Real-world Implementation of the Location Stack: The Universal Location Framework

David Graumann
Intel Corporation
david.graumann@intel.com

Walter Lara
Intel Corporation
walter.lara@intel.com

Jeffrey Hightower
Univ. of Washington
Dep't of Computer Science & Engineering
jeffro@cs.washington.edu

Gaetano Borriello
Univ. of Washington/
Intel Research Seattle
gaetano@cs.washington.edu

Abstract

Both the research community and developers in industry have identified the need for a clearly defined vocabulary and programming framework for location technologies. A layered Location Stack that provides appropriate abstractions, common terminology, and a clear API has been proposed in the last edition of WMCSA. This paper relates the experience gained in applying the Location Stack abstractions to the design and implementation of a location system using three separate location technologies integrated in a wireless computer. A single application interface, termed the Universal Location Framework, is designed to aggregate indoor, outdoor, and proximity sensor technologies. Our focus is on how well partitioning and parameter passing between the layers worked for our purposes as well as identifying system requirements not currently part of the stack. In closing, we provide evidence in support of the Location Stack's usefulness and we make some recommendations for its potential evolution.

1. Introduction

The Intel Universal Location Framework (ULF) is our implementation of location-aware computing on individual tablet, notebook, and handheld computers. This paper relates our experiences building a real system and the benefits realized by designing and thinking in terms of the Location Stack. The Location Stack, shown in Figure 1, is an existing six-layer design framework that establishes clearly defined abstractions building from raw sensor data up to context aware computing and activity inferencing [1]. This partitioning was proposed in the last edition of WMCSA after architectural review of several existing location-aware systems. Using the Location Stack as a guide, we establish a flexible framework capable of utilizing plug-in location sensing technologies that start or stop providing location data as the mobile device moves through various environments. Our primary focus is on the lowest three stack layers comprised of sensor, measurement, and fusion capabilities. We expose the Fusion layer directly to applications interested in geometric location. We expect that the additional layers (Arrangements, Contextual Fusion, and Activities) can be built independently and utilize the Fusion layer interfaces we provide. (See [1] for definitions and references to research on the upper layers).

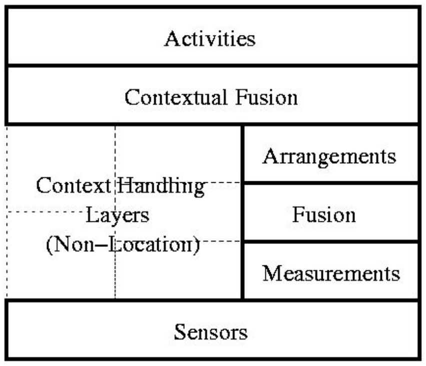

Figure 1. Location Stack.

Our design must meet the real needs of both the platform hardware and third-party applications. We identify three design criteria. First, we must provide a uniform location interface to applications allowing both push and pull access to location information. The API must be technology neutral yet provide access to low-level details if desired. Second, handoff between sensor technologies and simultaneous fusion of measurements from multiple sensor technologies must be possible. Finally, it should be easy to extend support to new location technologies without redesign and, most importantly, without modification of application code. Our prototypes are built on tablet computers with Intel® Centrino™ mobile technology and demonstrated in real-time using custom maps of our building and surrounding campus. We use three sensor technologies: GPS, 802.11 indoor positioning, and UC Berkeley sensor motes as proximity beacons. This collection of sensors enables a range of realistic configurations and environments including indoor sensing, outdoor sensing, and both hierarchical and geographical handoff between location sensing technologies. GPS provides outdoor location when the device is in view of the sky. 802.11 positioning provides coarse location indoors, in courtyards, and occasionally on city streets. The UC Berkeley motes are used as a flexible radio capable of mimicking a Bluetooth™ proximity sensor or providing crude GPRS – like location information. We primarily configured the motes to represent 10ft range Bluetooth™ proximity sensors.

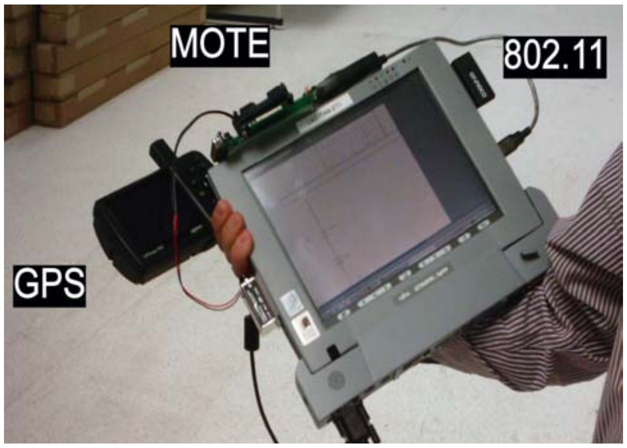

Figure 2. Demonstration platform.

To meet our goals, we implement the bottom three abstraction layers of the Location Stack: Sensors, Measurements, and Fusion. The Sensor layer encapsulates drivers to extract data from sensing hardware. The Measurements layer abstracts this raw sensor data into a common set of measurement types such as distance, angle, and velocity. The Fusion layer probabilistically merges measurements into a dynamic representation of an object's location and presents a uniform interface to this location information for higher-level layers. We expose the Fusion layer interface (with a few embellishments) directly to applications. As we built this system it became clear that we needed to deal with several real-world issues introduced by our selection of location sensing devices. For example, we have both sophisticated and simple sensor types. GPS units contain a complete navigation system while the UC Berkeley motes we used as proximity sensors provide just an ID value in a radio message. Between these two extremes lie 802.11 positioning technologies which can behave as simple proximity sensors or provide coarse real-time navigation. Looking beyond individual sensor integration, this effort has forced us to address implications on security, privacy, and real-time streaming as they relate to the Location Stack abstractions.

2. Applying location abstractions to ULF

We use the Location Stack abstractions to guide our design and integration of the three sensor technologies into the Universal Location Framework. Our internal canonical measurement representation consists of WGS-84 [2] latitude, longitude, altitude, ground velocity, heading from true north, elliptical arch or polygonal uncertainty along the earth's local tangential plane, a separate uncertainty for elevation, the technology type, and a timestamp. All location fixes are presented to the fusion layer in these terms. The two uncertainty values provide a flexible method to describe a location fix's horizontal error. The elliptical arch can describe a circle, ellipse, or sector. The polygonal shape is a list of location points with the last location tying back to the first. One of the two uncertainty descriptors in conjunction with the other location parameters completely describes the fusible location fix. Our fusion algorithm handles these complex shapes. The following sections relate the issues encountered integrating each technology under these design choices.

2.1 Global Positioning System stack integration

GPS is the most sophisticated sensing device we used. GPS units contain a complete navigation system capable of providing latitude, longitude, altitude, heading, velocity, highly accurate time, waypoints, and several variations of location uncertainty. This information is provided as a serial ASCII sentence defined by the NMEA standard [3] or a proprietary binary format. Their internal capabilities directly fulfill the requirements of the Sensor and Measurements layers of the Stack excluding the simple transformation into our chosen canonical measurement representation.

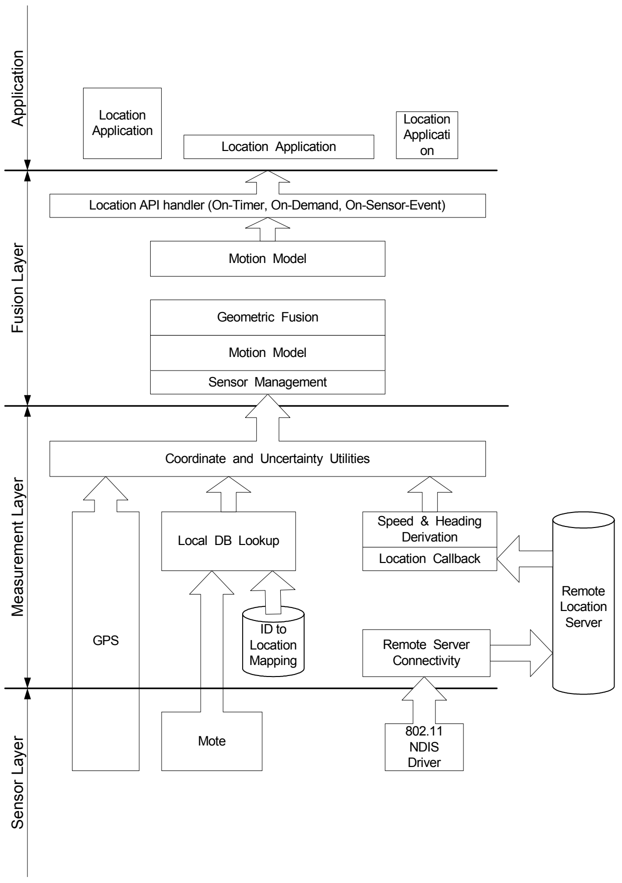

Figure 3. Universal Location Framework block diagram.

Though GPS integrated easily, the sophistication of GPS devices presented other problems. Two cases required interrogation of the GPS location report prior to use at the Fusion layer. The uncertainties of a GPS location report are described in multiple ways. In addition to the required Dilution of Precision (DOP) information, GPS vendors provide location uncertainty values that are more indicative of the errors experienced by the end-user. Unfortunately, these messages come with proprietary headers and at different times than the main location fix. In our implementation, we pay special attention to the uncertainty of the report because it is critical to the location's geometric fusibility. We resolve this by adding a bundling/parsing stage in the measurement system to collect a complete GPS location report before conversion to the canonical form. If the proprietary message is not present or is not recognized then we use the standard DOP. In this way, by abstracting the ability to recognize proprietary error messages within a configuration file, we are able to select the best possible representation of uncertainty without a priori knowledge of the brand of installed GPS device.

Along similar lines as the bundling stage, we need to account for erroneous location fixes being reported by GPS when operating in courtyards and transitioning from outdoors to indoors. Many GPS units continue to provide seemingly valid location reports even after complete loss of satellite tracking. These reports were often erroneous, especially when sharp turns were taken after satellite loss. By monitoring the number of satellites used to determine location as well as the number being tracked in the sky, we can better determine how to utilize a single GPS location fix in the presence of other sensing technologies. In adherence to the abstraction layers, we chose to propagate the sensor technology type, in this case GPS, and the satellite information up through the Measurements layer to the Fusion layer so that the fusion layer can best determine the course of action to take.

Our integration of GPS establishes the Sensor layer as the sole owner of the hardware driver. This is adequate for our needs but presents a foreseeable problem for Assisted GPS. Assisted GPS requires a mechanism to "seed" the satellite acquisition engine with up-to-date satellite information. However, the ability to relay externally acquired data down the stack did not have an obvious mapping to the abstractions.

2.2 Proximity sensor stack integration

We use UC Berkeley mote ID messaging, 802.11 MAC addresses, and received signal strength for proximity sensing. All these technologies map easily into the Location Stack model. Mote IDs and 802.11 MAC addresses are used as look up keys into a locally cached database. The database contains a record defining the canonical location form. The mote database format was intentionally defined to closely match the Bluetooth Local Positioning Profile allowing us to easily replace this sensor/database pair with a Bluetooth radio and a packet received over the communication channel. In our case, however, proximity sensors used in this way do not provide velocity or heading, therefore, in addition to passing the sensor type, these fields are set to N/A as opposed to zero to clarify their use up at the fusion layer.

We ran across an interoperability issue when interchanging proximity sensor client receivers. The notion that one can interchange similar technologies at the Sensor layer while maintaining the same Measurements layer did not hold true in practice with our first implementation of RF energy-based sensing devices. It was determined that the measured receive energy from either a mote or 802.11 access point is a function of the receiver's radio front-end circuitry. Due to component variations, and even assuming all other conditions are held constant, different sensors report the received signal strength differently. This varies dramatically across sensor vendors but also from unit to unit. Though this is an issue with the specific technology, by placing the burden on the Sensor layer to pass reference RF sensitivity information to the Measurements layer a normalized measurement database can be used in all cases. Thus, we had to slightly expand the interface between the Sensor and Measurements layers.

The static nature of proximity sensors presents an opportunity to include the concept of an exclusion region within which the client devices cannot reside. For example a mote placed against the outside wall of a second floor describes its uncertainty shape as a half circle. This introduces a non-Gaussian uncertainty that we termed an "embedded rail." Since we have already chosen methods to handle this descriptor at the Fusion layer, the "embedded rail" is naturally reflected in the final location resolution. This technique allows us finer granularity control of the proximity sensor's influence on our location belief than simply describing their emitted power pattern. In relating this to the Location Stack we see that an "embedded rail" conceptually maps to obstacles and object relationships residing at the Arrangement layer which we did not include in our design. This abstraction guidance does not seem appropriate for our needs. However, the "embedded rail" is very simple and does not obviate the need for a complete description of obstacles and arrangement.

One requirement for geometrically fusing location sensors is the need to resolve all location fixes to a common coordinate system. This restricts our system's extensibility to sensors whose information can be mapped to WGS-84 [2] positioning. For indoor positioning, this is not necessarily the simplest or most natural coordinate space. One simpler form is the horizontal plane coincident with the building floor along with a floor number. This information suffices to place the device on a local map without consideration for the earth's curvature or true North. However, using this coordinate system presents a problem when transitioning to outdoor sensors that report location in world coordinates. Without a means of relating the two coordinate systems a discontinuity occurs at the Fusion layer.

2.3 802.11 Navigation stack integration

We implemented 802.11 location and navigation using a state-of-the-art indoor positioning engine based on analysis of the electromagnetic characteristics of 802.11 beacons similar to the classic RADAR system [4]. This technology requires a calibrated database of access point signal characteristics, which in our case is stored on an external server. The Sensor layer is implemented as an NDIS™ driver interface while the Measurements layer provides the remote server connection and callback service (Figure 3). The server's response describes location in a local coordinate system that is converted to WGS-84 within the Measurements layer. This technology provides a location update every few seconds barring network congestion, with an accuracy ranging between 3-30 meters depending on access point configurations.

This particular 802.11 location technology has no provisioning for uncertainty, velocity, heading, or altitude parameters necessary to completely describe a fusible location. To compensate, we enhanced the Measurements layer to derive the necessary information. Formal evaluation revealed that the uncertainty is a function of location and can vary between 3-30 meters. We were unable to generalize these rules and simply use an accuracy of 7 meters derived from empirical testing. A Kalman filter was added to derive velocity and heading. The acceleration/deceleration uncertainty handles stationary to running conditions and the velocity noise was set comparable to the location uncertainty. The altitude was pre-assigned. These approximations are now under evaluation. However, at a minimum it is preferred to derive uncertainties directly from the sensor's real-time data.

Conveying location information between a server and client introduces an important security concern that proves difficult to adequately address. In a very simplistic manner, we apply the first level of security by tunneling the location content through a Virtual Private Network (VPN). This limits visibility to within the network but many more aspects must be considered that are outside the scope of this paper.

Wireless LAN 802.11 technologies are notorious for battery consumption and we seek to control power management capabilities of 802.11 hardware from within ULF. For example, if all running applications simply need on-demand location reports, then the scan rate of the 802.11 can be considerably reduced or even turned off until a location fix is requested. The Location Stack abstractions do not provide us with insights on how to best partition this or other similar mechanisms requiring control of the underlying sensors. This is a common weakness of many layered software models: they provide robust separation of concerns but make it difficult to handle crosscutting concerns.

2.4 Fusion layer implementation

The Fusion layer merges measurement reports into the best "belief" of the device's location. The core of our implementation stems from ongoing research in robotics and multi-sensor location estimation at the University of Washington [5,6,7]. We do not discuss the details of location estimation as it is outside the scope of this paper. In those activities, as in ours, Bayesian filtering in the specific form of particle filtering is used to manage location uncertainty and establish an optimal estimate of the true position. This algorithmic choice grants us the opportunity to resolve the richly described probabilistic models being provided by the Measurements layer (as we've seen above, they are not simple Gaussian models). After considering our particular sensors, we subdivided our Fusion layer into three stages: Sensor Management, Motion Modeling, and Particle Filtering.

Sensor Management interrogates the sensor type and qualifies the location report before submission to the other stages. Heuristics handle boundary cases including stale location reports, unbalanced rapid reporting of proximity sensors, and GPS reports derived from zero satellites. In the absence of multiple sensing technologies, any and all location fixes are used. However, when multiple location technologies are maturing a motion model and location belief, these anomalies degrade the system. To perform this task we keep track of the recently observed technologies, a timestamp tightly coupled to the raw data acquisition time, and some sensor specific details. We propagate these parameters up the stack to the Fusion layer, because here we have the vantage point of making informed decisions about multiple technologies. The choice still remains to move this information even farther up the stack for the application to inspect.

The Motion Model is applied to the internal state of the particle filter prior to the inclusion of the current location fix and prior to reporting a location to an application. We use a dynamic motion model that includes velocity and heading as provided by the sensors. When these observations are not available then the motion is updated assuming a walking human stochastic process. The use of motion estimation within this layer has the added advantage of allowing an application to obtain an accurate location fix asynchronously to the underlying reports from the sensors.

As the final Fusion stage, a Bayesian filter implemented as a particle filter is applied using the probabilistic models described by the location descriptor from the Measurements layer. The internal location representation of the particle space occupies a Cartesian coordinate system representing East-North-Up (ENU). The space also contains dimensions for velocity and heading. All measurement descriptors are aligned with ENU coordinates. The particle filter elegantly incorporates these irregular probability distributions into its state [5,6,7].

2.5 Application interface stack integration

One of the main goals of our ULF design is to insulate applications from the burden of direct interaction with low-level location technologies. We realize this goal by building our Application Programming Interface (API) on top of the Fusion layer. The API provides location information to the application in the form of *location reports* consisting of: timestamp, position (as per WGS-84), and uncertainty. In addition to reporting the geometric shape descriptor extracted from the full probability estimated by the particle filter, we provide a simplistic location and accuracy value derived from the average radius of the geometric shape created by the distribution of particles.

Another goal of our design was to deliver location information in a very flexible manner. Specifically, we want our API to be able to provide location information on-demand, periodically, or when new information is available. To accommodate these requirements our API provides the ability for the application to register interest in location reports through a callback mechanism. We created three different types of reports:

- *Automatic*. Generated whenever the Fusion layer has new information as a result of an update from the Measurements layer. This type enables applications to implement trigger mechanisms. For example, an application may be interested in setting an alarm if the detected position is outside a specific region.
- *Manual*. This report is triggered in response to a query. This type is used by applications that need to know the location only when the user requests it.
- *Periodic*. Generated at a time interval specified by the application. This type is most suitable for applications requiring smooth real-time tracking of the device's location but can also be used for

slow background logging of position over long periods of time.

We have identified the need for the API to support waypoint functionality. This involves the caching and comparing of location information within the stack. In some cases these waypoints do not need to represent physical locations in a coordinate system. Rather they could represent an identifiable sensor configuration or signature. An example of this is the latching of raw 802.11 access point MAC addresses, signal strengths, signal-to-noise ratios, and energy fluctuations as a uniquely discernable waypoint. This could be captured at the Fusion layer but does not need a physical representation or Measurements layer. We are currently considering this design option for future inclusion as a feature within the Location Stack.

3. Future work

Privacy and Security. We addressed privacy and security in a mild manner within the Measurements layer and recognize the need for a complete treatment at higher levels of the stack and in more flexible ways. We believe that users want to control who gets their location information and when. The ability to breach the user's wishes by inappropriate use of their location is a potential liability not addressed by the current stack. Advocating interchangeable layers introduces additional avenues for malicious programs to insert themselves into the system and the identification of stack capabilities that address this problem is ongoing.

Waypoint logging and triggering. The identification of a location waypoint can be achieved with or without a final geometric resolution. Unique characteristics of raw data within the Sensor layer can be just as meaningful as latitude, longitude, altitude when saving and comparing similarities between two locations. Evolving the location stack abstractions to encompass the creation, preservation, and management of raw waypoints would more completely describe the capabilities we contend are necessary for the Universal Location Framework.

Common Coordinate Systems. For the particular applications we are enabling on a notebook or handheld computer the managing of multiple sensor technologies with the Location Stack greatly benefits from the ability to relate all coordinate systems to one another. Promoting standards that allow the ability for local or proprietary coordinate systems to be transformable into one with universal acceptance could be beneficial to deploying a location stack with interchangeable technologies within the Sensor/Measurements layer. The plausibility of this requirement remains an open question.

4. Conclusion

The Location Stack abstractions provided meaningful guidance as we built a real-time demonstration platform unifying three location technologies under a common programming interface. We encountered several real-world issues that have matured our thinking on future directions for using the Location Stack on notebook, tablet, and handheld computing devices. From a system integration perspective, the Measurements layer requires the most attention. The biggest stack disruption occurs by placing remote server connectivity within the Measurements layer, which brings to light many issues surrounding stability and security. By designing this layer with a common set of coordinate transformation utilities, tightly coupled timestamps, and VPN tunneling, we were able to establish a unified fusible data format with a variety of plug-in technologies.

While the majority of our design maintains the Location Stack abstractions, we deviate in a few regards. We elected to implement the Fusion layer with knowledge of the technology type and a few attributes to assure proper removal of location anomalies. This choice does limit our ability to freely exchange layers without a better abstraction of technology behaviors, but at this time this is not seen as overly restrictive. We worked outside of the Location Stack abstractions when relaying programmatic control parameters up and down the stack. Because our design establishes the Universal Location Framework as the sole owner of the sensor hardware drivers, we have the extra burden of handling non-geometric sensor information. Though these requirements did not map directly to the abstractions presented in the Location Stack, they were simple to accommodate.

In summary, the Location Stack has proven itself in providing a valuable set of abstractions for building location-aware systems. Our success in enabling a tablet with the ability to use three disparate location sensing methods while leaving room for future evolution provides evidence of this. In the process, we have identified several new directions for development of our Universal Location Framework (and the Location Stack on which it is based). We look forward to investigating these new issues in making location-aware systems practical and ubiquitous.

5. References

[1] Jeffrey Hightower, Barry Brumitt, and Gaetano Borriello, "The Location Stack: A Layered Model for Location in Ubiquitous Computing", in Proceedings of the 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002), (Callicoon, NY), pp. 22-28, June 2002.

[2] EUROCONTROL: European Organization for the Safety of Air Navigation, "World Geodetic System 1984, Implementation Manual", Version 2.4, February 1998.

[3] National Marine Electronics Association, "NMEA 0183 Standard for Interfacing Marine Electronics Devices", Version 3.01, July 2000.

[4] Paramvir Bahl and Venkata Padmanabhan. "RADAR: An in-building RF-based user location and tracking system", in Proceedings of IEEE INFOCOM, volume 2, pages 775-784, Tel-Aviv, Isreal, March 2000.

[5] Dieter Fox, Jeffrey Hightower, Lin Liao, Dirk Schulz, and Gaetano Borriello, "Bayesian Filtering for Location Estimation", in Pervasive Computing, vol. 2, no. 3, IEEE Computer Society Press, July-September 2003.

[6] Dirk Schulz, Dieter Fox, and Jeffrey Hightower. "People tracking with anonymous and id-sensors using rao-blackwellised particle filters", in Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI), 2003.

[7] Cody Kwok, Dieter Fox, and Marina Meila. “Adaptive Real-Time Particle Filters for Robot Localization”, in Proceeding of the IEEE International Conference on Robotics and Automation (ICRA), 2003