

A Location-Aware Framework for Intelligent Real-Time Mobile Applications

The Location-Aware Information Systems Client (LAISYC) supports intelligent, real-time, mobile applications for GPS-enabled mobile phones by dynamically adjusting platform parameters for application performance while conserving device resources such as battery life.

One of the most valuable pieces of contextual information for an intelligent mobile application is the user's location. Because of the complexity of realizing location-aware capabilities for cellular devices, location-based intelligence is only now emerging in commercial mobile phone applications. Several factors are contributing to the renaissance of location-based services (LBS):

- Positioning technologies (such as high-sensitivity GPS) have improved accuracy and time-to-first-fix even in obstructed indoor environments.
- Mass production of GPS hardware in mobile phone chipsets has reduced the cost of embedded GPS hardware.
- The emergence of cross-platform application environments for mobile phones, such as Java Micro Edition (Java ME).

Java ME provides standardized programmatic access to location data through the JSR179 Location API and allows applications

to run in the background (through a multitasking virtual machine), both key requirements for widely deployable location-aware applications. Because a device-based location API is critical for supporting real-time location-aware mobile applications, third-party application developers must assume responsibility for efficiently managing location data in a location-aware system. Unlike other types of mobile applications, location-aware software often runs in the background for extended periods of time to monitor the device's real-time position. Because each position calculation using embedded GPS hardware expends battery energy, as does reporting this position to a server, battery life is a critical concern in mobile application design for LBS. As a result, location-aware applications require intelligent client software that conserves device battery energy while meeting application functionality requirements that can vary dynamically during execution.

The *Location-Aware Information Systems Client (LAISYC)* is a comprehensive, location-aware framework that supports intelligent and energy-efficient real-time distributed applications for Java ME. Any third-party software developer can implement LAISYC using standard IP-based networking protocols. The framework

Sean J. Barbeau, Rafael A. Perez,
Miguel A. Labrador,
Alfredo J. Perez, Philip L. Winters,
and Nevine Labib Georggi
University of South Florida

Related Work in Location-Based Services

Although many location-based services (LBS) architectures are documented in literature, none includes a comprehensive location-aware framework for intelligent, real-time applications for GPS-enabled mobile phones. Following the E911 mandate, some publications targeted the implementation of core positioning technologies by cellular carriers.¹ The focus of academic works then turned to general LBS including emergency² and commercial services either tightly coupled to the cellular infrastructure or maintained by a centralized entity.^{3–5} These architectures are of limited use to third-party mobile software developers because LBS functionality is tightly controlled by the centralized entity, and the scope of location data use and system settings are limited. Other location-aware architectures have focused on the Session Initiation Protocol (SIP), an application-layer protocol often used for Voice over IP,⁶ but SIP is currently not widely supported in Java Micro Edition.

Our research focuses on an architecture that can be implemented in its entirety by third-party application developers on most Java ME devices. The implementation of this architecture uses publicly available, standardized APIs and doesn't require programmatic interaction with a centralized system that controls all LBS for a cellular network.

In previous work, we presented a general architecture in support of interactive, multimedia, location-based mobile applications.⁷ Our primary focus was the integration of location data into multimedia messaging service (MMS) messages sent through a cellular carrier's publicly accessible gateway. The *Location-Aware Information Systems Client* (LAISYC) adds support for real-time, intelligent IP-based location-aware clients to this architecture. Our work differs from the Mobile Millennium project by Nokia and the University of California, Berkeley,⁸ in that

LAISYC focuses on delivering personalized, real-time services to users based on their real-time locations and travel history. The Mobile Millennium project focuses on collecting anonymous, aggregated GPS probe data from mobile phones as they cross virtual trip lines on highways to estimate travel times and disseminate this information to the general public.

REFERENCES

1. Y. Zhao, "Standardization of Mobile Phone Positioning for 3G Systems," *IEEE Comm.*, vol. 40, no. 7, 2002, pp. 108–116.
2. M. Mintz-Habib et al., "A VoIP Emergency Services Architecture and Prototype," *Proc. 14th Int'l Conf. Computer Comm. and Networks (ICCCN)*, IEEE Press, 2005, pp. 523–528.
3. M. Spanoudakis et al., "Extensible Platform for Location-Based Services Provisioning," *Proc. 3rd Int'l Workshop Web and Wireless Geographical Information Systems (W2GIS 03)*, IEEE CS Press, 2003, pp. 72–79.
4. A. Kupper and C. Linnhoff-Popien, "TraX: A Device-Centric Middleware Framework for Location-Based Services," *IEEE Comm.*, vol. 44, no. 9, Sept. 2006, pp. 114–120.
5. Y. Chen et al., "LORE: An Infrastructure to Support Location-Aware Services," *IBM J. Research and Development*, vol. 48, no. 5/6, 2004, pp. 601–615.
6. Z. Shah, R. Malaney, and N. Dao, "An Architecture for Location Tracking Using SIP," *Proc. IEEE Global Telecomm. Conf. (GLOBECOM 07)*, IEEE Press, 2007, pp. 124–128.
7. S. Barbeau et al., "A General Architecture in Support of Interactive, Multimedia, Location-based Mobile Applications," *IEEE Comm.*, vol. 44, no. 11, 2006, pp. 156–163.
8. S. Amin et al., "Mobile Century-Using GPS Mobile Phones as Traffic Sensors: A Field Experiment," *Proc. 15th World Congress on Intelligent Transportation Systems*, ITS America, Nov. 2008.

supports various types of location-aware applications—from real-time tracking to more delay-tolerant applications focused on recording accurate travel paths, or even hybrid applications with real-time and delay-tolerant features—by dynamically manipulating parameters according to real-time application needs. Because LAISYC's design is modular, we can integrate other work in client-side location intelligence. (See the "Related Work in Location-Based Services" sidebar for how LAISYC differs from existing research in LBS.) We have evaluated LAISYC by implementing it in several intelligent real-time

mobile applications, which are discussed in the "Evaluation" section.

LAISYC Communications Framework and Components

LAISYC's two-tiered communication protocol uses HTTP (or HTTPS, for secure transfer) to transport all nonlocation information required for application execution (application data), and UDP to transport location data.

HTTP supports application data transferred between the device and server through a device-initiated request-response model similar to RPC. Integrated development environments

(such as Netbeans) provide tools that enable rapid implementation of distributed functions using the HTTP POST method (that is, REST-ful Web services), which would be time-consuming to manually implement using TCP. The Java API for XML-based RPC (JAX-RPC), defined in the JSR172 Web Services API, is avoided due to overhead in SOAP, an XML-based messaging protocol, and the limited availability of JSR172 on commercially available mobile phones. The transfer of unnecessary XML data reduces mobile device battery life, as we demonstrate later.

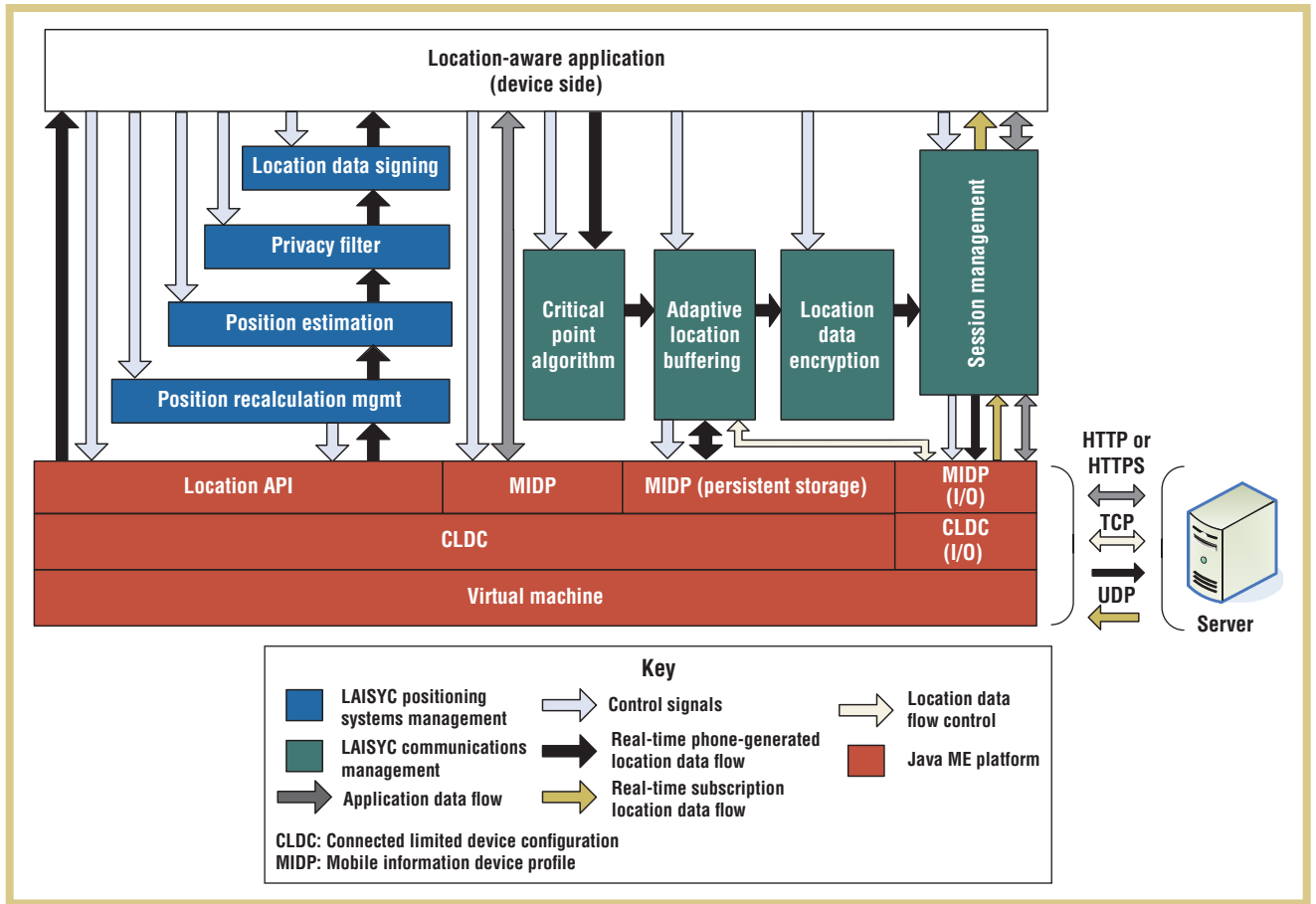


Figure 1. The Location-Aware Information Systems Client (LAISYC) framework mobile phone-based modules conserve device resources such as battery energy while meeting application functionality requirements that can vary dynamically during execution. Location data is intelligently processed on the device before being sent to a server.

LAISYC uses UDP, which is commonly used for services where timeliness is favored over reliability, to transport continuous location data updates from the mobile phone to a server. Location updates can occur as frequently as once per second for time-sensitive LBS, so a lightweight protocol is required for system efficiency and scalability, as well as to reduce the communication load on the mobile device's limited resources. Occasional messages from the server to the phone both confirm an open connection and pass specific-location dataflow-control commands to the phone.

The various framework components reside either in the device or on the server to support a complete distributed application.

Device-Side Components

LAISYC device-based components (Figure 1) are divided into two categories: positioning systems management and communications management. The application executes various types of controls, including activation and deactivation for all device-side modules, based on its real-time needs.

Location data flow from the positioning system (such as GPS) on the mobile device through the Location API (JSR179 or JSR293 in Java ME) and then into the bottom layer of positioning systems management (that is, the position recalculation management module). The location data are then propagated upward through each module of positioning systems management until they reach the application. If the

application deactivates certain modules, the location data pass through that module without the module modifying or acting on the data.

After location data are propagated through positioning systems management, the application passes location data that are candidates for wireless transmission to the critical point algorithm in communications management. The data then propagate until they reach the session management module, which activates the location data's wireless transmission over UDP. The application also directly passes application data to the session management module, which activates the transmission of this information over HTTP.

All modules are translucent to the location-aware application, meaning

that the application can still directly access the underlying APIs if necessary, to access functionality not controlled directly by the framework, but LAISYC is designed to help alleviate the application from this responsibility for most major functionality.

Position recalculation management.

This module intelligently adjusts the frequency of position recalculations to save battery energy when continuous device position calculations aren't required.¹ It realizes significant energy savings by increasing the time interval between GPS fix attempts. For example, if a mobile device is stationary for a long time, the interval between position recalculations can gradually be increased to enter a sleep mode and prevent repeated calculations of the same position information. The application can emerge from sleep mode and snap back to rapid position recalculation when it determines that the device is moving. This wake-up trigger can be based on the device's speed exceeding a certain threshold, or a certain distance between the most recent GPS fixes (that is, the distance the user walked since the previous GPS fix). We use a state machine to gradually progress from fully awake to fully asleep when LAISYC is unsure whether observed motion is true movement or a result of GPS drift, thereby preventing LAISYC from making large adjustments based on noisy outlier data. Embedded accelerometers, if available, can also wake up this module.

The position recalculation management module has a secondary navigation mode based on the distance to a goal (for example, the next turn for real-time driving directions or a remote device's location for real-time friend finders), which increases the frequency of position calculations as the mobile device nears the goal.

Other intelligence to dynamically manage the selection between multiple positioning technologies (for example, if no GPS signal exists) can also be

integrated here. For example, the PosIM middleware can facilitate dynamic location-technology switching at runtime based on rules created by the user at compile time.²

Position estimation. Device-side software can fuse data from multiple technologies (Wi-Fi, cellular signals, and so on) to estimate the device's current position when location data from primary positioning systems are unavailable. Although traditional dead reckoning relies on accelerometers to estimate the device's movement, the position estimation module uses real-time and historical data to produce an intelligent estimate of the user's real-time position. For example, a computationally inexpensive probabilistic algorithm might provide an intelligent guess at the phone's current position based on past travel behavior (such as time of day or travel patterns). This module is a candidate for different types of research into intelligent, on-board, position estimation.

For example, Skyhook's XPS hybrid positioning technology,³ used by Apple's iPhone, can be integrated into LAISYC's modular design. Skyhook's XPS system synthesizes location data from GPS, Wi-Fi, and cellular radio broadcasts that are sensed on the device and then cross-referenced with a database of cell tower and Wi-Fi locations.

Privacy filter. The Java ME security model for the location API has only blanket options for user approvals: *allow this time*, *always allow*, *allow until exit*, or *never allow*. Therefore, users must permit all location requests by the application or they're prompted each time the location-aware application accesses device location. Instead of these extremes, it's desirable to let the user define conditional approvals based on preferences and device location.

The privacy filter lets the application define conditional permissions for location requests, such as time limitations

(for example, requests permitted only during business hours for business employees) or sensitive location restrictions (for example, no requests allowed near privacy zones such as the user's home). For extremely privacy-sensitive applications, the filter is inverted to deny all location requests except those falling within defined public areas (such as major interstates). Virtual trip lines, which trigger updates to the server only at certain highway locations in the Mobile Millennium project, are one example of this type of privacy filter.⁴

Location data signing. Businesses and government agencies increasingly use GPS data to support key operations (for example, mileage and time verification for workers, or confirmation of duration and location of car use for pay-as-you-drive insurance and taxes). However, these uses of GPS data have a key weakness: GPS data are falsifiable through tampering and can't be independently verified.

Location data signing uses asymmetric cryptography to digitally sign data related to a GPS fix. These data can include the latitude, longitude, altitude, speed, GPS time stamp, system time stamp, device phone number, and identifying information for the phone and user to prove that a particular GPS fix occurred on a particular phone with a specific user logged into the application, at a specific time. Because this information is hashed and signed by the application using a private key, the data's integrity can be verified using the public key and a hash of the message. Therefore, it can easily be shown that a GPS fix is unaltered from the data originally calculated by a specific GPS-enabled mobile phone application.

Although there will likely be some impact on device battery life, lightweight cryptographic methods such as the elliptic-curve digital signature algorithm use small key lengths, which translate to better performance. Additional experimentation is necessary to

quantify the impact of frequent location data signing on a mobile device.

Critical point algorithm. Because GPS generates a large amount of location data, this data must be carefully managed to avoid wasting resources (such as battery energy) by transferring fixes to a server that might not contain useful information (for example, repeated GPS fixes when the user is standing still or fixes lying on the same vector when the user is traveling in a straight line). The user's path can be accurately represented using only small portions of GPS data generated by mobile phones.

The critical point algorithm (CPA) uses the change in direction between sequential points as well as the user's speed to filter noncritical points from a set of GPS data, so only critical points representing the user's path remain.¹ LAISYC then transfers these critical points to a server for storage and analysis. By prefiltering GPS data before it leaves the device, CPA saves battery energy, reduces data transfer costs, and saves network bandwidth. It also reduces the load on the server because it processes a fraction of the total GPS data generated by mobile devices. CPA also contains several conditional evaluations that simulate other position update methods including polling, periodic updates, and distance-based updates.

CPA is a variation of the perpendicular distance routine algorithm, which is an accurate approximation of more complex line-simplification algorithms (for example, Douglas-Peucker). CPA has $O(n)$ complexity (where n is the number of GPS fixes) and approximates the user's travel path in real time as GPS data is generated, as opposed to Douglas-Peucker's $O(n^2)$ running time, which requires the entire point dataset before beginning line simplification.⁵

CPA is also replicated on the server to filter data that hasn't been prefiltered onboard the device if real-time remote tracking is critical to the application.

Adaptive location data buffering. Because UDP is used for efficient location data transport, no end-to-end reliability (such as TCP) exists. In real-time tracking, the loss of occasional location fixes is acceptable because another location update will soon follow. However, because location data is often referenced after the fact to provide metrics (such as distance traveled) and reconstruct users' paths, the loss of many contiguous fixes introduces significant problems. Extended gaps can result from lack of support for simultaneous voice and data services or no cellular signal.

Adaptive location data buffering increases the probability that most location data points will arrive at the server. Before each location data UDP transmission, device-side APIs are checked to assess the current level of cellular signal and determine if a successful UDP transmission is probable. If not, the location data is buffered to either main memory or persistent storage (for example, MIDP RecordStore). Once it's detected that UDP transmissions will likely succeed, the buffered data is sent via UDP and deleted from the device.

Although this method increases the probability that the device will successfully issue a UDP transmission, it doesn't necessarily improve the chances that a UDP packet that leaves the phone will be received by the server. Adaptive location data buffering provides two methods to occasionally confirm an open end-to-end connection with the server. If the device's IP address is publicly addressable, the server can occasionally send alive messages via single UDP packets to the phone. If, for security or capacity reasons, the cellular provider doesn't allow publicly addressable IPs, the adaptive location data buffering module occasionally opens a TCP connection from the phone to the server to determine if there is a successful alive response from the server. Using either method, the phone will continue to transmit

UDP data to the server as long as it continues to receive alive messages from the server. If it doesn't receive an alive message (for example, if the phone is off network, the user is on a voice call, or the server is down), the phone begins buffering location data until it receives the next alive message.

Adaptive location data buffering is only intended to increase the probability that most location data will arrive successfully at the server; it doesn't guarantee location data reliability. Therefore, the ratio between the number of location data transmissions from the phone to the server via UDP and the number of alive messages received by the phone from the server should be carefully balanced based on the application's reliability requirements.

Through our experiments, we've found that UDP is the preferred primary transport protocol for location data, with adaptive location data buffering primarily preventing large losses of contiguous location data due to atypical phone or network conditions.

Location data encryption. The interception of location data during transfer over the Internet is a significant security threat to LBS. Although secure TCP connections are implemented within Java ME through the Secure Sockets Layer (SSL), application developers must implement secure UDP.

Location data encryption handles the encryption of location data in the UDP datagram's payload to enable end-to-end security between the mobile device and a server. Symmetric encryption, which uses a shared key between two parties, is generally more efficient than asymmetric encryption. Therefore, asymmetric encryption using public and private keys can protect the initial shared key exchange using SSL and HTTPS. Symmetric encryption can then be used to encrypt the subsequent location data passed over UDP during the session.

The Advanced Encryption Standard (AES) appears preferable to RC4 as an energy-efficient encryption algorithm for small data packets from laptop-based experiments.⁶ However, future experimentation will determine whether these results are transferrable to GPS-enabled mobile phones.

Session management—client-side application support. This module is a client-side counterpart to the server-side session management module, which maintains information (including current IP address) for each connected device. Together, the client and server-side session management modules help the mobile and server-side applications function together as one distributed application. The client module initiates a session for a device by calling a `createSession()` Web method and passing various authorizing information (such as username, password, and phone number). The server responds with a unique session identifier that will let it match future location data received over UDP and application-specific data received over HTTP with a single session. To signal to the server that a session is finished, the module initiates a `destroySession()` Web method.

The client session management module implicitly controls the creation and destruction of sessions surrounding the transfer of application and location data to the server and relieves the application from direct session management. For example, an application using LAISYC can simply instruct the framework to send a GPS fix to the server. This client-side module will then perform appropriate checks to ensure that a session already exists, and if not, it will automatically initiate a session with the server via HTTP and then send the data via UDP.

Server-Side Components

LAISYC's server-side components (Figure 2) are divided into two categories: communication management and location data analysis.

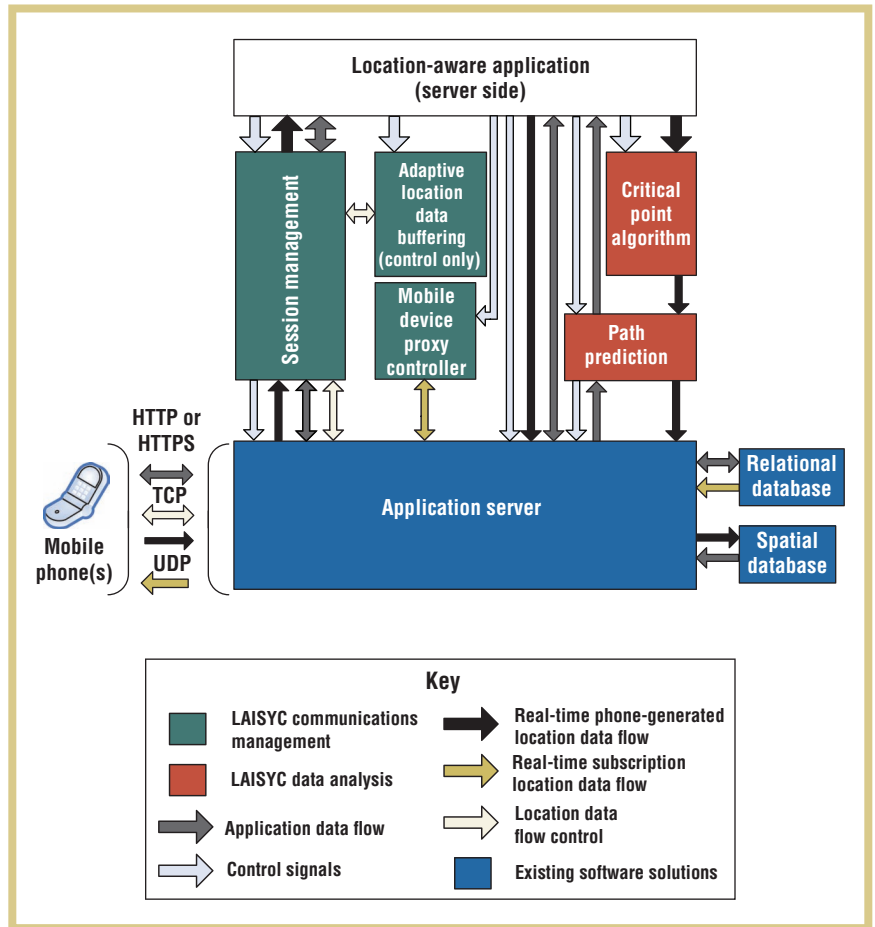


Figure 2. The LAISYC framework server-based modules support the server-side portion of the distributed application to maintain individual device sessions, access persistent relational and spatial database storage, predict the user's real-time path, and control location data flow between device and server.

As with the device-side application, the server-side application asserts various control signals to each component to activate or deactivate modules. The client device initiates all application data communication to the server using the HTTP request-response model. Information flow (that is, session requests, application-specific Web services, and location data) coming into the server-side communication management enters through the session management module and propagates directly up to the application. The application can then initiate location data analysis by passing the location data into the critical point algorithm, which propagates to the

path prediction module. For subscription services, the mobile device proxy controller module sends location data to the device via UDP. The server-side application also interfaces with both traditional SQL relational and spatial databases.

Session management—server-side application support. A session identifier, passed to the device in response to a session creation request, links multiple Web service calls over HTTP with location data sent via UDP and is included in all subsequent device-initiated communication. LAISYC uses HTTPS to encrypt Web service calls from the phone for secure services.

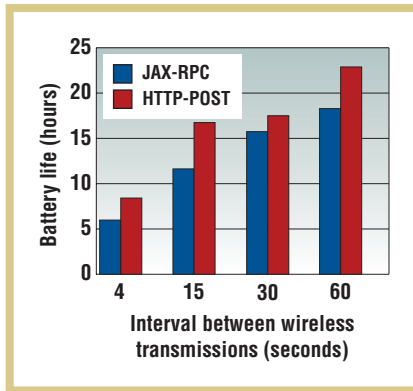


Figure 3. Comparison of Java API for XML-based Remote Procedure Calls (JAX-RPC) and HTTP POST operations on battery life. SOAP, an XML-based messaging protocol used in JAX-RPC, adds a significant amount of unnecessary XML overhead to communications, which has a significant negative impact on battery life, as well as other device resources.

A limited amount of information for each session is kept in main memory inside the application server to enable rapid response to the device based on incoming data. Although extremely time-sensitive tasks (such as real-time navigation) must still be handled by software on the mobile device, near-real-time modules that access large databases can be realized on the server side. The disk-based database contains a record of all the users, sessions, and location information, and serves as a back-up of information contained in the application server memory. This module also automatically manages session expirations to ensure efficient memory usage. For example, if no data has arrived from a particular device after a certain amount of time, information for that session will be removed from memory and marked as “expired” in the database. If a UDP packet arrives with a session ID that doesn’t exist in memory, this module checks the database to see if an expired session with that ID exists, and reactivates that session by moving it back into server memory. Removing unused sessions from memory is an important performance feature to avoid accumulating orphaned sessions as a result of

device malfunction, including power-off due to low battery levels during application execution.

Mobile device proxy controller. To support mobile device subscriptions to the real-time location of other moving entities (such as buses) during real-time cross-referencing LBS, the mobile device proxy controller proactively forwards each location update for a moving entity directly to the mobile device via UDP. This methodology avoids the latency and overhead of repeated device-initiated HTTP requests to obtain the newest location information from a server database. The mobile application subscribes and unsubscribes via HTTP Web methods.

Path prediction. Intelligent location-based services should be highly relevant and precisely targeted to each user based on the user’s real-time position and predicted path. To let users plan accordingly, location-based alerts (traffic accident notifications, advertising, and so on) should be distributed to travelers before they reach the alert area.

Because human travel behavior is highly repetitive in both space and time, path prediction uses spatial representations of a user’s historical trips along with their current position to predict the paths they might take in the immediate future.⁷

This module uses a spatial database to perform a series of intersection queries between the user’s real-time location/path and the buffers surrounding the user’s previously recorded paths. Each detected buffer represents a predicted path that the user might follow. Additional spatial queries are then executed to discover alerts relevant to the user’s predicted path and personal preferences.

Evaluation

Our research team has used LAISYC in several mobile applications, including

- Trac-It, a personal travel coach that both helps users reduce their “travel

footprint” by showing their travel history and providing travel suggestions as well as real-time traffic alerts through personalized path prediction;⁸

- Travel Assistance Device (TAD), a real-time transit navigation application that prompts transit riders to exit the bus at the proper stop and delivers real-time, estimated bus arrival times;⁹ and
- Tactical Local Area Network (TACLAN) Real-Time Location and Multimedia Messaging System, a tactical LBS system for real-time battlefield tracking and messaging between mobile devices and a centralized dispatch station.

These and other applications based on the LAISYC framework used Glassfish as the Java application server, while Microsoft SQL Server and PostGIS served as the primary relational and spatial databases, respectively. We used Netbeans to define Web services using the Java API for Web Services (JAX-WS) 2.0. We also used Netbeans to autogenerate the code stubs for the phone and server that implement HTTP POST methods that mirror input and output of the JAX-WS. We implemented adaptive location buffering using device-initiated TCP connections, because not all Sprint-Nextel phones have public IP addresses.

To evaluate framework modules for energy efficiency, we created a battery life benchmarking application that measures how long the phone battery lasts while operations are repeated at fixed intervals (for example, GPS fixes and wireless transmissions). By comparing the resulting battery life from each execution, we can determine each operation’s energy cost at different frequencies.

We demonstrated the efficiency of the HTTP POST implementation of Web services on a Motorola i580 on Sprint-Nextel’s iDEN network (see Figure 3). At 60-second transmission intervals, battery life when using HTTP POST is

more than 24 hours and only 19.3 hours using JAX-RPC. We found similar results at lower intervals, thus justifying the choice of HTTP POST over JAX-RPC as the application-layer protocol in LAISYC.

Position recalculation management provides significant energy benefits by dynamically adjusting the JSR179 LocationListener position recalculation interval based on whether the user is moving or stationary (see Figure 4). Testing with a Sanyo Pro200 on Sprint-Nextel's code division multiple access (CDMA) EV-DO Rev. A network shows that this module can extend battery life from 8 hours (4-second intervals) to more than 14 hours at 30-second intervals, and upward of 41 hours at 300-second (5-minute) intervals. The large jump in battery life between the 150-second and 300-second measurements indicates that a hardware component in the phone (such as CPU or GPS) can reach a low-power state when there are at least 300 seconds between GPS fix attempts. Because the hardware can spend more time in a low-power state between GPS fix attempts, the battery life is extended significantly at 300-second intervals. Therefore, position recalculation management can save the most energy by properly identifying when a user has stopped moving and quickly transition into large intervals between GPS fix attempts.

Figure 5 demonstrates the energy benefits of the CPA. Increasing the interval between UDP transmissions from 15 to 30 seconds extends battery life from approximately 9 hours to more than 17 hours. Increasing the interval to 60 seconds extends battery life to approximately 30 hours. Battery life is therefore directly proportional to the length of transmission interval, meaning that less frequent wireless transmissions significantly increase battery life. Energy levels shown on the y-axis of Figure 5 refer to battery-level values recorded from the Sprint Extensions API by the application

during the experiment (where 4 indicates a full battery and 0 indicates no power).

Because the benefits of using UDP (such as when timeliness and scalability are critical) are well understood, we focus on evaluating the energy footprints of UDP and TCP when transferring location data from the phone to better understand adaptive location data buffering and the potential trade-offs between reliability and power consumption.

To evaluate power consumption differences, we used an Agilent E3631A power supply to measure the current drawn by a Sanyo 7050 phone while running our test applications. Figure 6a shows the power consumption when the application is transmitting location data at 4-second intervals during separate TCP and UDP tests. The hardware is constantly active during both protocols, so there is a negligible energy difference in using UDP versus TCP at 4-second transmission intervals. However, as Figure 6b shows, with transmission intervals as low as 10 seconds, a clear energy benefit becomes evident, as the red TCP graph shows current flow when UDP (transparent blue graph) isn't consuming any energy. The approximate energy used during transmissions is 110 joules for UDP and 152 joules for TCP, yielding an average energy use of approximately 3.68 joules per transmission for UDP and 5.08 joules per transmission for TCP. Applications can now determine if occasionally querying the server to verify an end-to-end connection is worth the extra energy cost.

Because UDP is used to transport location data in a wireless environment, we also evaluated the potential loss of location data between the device and server. In extended testing performed with several Sanyo 7050s in Tampa, Florida, the server received 45,525 (97.3 percent) of the phones' 46,785 UDP transmissions. We performed these tests using ideal communication scenarios

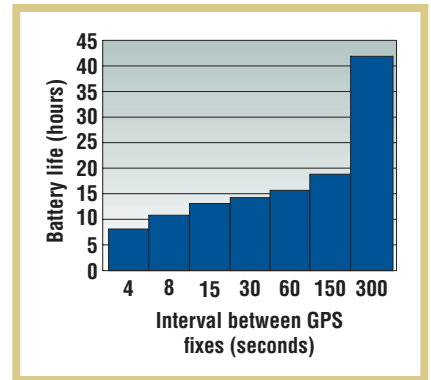


Figure 4. Impact of GPS interval on battery life. By transitioning from small sampling intervals when the user is moving to large sampling intervals when the user stops moving, the position recalculation management module can save significant battery energy.

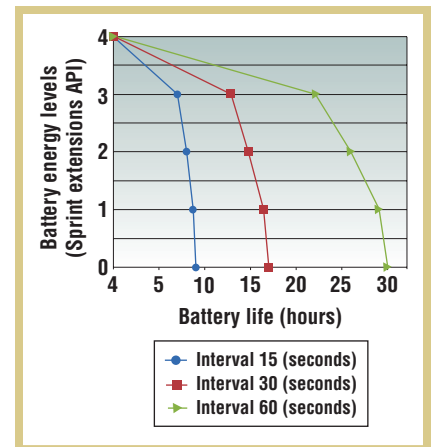


Figure 5. Impact of wireless transmission interval on battery life. By only sending critical location points needed to reconstruct a user's path, and thus reducing the frequency of wireless transmissions, the critical point algorithm can save significant battery energy and data transfer costs. For the battery levels, 4 is full, 3 is half full, 2 is low, 1 indicates a warning, and at 0, the device powers off.

(that is, in suburban areas outdoors with adequate cellular signal coverage, with a server under a very light processing load, and using test phones with no incoming or outgoing phone calls).

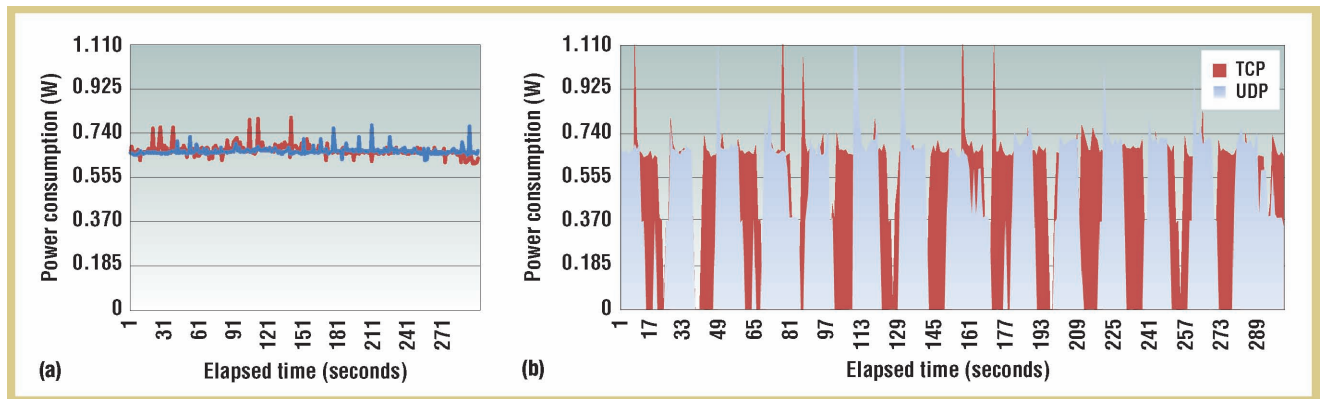


Figure 6. Comparison of power consumption when transferring location data using TCP and UDP. (a) Whereas at 4-second transmission intervals, TCP and UDP have similar power consumption, (b) at 10-second transmission intervals, TCP consumes approximately 38 percent more power than UDP.

Adaptive location data buffering can increase the probability of success for UDP transmissions during real-world use on phones that can't support simultaneous voice and data sessions, in rural or indoor areas with intermittent or sparse cellular coverage, and with servers under greater processing loads.

We're currently performing experiments with LAISYC framework components on the Google Android platform, which has become the leading cross-platform application environment for smartphones. All LAISYC design principles discussed in this article in the context of Java ME also apply to Android, with some changes in terminology for certain platform features (such as SQLite for on-device persistent storage, instead of the Java ME MIDP RecordStore). In fact, early experiments with several Android devices indicate that the energy challenges discussed in this article are even more significant on smartphones. For example, battery life when sampling GPS at 4-second intervals was approximately five hours on an HTC Hero with Android 2.1 update1, compared to eight hours on a Sanyo Pro 200 with Java ME. As a

result, the energy-efficient components in LAISYC become even more important on smartphones.

Smartphones also introduce additional capabilities and challenges when compared with typical Java ME feature phones. Additional radios for Bluetooth, Wi-Fi, and 4G communication all increase connectivity and positioning system options as well as network speeds, but require intelligence to efficiently use each technology without increasing the overall power consumption of the device and applications. Internal accelerometers, gyroscopes, barometers, and magnetic compasses all provide additional data to derive a user's position and orientation, but additional intelligence is required to transform the data into contextual information. Therefore, although smartphones provide many new and exciting technologies, these devices also open many new research areas for intelligent location-aware applications. ■

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation under grant 0754537 and the Florida and US Departments of Transportation through the National Center for Transit Research under grants BD-549-35 (TRAC-IT Phase 3) and BD-549-33 (TAD). We acknowledge the Sprint Application Developer Program for providing mobile

devices and service, and Josh Kuhn for his assistance with the energy benchmarking tests. Patents are pending on LAISYC modules by the University of South Florida, 2009.

REFERENCES

1. S. Barbeau et al., "Dynamic Management of Real-Time Location Data on GPS-Enabled Mobile Phones," *Proc. 2nd Int'l Conf. Mobile Ubiquitous Computing, Systems, Services, and Technologies (UBICOMM 08)*, IEEE CS Press, 2008, pp. 343–348.
2. P. Bellavista, A. Corradi, and C. Giannelli, "The PoSIM Middleware for Translucent and Context-Aware Integrated Management of Heterogeneous Positioning Systems," *Computer Comm.*, vol. 31, no. 6, 2008, pp. 1078–1090.
3. Skyhook Wireless, "XPS Overview," 2008; www.skyhookwireless.com/howitworks.
4. S. Amin et al., "Mobile Century-Using GPS Mobile Phones as Traffic Sensors: A Field Experiment," *Proc. 15th World Congress on Intelligent Transportation Systems*, ITS America, CD ROM, 2008.
5. W. Shi and C. Cheung, "Performance Evaluation of Line Simplification Algorithms for Vector Generalization," *The Cartographic J.*, Mar. 2006, vol. 43, no. 1, pp. 27–44.
6. P. Prasithsangaree and P. Krishnamurthy, "Analysis of Energy Consumption of RC4 and AES Algorithms in Wireless LANs," *Proc. IEEE Global Telecomm. Conf. (GLOBECOM 03)*, vol. 3, IEEE Press, 2003, pp. 1445–1449.



Sean J. Barbeau is a research associate at the Center for Urban Transportation Research, a founding faculty member of the Location-Aware Information Systems Laboratory, and a PhD candidate in computer science, all at the University of South Florida. His research interests include the design and evaluation of intelligent location-based services and online communities for user-generated spatial data. Barbeau has an MS in computer science from the University of South Florida. He is a member of the IEEE Computer Society. Contact him at barbeau@cutr.usf.edu.



Rafael A. Perez is a professor of computer science and engineering and associate dean for academics in the College of Engineering at the University of South Florida. His research interests include intelligent systems. Perez has a PhD in electrical engineering from the University of Pittsburgh. Contact him at perez@cse.usf.edu.



Miguel A. Labrador is an associate professor in the Department of Computer Science and Engineering at the University of South Florida. His research interests include energy-efficient mechanisms for wireless sensor networks, and location-based services. Labrador has a PhD in information science with a concentration in telecommunications from the University of Pittsburgh. He is a senior member of the IEEE Communications Society and a member of the ACM SIGCOMM and SIGCSE, the ASEE, and the Beta Phi Mu honor society. Contact him at labrador@cse.usf.edu.



Alfredo J. Perez is a PhD candidate in the Department of Computer Science and Engineering at the University of South Florida. His research interests include mobile sensor networks, location-based systems, evolutionary algorithms, and multi-objective optimization. Perez has an MS in computer science from the University of South Florida. He is a member of the IEEE Computational Intelligence Society and the Location-Aware Information Systems Laboratory at USF. Contact him at ajperez4@cse.usf.edu.



Philip L. Winters is the Transportation Demand Management (TDM) Program Director in the Center for Urban Transportation Research at the University of South Florida. His research interests include TDM research, planning, operations, training, and evaluation. Winters has a BS in civil engineering from Virginia Tech. Contact him at winters@cutr.usf.edu.



Nevine Labib Georggi is a senior research associate at the Center for Urban Transportation Research at the University of South Florida. Her research interests include the use of intelligent transportation systems (ITS) applications to collect travel data, enhancing the transit rider experience, alcohol-related safety research, transportation survey design and analysis, and project development and environmental studies. Georggi has an MS in civil engineering from the University of South Florida. Contact her at georggi@cutr.usf.edu.

7. N. Persad-Maharaj et al., "Real-Time Travel Path Prediction Using GPS-Enabled Mobile Phones," *Proc. 15th World Congress on Intelligent Transportation Systems*, ITS America, CD ROM, 2008.
8. S. Barbeau et al., "TRAC-IT: A Software Architecture Supporting Simultaneous Travel Behavior Data Collection and Real-Time Location-Based Services for GPS-Enabled Mobile Phones," *Proc. Nat'l Academy of Sciences' Transportation Research Board 88th Ann. Meeting*, Transportation Research Board, 2009, paper 09-3175.
9. S. Barbeau et al., "The Travel Assistant Device: Utilizing GPS-Enabled Mobile Phones to Aid Transit Riders with Special Needs," *Institution of Eng. and Technology (IET) Intelligent Transportation Systems*, vol. 4, no. 1, 2010, pp. 12-23.



Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.



Call for Articles

IEEE Software seeks practical, readable articles that will appeal to experts and nonexperts alike. The magazine aims to deliver reliable information to software developers and managers to help them stay on top of rapid technology change. Submissions must be original and no more than 5,400 words, including 200 words for each table and figure.

Author guidelines: www.computer.org/software/author.htm
 Further details: software@computer.org

www.computer.org/software

IEEE Software