An Open Platform for Location-Aware Services Development and Sensor Data Collection

Yao Hua Ho, Yao Chuan Wu, and Meng Chang Chen
Institute of Information Science
Academia Sinica, Taiwan
{yho, ycw, mcc}@iis.sinica.edu.tw

Abstract— In this paper, an open platform, called PLASH (Platform for Location Aware Services with Human Computation), for developing and hosting location-aware application is presented. This open platform provided number of fundamental services and APIs for a 3rd party developer to integrate and share both functionality and collected data. The two missions of this project: First, this platform is designed to help location-aware service (LAS) providers deploy their applications easily so that users can contribute their efforts and location-related data by using the services, which is the main difference from traditional location-aware services. Second, the collective efforts and data that may be generated by humans or sensors (or both) can be used to help researchers to solve difficult location-aware problems, such as real-time surface traffic estimation, city panoramas and social networking analysis. The PLASH platform comprises three layers, namely, Communication, Data and Service Layers. An LAS application developed on PLASH can exploit the functions provided in each layer by using an application specification language provided by PLASH. In addition, application builders can contribute their mature applications as a service accommodated in the Service Layer for other application builders to expand and create more sophisticated applications.

Keywords- Mobile Device; Mobile Application; Human Computation; Location-aware Service; Open Platform.

I. INTRODUCTION

Location-aware services (LAS) have emerged as smart mobile devices and IP wireless networks (e.g., 3G and WiMAX) have become popular and ubiquitous. An LAS application provides a geographical position-specific service to the mobile devices through the IP wireless network. Typical examples of LAS services are identifying the location of a person or an object (e.g., the whereabouts of a friend or the nearest gas station) and location-related recommendations (e.g., driving route recommendations and the best restaurants in an area). Foursquare [9], GeoLife [30], Reality Mining [7], Shanghai Grid [16], Cartel [25], and PRISM [5] are examples of existing LAS systems and research projects.

The major difference between PLASH [14] and other approaches is that it emphasizes the support of volunteers and application providers. In PLASH, users can contribute their efforts and location-related data voluntarily; and LAS builders can use the PLASH framework and provided functions to deploy their applications easily. Moreover, with the support of PLASH, an ordinary user can easily share his personal LAS with other users. The “H” in PLASH stands for “human computation,” which is a technique (e.g., CAPTCHA [29] and Spam Prevention [25]) that uses large groups of humans to solve problems that current computers cannot process. The collective efforts and data generated by humans or sensors (or both) can be used to help solve difficult location-related problems, such as real-time surface traffic estimation, city panoramas, and location-aware social networking analysis.

The PLASH platform comprises three layers, namely, Communication, Data, and Service Layers. The Communication Layer supports various wireless communication protocols (e.g., WLAN, WiMAX, and 3G) and networking contexts (e.g., Vehicle to Infrastructure, Vehicle to Vehicle, and Vehicle to Mobile device). The Data Layer is responsible for geo-location data representation, storage and access; and the Service Layer provides various services to support PLASH applications. The layered architecture allows LAS application builders to create their systems by simply using APIs to access the Service Layer or by employing an Application Specification Language (ASL) to build the systems. To demonstrate the capability of the PLASH platform, we have built a number of prototype LAS applications for Android, iPhone, and web browsers.

The contributions of this paper are as follow: (1) we introduce PLASH, an open platform for location-aware services with human computation; (2) we propose a multi-layer platform and language for application builders to develop their applications; and (3) we present an operating model of PLASH components and applications; (4) we demonstrate an effortless way to build a location-aware application on PLASH’s open platform; and (5) we present some of initial research topics/results in PLASH project.

The remainder of the paper is organized as follows. Section II provides a review of related works. In Section III and Section IV, we introduce the multiple-layer platform and PLASH framework, respectively. In Section V, we discuss the results of experiments conducted to evaluate the performance of PLASH. In Section VI, we present a number of location-aware applications that we have developed. Section VII contains some on-going research projects build on top of PLASH platform. Finally, we drew our conclusion and future works in Section VIII.
In this section, we review some related works in the literature.

A. Location-Aware Services and Research

A location-aware/location-based service exploits information about the geographical position of people and objects. For example, Foursquare [9] is a type of social networking application that runs on mobile devices and lets friends track each others’ whereabouts on the Foursquare map. It combines game-like objectives, such as collecting points, prize, badges, and eventually coupons, as users go about their everyday business. Foursquare surpassed 500,000 registered users in March, 2010 and about 275,000 users are signed-in at any one time.

Microsoft Research’s GeoLife [30] is a location-based social network service on Microsoft Virtual Earth. It enables users to share their life experiences using their location history. Specifically, GeoLife mines the correlation between users and their location based on user-generated GPS trajectories. It provides three applications, namely, generic travel recommendations, personalized friend recommendations, and location recommendations, which leverage the GPS information and enable users to share their daily experiences.

MIT Media Lab’s Reality Mining [7] is a system for sensing complex social systems with data collected from 100 mobile phones over the course of nine months. It measures information access and usage in different contexts, recognizes social patterns in daily user activities, infers relationships, identifies social significant locations, and models organizational rhythms. Personal reality mining infers human relationships and behavior by applying data-mining algorithms to information collected by cell phone sensors that can measure location physical activity.

Shanghai Grid [16] has established an information grid to support the city’s traffic system, and continues to develop Grid-based applications for traffic congestion control and guidance. One application, called the Intelligent Traffic Information Service, is designed to achieve effective traffic management. It processes real-time traffic data and provides various traffic services, such as vehicle tracking and traffic status estimation.

MIT’s Cartel [25] is a distributed mobile sensor network. Applications built on top of the system can collect, process, deliver, analyze, and visualize data from sensors located on mobile units, such as mobile devices. One of the applications monitors traffic conditions based on GPS data contributed by users. Using the GPS data, the authors developed a predictive delay model and traffic-aware route planning algorithm.

Microsoft’s PRISM (Platform for Remote Sensing using Smart phones) [5] is a platform to support opportunistic sensing applications on mobile smart phone. PRISM responsible to push different sensing applications out to an appropriate set of phones based on a specified set of predicates. Three applications have been built on PRISM. Each very in the set of sensor they use and in their mode of operation. PRISM focused on a number of challenges regarding to security, scalability, and resource exhaustion.

B. Location-Aware Services Platform

The number of mobile device users has increased rapidly in recent years. Among smart phone users (e.g., iPhone or Android users), location-aware applications are one of most popular downloaded applications. Location-aware applications, such as Foursquare [9] and Gowalla [11], allow users to connect between the “real-world” and a virtual world. Thus, many platforms have been developed to support location-aware applications. For example, the Symbian platform provides a set of APIs for Symbian devices to store and access location information. In contrast to Symbian, PLASH stores location data and provides a development environment for application builders to deploy their applications.

The NEXUS platform ([15] and [27]) focuses on providing efficient data management services by distributing data and handling queries from location-aware applications. In PLASH, the data management service is just one of many fundamental services (i.e., authorization control services). PLASH also allows other application developers to contribute by sharing services (i.e., shared services) and applications that they have developed.

IHP Technology developed a location-aware service platform based on Java RMI/Jini [6] to provide the basic functions required by a location application. In the IHP platform, an aura concept is used to define a region of interest for a mobile terminal or a service available at a specific location. A lookup service is also provided to determine where a mobile device can offer its services to other users and vice versa. Unlike IHP’s approach, most of the services provided by PLASH are located on servers. This strategy reduces the resource requirements of mobile devices (e.g., computation power, bandwidth, storage, and battery power). Moreover, in the PLASH platform, a shared service is not the result of a region of interest query (i.e., a location query) but an actual functionality of a service (i.e., an authorization control service) that can be used to develop phone applications.

C. Platform for Hosting Application

PLASH also aims to provide a platform for application builders to develop and host their applications with PLASH APIs and services. In service platform, there are several providers/solutions based on cloud computing technology: Google App Engine [11] is a platform for developing and hosting web applications in Google managed data centers. Based on cloud computing technology, it virtualizes applications across multiple servers and data centers. App Engine provides infrastructures to make it easy to write scalable applications by removing many system administration and development challenges of building applications.

Amazon Web Services [1] is a collection of web services in a cloud computing platform. Based also on the cloud computing technology, it provides online services for other web sites or client-side applications. Most of these services are not exposed directly to end users, but instead offer functionalities that other application builders can use.

Microsoft’s Azure Service Platform [23] is an application platform in the cloud that allows applications to be hosted and run at Microsoft datacenters. It provides a set of services that

*This project is partly supported by NSC Taiwan under grant 98-2219-E-001-001.
allows development, management and hosting of applications off-premise. Azure Service Platform provides an API that allows a builder to interact with the service provided by Window Azure.

III. THE LAYERED ARCHITECTURE OF PLASH

In this section, we introduce the PLASH system architecture. The system is an application platform built in JAVA based on Mule (an enterprise service bus). Figure 1 shows the three layers of the architecture, i.e., the Communication Layer, the Data Layer, and the Service Layer. APIs are defined between adjacent layers.

![Figure 1. PLASH’s Multi-layer Platform](image)

A. Communication Layer

The Communication Layer is designed to support wireless communications (e.g., 3G, WiMax, WLAN) and networking contexts (e.g., V2I - Vehicle to Infrastructure, V2V - Vehicle to Vehicle). The layer supports layered coding to overcome the limitations of some mobile devices (e.g., small screens, low bandwidth, and limited power and storage capacity). The PLASH platform allows application builders to customize the communication requirements (in terms of bandwidth or communication protocols) based on the needs of different applications through the APIs in the Communication Layer. In this layer, RESTful (REpresentational State Transfer) serves as a common protocol for different servers (e.g., application servers, database servers). The various services provided by application servers can be accessed via an HTTP request using methods like GET and POST to describe the necessary create, read, update, and delete actions. The PLASH platform can support different devices (e.g., iPhone and Android devices) with REST/XML APIs.

B. Data Layer

The Data Layer is responsible for data representation, storage, and handing access requests. The large volume of geo-location related data generated and uploaded by mobile devices is stored in the geo-location database. In addition to the PLASH geo-query processor, we currently use PostgreSQL with PostGIS [23] as the database to support geographic object and query processing. The location information uploaded by a user contains spatial and related information. PLASH supports different types of location-dependent spatial queries (LDSQs), including select queries, range queries, moving object queries, continuous queries, and k-NN queries. Thus, an application can query the geo-location database about the locations of objects or people, such as gas stations within 50 meters of the user’s current location or the five friends that are closest to his/her home. PLASH also maintains and stores user information, such as username, password, and friendship information in the Social Core database. Between the Data Layer and the Application Layer, a set of standard APIs is defined for querying location information and user information. To protect the user’s privacy, the location data collected by PLASH is filtered before it is processed (e.g., the username is removed from the location information).

C. Service Layer

The main function of the Service Layer is to provide fundamental services and applications contributed (i.e., shared services) by other users through APIs. In addition, PLASH application builders can use PLASH Application Specification Language (discussed in Section IV.D) to access the services. The layer also provides fundamental services, such as authorization control services (e.g., login, logout, register, and friendship), database access services (e.g., select and insert), and LDSQs services. The PLASH platform is a friendly and efficient environment designed to help users and application builders utilize LASs. Therefore, PLASH also accommodates mature LAS applications as services. Through this layer, application builders can develop and deploy applications with low complexity.

IV. STRUCTURE OF THE PLASH FRAMEWORK

In this section, we discuss the components of the PLASH operating model, and application development of PLASH.

A. Components of the PLASH Framework

The operating model is divided into two parts: PLASH Application and PLASH Platform, as shown in Figure 2. Various communication protocols are used to communicate between different layers. HTTP/RESTful (REpresentational State Transfer) is used between the Presentation Layer, Logic Layer, and Access Managers in the Service Layer. Java Message Service (JMS) is used within different components in Service Layer between the Access Managers and the Fundamental Services; and Java Database Connectivity (JDBC) is used to communicate between the Service Layer and different databases in the Data Layer. All the detailed information about the layers will be described in the latter article.

![Figure 2. The PLASH Operating Model](image)
1) **PLASH Application**

PLASH Application is further divided into the Presentation Layer and the Logic Layer.

**Presentation Layer:** Client software is responsible for user-side presentations and interactions in this layer. It can be implemented on different types of mobile devices, such as a mobile phone or a vehicle’s on-board unit, as shown in Figure 3. Users can subscribe to many PLASH applications by downloading the corresponding client software to their mobile devices. The subscribed application accesses the PLASH Platform through the Access Manager, and gets the responses directly from the Access Manager or its Application Server by its application logic.

![Figure 3. The Presentation Layer.](image)

**Logic Layer:** Application logic and databases are implemented in this layer. An input of logic layer is a response of a client application request which returned by the Access Manager in the Service Layer, as shown in Figure 4. Inputs are processed by the corresponding application server and then returned back to the requested client application. Depend on the application server, the communication protocol between the Access Manager and the Logic Layer can be a variety of SOAP, RESTful, etc. Both application logic and database can be hosted on an application builder’s server or on the PLASH platform/server. Therefore, application builders do not require maintaining their own application servers.

![Figure 4. The Logic Layer.](image)

2) **PLASH Platform**

PLASH Platform is divided into the Service Layer and the Data Layer (see Figure 2).

**Service Layer:** This layer consists of the Access Manager and the Fundamental Services. The Access Manager is responsible for the object validation and flow control. There are two types of Access Manger –Client Access Manger (CAM) and Server Access Manger (SAM). Both share the Request Router and the Waiting Queue which implemented using Java Message Service (JMS) standard, as shown in Figure 5.

![Figure 5. The Access Manager in the Service Layer.](image)

CAM receives requests from clients using HTTP Input Service which listens to a certain port through the HTTP protocol. In CAM’s HTTP Input Service, the request is processed as follows:

1. The HTTP Input Service converts the query string of the request into a map object which is a key-value pair data structure.
2. The input service checks whether the API key is valid and give it a time stamp.
3. The map object is transformed into correct message type (i.e., *MapMessage* type) depend on its service request.
4. The map object is sent to the Waiting Queue.

The Waiting Queue is followed the queue service of JMS standard which is provided by Apache ActiveMQ [2] – a message broker for remote communication between systems using the JMS. The queue holds objects (i.e., requests) from the HTTP Input Service and objects (i.e., results) returned by fundamental services. Those objects in the queue are waiting to be sent to the Request Router. Before sending to the router, each object is transformed from *MapMessage* into Map type.

The Request Router determines where the destination of an object based on its Application Specification Language (ASL) ID and its current step. We will discuss ASL in more detail later in Section IV.D. The destination of an object can be another fundamental service for an addition service required by the object or Response Queue in SAM. The Response Queue is used to store objects (i.e., results of requests) that are waiting to be return to the Application Server or clients depend on objects’ setting.

Another component in the Server Layer is the Fundamental Services which provide services to process the requests. There are three types of Fundamental Services: Upload, Geo-Query, and Social Services. As show in Figure 6, each Fundamental Service is consisted by a Service Queue (with JMS standards) and a Service Core.

The Service Queue holds the objects (i.e., requests) which are sent from the Request Router in the CAM and waiting to be processed by its Service Core. The Service Core implements the service logic. Before sending to Service Core, objects need to transform from JMS to the required form (e.g., javax.jms.TextMessage into java.lang.String) according to the different service logic. For Upload Service and Geo-Query
Service, objects are pre-processed by the Data Pre-Processor to optimize the performance. To improve database’s performance, Data Pre-Processor adapts different techniques, such as compressing and filtering incoming data before sending it to the Geo-Location Database. For Geo-Query Service, data retrieved from the database is check for geo-related criteria at Geo-Processor. Before an object (i.e., a result of a request after a fundamental service) is sent back to the Waiting Queue in the Access Manager, it needs to convert to JMS Message type. We will discuss Upload, Geo-Query, and Social Services in more detail later in next section (Section IV.B).

**Data Layer:** All data of the PLASH platform are stored in this layer. As show in Figure 7, the Data Layer consists of a Connection Pool, Geo Core, and Social Cores. Fundamental Services gets the connection in the Connection Pool in order to access the Databases. The Geo-Location Database of the Geo Core stores Geo-Location information, such as longitude, latitude, timestamp, etc. In Social Core, User Database and Friend Database store the detailed user information and friend relation, respectively.

![Diagram of Data Layer](image)

**B. Services on the PLASH Platform**

In the PLASH platform, three types of services, namely, Social Service, Upload Service, and Geo-Query Service, enable PLASH Applications to access and share the services and databases. The Social Service allows the user to login/logout and maintains personal information (i.e., friendship/membership, username/password information). If the login is successful, the Upload Service (i.e., user’s location/sensor information) is initiated. During this procedure, application dependent information (i.e., location associated information, Application ID/API Key, and the username) is sent to the PLASH Platform. The Geo-Query Service may be activated to check if the given geo-query can be satisfied. Depending on the Application ID and its logic, the PLASH Platform triggers associated services and sends the returned information to the Application Server or the Client. The Application Server, if received the returned information, then processes the returned information based on its application logic and delivers the aggregated data to the mobile device. Finally, the client software on the mobile device displays the result.

We use an example, called the “e-Coupon” application, to help explain the interactions between the components. The application server gives an e-Coupon to registered users within 100 meters of a shop. First, a user logs into the e-Coupon application through the Client Access Manger (CAM) to access the Service Layer (Step 1 in Figure 2). The CAM is responsible for load balancing and enforcing authentication, authorization, and security policies to manage different levels of access to resources. After the CAM grants access, the login information is passed through the Social Service to the User Database in the Social Core to check the user’s authorization (Step 2). Next, the Social Service in the Service Layer queries the databases in the Social Core (as Step 3). If the login is successful, a successful login message is sent back to the Server Access Manager (SAM) and directed to the e-Coupon Client (Steps 4 and 5). The SAM is responsible for routing the returned information from the PLASH platform to the PLASH Application, including the Application Server and the Client. Finally, the e-Coupon user receives a successful login message from the SAM (Step 6).

After the user logs in successfully, the mobile device can start sending its GPS data to the PLASH platform via the Upload procedure. The location information is passed through the CAM and Upload Service to do the data pre-processing (Steps 1 and 2 in Figure 2). Then the information is sent to the Geo-Location Database (Step 3) and the response is passed through the Upload Service, SAM to the Client (Steps 4 to 6). Besides, the “e-Coupon” service continuously checks if the user is within 100 meters of the shops have coupons. The location information along with the user information and application ID (i.e., the e-Coupon Application) is passed to the Geo-Query Service (Steps 1 and 2). If the condition is satisfied, the result is forwarded to e-Coupon Server via the Geo-Query Service and the SAM (Steps 4, 5, and 7). The Geo-Query Service is the outgoing interface of Geo-Core, which processes and sends the result to the SAM. If the response is positive, the e-Coupon Application Server then sends the corresponding e-Coupon to the user (Step 8).

**C. Developing Applications on the PLASH Platform**

A PLASH application comprises the client software on the mobile device and the server software of the application server. To build a PLASH application, builders are required to have a registered account and service agreement with PLASH. After signing up, application builders can design their own LAS
applications by using PLASH’s services, client and server software repositories, and databases.

Client software can be implemented in various programming languages such as Java or Objective-C. Besides providing GUI for user interactions, Client software has a function that collects geo-location and/or sensor data from devices as well as the user’s input data, which it then uploads to the PLASH platform. The server software, which currently supports the application logic implemented in JAVA, maintains user states and interacts with PLASH’s services through the Server Access Manager via APIs. As a result, application builders can combine the services with their own application logics. For example, an application server’s software can combine different services including geo-queries, user authentication, and social network management to develop its application logic. Builders can use PLASH’s databases (e.g., the geo-data base, friend database, and user database) and also maintain their own databases for specific uses.

In addition to construct an application server, builders can implement own services as “Shared Services” based on the service logic and Fundamental Services. After the builder registers the service on the PLASH platform, the platform takes care of the other things, including constructing corresponding service queue or the object transformations between the logic and the queues. While the service is assembled onto the PLASH platform correctly, users and other builders are then able to use the Shared Services or mashup with others like the Fundamental Services.

Furthermore, builders can easily create an Application Specification Language file, without writing any program codes, to describe the database schema, service usage and their interactions with other services. After uploading the ASL file, the platform constructs the databases, approves the usage of the services, and creates an application server.

D. Application Specification Language (ASL)

In PLASH, application builders can construct their application servers by using the ASL description file and construction file. Figure 8 shows an example of the ASL description file for the e-Coupon service. In the following, we describe how the file helps users determine the e-coupons available near their current positions. Then, we explain how other services can easily mash up with the e-Coupon service by using an ASL construction file.

1) ASL Description File

Before constructing an ASL description file, a builder needs to obtain an API Key for the e-Coupon service from PLASH. The key is used to control access to PLASH’s interfaces. An ASL description file comprises three parts: the service usage declaration, the application description, and the database definition. (Note that the numbers on the left-hand side of Figure 8 and Figure 9 are the real line numbers of the ASL files. Some lines may be omitted to save space.)

Service Usage Declaration: This defines the services used by the current service. In lines 3 to 20 of Figure 8, services like geo-query and personal information query are used in the e-Coupon service.

Database Definition: This defines the schema of the private database used by the service. In lines 21 to 58, the e-Coupon service defines its own private database (i.e., line 22: “coupon”) to store information, such as the location of stores/coupons, the description of the coupon, and a link to the coupon image.

Application Description: This provides general information about the service as well as the definition of the shared service so that other applications can “mash it” up with their own services. In Figure 8, the application description of the e-Coupon service is given in line 60, and the shared service definition is given in lines 62 to 88.

In e-Coupon’s application description, line 61 sets the attribute “InternalDatabase” to indicate that the database is private. It defines shared services, including its description (line 63), the handle (i.e., “/eCoupon/FindCoupons”), and the input parameters and return results (lines 65 to 87). When other builders apply to e-Coupon’s shared service (i.e., FindCoupons), they only need to provide location information such as the longitude and latitude, as an input. According to the location information, e-Coupon’s FindCoupon shared service returns a set of coupons.

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;?xml version=&quot;1.0&quot; encoding=&quot;utf-8&quot;?&gt;</td>
</tr>
<tr>
<td>2</td>
<td>&lt;PlashServer apiKey=&quot;f3e8670a320590d........&quot;/&gt;</td>
</tr>
<tr>
<td>3</td>
<td>&lt;Services&gt;</td>
</tr>
<tr>
<td>4</td>
<td>&lt;GeoQuery&gt;</td>
</tr>
<tr>
<td>5</td>
<td>&lt;Type&gt;RangeQuery&lt;/Type&gt;</td>
</tr>
<tr>
<td>6</td>
<td>&lt;Range&gt;100M&lt;/Range&gt;</td>
</tr>
<tr>
<td>7</td>
<td>&lt;/GeoQuery&gt;</td>
</tr>
<tr>
<td>8</td>
<td>&lt;/Services&gt;</td>
</tr>
<tr>
<td>9</td>
<td>&lt;Database&gt;</td>
</tr>
<tr>
<td>10</td>
<td>&lt;Table name=&quot;coupon&quot;&gt;</td>
</tr>
<tr>
<td>11</td>
<td>&lt;Column&gt;</td>
</tr>
<tr>
<td>12</td>
<td>&lt;Name&gt;id&lt;/Name&gt;</td>
</tr>
<tr>
<td>13</td>
<td>&lt;Name&gt;name&lt;/Name&gt;</td>
</tr>
<tr>
<td>14</td>
<td>&lt;/Column&gt;</td>
</tr>
<tr>
<td>15</td>
<td>&lt;/Table&gt;</td>
</tr>
<tr>
<td>16</td>
<td>&lt;/Database&gt;</td>
</tr>
<tr>
<td>17</td>
<td>&lt;Application&gt;</td>
</tr>
<tr>
<td>18</td>
<td>&lt;Desc&gt;This service discovers nearby coupons.</td>
</tr>
<tr>
<td>19</td>
<td>&lt;/Desc&gt;</td>
</tr>
<tr>
<td>20</td>
<td>&lt;InternalDatabase&gt;</td>
</tr>
<tr>
<td>21</td>
<td>&lt;Private&gt;</td>
</tr>
<tr>
<td>22</td>
<td>&lt;/Private&gt;</td>
</tr>
<tr>
<td>23</td>
<td>&lt;/InternalDatabase&gt;</td>
</tr>
<tr>
<td>24</td>
<td>&lt;SharedServices&gt;</td>
</tr>
<tr>
<td>25</td>
<td>&lt;Desc&gt;This service discovers nearby coupons.</td>
</tr>
<tr>
<td>26</td>
<td>&lt;/Desc&gt;</td>
</tr>
<tr>
<td>27</td>
<td>&lt;Link&gt;/eCoupon/FindCoupons&lt;/Link&gt;</td>
</tr>
<tr>
<td>28</td>
<td>&lt;InputParameters type=&quot;SET&quot;&gt;</td>
</tr>
<tr>
<td>29</td>
<td>&lt;Desc&gt;PLASH Login&lt;/Desc&gt;</td>
</tr>
<tr>
<td>30</td>
<td>&lt;Name&gt;plaslogin&lt;/Name&gt;</td>
</tr>
<tr>
<td>31</td>
<td>&lt;Desc&gt;Location&lt;/Desc&gt;</td>
</tr>
<tr>
<td>32</td>
<td>&lt;Name&gt;latitude&lt;/Name&gt;</td>
</tr>
<tr>
<td>33</td>
<td>&lt;Name&gt;longitude&lt;/Name&gt;</td>
</tr>
<tr>
<td>34</td>
<td>&lt;InputParameters&gt;</td>
</tr>
<tr>
<td>35</td>
<td>&lt;/InputParameters&gt;</td>
</tr>
<tr>
<td>36</td>
<td>&lt;ReturnResult type=&quot;SET&quot;&gt;</td>
</tr>
<tr>
<td>37</td>
<td>&lt;Desc&gt;Information of the coupon.&lt;/Desc&gt;</td>
</tr>
<tr>
<td>38</td>
<td>&lt;Name&gt;category&lt;/Name&gt;</td>
</tr>
<tr>
<td>39</td>
<td>&lt;Name&gt;brand&lt;/Name&gt;</td>
</tr>
<tr>
<td>40</td>
<td>&lt;/ReturnResult&gt;</td>
</tr>
<tr>
<td>41</td>
<td>&lt;/SharedServices&gt;</td>
</tr>
</tbody>
</table>

Figure 8. Examples of PLASH ASL Description File.
2) ASL Construction File

With the ASL construction file, PLASH application builders can mash up services provided by other builders. To explain the process, we use an example called Friend&Coupon, where a builder wants to create a service to track his friends and discover coupons near those friends at the same time. We assume that the builder can also access a service called Map’n Track Friends, which allows users to share their current locations and trajectories with friends. The builder can easily mash up with the Map’n TrackFriends and the e-Coupon service to create the new service by using the ASL construction file (as Figure 9).

```
  <xml version="1.0" encoding="utf-8">
    <MashUpPlashServer apiKey="v232x093m1koa19"/>
    <Services>
      <Service name="ShowFriends">
        <InputParameters>
          <Desc>Map'n TrackFriend's ShowFriends</Desc>
          <InputParameters>
            <Link>/MapnTrackFriend/ShowFriends</Link>
            <ReturnResult type="SET">
              <Desc>Friend's Name</Desc>
              <Name>fname</Name>
              <Desc>Friend's Location</Desc>
              <Name>latitude</Name>
              <Name>longitude</Name>
            </ReturnResult>
          </InputParameters>
        </Input>
      </Service>
      <Service name="FindCoupon">
        <InputParameters type="FOREACH">Name</Name>
        <Desc>FindCoupon's FindCoupons</Desc>
        <InputParameters>
          <Link>/eCoupon/FindCoupons</Link>
          <ReturnResult type="SET">
            <Desc>Friend's Name</Desc>
            <Name>fname</Name>
            <Desc>Location of the coupon</Desc>
            <Name>latitude</Name>
            <Name>longitude</Name>
          </ReturnResult>
        </InputParameters>
      </Service>
    </Services>
  </xml>
```

Figure 9. Examples of PLASH ASL Construction File.

Friend&Coupon’s ASL construction file defines the input parameters, mashed up services, and output parameters. Each mashed up service (Map’nTrackFriends’ ShowFriends, and e-Coupon’s FindCoupon) accepts a set of input parameters and returns a set of results, as defined in their respective ASL description files. The ShowFriends service accepts username, password, and location information as input and outputs a list of friends’ names and their locations. Meanwhile, FindCoupon accepts location information and returns a list of coupon locations. By using the ASL construction file, PLASH application builders can develop new applications by utilizing shared services to save time and effort, and thereby reduce the entry barriers to providing personalized LAS services. In the future, builders will be able create new LAS applications through the PLASH GUI Platform.

E. GUI for Application Builder

PLASH application builders can upload the ASL files (i.e., Description File and Construction File) through the provided interface (shown in Figure 10). Once login, builders just have to choose the file, input the email address and the description, and click the "Send" button. After the validation, the application servers will be constructed.

![Figure 10. PLASH GUI for upload ASL files.](image)

Application builders are able to create new LAS applications without writing ASL files through the PLASH GUI — a simple drag-and-drop interface (shown in Figure 11). The Fundamental Services listed on the left side allow application builders to construct their own service by drag the desired services and drop into “Your Service” column (shown in Figure 12). Builders only need to name the created service, weather to share the service as Shared Service, and descriptions on the services. After that, a new service is mashed-up and published with simple drag-and-drop.

![Figure 11. PLASH GUI for Drag-and-Drop Interface.](image)

![Figure 12. PLASH GUI for Drag-and-Drop Interface.](image)
V. Evaluation

In this section, we describe the evaluation experiment. We examined the performance of PLASH on a machine equipped with dual Intel XEON 2.5 GHz Quad-Core Processors and 3G of RAM.

A. Performance of Geo-Location Database

We experiment with different types of queries on different data sets on PLASH server and database. First, we study a k-NN query where \( k = 10 \) and the number of Points of Interest (POI) is 1,000,000 pairs of latitude and longitude. With no indexing on the location point, the response time is 4,594 (milliseconds). However, with the Generalized Search Tree (GiST) indexing structure, we can reduce the response time to 486 (milliseconds); and by using KNNGiST index [18], the response time can be further reduced to 3.15 (milliseconds). The time cost of building the KNNGiST index is around 80,000 (milliseconds) and 69 (MB) of memory. However, KNNGiST is 8 to 11\% slower on insertion queries and 5\% slower on non- k-NN queries. Thus, we only use an indexing structure on data that are updated less frequently, such as POIs (e.g., gas-stations and restaurants). Since a user location is usually retrieved by using the trip id, we do not use any indexing structure on users’ locations. Without indexing, the maximum number of location updates that the PLASH platform server can support is around 120,000 (inserts/minute) or 2 (inserts/milliseconds). The query performance is summarized in Table 1. When measuring the response time of the Map’n Track Friends application (see Section VI.A), we observe that there is a bottleneck in the communication layer when we compare the round trip time of the three major 3G phone carriers. The average round trip time per-data packet is 290 (milliseconds) and the average total response time is less than 300 (milliseconds).

<table>
<thead>
<tr>
<th>Query Type</th>
<th>Indexing Type</th>
<th>None</th>
<th>GiST</th>
<th>KNNGiST</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-NN query (ms)</td>
<td></td>
<td>4,594</td>
<td>486</td>
<td>3.15</td>
</tr>
<tr>
<td>Build Index (ms)</td>
<td></td>
<td>0</td>
<td>68,000</td>
<td>80,000</td>
</tr>
<tr>
<td>Insertion query (ms)</td>
<td></td>
<td>0.5</td>
<td>0.6</td>
<td>0.6</td>
</tr>
<tr>
<td>None k-NN query (ms)</td>
<td></td>
<td>0.8</td>
<td>0.824</td>
<td>0.845</td>
</tr>
<tr>
<td>Insertion query (insert/min)</td>
<td></td>
<td>120,000</td>
<td>108,000</td>
<td>102,000</td>
</tr>
</tbody>
</table>

B. Performance of Response Time on PLASH Platform

In this study, we evaluate the response time of an application scenario for a user to find his nearest friend based on his location information. This scenario requires different Fundamental Services and multiple accesses of different databases.

Fundamental Services required in the scenario are as follows:

- Login – Social Service
- Access Friend List – Social Services
- Access Friends’ Locations – Geo-Services
- Query Nearest Friend – Geo-Services

Database Accesses in the scenario are as follows:

- User Database in Data Layer’s Social Core for user login
- Friend Database in Data Layer’s Social Core for user’s Friend List
- Geo-Location Database in Data Layer’s Geo-Core for finding the location of each friend.

We compare the following three platforms under 3G and GPRS mobile network connections:

- **PLASH (Open) Platform** – It mashes-up four Fundamental Services into one service (i.e., Shared Service) by utilizing the ASL file. Clients only need to request the Shared Service once in order to find users’ nearest friend.

- **Traditional (Service Oriented Architecture) Platform** – Like traditional API provider, clients request one service, wait for the response, and request another one, if there is. In order to find the nearest friend, Clients have to send four requests.

- **Closed Platform** – Implementation of the service (i.e., four Fundamental Services) for finding nearest friend are combined under programming level. Although the Closed Platform eliminates the overhead of the Access Manager of the PLASH Platform, application builders are not able to use existing services/function or share services they have created.

Figure 13 represents the response time over different number of requests under 3G.
connections for every request. On the other hand, the PLASH Platform is robust to the number of requests.

![Response Time Graph](image1)

Figure 14. Response time over different number of requests under GPRS.

Figure 14 shows the response time over different number of requests under the GPRS (i.e., 2G) network connection. GPRS has slower connection speed and lesser stable than 3G which are reflected on the response time performance. Again, the PLASH Platform has the similar response time with the Closed Platform due to the low overhead and less database connections required. The Traditional Platform still performs the worst of all three with the same reason as discussed. Notice that the slow connection speed of GPRS overshadows the issues (i.e., number of requests to different services, wait for the results, and establish number connections to different databases) in Traditional Platform. As the results, the PLASH Platform is robust to number of request while maintain an open platform for application builders to integrate and share both services and data.

C. Open Platform v.s. Copy and Paste Programming Practices

PLASH’s open platform provides a set of standard application programming interface (API) and application specific language (ASL) for number of fundamental services. This allows a 3rd party application builder to integrate with our platform’s functionalities to their applications. In addition, application builders are able to contribute added services that are not yet existed in PLASH platform. Therefore, the open platform allows the application builders to share some of useful services they have created to others.

Without PLASH’s open platform, LAS application builders often copy and paste code from various locations: documentation, someone else’s code or their own code. However, the copy and paste (C&P) code has the potential to create unnecessary duplicates in a code base. According to [10] and [16], authors recommended that programmers should avoid creating code duplicates, which often created by C&P, because such duplicates can be difficult to maintain. For example a bug can be propagated to scattered places when the code is copied. A number of clone detection tools ([2], [7], [17], [18], and [20]) have been developed to help programmers automatically located code duplicated and refactor existing duplications.

In this study, we investigate two of applications developed under PLASH project (see Section VI) - Map’n Track (MTF) and Laser Flag Game (LFG) and their service codes. MFT and LFG used number of fundamental services such as Social Service (e.g., user registration, user log in and out, user notification, or friend relationship), Upload Service (e.g., user location update, game status, or players’ HP), and Geo-Query Service (e.g., k-nearest friends or players within 50 meter). We observed that 76% of code is shared between MTF and LFG (see Figure 15).

![Line of Code Graph](image2)

Figure 15. Line of Code shared by MTF and LFG.

MTF is the first location-aware application developed with our open platform. We made the most of MTF’s functionalities become fundamental services. The other applications can be built based on those fundamental services. Depend on the application’s variation, application builder can mash up fundamental services with customize services. When a customize service becomes more desirable by others, we will do a code checking for any performance and security issue before it is release as either a shared service or a fundamental service. The main difference between shared serve and fundamental service is that a shared service is developed and maintained by 3rd party application builder who wants to share the service. 3rd party application builders can charge a fee for develop such service for other application builders who wants use it. Fundamental services are developed and maintained by PLASH team members.

![Percentage of Services Graph](image3)

Figure 16. Percentage of Services in Different Applications.

In Figure 16, 92% of MTF’s services are based on fundamental services. In Friend Compass (see Section VI.B), 11% of customize services are functions required by
augmented reality. Those customize services were transfer later to shared services and used by LFG. In Music Spirit (see Section VI.G), the most of the functionalities are existed in the PLASH’s open platform. Thus, application developers only need to implement the motion detection service. In fact, Music Spirit was developed and completed by two summer interns at Academia Sinica within one month with limited knowledge of java and no prior experiences in implementing smart phone application. In Academia Sinica Discovery (ASL – see VI.L), 88% of services are made up of fundamental and shared services. Using PLASH’s ASL, application builders only need to learn about existing services and be able to create interesting applications by mash up different services.

VI. LOCATION-AWARE APPLICATIONS

In the following sections, we introduce some LAS applications that we have developed on top of the PLASH platform within one year. Those applications are incorporate into the Service Layer as shared services so that other application builders can use them directly or include them in their applications.

A. Map’n Track Friends

One of the applications is called “Map’n Track Friends.” It is a location-based service application that allows users to share their location trajectories with authorized persons. Figure 17 shows how a person lets two of his friends know his location. One of the advantages of the PLASH platform is that it can support different types of mobile devices without changing the application logic or server software. To authenticate users, the server software of Map’n Track Friends uses the authorization control services (i.e., register, login or logout, change and lookup password, or send notification to users) provided by the Service Layer. In Map’n Track Friends, users can store their trip information, share their current trip information with friends, or browse and track their friends’ trip information.

B. Friend Compass

In FriendCompass, shown in Figure 18, we combine Augmented Reality with the location information of friends and points of interest (POIs). Augmented reality means a live view (i.e., obtained by using a camera on a mobile device) of a physical real-world whose elements (e.g., a friend’s location or a POI) are augmented by virtual computer-generated imagery. In FriendCompass, users are able to use their current location, their orientation, and their mobile phone’s camera to view the surrounding information, such as friends’ current locations or POIs. The figure shows how FriendCompass locates a friend and draws his location (i.e., the green human icon) on the camera screen. The server software of FriendCompass uses the fundamental services provided by the Service Layer (e.g., authorization control services and LDSQs services). By using the provided services, it is easy to develop LAS applications on PLASH.

C. Tour Route Recommendation

In the Tour Route Recommendation (TRR) application (see Figure 19), we take advantage of human computation and data mining techniques to provide personalized recommendations. By combining the location and preferences of a user with trip history information gathered from other users, TRR can provide an on-demand recommendation. In the TRR server software, application builders only need to focus on the application logic, i.e., technologies that offer route recommendations to the user. All other required services are provided by the PLASH Service Layer. Note that all the server software of the TTR application server can be hosted by our application server.

D. VProbe

VProbe ([22] and [28]) is an application that uses sensing technologies (e.g., accelerometers and proximity sensors) on mobile devices to evaluate users’ daily driving experiences or behavior (see Figure 20). Users can evaluate their driving behavior afterwards. They can also compare their driving results with those of friends and share their driving experiences

Figure 17. Map’n Track Friends in Andorid, iPhone, and a Web broser.

Figure 18. FriendCompass showing a friend’s location.

Figure 19. Tour Route Recommendation Application.
on a social network service, such as Facebook. Figure 20 shows that the user’s driving behavior is evaluated with six stars and shared with his friends in Facebook. The PLASH platform allows developers to build their own applications and combine them with web services like Facebook.

E. To-and-For (TAF) in Vehicular Network

In many situations on vehicular networks, users are primarily interested in information from a nearby source. To-and-For (TAF) [1] is an application that allows users (i.e., observed user) to share local information, such as road surface conditions, traffic jams, or car accidents (see Figure 21). For more critical information, e.g., a car accident, TAF ensures that all relevant users within the area are aware of the information. A special type of local broadcasting protocol (i.e., geo-casting) that decides to forward relevant information to appropriate users is implemented in the Communication Layer.

F. Literacy Enabling Service

The Literacy Enabling Service (LES) detects and reads text from a mobile device’s camera. Currently, LES supports Chinese and English language. Using the detected text and the user’s current location (and targeted locations), LES is able to provide related information from the Internet. In Figure 22, client software on mobile devices first needs to detect the text area on a live view. Then, the client software sends the detected area (as a small picture of the selected area) and location information to the LES’s application server (server software). The LES’s server software then analyzes the received image to extract the text. The server software is able to use the extracted text and the user’s location information to search the Web and PLASH’s geo-location database.

G. Music Spirit

The Music Spirit is a smart music player application on mobile phone that uses both user’s location and user’s mode to select/suggest a song for user (see Figure 23). Application uses sensing technologies (e.g., accelerometers and proximity sensors) on mobile devices to evaluate users’ mode. For example, if a user’s location is at a park and user’s mode (i.e., movement) is running, Music Spirit application will select a fast-paced music according to genre tag in the music files.

H. Laser Flag

Laser Flag is an augmented reality game that combines laser tag and capture the flag (see Figure 24). Players used his cell phone to upload location information to PLASH server. Players use their cell phones’ camera to find and shot opponents. Laser Flag application uses PLASH’s fundamental services for login and create/join teams with friends. Also, the application use PLASH’s geo-query services for in game action, such as shooting opponents and capturing flags. We recorded all the sensor data (i.e., G-Sensor, compass, proximity sensor, etc.) of players’ smart phone which is used in different research to determine user’s mode (e.g., running, walking, or jumping).
I. Academia Sinica Discovery

Academia Sinica Discovery (ASD) is a phone application that developed for the Open House of Academia Sinica in 2010 (see Figure 25). This is one of the examples that an application is completed in less than one month with developers have no experience on developing phone application and has general knowledge on Java programming. ASD is a sample game/tour guide with missions of discovers ten different locations (i.e., department/building) on Academia Sinica campus and takes pictures at different locations. The pictures are uploaded to the Google’s Picasa for user to view and download. It is easy to see that we can mash up with different services and create new applications in a more efficient way.

![Figure 25. Academia Sincia Discovery application.](image1)

VII. COOPERATION WITH DIFFERENT RESEARCH

In this section, we describe some of on-going research projects build on top of PLASH platform. Our platform allows researchers to easily create different type of services to support the needs and collect useful information for different research areas. PLASH is a useful tool for data collection and data processing. With some of the services built by different research projects, PLASH is able to produce more interesting mobile applications.

A. City Profiling and Traffic Pattern Analysis

Using PLASH and authorized data of the public transportation system, we are able to analysis traffic pattern of different street segments. In order to understand the traffic condition in a city, we must collect accurate traffic data from the wide-area road network with long period of time. In PLASH, we have been collect two types of data from both mobile application users and public transportation system. After traffic data is collected, we analysis different traffic pattern. In Figure 26, we compare the difference of travel route between buses and taxis. We are able to analysis the relationship between the public transportation and the real traffic in the road network in Figure 27. In Figure 28, we compare the average speed of different buses at a road segment. We observed that the average speed of buses is depended on their route (i.e., different bus stops), number of passengers (i.e., passengers need to get on or off a bus), and current traffic condition. Notice that Bus ID 117-AD’s travel speed is different than other three buses (i.e., 202-AD, 507-AD, and 548-AD) in Figure 28. Thus, the data collected from the buses cannot represent the current traffic condition of a road segment.

![Figure 26. Traffic data of Bus and Taxie.](image2)

(a) Bus Trace

(b) Taxie Trace

![Figure 27. Bus and Taxie Trace Overlay on Map.](image3)

![Figure 28. Average Speed of Buses at Road Segments.](image4)

B. Dynamic Routing Diversion using Real-time Traffic information

In this research, we use the GPS coordinates collected from PLASH users who are driving on the city road network to estimate the traffic condition of road segments. We are able to use the PLASH platform to build a speed pattern model to describe traffic condition in the complex city road network. We build a classification-based guidance model by learning the traffic data history provided by PLASH’s geo-location database. Using the PLASH platform, we are able to compare the accuracy of predicted traffic data of the proposed technique and three other techniques [add related works] with real traffic data. Figure 29 shows the average accuracy of four route guidance methods. In Figure 30, we showed that the accuracy of traffic predication improved when there are more data collected by PLASH.
C. Study of Network Connectivity under High Speed Train

In this research, PLASH platform is able to help to identify the network performance alone the Taiwan’s High Speed Rail (THSR). Researchers are able to build a monitoring service on top of our platform to observer the packet drop and round trip time when PLASH user takes THSR. This helps researchers to identify the cause of decrease of network performance due to speed of THSR or physical environment (i.e., tunnels). In Figure 31, we are able to identify the cause of long round trip time (RTT). Between packet IDs of 21182 to 24868, the cause of long RTT is due to in and out of tunnels when we compare the packets’ location information (see Figure 32). We also able to identify the long RTT are caused by high speed of THSR (i.e., 286 km/h) between packet IDs of 2507 to 36142. Notice that Figure 31 also shows difference of performance between different 3G/phone carriers. This is result of the coverage of the network of different carriers.

D. Routing Protocols for Vehicular Network

With the data provided by PLASH, we are able to observe that the network connective of wireless network is depending on environment. We proposed a resilient VANET (RVNET) [13] that can effectively retain the coverage and capacity of the original network. Using the data collected and provided by PLASH, we are able to model and simulate the proposed approach. Our results indicated that our technique (RVNET) using virtual router concept can effectively withhold most of its original quality of service. Figure 33 shows the effect on the fraction of packet deliver rate by reducing number of mesh nodes that are accessible. We observe that the proposed resilient mesh network can sustain mesh node failures very well. Even when there are 8 mesh node failures, the delivery rate degrades by only 20%.

E. Monitor Network Performance of Different Phone Carriers

Use PLASH platform, researchers are able to identify and monitor the performance of different phone carriers. Based on the PLASH users, we are able to compare the successful delivery rate, round trip time, data rate of three major 3G carriers in Taiwan.

<table>
<thead>
<tr>
<th>Performance Metric</th>
<th>Carrier</th>
<th>Far East Tone</th>
<th>Chungwa Telecom</th>
<th>Taiwan Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Successful Delivery Rate</td>
<td>99.64%</td>
<td>99.49%</td>
<td>89.60%</td>
<td></td>
</tr>
<tr>
<td>Average RTT</td>
<td>0.1363279</td>
<td>0.29634813</td>
<td>0.27645841</td>
<td></td>
</tr>
<tr>
<td>Average Drop Connection</td>
<td>2.34</td>
<td>1.63</td>
<td>1.32</td>
<td></td>
</tr>
<tr>
<td>Average Retransmission</td>
<td>108</td>
<td>150</td>
<td>143</td>
<td></td>
</tr>
<tr>
<td>Out of Order</td>
<td>11</td>
<td>9</td>
<td>13</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Performance of Different Phone Carriers.
F. Sensor Network Research

In this research, volunteers use the VP probe application described in VI.D to contribute their phones’ sensing capabilities and gather, and analyze, and share local knowledge about their surroundings. Researchers proposed a novel Comfort measuring System (CMS) [22] for public transportation system. CMS exploits the GPS and 3-axis accelerometer functions of modern smart phones to measure the comfort level of vehicle rides. It then mashed up with the sensed data with the authorized data of the public transportation system and provided a comfort statistics.

VIII. CONCLUSIONS AND FUTURE WORKS

In this paper, we have proposed an open Platform for Location-Aware Service with Human Computation (PLASH). The multi-layer platform and application specification language provide a convenient environment for application builders to develop and host location-aware applications. Builders can also contribute their mature applications as shared services accommodated in the Service layer for other application builders to expand and create more sophisticated applications. PLASH users can enjoy the LAS applications and simultaneously contribute their data and intelligence/efforts to help solve difficult location-related problems, such as real-time surface traffic estimation, city panoramas, and location-aware social networking analysis.

A number of issues need to be addressed in our future work. (1) Scalability: currently, the PLASH platform cannot handle a large volume of requests. Advanced computing mechanisms, such as cloud computing, may help alleviate the scalability problem. (2) Efficient Geo-query Processor: a Geo-query processor determines the response time of PLASH as each upload of geo-location data will trigger the examination of each geo-query of applications. More efficient algorithms and resource management are required. (3) Privacy: PLASH only applies minimal privacy protection by hiding the users’ IDs; hence, there are still many privacy issues to be addressed.

ACKNOWLEDGMENTS

We wish to thank Ling-Jyh Chen, De-Nian Yang, Sheng-Wei Chen, Jan-Ming Ho, Jane Liu and all other PLASH team members for their contributions to this project.

REFERENCES