend service average among the location results from each chunk to obtain the final fix. As illustrated in Figure 9, the chunk and gap parameters define the duty cycling of the receiver when sensing one location. Table 1 shows the three scenarios we evaluated to determine the appropriate amount of data to use to estimate the receiver's position. Each row is a different combination of the number of chunks, the chunk duration in milliseconds, and the time gap (or sleep period) between each chunk.

Table 1. Scenarios of Evaluation

	# of chunks	chunk length (ms)	gap length (ms)
1	1	{2, 4, 6, 8, 10}	0
2	{1, 2, 3, 4, 5}	2	0
3	5	2	{0, 10, 50, 100}

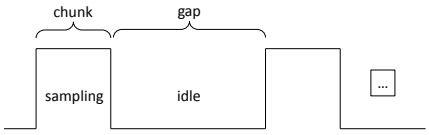


Figure 9. Duty cycling in experimental evaluation. After an idle period (called a gap), the receiver collects a chunk of raw data.

We extract a set of chunks of at least 2 ms duration and average the position outcomes derived from them. When the chunk length is longer than 2 ms, we use the first 2 ms for acquisition and the rest for the tracking loop to refine the code phase and Doppler results.

Figures 10 and 11 show the influence of these parameters on the number of visible satellites detected and in the results in terms of absolute error, respectively. Since our goal is to evaluate acquisition results, when multiple chunks are used, we select the chunk of best quality (in terms of having the maximum number of satellites) for location calculation. We evaluate the effect of averaging locations from multiple chunks in the next set of experiments.

Figure 10(a) shows that increasing the chunk length from 2 ms to 10 ms does not yield an increase in the number of satellites in view, since the acquisition is only done once. In Figure 11(a) we also observe that longer chunks do not improve the error. Therefore, the tracking loop, for up to 10ms long, does not provide any benefits to location accuracy. This implies that we should duty-cycle the GPS circuitry to save energy, by intercalating sleeping intervals between each sensing interval.

Figure 10(b) shows that the number of visible satellites detected can be improved if we take multiple chunks of data. Observe that the acquisition results using a contiguous 10 ms chunk of data are different from the results when we split this data into 5 chunks of 2 ms. The acquisition algorithm considers a satellite to be visible when it is stable along the entire signal, so it is more likely to recognize transient satellites in separate chunks of 2 ms than in a single 10 ms chunk. These transient satellites may help improve the position calculation. Thus, we can select only the chunk that yields the highest number of satellites in view and use this chunk to estimate the receiver's position. In Figure 11(b) we can observe that when only the best chunk is used among multiple chunks, the location accuracy only improved slightly. In fact, the average error barely improves, while the variance is lower with more chunks collected.

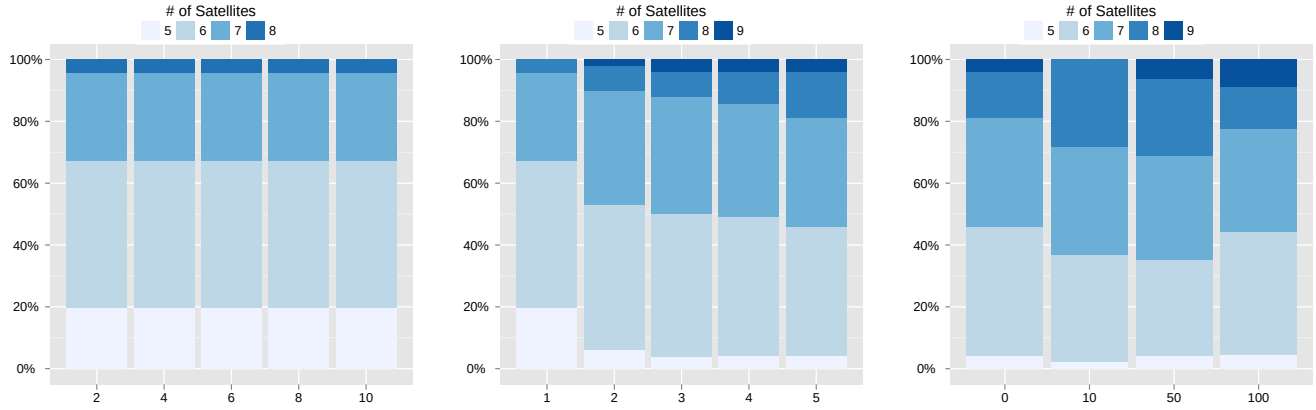
Figure 10(c) shows that we are also able to increase the number of satellites in view when we separate the sampling intervals by intercalating some sleep time (the gap duration). We see that for this parameter, the number of satellites in view generally increases as gap length increases, but not monotonically. This is because the signal can change over time in ways that are not always directly related to the gap duration. For example, a moving receiver may be blocked briefly by trees, buildings, bridges, or tunnels. Atmospheric conditions may also change slightly, and for large gap intervals, the satellite and/or receiver movement can result in a different satellite arrangement relative to the receiver. Empirically, we found that 50 ms appears to be a reasonable gap value as the movement of the satellites and the receiver can be considered negligible, and the obstacles and shadowing are not likely to change significantly. We also observe in Figure 11(c) that the error slightly decreases accordingly.

4.2 Overall Location Accuracy

Figure 12 presents graphs of satellite visibility and error metrics. Figure 12(a) shows the number of visible satellites recognized when each location request contained 5 chunks of 2 ms with a 50 ms gap between them. To improve the location accuracy, we use each chunk independently and then average the result location to obtain the final location.

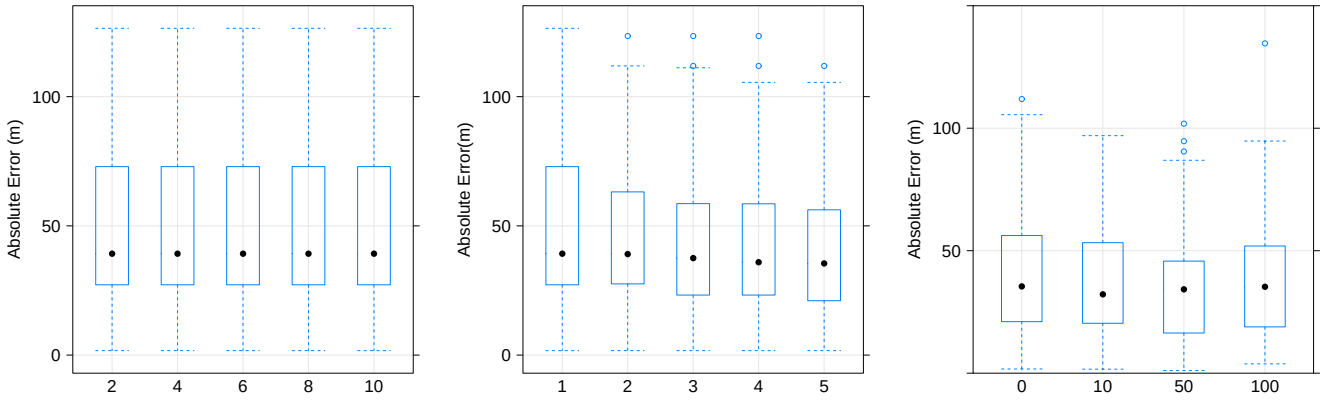
For our data, the receiver often has between 6 and 8 satellites in view, with 7 being the most frequent count obtained (Figure 12(a)). We see that in general, the more satellites in view, the more accurate the individual location fixes are (Figure 12(b)). As CO-GPS uses CTN, at least five satellites are required, and the accuracy is greatly improved when more satellites are available. Figure 12(c) compares the the error histogram when we use the single chunk approach, (i.e., when we process only the chunk that yields the largest number of visible satellites and discard the other 4 chunks) versus the error histogram from averaging results from multiple chunk. We observe that when we use multiple chunks, the results have a smaller variance and thus are more accurate.The significant improvements are because we are using more information to calculate the position while the samples are independent. Table 2 presents some statistics of the absolute error corresponding to the single and multiple chunk approaches. Observe that the mean error is about 20% smaller when the multiple chunk approach is adopted.

Figure 13 visualizes some outcomes of CO-GPS's location estimation for the 6 locations we evaluated. We plot a circle of radius 100 m around the ground truth, represented by a push pin, to give the sense of a block level accuracy (accurate to within a city block). As we can see, on average CO-GPS can achieve < 35m location accuracy with 10ms of data.



(a) Chunk Length (b) # of Chunks (c) Gap Duration

Figure 10. The number of acquired satellites in various experiment settings.



(a) Chunk Length (b) # of Chunks (c) Gap duration

Figure 11. Location error distribution in various experiment settings when single chunk is used for location calculation. In (b), we select the "best" chunk according to the number of satellites in view.

4.3 Time Sensitivity

Two pieces of information are required in order to obtain good results when CTN navigation is adopted: the initial position, and the time stamp corresponding to the moment that the GPS signal was collected. In order to fine-tune CO-GPS's time synchronization mechanism, we evaluated how the error changes as the time drift increases. This is a key parameter that influences how tight the time synchronization must be on a CO-GPS implementation.

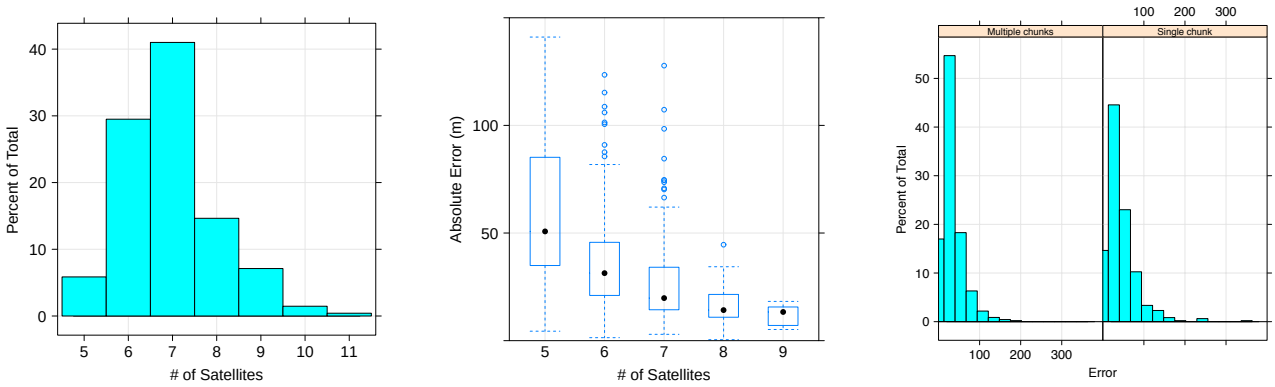
Figure 14 shows the error boxplots when the time drift increases from 0 up to 300 s. Observe that the error does not change significantly when the time drift varies from 0 to 60 s. After that, the error increases sharply, eventually reaching 10^6 m. This is due to the fact that under bad initial conditions, CTN navigation is not able to estimate the pseudorange millisecond part properly. Thus, as light travels about 3.10^5 m/ms, errors of the same order of magnitude are expected due to integer rollover.

5 Platform Implementation

Based on the CO-GPS principle and web service, we built a GPS sensor platform, code-named CLEO[7]. It is a reference design for low power embedded sensing nodes that can log time and the GPS signals received by a mobile object at a high sampling rate.

This reference platform consists of a GPS receiver (Maxim MAX2769), a microcontroller (TI MCU-MSP430F5338), a WWVB receiver module for time synchronization, and a serial flash chip for storage and some glue logic. The 8Mbit flash enables storing up to 1000 GPS sample points. The goal of the design is to demonstrate the low energy consumption per GPS sample. In addition, the platform also has a solar cell, a thin-film Micro-Energy cell battery, and a Hi-Jack[13] inspired audio communication port. For the purpose of this paper, the energy evaluation is

[7]CLEO: Cultivating the Long tail in Environmental Observations, for the potential applications of the platform in eScience.



(a) Statistics of the # of Satellites in view in 100 traces

(b) Individual location error statistic as a function of satellites in view

(c) Error distribution when results from multiple chunks are averaged to get a final location.

Figure 12. Overall location accuracy distribution.

Table 2. Error statistics

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Single Chunk	0.96	19.00	32.45	43.16	56.79	349.70
Multiple Chunk	0.45	16.56	26.67	33.77	42.17	181.10

on the GPS sensing part.

5.1 GPS Sensing

We selected the MAX2769 for the GPS receiver chip due to the relatively low power consumption (18mA in active mode), support for multiple GNSS standards (GPS, GLONASS, and Galileo), and the simple receiver design with fewer external components. MAX2769 includes a radio front end, RF down converter, and an ADC that generates a bit stream of the sampled down converted RF signal. We used the pre-configured mode 2 (c.f. MAX2769 datasheet[17]) of the receiver which uses a 2 bit ADC to generate 3 output signals : I_0, I_1, and GPS data clock. Each of these generates data at 16.368 Mbits/s.

Capturing the GPS data at this high data rate directly will require the microcontroller to operate at a very high frequency. For example, even a DMA based transfer requires 5 clock cycles on the MSP430, requiring a $\simeq$ 80MHz clock at the microcontroller. This is beyond the operating frequency of the MSP430. Microcontrollers operating with that high frequency will lead to huge power consumption in comparison. Hence, we first use a serial-to-parallel converter glue logic to reduce the microcontroller-side data transfer rate.

Figure 16 shows the serial to parallel glue logic. Each of the two data signals, I_0 and I_1, is connected to an 8 bit serial-to-parallel converter IC (74LV595) clocked by the 16.368 Mbits/s data clock. The data clock signal is divided by 8 using a binary counter IC (74LV161) to generate a 2.046 MHz signal which is used as a DMA trigger input to the microcontroller as well as a parallel load signal to the shift registers. Effectively, the glue logic converts the GPS signal to a 16bit parallel data stream and a GPS trigger signal that updates at 2.046 MHz. We use DMA, operating at 12MHz, for capturing and buffering this data inside the MSP430.

5.2 Time Synchronization using WWVB

Correct time stamping is a fundamental requirement to make CO-GPS work. To achieve low-power, high-accuracy time synchronization, we borrowed the approach of using WWVB signals, which is previously exploited in [6]. A CME6005 WWVB module offers universal time signal decoding from dedicated radio stations around the globe, such as WWVB (in US), DCF77 (in Europe), JJY (in Japan), BPC (in China) etc. Experimental evaluations in [6] have shown that at 2,400 km away from the WWVB station, the signal is available 47% of the time indoors and 75% of the time outdoors, while at 700 km away from the DCF77 station, the signal is available 97% of the time indoors. The time synchronization can achieve an accuracy of 3.9ms (indoor) or 4.3ms (outdoor). Furthermore, the power consumption of CME6005 is low, consuming less than 100μA in active mode and 0.03μA in shutdown mode.

To further reduce energy consumption, CLEO uses a real-time clock (RTC) in the MCU to manage time between synchronizations with WWVB. A normal 32,768 Hz crystal for RTC has an accuracy of 20ppm, which may induce a maximum of 1.728s drift over the course of a day. In the previous section, we have validated that CO-GPS can tolerate up to 60s time drift without increasing the location calculation error. This means that CLEO only needs to perform time synchronization once a month, which greatly reduces the energy consumption for receiving WWVB signals. Although the receiver itself only consumes 100μA active power, the MCU needs to be active to decode the time synchronization messages. Assuming that the device needs to run continuously for 12 hours to successfully receive the WWVB signal, amortizing the cost over 30 days will result in only 30μA average power consumption. Further reduction of WWVB sig-

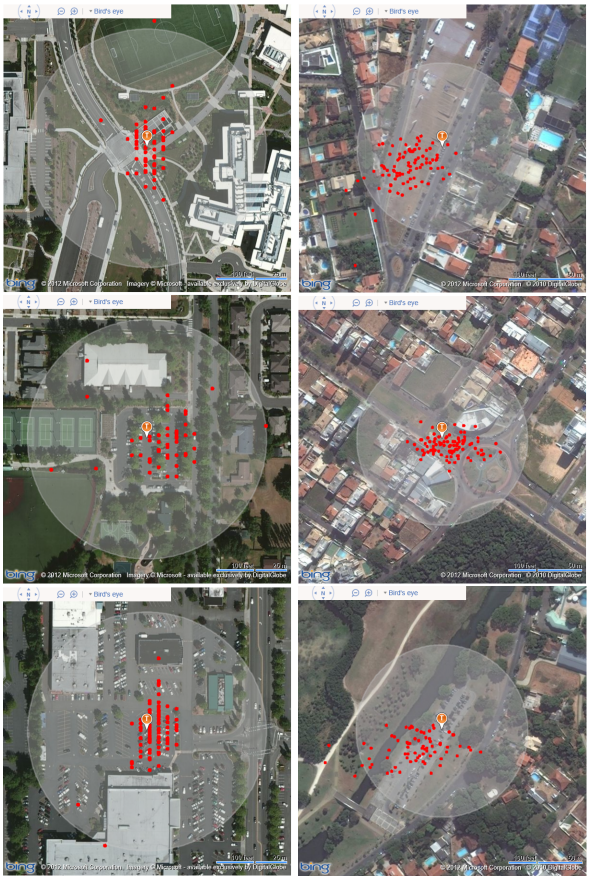


Figure 13. Overall results from 6 locations. The shadow is 100m in diameter. We see that there are bias errors in some cases.

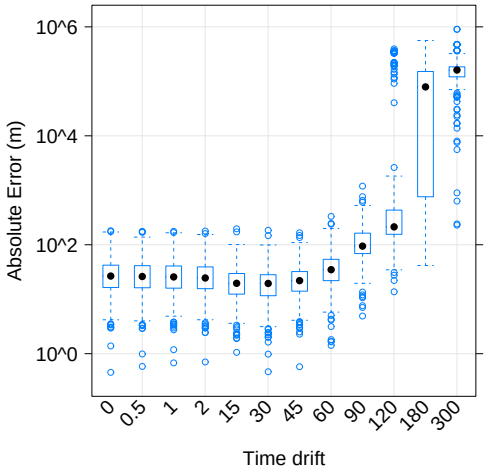


Figure 14. Errors due to time drift (in seconds).

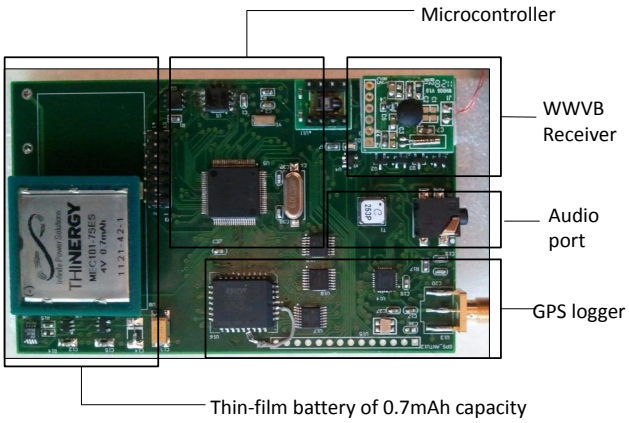


Figure 15. CLEO hardware platform and description.

nal searching is also possible by predicting the signal available time and duty-cycling the listening time.

5.3 Energy Consumption Evaluation

We use several techniques to control the power consumption of the platform. First, we turn off the power to modules with high leakage currents and make use of different power modes. We completely turn off the GPS receiver and the glue logic by using the *Enable* pin of the dedicated voltage regulator. We use glue logic with low-leakage power off capability to reduce the leakage current between the glue logic and the microcontroller. When not active, we operate the microcontroller at the LP3 sleep mode, with only the real time clock active.

Second, we use 3 different clock domains on the Microcontroller to reduce the average power consumption. Microcontroller DMA module operates at 12MHz to support GPS data rate, as well as for high speed burst writes to flash. Since the microcontroller CPU is only responsible for setting up of data transfers between GPS and flash modules through the internal RAM, and for decoding low-frequency WWVB data, the CPU core uses a $\simeq$ 2MHz internal low-accuracy clock. We use a low-power 32kHz real-time clock for maintaining system time.

To measure the power consumption of the CLEO platform, which runs at $3V$, we connect its power supply through two resistors (15Ω and 100kΩ) and short circuit the larger resistor at system power up and during certain operating modes; we measure the voltage drop across these resistors to calculate the current. The measured current draw at different operating modes is shown in Table 3. The average current for GPS receiving is significantly higher than the average operating current (27mA) due to the large in-rush current spike to charge the capacitors. There are opportunities to further optimize this factor.

Figure 17 shows an active working cycle for sampling and storing 2ms of GPS signal. The process starts from an idle state. It turns on the GPS receiving module for 2ms, and spends 28.8ms to write them into the flash chip. The total energy consumption of the process is $3V \times [1.5mA \cdot 28.8ms + 42mA \cdot 2.2ms] = 0.407mJ$. By comparison, an A-GPS on mobile phones takes about 1J for the first location fix. We achieve a gain of more than three orders of magnitude in device-side energy efficiency.

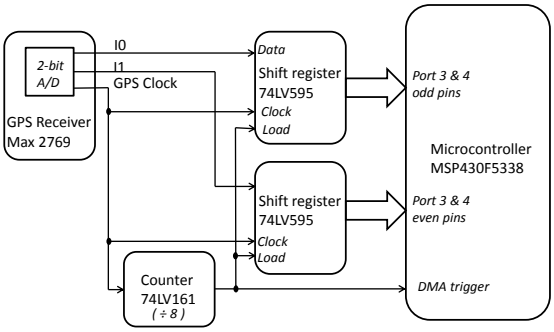


Figure 16. Glue logic between the GPS receiver and the microcontroller.

Mode	Current (mA)	Duration(ms)
System idle (RTC on)	0.01	–
MCU only (@2MHz)	0.65	–
Flash writing (GPS off)	1.5	28.8
GPS receiving (average)	42	2.2
WWVB (GPS off)	1.9	$<$ 12hrs/month
Total (logging 2ms GPS data)	4.37	31

Table 3. The power consumption of major components on CLEO

6 Related Work

Location sensing is a basic service in sensor networks. In most outdoor environments and for stationary sensors, researchers usually assume the locations are set using GPS at deployment time. For mobile sensors, there are two classes of solutions: one is to use public infrastructure and the other is to use deployed infrastructure. Public infrastructure includes GPS, WiFi access points[8], and FM radio stations [5]. When the system includes deployed nodes to assist localizing mobile nodes, signals like RF [7], sound/ultrasound [22, 3, 25], and magnetic coupling [15, 11] can be used as propagation media to provide distance or angle measurements. Our method falls into the first category of using publicly available infrastructure.

Although our solution is the first to propose cloud-offloaded GPS for embedded sensing, it is based on a rich body of work in GPS [18], A-GPS [27], and time synchronization [19]. With their integration into mobile phones, GPS and A-GPS have increasingly become low cost, low power and highly accurate. Commercial services, such as from Skyhook, Google, Apple and Microsoft, are available and may use WiFi and other signatures to improve location sensing coverage and latency. However, most previous work focuses on how to assist the mobile device in obtaining its own location. LEAP [23] is a first attempt to move GPS location calculation to the cloud. In contrast to CO-GPS, LEAP relies on the local processing power on mobile phones to derive the code phases. In our embedded sensing applications, the device may not have the processing power or energy to compute code phases locally. So we allow the device to simply store the raw data and later send it to the cloud for offline processing.

[8] e.g., `http://www.skyhookwireless.com/`

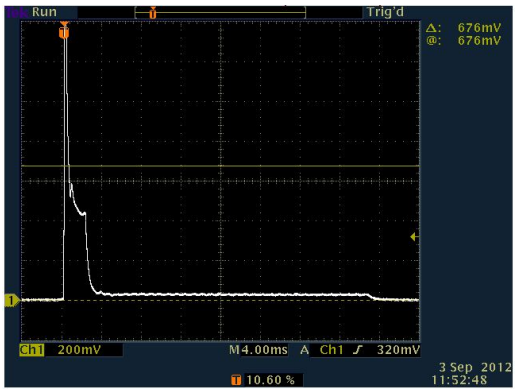


Figure 17. The current draw of a GPS sensing cycle

Our approach of using Doppler shifts to estimate the rough location of the receiver is related to the approach used in mobile transmitter tracking systems such as Argos[9], which is used in applications such as wildlife tracking and environmental monitoring. Argos uses multiple signals sent by a mobile transmitter to a single satellite over a given time interval. The satellite uses the varying Doppler shifts of these signals to infer the angles of arrival, which define cones with the satellite at their apex at each signal time. The intersection of these cones gives the location of transmitter. In general, the accuracy of Argos is within a few kilometers [2]. CO-GPS uses the same principle in the reverse way. We use multiple simultaneous signals sent by different satellites, and from these we determine the Doppler shifts, angles of arrival, and the cones that we intersect to guess the location of the GPS receiver.

Time synchronization is another rich topic in sensing systems. Most previous work focuses on synchronizing clocks within the system to achieve communication efficiency or activity coordination [8, 9, 16]. Since we process infrastructure signals, we need to synchronize with the global clock. Our application of WWVB receiver is learned from [6]. In indoor environments, [24] proposes to synchronize sensor clocks through observations of power line interference.

There are many options to choose from in communicating between sensors and data uploading devices. Our design is inspired by Hijack [13]. Alternatively, one may consider using Bluetooth, USB, or flash storage cards (e.g. MicroSD cards) that can be read by a computer.

7 Conclusion

Motivated by the possibility of offloading GPS processing to the cloud, we propose a novel embedded GPS sensing approach called CO-GPS. By using a coarse-time navigation technique and leveraging information that is already available on the web, such as satellite ephemeris and Earth elevations, we show that 2ms of raw GPS signals is enough to obtain a location fix. By averaging multiple such short chunks over a short period of time, CO-GPS can achieve $<$ 35m location accuracy using 10ms of raw data (40kB). Without the need to do satellite acquisition, tracking and decoding, the

[9] See `http://www.argos-system.org/`

GPS receiver can be very simple and aggressively duty cycled. We built an experimental platform using WWVB time synchronization and a GPS front end. On this platform, sensing a GPS location takes more than 3 orders of magnitude less energy than GPS on mobile phones.

The initial success of CO-GPS motivates us to extend the work further. On the sensor platform, we will explore other microcontrollers, especially those with FRAMs to further reduce the energy required to store the data. We will also exploit various compression techniques, especially those based on compressive sensing principles, to further reduce the storage requirements. On the web service side, we are exploring further optimizations to improve processing speed and location accuracy.

References

[1] I. Amundson, X. Koutsoukos, J. Sallai, and A. Ledeczi. Mobile sensor navigation using rapid rf-based angle of arrival localization. In *Proceedings of the 2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium*, (RTAS '11), pages 316–325. Chicago, IL, April 11-14, 2011.

[2] Argos Systems. *Argos User's Manual*. CLS group, 2011.

[3] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: a wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5):605–634, Dec. 2004.

[4] K. Borre, D. M. Akos, N. Bertelsen, P. Rinder, and S. H. Jensen. *A Software-Defined GPS and Galileo Receiver: A Single-Frequency Approach*. Birkhäuser, 2006.

[5] Y. Chen, D. Lymberopoulos, J. Liu, and B. Priyantha. FM-based indoor localization. In *Proceedings of The 10th International Conference on Mobile Systems, Applications and Services*, (MobiSys '12). Lake District, UK, June 25-29, 2012.

[6] Y. Chen, Q. Wang, M. Chang, and A. Terzis. Ultra-low power time synchronization using passive radio receivers. In *Proceedings of the 10th International Conference on Information Processing in Sensor Networks*, (IPSN'11). Chicago, IL, April 12-14, 2011.

[7] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan. Indoor localization without the pain. In *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, (MobiCom '10), pages 173–184. Chicago, IL, Sept. 20-14, 2010.

[8] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, (OSDI '02), pages 147–163. Boston, MA, Dec. 9-11, 2002.

[9] S. Ganeriwal, D. Ganesanl, M. Hansen, M. B. Srivastava, and D. Estrin. Rate-adaptive time synchronization for long-lived sensor networks. In *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, (SIGMETRICS '05), pages 374–375. Banff, Alberta, Canada, June 6-10, 2005.

[10] M. Horemuz and J. V. Andersson. Polynomial interpolation of gps satellite coordinates. *GPS Solut*, 10:67–72, 2006.

[11] X. Jiang, C.-J. M. Liang, F. Zhao, K. Chen, J. Hsu, B. Zhang, and J. Liu. Demo: Creating interactive virtual zones in physical space with magnetic-induction. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, (SenSys '11), pages 431–432. Seattle, WA, Nov. 4-7, 2011.

[12] E. D. Kaplan and C. J. Hegarty. *Understanding GPS: Principles and Applications*. Artech House, second edition, 2005.

[13] Y.-S. Kuo, S. Verma, T. Schmid, and P. Dutta. Hijacking power and bandwidth from the mobile phone's audio interface. In *Proceedings of the First ACM Symposium on Computing for Development*, (ACM DEV '10), pages 24:1–24:10. London, United Kingdom, Dec. 17-18, 2010.

[14] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao. Energy-accuracy trade-off for continuous mobile device location. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, (MobiSys '10), pages 285–298. San Francisco, CA, June 15-18, 2010.

[15] A. Markham, N. Trigoni, S. A. Ellwood, and D. W. Macdonald. Revealing the hidden lives of underground animals using magneto-inductive tracking. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, (SenSys '10), pages 281–294. Zurich, Switzerland, Nov. 3-5 2010.

[16] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi. The flooding time synchronization protocol. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, (SenSys '04), pages 39–49. Baltimore, MD, Nov. 3-5, 2004.

[17] Maxim Integrated. Max2769 universal gps receiver.

[18] P. Misra and P. Enge. *Global Positioning System: Signals, Measurements, and Performance*. Ganga-Jamuna Press, 2006.

[19] National Institute of Standards and Technology. *NIST time and frequency radio stations: WWW, WWVH, and WWVB*. Books LLC, Reference Series, 2011.

[20] M. Olson, A. Liu, M. Faulkner, and K. M. Chandy. Rapid detection of rare geospatial events: earthquake warning applications. In *Proceedings of the 5th ACM International Conference on Distributed Event-based System*, (DEBS '11), pages 89–100. New York, NY, July 11-15 2011.

[21] J. Paek, J. Kim, and R. Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, (MobiSys '10), pages 299–314. San Francisco, Ca, June 15-18 2010.

[22] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, (MobiCom '00), pages 32–43. Boston, MA, Aug. 6-11, 2000.

[23] H. S. Ramos, T. Zhang, J. Liu, N. B. Priyantha, and A. Kansal. LEAP: a low energy assisted gps for trajectory-based services. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, (UbiComp '11), pages 335–344. Beijing, China, Sept. 17-21, 2011.

[24] A. Rowe, V. Gupta, and R. R. Rajkumar. Low-power clock synchronization using electromagnetic energy radiating from ac power lines. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, (SenSys '09), pages 211–224. Berkeley, CA, Nov. 4-6, 2009.

[25] Z. Sun, A. Purohit, P. De Wagter, I. Brinster, C. Hamm, and P. Zhang. Poster: Pandaa: a physical arrangement detection technique for networked devices through ambient-sound awareness. In *Proceedings of the ACM SIGCOMM 2011 Conference*, (SIGCOMM '11), pages 442–443. Toronto, Ontario, Canada, Aug. 15-19, 2011.

[26] B. Thorstensen, T. Syversen, T.-A. Bjørnvold, and T. Walseth. Electronic shepherd - a low-cost, low-bandwidth, wireless network system. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services*, (MobiSys '04), pages 245–255. Boston, MA, June 6-9, 2004.

[27] F. van Diggelen. *A-GPS: Assisted GPS, GNSS, and SBAS*. Artech House, Boston/London, 2009.

[28] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi. Hardware design experiences in zebranet. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, (SenSys '04), pages 227–238. Baltimore, MD, Nov. 3-5, 2004.