Toward Standardized Localization Service

Filip Lemic*†, Vlado Handziski*, Nitesh Mor†, Jan Rabaey†, John Wawrzynek†, Adam Wolisz*
*Telecommunication Networks Group (TKN), Technische Universität Berlin, Germany
†Qualcomm Ubiquitous Swarm Lab, University of California at Berkeley, USA
Email: {lemic,handziski,wolisz}@tkn.tu-berlin.de, {mor,jan,johnw}@eecs.berkeley.edu

***Abstract*—Localization services available on today's mobile devices are proprietary and leverage a limited set of sources of location information. Integration of new location estimation methods is therefore cumbersome, requiring adaptation to the specific interfaces of the proprietary location service. In addition, location-based applications are tightly interwoven with the location service that is typically provided by the operating system, hence these applications require significant restructuring to be able to run with another location service. To address these problems, we propose a modular localization service architecture that consists of location-based applications, an integrated location service enabling a fusion of so-called elementary location services, and resources for generating location information. A unified style of interaction among these components is enabled by a set of well-defined Application Programming Interfaces (APIs). The practicability and advantages of the proposed design is demonstrated by outlining how the APIs can be realized using modern types of component interactions.**

I. Introduction

The availability of information for the physical location of a mobile device is a driver for a plethora of Location Based Services (LBSs) and therefore a "hot" research subject. While for outdoor environments the Global Navigation Satellite System (GNSS) is the primary source of location information, even there we observe the tendency of enhancing it by inputs from other sources, notably WiFi and cellular telecommunication systems. Such a combined usage of different sources of information is attractive from the point of view of both improving the accuracy and balancing the required accuracy with the unavoidable latency and power consumption.

The situation is more complex in indoor environments where the GNSS is not reliable, but in which people spend most of their time. Development of simple, reliable, and accurate indoor localization methods is the subject of active research [1], [2]. This efforts yielded a plethora of methods differing substantially in the types of signals used for localization, the processing of these signals, and, last but not least, in the way the location information is delivered to the applications, i.e. in the APIs for providing location information.

All "elementary" location information provisioning services have limited applicability because each of them provides desirable performance only in environments with specific characteristics. In environments with different characteristics, other elementary services are likely to offer better performance [3], [4]. Hence, it can be anticipated that in the near future popular devices will interact with several elementary services. We recognize a need for an approach enabling dynamic addition and/or exchange of elementary services, as well as fusion of their results in a seamless way from the perspective of location-based applications running on these devices.

Development of such an approach is, indeed, the major goal of this paper. There have been several existing proposals in this direction [5], [6], however they are missing important features, for example dynamic selection or straightforward addition of new elementary services. We present a full proposal for a modular standardized localization service architecture that aims at a comprehensive solution of the above mentioned interoperability and integration problems. The proposed architecture features composition of elementary location provisioning services by a newly introduced component called an "integrated location service". It offers to the location-based applications a single Standardized Localization Interface (SLI) that is independent from the used individual elementary services and their possible fusion. This standardized interaction enables full portability of the location-based applications, by design, across specific elementary location information provisioning services. The concept of an integrated location service is a powerful abstraction allowing for fine-tuning of the usual trade-offs between the accuracy, availability, latency, and power consumption of location information provisioning dependent on the requirements from the applications.

At the same time, a second interface, namely the Elementary Service Interface (ESI), assures modular addition of new elementary services, as well as dynamic binding between an elementary service and integrated location service, e.g. in case a mobile device transits an environment served by the elementary service. Our architecture covers another important aspect concerning resources used by the elementary services for generating location information. These useful resources are generally sensor readings (e.g. accelerometer readings, signals strengths received on radio interfaces, etc.) that might be available both on a mobile device or constrained to an environment potentially surrounding the device. A good example for the later is a signal strength reading of WiFi beacons emitted by the mobile device by a set of Access Points (APs) deployed in a building. These resources are today tightly coupled with an individual elementary service, which is suboptimal in the context of sharing of such resources across diverse elementary services. To meet these needs, we introduce the notion of resources and propose an API, namely the Resource Interaction Interface (RII), enabling their usage in an unified fashion. Finally, to demonstrate the practicability of the proposed architecture, we outline how the defined APIs can be implemented using emerging and scalable types of interactions in today's distributed systems.

II. State of the Art Localization Services

Location awareness is an integral part of today's leading smartphone operating systems, Android and iOS.

In Android, location information is exposed to the consuming applications through the Google Play services location APIs [7]. The *LocationRequest* class is used for requesting location information with certain Quality of Service (QoS) from the operating system. These parameters pertain to the desired location information accuracy (best, "block", and "city" level accuracy), provisioning duration (expiration duration or time), provisioning frequency, maximum wait time for location updates, number of updates to be reported, smallest displacement for location information updates, and priorities for reporting location information (which have impact on the accuracy vs. power consumption trade-off). Current location information can be requested with the *getLastLocation* method. Accessing location information updates requires implementation of the *LocationListener* interface. The interface provides the *onLocationChanged* method for receiving updates upon a change of location information. The *Location* object represents location information consisting of a latitude, longitude, timestamp, and additional information (e.g. bearing and altitude). Location information context, such as an environmental map or a translation of location information to an address can be requested using the Google Maps Geocoding API.

Similar functionalities are provided in the iOS's Core Location framework [8]. The standard location service provides location information and its changes for a specified level of accuracy and distance filter parameters. Current location information can be requested using the *requestLocation* method. Provisioning of location information can be requested using the *startUpdatingLocation* method. Furthermore, location information updates can be based on region monitoring service. Region monitoring can be started using the *startMonitoringForRegion* for geographical and the *startRangingBeaconsInRegion* method for beacon regions. Moreover, location information updates can be based on visit monitoring, where such updates are delivered upon entering or exiting frequently visited locations, such as home or work. Visit monitoring can be started using the *startMonitoringVisits* method. The significant-change location service's method *startMonitoringSignificantLocationChanges* provides location information updates upon specified significant change of such information, where the change is larger than the one covered with region or visit monitoring. Contrary to Android's API where location information updates are requested for a certain time period and upon its expiration have to be requested again, provisioning of location information updates in the iOS's framework has to be stopped explicitly using appropriate terminating methods. Similar to the Android's Geocoding API, iOS provides its own Geocoder for converting location information into names, addresses, placemarks, etc. Finally, iOS provides applications access to environmental maps though the Map Kit framework.

The localization service architectures in both Android and iOS are tightly integrated with the operating system and provide a well-defined interface for the location-based applications to access location information. Clearly, due to differences in the interfaces, portability of the application across operating systems from the location information provisioning perspective is not straightforward. In both architectures, the operating system is an entity that fuses location information provided by different elementary services. In particular, the elementary services are based on GNSS, cellular, and WiFi sensors, with the addition of iBeacons for the iOS's framework. In general, elementary service in both architectures evolve over time (e.g. addition of iBeacons in the iOS's framework), but their addition is slow and depends on releasing new versions of the operating system or of the core services. Therefore, the addition is not happening "on-demand" and it is hence unable to keep pace with the rapidly evolving elementary services.

These elementary services could be continuously available (e.g. running on a mobile device), but could as well have a locality feature, i.e. provide service for a given environment only. An example of the latter is a WiFi fingerprinting-based elementary service, where a training database, i.e. a sampling of a served WiFi environment at known locations, is required for generating location information of a mobile device [9]. In this case, a fingerprinting server containing the training database is usually deployed in a served environment, while a fingerprinting client deployed on a mobile device interacts with the server for obtaining location information. Moreover, useful resources for generating location information can also be embedded in an environment potentially surrounding a mobile device. For example, surveillance cameras can be used for distinguishing a case of a person (hence device) being in that space, in contrast to nobody being there.

Our goal is to provide a localization service architecture that captures the outlined functionalities of the state of the art services, but also by design enables dynamic usage of both elementary services and resources in case they are available. This includes a scenario in which a mobile device enters an environment served by an elementary service or in which a resource is available in an environment. In that case, the elementary service should be able to start providing location information of the device and this information should be fused with location information provided by other services or, in case of a resource, this resource should by available to the elementary services. Furthermore, the goal is a straightforward addition of new elementary services in a way that does not require versioning of the operating system. Our aim is to achieve that in an unified way that enables portability of consuming applications, elementary services, and resources.

III. Localization Service Architecture

The proposed architecture of a standardized localization service is depicted in Figure 1. A *location-based application* (shortened to an *application*) is a module that requires location information of a mobile device. An *elementary service* is an abstraction of different types of services that can provide mobile device's location information. An *integrated location service* represents a module that supplies the consuming

applications with location information on one side, and requests such information from the elementary service on the other. The integrated location service manages the selection of elementary services to be invoked, given the requirements from the applications and the accuracy/cost trade-off of each elementary service. The integrated location service provides a setting for fusion of location information provided by different elementary services. It further allows cashing of such information and its provisioning to many applications at once, as well as a setting for calculating parameters related to location information, such as the speed or direction of movement. It also offers redundancy of such information in case some elementary services fail to provide it. The applications are thus agnostic from a specific elementary localization service. Furthermore, the elementary services are agnostic from the perspective of location information fusion, which simplifies their introduction in a mobile device. A *mobile device and environmental resources for generating location information* (or shortly *resources*) is an abstraction of input data needed by the elementary services for generating location information.

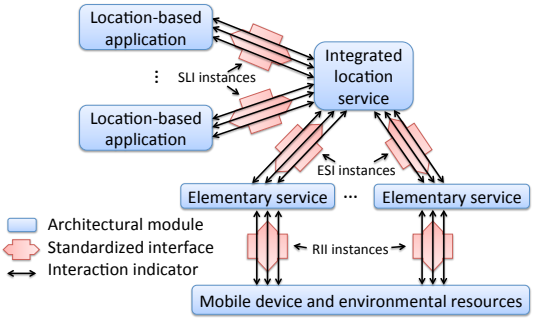

Fig. 1: Localization service architecture

A naive procedural overview is given as follows, with more details provided in the following section. The procedure begins by the application requesting location information of certain features from the integrated location service. Both the application and the integrated location service are envisioned to run on a mobile device. In case the requested location information is not available at the integrated location service level (i.e. not cashed), the integrated location service issues a request for location information provisioning of desired features to one or more registered elementary services. The elementary services do not necessarily have to be standalone services running entirely on a mobile device, as long as they registered themselves to the integrated location service. Each elementary service responds with an "offering", i.e. it bids a set of location information features that can be provided to the integrated location service. This offering is based on the availability and quality of resources needed for generating location information. These resources can be available both on a device and in an environment. In case the provided offering is satisfactory, the integrated location service requests a service from the elementary service. The service is then acknowledged or rejected by the referred elementary service for a certain time period. The procedure continues by the elementary service accessing the resources, followed by generating and reporting location information to the integrated location service. The integrated location service fuses all such information and reports final location information to the application.

IV. Standardized Localization Interface

As shown in Figure 1, the SLI defines parameters and actions required for enabling interaction between the location-based application and the integrated location service. The ESI standardizes interaction between the integrated location service and the elementary service. The RII defines a set of standardized actions for accessing resources needed by elementary services for generating location information.

A. Standardized Localization Interface (SLI)

The *Specify policy* action of the SLI is used by the applications for specifying the desired priority of location information provisioning. The only parameter of the action is *policy*, which can either specify "accuracy", "latency" or "power" as the desired provisioning priority. This priority specification is then used by the integrated location service maximizing the utility function of elementary services invoking based on the policy specified by each application and the accuracy, latency, and power consumption parameters of each elementary service.

Furthermore, the SLI's actions cover the requesting and reporting location information of desired features, i.e. the location type and dimensionality (2D or 3D), its desired accuracy, the period of its provisioning, and the desired provisioning duration. Furthermore, the SLI includes actions for exchanging the location information context. A summary of the SLI's actions, with an indication of the directionality of each action, i.e. from the location-based application (LA) to the integrated location service (IL) and vice versa, is given in Table I.

The application can request location information of certain features from the integrated location service using the *Request location* action. If such information is available, the integrated location service can report it using the *Report location* action. In case such information is absent at the integrated location service level, the integrated location service initiates the interaction with the elementary services for obtaining it, which is standardized though the ESI and discussed in the next section.

Location information can be reported as a global, local or semantic location, which is defined with the *location type* parameter of the *Request location* action, its type 0, 1, and 2 referring to global, local, and semantic location information, respectively. The global location, comprised of a longitude, latitude, and altitude information, defines a location in the WGS 84 global coordinate system. The local location, represented in a (x, y, z) notation, is understood as a location in relation to some predefined location coordinate, i.e. zero-point, usually in indoor spaces. In addition to their 3D notations, both types are in practice also often used for 2D localization, so the altitude/z-coordinate is not compulsory. In case the application requires location information in a 2D space, the *dimensionality* parameter of the *Request location* action is set to 0, otherwise it is set to 1. If location information is requested for a 3D space, and only 2D location information is available to the

TABLE I: Summary of the Standardized Localization Interface

Action	Description	Direction	Parameters
Specify policy	Specify trade-off policy	LA→IL	policy
Request location	Request location information	LA→IL	type, dimensionality, accuracy, period, on_event, step, duration, movement
Report location	Report location information	IL→LA	location information, distance, confidence interval, movement vector
Request renewal	Request provisioning renewal	LA→IL	duration
Request context	Request context of location information	LA→IL	context type, parameters
Report context	Report context of location information	IL→LA	map (zero-point, map vs. physical sizes) or location types translation

integrated location service (after consolidating the elementary services), this should be reported back to the application to be handled in application-specific manner. In case of local and global location, the confidence interval parameter of the *Report location* action is expressed as the expected average error, i.e. the average offset from the ground-truth location. In contrast to geometrical coordinates that express location information as a point in a particular coordinate system, the semantic location type is a string object used for expressing location information as being inside of a space or in the proximity of an object. This format gives a more contextual meaning to the provided location information. The *inside(space)* expression is used for expressing a location being inside of a space, while for the proximity to an object the *proximity_to(object)* expression is used. In case of the semantic location, the *confidence interval* of the *Report location* action is defined as the correctness of location information and it is expressed as an expectancy (ratio) of the number of correct and all possible information. The *distance* parameter in the *Report location* action is used exclusively in case of the semantic location type, particularly in the *proximity_to(object)* expression to specify the proximity to the object more precisely. This parameter defines the distance implied by the proximity to an object. In addition to reporting location information, the integrated location service can report the *movement vector* parameter, which specifies the movement speed and orientation of a mobile device. This parameter is reported if the integrated location service can calculate it from the historical location data and if the binary parameter *movement* of the preceding *Request location* action is set to its non-default value 1.

The *accuracy* and *period* parameters of the *Request location* action define the desired accuracy of location information and the period of its provisioning. The *on_event* parameter is a binary indicator, with its default value 0 indicating that location information should be provided periodically, with the period specified with the *period* parameter. The value 1 indicates that location information should be provided when it changes from a preceding location information, with the size of change indicated by the *step* parameter of the action. These parameters are defined to better support the broad range of different requirements that location-based applications have. For example, the application may benefit more from receiving location information with a shorter period and/or power consumption, given that its accuracy is above a certain threshold, rather than obtaining a more accurate information, but with a larger period and/or power consumption [2], [3]. E.g. for asset tracking scenarios periodically receiving location information updates is required, while for some other, e.g. person walking in assisted living and health-care scenarios, such information is required on event, e.g. the person left the room [10], [11].

The *duration* parameter defines a time interval for which the application requires location information updates. For some applications and usage scenarios location information is needed just once, which can be defined by setting the *duration* parameter to 0. Setting this parameter to -1 implies that location information should be provided until further notice. Location information can also be requested for a defined amount of time by setting the *duration* parameter to a desired value (in seconds). For the defined time interval, location information updates should, either periodically or on event, be reported to the application. In case location information updates are required longer than the originally defined duration, the application can request a renewal. This is defined by the *Request renewal* action, with the *duration* of the new provisioning as the only parameter in the action. By setting the *duration* parameter to 0, the application indicates that the service is no longer needed and can be terminated.

The SLI also defines a set actions for requesting and reporting a context of reported location information. Two type of context information are envisioned, i.e. provisioning of the environmental map and translation from one location type to another. The *Request context* action is used by the application for requesting the context, while the *Report context* action provides the requested context. The *context type* parameter in *Request context* action defines the type of context requested by the application, with two possible context types envisioned at the moment. If the *context type* parameter is set to 1, in the *Report context* action the integrated location service will report the utilized *map*. Together with the map, this response has to define the zero-point, i.e. the point used as a reference location, the mapping between map sizes in its default resolution and physical sizes of an environment. If the *context type* parameter is set to 2 in the *Request context* action, a translation between location types is requested from the integrated location service. In this case, additional *parameters* resource has to be defined in the *Request context* action, i.e. the input and output location types and input location information. The integrated location service responds with the *location types translation* parameter, i.e. location information of the requested output location type. If the map is not cashed or the translation cannot be carried by the integrated location service, the integrated location service requests this context information from the elementary service. This is standardized through the ESI and discussed in the next section. The sequential numbering of the *context type* parameter is done to simplify the extension in case additional context information is required in the future.

In case of a translation to the semantic location, multiple semantic locations can be representative of one local or global location, hence the response can consist of multiple semantic locations. Similarly, if a translation is from the

TABLE II: Summary of the Elementary Service Interface

Action	Description	Direction	Parameters
Register service	Register location provisioning service	ES→IL	address, location information type
Discover service	Discover provisioning service's features	IL→ES	address
Offer service	Offer provisioning service of certain features	ES→IL	dimensionality, accuracy, period, power consumption
Request service	Request provisioning service	IL→ES	address, period, on_event, step, duration
Report service	Report provisioning service	EP→IL	service_granted, duration
Request renewal	Request service renewal	IL→ES	address, duration
Request location	Request location information	IL→ES	address
Report location	Report location information	ES→IL	location information, distance, confidence interval
Request context	Request context of location information	IL→ES	address, context type, parameters
Report context	Report context of location information	ES→IL	context_indicator, address, map or location types translation

semantic location to the local/global location type, reported location information can be a compound of multiple location coordinates. These segments can then, in the application, be connected to create shapes, such as rooms, buildings, etc. The limitation is that such shapes have to be of a polynomial type, i.e. without curved parts, which is sufficient for most of the space shapes used in practice, noting that curved parts can be accurately approximated by polynomials with small "steps".

B. Elementary Service Interface (ESI)

The ESI defines actions and parameters exchanged between the integrated location service (IL) and the elementary services (ES). An overview of the ESI is given in Table II, with an indication of the directionality of actions.

Firstly, the elementary service has to register itself to the integrated location service as defined by the *Register service* action. The parameters are the *address* where the elementary service can be invoked and the *location information type* that the elementary service can provide. The integrated location service can request the "discovery" of elementary service's features with the *Discover service* action, with the only parameter being the *address* of the elementary service. With the *Offer service* action, the invoked elementary service reports its ability to provide location information and the anticipated features of this information. A registered service may not be available or may not be able to provide location information of desired features. For example, a GNSS-based elementary service will usually not adequately provide location information in indoor environments. Moreover, the WiFi sensor may be disabled, thus a WiFi-based elementary service will not be able to provide location information. The parameters exchanged in the *Offer service* action are the likely *accuracy*, *period*, and *power consumption* of location information provisioning. Except for the power consumption, these are the same parameters as in the *Request location* action of the SLI. This is done so that the integrated location service can assume a central role in the elementary service invoking decisions, based on the accuracy, period, and power consumption trade-offs. In other words, while the *accuracy* and *period* parameters are essential for applications and should be specified by the applications, we believe the power consumption related optimizations should be performed by the integrated location service.

If a satisfactory elementary service has been selected, the integrated location service requests the service with the *Request service* action. In this action, the integrated location service specifies the address of the elementary service using the *address* parameter. Similar to the SLI case, the *on_event* parameter is used to specify if location information should be provided periodically, with the period defined with the *period* parameter, or on event, in case location information changes more than specified with the *step* parameter. The *duration* parameter of the action is used for specifying a time interval for which the service is required. Setting the *duration* parameter to 0 implies that location information should be provided once. Setting this parameter to -1 implies that location information should be provided for a maximum time allowed for service provisioning, as defined by the elementary service. The provisioning service accepts the service using the *Report service* action, with the only parameter being the granted provisioning *duration*. In case the service is required for a longer period of time, it has to be renewed by the integrated location service using the *Request renewal* action, with the parameter being the *address* of the elementary service and the renewal *duration*. In that case the service is either rejected or granted anew for a certain duration using the *Report service* action. The service can be rejected by setting the *service_granted* parameter to its non-default binary value 1. The *Request renewal* action with the *duration* parameter set to 0 can be used to terminate service provisioning before the expiration of previously defined duration.

Once an agreement for a service has been established, the interaction continues by the integrated location service requesting location information provisioning. This is defined by the *Request location* action, with the only parameter being the *address* of the evoked elementary service. The elementary service responds using the *Report location* action, with the parameters being the *location information* in the pre-specified format, the *distance* in case of the semantic location type, and the *confidence interval* of the provided information. These are the same parameters as for the SLI's *Report location* action.

In a similar fashion as for the SLI, the integrated location service can request the context from the elementary service using the *Request context* action, while the requested context is reported using the *Report context* action. The integrated location service can request the utilized map or location types translation from the elementary service. In contrast to the SLI, the *Request context* action also specifies the address of the elementary service where the request for context is directed. The elementary service can either *per-se* provide the context, the context provisioning cannot be granted, or the elementary service can redirect the request to some other context provisioning service. This is defined using the *context_indicator* parameter of the *Report context* action. In case this parameter is set to 0, the context is provided to the integrated location

service by the elementary service, while in case this parameter is set to 1 the context information cannot be provisioned. If the *context_indicator* is set to 2, the requested context can be provisioned by another service, with its address defined with the *address* parameter of the *Report context* action. In this case, the *Request context* action has to be directed to a new address and the addressed context provisioning service will report the context using its own *Report context* action. This scenario is anticipated in case one context provisioning service is a representative for a set of elementary services.

C. Resource Interaction Interface (RII)

The RII defines a set of actions for exposing the resources needed by the elementary services for generating location information. The actions are related to either getting or configuring a resource by the elementary service. The *technology* parameter defines the technology used by a particular service for generating location information. The *feature* parameter defines a specific feature of the technology used by the elementary service. These two parameters can also specify a non-technology specific resource (e.g. an action required from the user of a mobile device). A summary of the RII parameters and actions is given in Table III, with an indication of actions' directionality (resources (RS) to the elementary service (ES) or vice versa). Elementary services can require resources from multiple technologies or multiple features of the same technology [12], [13]. In this case multiple *Get resource* and/or *Configure resource* actions should be issued.

TABLE III: Summary of the Resource Interaction Interface

Action	Direction	Parameters
Get resource	ES→RS	technology, feature
Report resource	RS→ES	result
Configure resource	ES→RS	technology, feature
Report success	RS→ES	success

The *Get resource* action serves as a request for a resource of specific technology and feature, while the *result* of the *Report resource* action contains the requested resource. Some examples are provided in Table IV. The requested resource can for instance be Received Signal Strength Indicator (RSSI) values from the WiFi beacon packets or a Bluetooth ID of a mobile device. Similarly, the *Configure resource* action can be used for configuring a resource by the elementary service. Example wise, a request can be directed toward calibrating a gyroscope by shaking a mobile device or synchronizing a mobile device to the Ultra-Wideband (UWB) anchor in a given environment. In this case, the *Report success* action is issued to report the *success* of the requested operation. The resources can be available on a mobile device's and accessible though a request to the device's operating system. The resources can also be available in an infrastructure and accessible remotely.

Since a plethora of elementary services' types exist and they are in general based on a large variety of technologies and features required for generating location information [14], the parameters in the RII actions are left abstract, defined only as a one or set of tuples *(technology, feature)*. The exact specification of the tuples is left for the future, because at this point it would be merely a speculation. The reason is that the ubiquitous deployments of elementary services in different types of environments are still not widely available. Once such ubiquitous deployments are in place, this will yield a set of parameter tuples that are practically used, hence these tuples will thereafter be standardized through the RII. A non-exhaustive set of examples used in practice is given in Table IV. An exhaustive set of smart-phone sensors that can be used for generating location information is given in [15].

TABLE IV: Example resources for generating location information

Technology	Feature
WiFi	RSSI values of observed beacon packets
Bluetooth	Mobile device's Bluetooth identifier
Cellular	Observed cellular base-stations
GNSS	Observed GNSS fixes
None	Calibrate gyroscope by shaking a mobile device

V. Additional Considerations

If the integrated location service is not desired, the SLI can still be used for interaction directly between the application and the elementary service. This direct interaction can be achieved because the SLI parameters and actions are designed as a subset of the ESI ones. This assumes constrained interaction, where location and context information can be requested, but the features of provisioning (e.g. periodically/on event, 2D/3D, etc.) have to be defined ahead.

Both the SLI and ESI are in essence similar to the location information provisioning APIs of the two previously discussed state of the art services. Both Google Play services location APIs and iOS's Core Location framework specify the functionality of getting the last known location, as well as the functionality of receiving location updates. The SLI's and ESI's actions, with a specific configuration of parameters, can be used for achieving the same functionalities. Android's framework defines the duration of provisioning, where after its expiration the service has to be requested again. The same functionality can be achieved with the SLI's and ESI's actions, same as the termination of service provisioning which is in line with the iOS's explicit termination functionality. Moreover, both Android's and iOS's interfaces provide functionalities of either displaying a location address or translation of coordinates to regions. These are specific types of translation between the global and the semantic location type in context information exchange in the proposed architecture. Both interfaces support provisioning of maps to the application, which is equivalent to the map provisioning in the proposed architecture. This means that the Google Play services location APIs and iOS's Core Location framework can functionally be "plugged" into the integrated location service and therefore can be viewed as an elementary service, as well as a valuable source of context information in the proposed architecture.

Actions defined by the SLI are asynchronous, i.e. an action does not block further execution until a response to the action has been received. Asynchronous interaction is utilized since the provisioning of location information can be requested on event, hence the information can be provided only once in a relatively long period of time. It is reasonable to allow

further processing and timely reaction on external events. The ESI's actions are also asynchronous, since e.g. a request for service can be issued by the integrated location service to many elementary services. Not all parameters are needed for the integrated location service to be able to provide this information to the application, hence the asynchronous interaction. The RII's actions can be both synchronous and asynchronous, depending on the needs of the elementary services.

The proposed localization service architecture and the standardized interaction between its modules allow portability of applications across operating systems. Having a well defined interface between the application and the integrated location service simplifies the development of applications by exposing to the developers only a small set of common actions needed for accessing location and appropriate context information. The integrated location service takes care of interaction with the elementary services and serves as the central point for both service selection and distributing location information. Therefore, through cashing, potentially multiple applications can benefit from one location or context information available on the integrated location service level. Moreover, supporting fusion of location information from multiple elementary services is enabled by design because of the availability of all such information at the integrated location service level. Upon request, elementary services generate and report location information to the integrated location service through a set of standardized actions. Developers of elementary services only have to deal with the registration of their service with the integrated location service and, by following the standardized interaction, their service can provide location information to many applications at once. On the lower level, through standardized access and arbitration to the underlying resources, multiple elementary services can benefit from obtaining the same resource in a given time instance. This also makes the elementary service reusable across OS ecosystems.

Elementary service developers have to define the expected accuracy, period, and power consumption of provisioning such information. Therefore, a means for objectively accessing these parameters is required. This can be done by evaluating elementary services in existing testbed environments (e.g. [16]) or by using publicly available data traces for such evaluation (e.g. [15], [17]). Also, predictions of accuracy, period, and power consumption in dependence to environment type will be necessary (e.g. [4]). Finally, the internal mechanisms of the elementary service for predicting accuracy, period, and power consumption based on the availability and quality of the available parameters will be required (e.g. [13]).

VI. Implementation using the Global Data Plane

In the following we demonstrate how the proposed architecture can be implemented in a modern software environment. Typically such environment includes some middle-ware featuring one of the popular styles of communication, for example single writer-log, publish/subscribe, and Representational State Transfer (REST). For the same of our implementation we have chosen the recently developed Global Data Plane (GDP) platform [18]. The GDP is designed as a universal platform, offering emerging types of a private and secure data access, independent of the data locality. The GDP natively supports a single-writer log-based messaging. The GDP also provides a publish/subscribe and REST interfaces for accessing data. Thus, in contract to a natively provided single-writer log, if a developer wants to apply these two interaction patterns, they are also enabled by the GDP. Logs are in the GDP encrypted and only applications having proper encryption keys can access their data, which ensures data privacy and security. Therefore, the GDP offers a good foundation for supporting software implementation of the proposed architecture.

Distributed single-writer log is an emerging interaction pattern grounded on a distributed, partitioned, replicated commit log service. The GDP also provides support for the discovery of such logs and or granting reading access to a particular log. A log is named by an opaque 256-bit number and consists of a series of records, where each record consists of a record number, a commit timestamp, and data. The mapping of the architecture to the abstractions provided by the GDP is performed by modeling each action by a log which has one owner module allowed to write new data. Other modules can read and be notified when data is being written.

An inherent feature of the GDP-based messaging is the support for cashing and historical information storage, since the timestamped location information is permanently stored in a log. Inherent storage of historical location information is beneficial for many applications and can serve for connectivity optimization (e.g. [19]). The GDP provides the possibility of addressing elementary services of the same type at once, using just one entry in an appropriate "service discovery" log. In contrast to individual addressing, group-based one is beneficial in terms of both period and power consumption.

Interaction between the integrated location service and each location-based application is implemented as a set of GDP logs. Due to distributed messaging enabled by the GDP, the application can access location information provided by the integrated location service even in case the application is not running on a mobile device. This enables a variety of location-aware networking (e.g. location-aware routing [20]) and location sharing scenarios [21]. The application is envisioned to be the only writer to its log for requesting location, with the entry data being the parameters defining the desired features of location information (location type, dimensionality, accuracy, period, on_event, step, duration). The integrated location service is subscribed to that log and upon request reports location information by writing it to a log for reporting location information to which the application is subscribed. If the requested information is not available, the integrated location service indicates that to the application by writing a NULL entry into the log. A renewal of location information provisioning can also be requested by writing in the log for requesting location information, with the only parameter being the renewal duration. As specified in the SLI, the duration parameter set to 0 indicates immediate termination of provisioning. Context information is requested by writing an

entry in a log for requesting context, where each entry defines the requested context type and additional required parameters. The requested context is then reported as an entry into a log for reporting context to which the application is subscribed, where again a NULL entry indicates its unavailability.

On the ESI level, the registration of the elementary service is performed by the elementary service granting the integrated location service a read permission to its "reporting service log", where the first entry to that log is the address of the elementary service and the location information type that can be provided. The integrated location service is a writer to three discovery logs, one for each location information type. The readers to these logs are all elementary services that can provide a particular type of location information. For implementing the ESI's action for discovering service, the writer writes any entry to a desired discovery log. The elementary service subscribed to that log offers service by writing into a log for reporting service, where an entry defines an offering (i.e. dimensionality, accuracy, period, and power consumption). The service is then requested by writing into a log for requesting service to which the elementary service is the only reader, where the entry specifies the period, on_event, step, and duration. The service is acknowledged by writing into a log for acknowledging service, where an entry indicates the provisioning duration. Service can be renewed by again writing into the log for requesting service, with the only parameter of an entry being the renewal duration. The elementary service then acknowledges the service by specifying the renewal duration in the log for acknowledging service. The procedure continues in a similar way as for the SLI, with the difference that each entry in a log for reporting context has to specify the *context_indicator* and the *address* in case another service is provisioning context.

For getting and configuring a resource the elementary service writes an entry in appropriate logs, specifying the (technology, feature) tuple as defined by the RII. Each tuple has its log in which either a value of a resource is written or a success of configuring a resource is reported. Elementary services are readers of these logs, hence they are able to access a resource or obtain a notification of a resource being successfully configured.

VII. Conclusion

In this paper, we defined a standardized modular location service architecture and interaction requirements between the introduced architectural modules. Although the focus was on proposing a localization service architecture, we believe the same principles apply for other sensor-based distributed service architectures, e.g. activity recognition and ambient sensing. A module with similar functionalities as the integrated location service, i.e. selection of services, cashing and processing of information, and information distribution to applications, can serve as a central point in such generalized architecture. This clear decoupling between modules also allows for global optimization and management of resources, which could yield substantial improvements in comparison to today's architectures in terms of operating delays, power consumption, and resource utilization. Future work will be oriented toward further assessment of this hypothesis.

As specified by the RII, resource types to be reported to the provisioning service are currently defined in an abstract way using a parameters tuple (technology, feature). Future work includes defining a full set of such resources, once when in becomes clear what are the technologies and features with practical utilization for generating location information. Moreover, the context of location information will possibly be extended, which is already envisioned in the SLI's and ESI's design. A GDP-oriented implementation of the proposed standardized architecture for a cognitive radio-based scenario is envisioned, which will give us a retrospective on the feasibility of the proposed modules and interactions between them. Therefore, the APIs are still subject to change and additional standardization efforts should be anticipated.

Acknowledgments

Support for this work has come from the EU Project eWINE (grant No. 688116), the German Academic Exchange Service (DAAD), the UC Berkeley Swarm Lab and the Berkeley Wireless Research Center (BWRC). This work was also supported in part by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

References

[1] D. Lymberopoulos *et al.*, "A Realistic Evaluation and Comparison of Indoor Location Technologies: Experiences and Lessons Learned," in *Information Processing in Sensor Networks (IPSN'15)*, 2015.

[2] F. Lemic *et al.*, "Experimental Evaluation of RF-based Indoor Localization Algorithms Under RF Interference," in *Localization and GNSS*, 2015.

[3] T. Van Haute *et al.*, "The EVARILOS Benchmarking Handbook: Evaluation of RF-based Indoor Localization Solutions," in *MERMAT'13*, 2013.

[4] F. Lemic *et al.*, "Toward Extrapolation of WiFi Fingerprinting Performance Across Environments," in *Mobile Computing Systems and Applications*, 2016.

[5] S. Couronné *et al.*, "LocON-A Platform for an Inter-Working of Embedded Localisation and Communication Systems," in *SECON Workshops*, IEEE, 2009.

[6] S. Lempert *et al.*, "Towards a Reference Architecture for an Integration Platform for Diverse Smart Object Technologies," *Proceedings of MMS*, 2011.

[7] Google, *Location APIs*, https://developers.google.com/android/reference/com/google/android/gms/location/package-summary, [Online; 5/1/16], 2016.

[8] Apple, *Core Location*, https://developer.apple.com/library/ios/documentation/CoreLocation/Reference/CoreLocation_Framework/, [Online; 5/1/16], 2016.

[9] F. Lemic *et al.*, "Experimental Decomposition of the Performance of Fingerprinting-based Localization Algorithms," in *IPIN'14*, 2014.

[10] P. Barsocchi *et al.*, "Evaluating AAL Solutions through Competitive Benchmarking," *IEEE Pervasive Computing*, vol. 12, no. 4, pp. 72–79, 2013.

[11] T. Van Haute *et al.*, "Performance Analysis of Multiple Indoor Positioning Systems in a Healthcare Environment," *Health Geographics*, 2016.

[12] J. A. Castellanos *et al.*, *Mobile Robot Localization and Map Building: A Multisensor Fusion Approach*. Springer Science & Business, 2012.

[13] N. Wirström *et al.*, "Localization using Anonymous Measurements," in *Distributed Computing in Sensor Systems*, IEEE, 2015.

[14] H. Koyuncu *et al.*, "A survey of Indoor Positioning and Object Locating Systems," *Computer Science and Network Security*, no. 5, pp. 121–128, 2010.

[15] N. Moayeri *et al.*, "PerfLoc: An Extensive Data Repository for Development of Smartphone Indoor Localization Apps," in *PIMRC'16*, IEEE, 2016.

[16] F. Lemic *et al.*, "Infrastructure for Benchmarking RF-based Indoor Localization under Controlled Interference," in *UPINLBS'14*, 2014.

[17] T. Van Haute *et al.*, "Platform for Benchmarking of RF-based Indoor Localization Solutions," *IEEE Communications Magazine*, 2015.

[18] N. Mor *et al.*, "Toward a global data infrastructure," *IEEE Internet Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[19] C.-C. Tseng *et al.*, "Location-based Fast Handoff for 802.11 Networks," *IEEE Communications Letters*, vol. 9, no. 4, pp. 304–306, 2005.

[20] W.-H. Liao *et al.*, "Grid: a fully location-aware routing protocol for mobile ad hoc networks," *Telecommunication systems*, vol. 18, no. 1-3, pp. 37–60, 2001.

[21] N. Li and G. Chen, "Sharing Location in Online Social Networks," *Network, IEEE*, vol. 24, no. 5, pp. 20–25, 2010.