An architecture for user-managed location sharing in the Future Internet of Services

Ana I. González-Tablas
Computer Science and Engineering Department
University Carlos III de Madrid
Leganés, Spain
AnaIsabel.Gonzalez-Tablas@uc3m.es

Mohammad Alam
Fraunhofer Institute for Secure Information Technology
SST Department
Munich, Germany
Mohammad.Alam@sit.fraunhofer.de

Mario Hoffmann
Fraunhofer Institute for Secure Information Technology
SST Department
Munich, Germany
Mario.Hoffmann@sit.fraunhofer.de

***Abstract*—In this paper we analyse the problem of providing an user-managed system for sharing the user's location information in the Future Internet of Services, and propose some architectural mechanisms to support this kind of system. Our approach is based on the work done within Kantara's UMA WG. Furthermore, we highlight open issues that still need to be addressed in location information sharing scenarios.**

I. Introduction

The Internet of People, Things and Services will enable a future ambient lifestyle where everything and everyone could be interconnected everywhere and at anytime. Context-aware personalised services will support daily life in intelligent environments around us. One of the most popular context information already today is location. Sustainable success, however, depends strongly on how users can keep under control who and what can get access to personal information when and where.

Nowadays, location information of mobile entities can be obtained very easily. GPS-based receivers are available in most modern mobile handsets. Furthermore, the location of a mobile device can also be determined, with more or less accuracy, using as reference the nearest cellular or WiFi base stations.

As technologies mature and become available, there is an increasing use of the location information. With more or less success, a bunch of location based service are currently deployed and can be accessed through Internet. Some of these services help the user to use her own location information for her own benefit (navigation, find nearby restaurants, etc.) and others allow the user to share her location information with other users or applications like Google Latitude [1], Fire Eagle [2] or Dopplr [3]. Examples of specific services offered by these Web applications are seeing in a map where your friends are, receiving an alert when you are close to some friend, receive suggestions specific to where you are, etc.

When users store information at some Internet node (within the cloud) and want to share it with other applications and users, privacy is an issue [4]. Regarding location information, Duckham and Kulik define location privacy as *"a special type of information privacy which concerns the claim of individuals to determine for themselves when, how, and to what extent location information about them is communicated to others"* [5]. For example, a user may want to share her specific location while shopping with some of her friends but share only the neighborhood where she is with other friends. Or you may want to share your location with some colleague temporarily till you meet with him but not further in time.

In the last decade, location privacy has been extensively addressed by academia. Duckam and Kulik identify four types of location privacy protection strategies in their survey on location privacy [5]: regulatory strategy, privacy policies, anonymity and obfuscation. Several of them can be used at the same time by privacy protecting systems.

The first approach considers the development of regulatory frameworks such as the EU Directive 2002/58/EC on privacy and electronic communications [6]. These frameworks provide general rules to protect citizens privacy and they are usually based on the same principles [7]. The second approach considers the definition of privacy policies that specify who can access the user's data, in which manner and under which conditions. Ardagna *et al.* present in [8] a good description of some representative privacy-based access control systems for location information.

Third approach considers anonymizing the location information for a user. A first group of these techniques dissociate the location information from the user's real name by using pseudonyms instead. A second group of anonymizing techniques create ambiguity about which entity, among a group, the reported location information belongs to. Fourth approach to protect location privacy encompasses techniques that degrade the quality of the location information, such as providing the city where a user is instead of a more specific location information. A survey of representative techniques following the third and fourth approaches can be found in [9].

Description of the problem. Excluding regulatory frameworks (they establish very general principles and are usually independent from technology), existing solutions do not fully satisfy the requirements imposed by the open Web 2.0 environment and much less the ones imposed by the foreseen Future Internet. In the anywhere, anytime and any way Future Internet, a full world of location-aware services will be available for users. Users will want to share their location information with these services and also that these services cooperate between

them, then sharing the user location information if necessary, to provide users with a better personalized experience.

Anonymization techniques may be an option for some type of location based services, but in the location sharing scenarios we address in this work, a user specifically wants that other users know her location. This means that the users receiving the location information can identify somehow the user whose location is being shared (that is, her real identity, one of her email accounts, a pseudonym in some system, etc.).

A combination of privacy policies and obfuscation techniques seems appropriate for allowing a user-managed location information sharing. In fact, currently deployed location-based web services that allow location information sharing usually provide their users with this type of privacy preserving mechanisms. For example, Google Latitude allows to share the best available location or the city level with other users or applications. Fire Eagle allows its users to share the most precise available location information or one of the levels in the structure Fire Eagle uses to describe location information (country, state, town...).

Unfortunately, these mechanisms are not enough. Location-based web applications are already deploying mechanisms that allow users to share in an automatic and transparent manner their location information with other web applications, most of them using the OAuth protocol [10]. For example, Dopplr users can share their location information with Facebook and Fire Eagle very easily; other applications allow to share a user's Fire Eagle location with Facebook [11] and Twitter [12] in both directions; and others allow updating Fire Eagle location from Google Latitude.

Users may end up sharing their location information among several web applications (see a real example in [13]) and they may lose control of who they are sharing their location information with and in which manner. Furthermore, currently, users of location sharing web services must configure the preferred sharing policies in each of them, having to cope with possible different policy options and terms. If users wish to make any change in their sharing policies at a global level (for example, stop sharing best available location and share city level instead), they have to propagate this change manually in all the location sharing web services they have an account. This situation is unacceptable in the foreseen Future Internet scenario. Users should have a mechanism that allows them to easily manage in a personalized way the sharing of their location information between services and users, and to have a global and understandable view of which location information they are sharing with and how this information is shared.

Goal and contributions. In this work we have as goal to provide some light regarding the design and integration of adequate personalized privacy preserving mechanisms for sharing location information between Future Internet web services and other users.

Our proposal is based on the work developed within the User-Managed Access (UMA) Working Group[1] [15] and intends to contribute to it. The purpose of the UMA WG is to develop specifications that let an individual control the authorization of data sharing and service access made between on line services on the individual's behalf, and to facilitate interoperable implementations of the specification. The specifications developed so far are the identification of the requirements [16], the elaboration of a set of scenarios and use cases (including at the moment a basic description of a location sharing scenario) [17], the definition of protocols and protocol's profiles [18]–[20] and data structures [21] needed to satisfy the elicited requirements. Furthermore, a discussion of legal implications [22] is also being elaborated. Most of them are still work in progress.

In this paper we analyze the problem of providing an user-managed system for sharing the user's location information in the Future Internet, and propose a UMA based architecture to support this kind of system. To do so, we present a set of uses cases where users share their location information with location-aware services and other users, and study the specific new requirements that arise in them compared to the requirements already considered within the UMA work. Then, for each use case, we propose and discuss possible architectural mechanisms that would allow to address the identified requirements. Furthermore, we highlight other open issues that would be interesting to address within the UMA framework, for location information sharing scenarios and other scenarios which may have the same needs.

Organization of the paper. The paper is organized as follows. Section II briefly describes the UMA architecture and the UMA 1.0 Core Protocol. Then, in Section III we analyze the use cases, identify their requirements and propose some architectural solutions. Section IV presents a summary of our work and a description of some identified open issues.

II. User-Managed Access Architecture and Protocol

In the UMA architecture, the *Authorizing User* (see Figure 1) manages a set of resources at one or several *Hosts* within Internet. *Requesters* are the entities that want to access (read, write, etc.) a specific resource owned by the Authorizing User and stored at some Host. Requesters may act on behalf of some *Requesting Party*, which can be a web user, a corporation or any other legal person.

The Authorizing User delegates the decision of authorizing or not a request submitted by Requesters to a distinct entity called the *Authorization Manager (AM)*. The Authorizing User controls the authorization decisions taken by the Authorization Manager by configuring a set of policies that establish for each resource owned by the Authorizing User at each Host, the conditions that must be met in order to grant access.

In comparison with the XACML model [23], there is the following relation between the used terms (showing first the UMA term and, second, the XACML model term): Authorizing User↔Policy Administration Point (PAP), Authorization Manager↔Policy Decision Point (PDP), Host↔Policy Enforcement Point (PEP), and Requester↔Subject.

[1]The UMA WG is part of the Kantara Initiative [14]

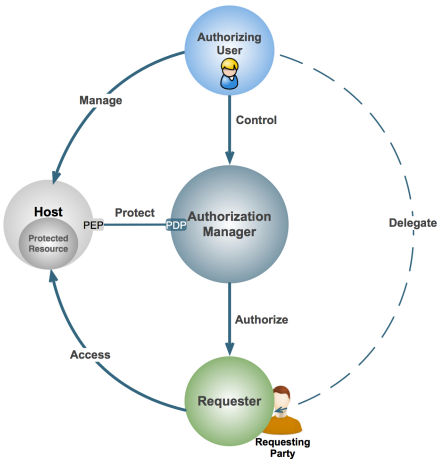

Fig. 1. This figure shows the entities of the UMA architecture and their relations. Source: UMA WG [15]

The UMA 1.0 Core Protocol has three main steps which are briefly described next (an illustration of its flow can be found at [24]). It is built on top of two OAuth 2.0 flows [10].

- **Step 1: Authorizing User introduces Host to AM.** In this step the Authorizing User grants the Host the right to use the AM services in order to obtain authorization decisions regarding the resources owned by the Authorizing User and that are stored at the Host. To acomplish this, the Host has to register as an OAuth client at the AM, obtain the authorization grant from the Authorizing User in the form of an OAuth authorization code and, using this authorization code, obtain an OAuth access token (known as the host access token) from the AM that would allow the Host to use the AM authorization services in next interactions with this entity. This would be the first OAuth flow. At this point, the Host may also register the protected resources at the AM as scopes [20].
- **Step 2: Requester gets access token from AM.** This step and the third one form the second OAuth interaction considered in the UMA 1.0 Core Protocol. In this second step, it is assumed that the Requester has registered at the AM (if not, a dynamic registration method has been proposed [19]). Therefore, it is possible for the Requester to obtain a requester access token from the AM in order to access a specific resource at some Host (scope at the AM), provided that the conditions determined by the Authorizing User are met (which may include the presentation of specific claims by the Requester). Otherwise, the authorization request is rejected. The Requester may try first to access the resource at the Host without presenting a requester access token, then he is redirected to the appropriate interface of the AM to perform the process described previously in this step.
- **Step 3: Requester wields access token at Host to gain access.** In this step, the Requester tries to access the protected resource at the Host by presenting the access token obtained in step 2. The Host then can validate the token locally or request the AM to validate it. After an authorization decision has been made by the Host or the AM, the access to the resource is allowed or denied.

Most of the UMA specifications are still work in progress and a description of some of the open issues that are currently being addressed can be found in [25].

III. User-managed location information sharing: Analysis of uses cases and proposal of architectural mechanisms

In this Section we analyze a set of location information sharing use cases, identify the new requirements they arise and propose some architectural solutions.

A. Administration of sharing policies from the user's mobile

Alice lives in the Vancouver area and because of work reasons she travels quite frequently inside this area and along the West coast, some times by car or bike, some times by airplane and others using the public transport. Additionally, because of the nature of her work, she may enjoy several unexpected holiday days from time to time. She usually takes advantage of these days by making a trip with her partner, Teo, who works as a freelance photographer and can make the most of these trips by taking some nice photos.

As her movements are a little bit unplanned, Alice wants to share her actual location with some people and services. In first place, Alice wants to share her location with her partner. To do so, she uses the Google Latitude mobile application on her Android mobile. This application can determine the mobile's location by several means and send it to the Google Latitude service. In order to share her location information with her friends and allow her friends to see where she is on a map or receive an alert when they are close to each other, Alice uses a Location Sharing Authorization Manager (LSAM) service. Alice configures in the LSAM whom and under which circumstances she wants to share her location information stored at Google Latitude [2].

Her partner, Teo, has also an Android mobile, so he likes too to use Google Latitude to see where Alice is. Teo also allows Alice to see his location and, by default, both of them allow each other to see the neighborhood[3] where they are. Sometimes they want the other to know their best available location so they can meet more easily. In order to do so, they change temporarily the policy at the LSAM from their mobiles.

1) Discussion and architecture: Most of the requirements of this scenario can be solved by the current UMA 1.0 Core Protocol. In this case, Alice is the Authorizing User, the Host is Google Latitude, Teo is the Requesting Party, the Google Latitude Mobile Application at Teo's mobile handset is the Requester (that is, the Requester is also Google Latitude via

[2]Currently, Google Latitude allows its users to configure basic privacy policies for sharing their location information with other Google Latitude users. Note that we are not describing what currently happens, but what would happen if services would be UMAnized (that is, UMA compliant instead of OAuth compliant, that is what happens now)

[3]Choosing neighborhood as location granularity for sharing is not possible at the moment in Google Latitude.

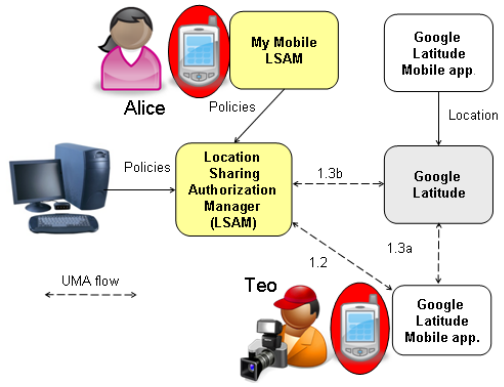

Fig. 2. Illustration of the administration of sharing policies from the user's mobile use case: Entities and their interactions.

the Google Latitude Mobile Application at Teo's handset) and the Authorization Manager is the Location Sharing Authorization Manager. The involved parties for this use case and their interactions are shown seen in Figure 2. Assuming that Google Latitude has already registered at the LSAM, step 1.2 in Figure 2 represents the interactions between the Google Latitude Mobile Application at Teo's handset and the LSAM to obtain the access token, step 1.3a represents the interactions between the Google Latitude Mobile Application at Teo's handset and Google Latitude service for accessing Alice's location, and step 1.3b represents the interactions between the LSAM and Google Latitude service to validate the access token presented by the Google Latitude Mobile Application at Teo's handset in its request.

A further requirement illustrated by this scenario is that, as authorizing users are assumed to be moving and have a "locatable" mobile device, it would be nice that they can change their location sharing policies from these devices. One way to solve this issue is that the LSAM provides a mobile compliant web interface. The main problem with this solution is that, if users want to link several location sharing services and have personalized policies for each of them, it can be quite complex to show this information in a usable way through the mobile interface.

The alternative solution that we propose is to deploy an specific application in the mobile handset (shown in Figure 2 as MyMobileLSAM) capable of dealing with the authorization management (for this specific use case, only policy administration functionalities are delegated) in this type of resource-constrained platforms and which interacts with the authorization manager. One approach for the design of the MyMobileLSAM is to have at the handset a copy of the location sharing policies. MyMobileLSAM should be capable of showing them in an appropriate way to Alice depending of the resources available in the handset. Alice then would be able to change her location sharing policies locally, as she would do at the LSAM, or by selecting between a predetermined set of policies she would have previously defined at the LSAM. Changes made locally would be submitted by MyMobileLSAM to the LSAM using the most efficient channel available for that specific handset. This update can be made similarly in the other direction if changes are made at the LSAM.

Deploying a local LSAM with policy administration functionalities delegated by the LSAM, rises several security requirements that should be considered (authentication of MyMobileLSAM to LSAM, management of the authorization regarding the delegated functionalities, etc.). However, it is worth that authorizing users have this alternative to administering their policies at the LSAM from their mobile hadsets through Web interfaces appropriate for constrained-resource platforms because of efficiency reasons (it would be more efficient to make changes locally and then send them using whatever connection technology) and the possibility of taking full advantadge of specific handset features that would provide a better user experience while managing their policies.

Furthermore, most of the location updaters used by the mentioned location-based web services assume or recommend a continuous Internet connectivity at the handset (via WiFi or with specific data plans contracted to mobile operators) in order to determine the location and update it at the location-based web service. Not all users and handsets may be able or willing to meet the continuous Internet connectivity requirement, but in the case of mobile telephone handsets, location-based web services are still possible as its location can be determined by the mobile operator, which would act as Host for the rest of location-based web applications. In this situation, designing a non Web-based local LSAM is an advantage.

B. *User-managed sharing between multiple hosts*

Apart from her partner, Alice has also some other friends who she wants to share her location information with, for example, her best friend Martha. Martha prefers to use Yahoo's Friends on Fire! application for Facebook to see her friends whereabouts, as she is usually logged in Facebook at her notebook and, therefore, finds this application more convenient. To share her location with Martha and other friends that use Friends on Fire!, Alice has also an account at Fire Eagle location-based web service. Fire Eagle acts as a broker for its user's location information, accepting several location updaters and consumers. In fact, from time to time, Alice chooses to use the Android Fire Eagle Updater in her mobile handset, which has a better performance in some situations. It updates her location at Fire Eagle instead of at Google Latitude service in first instance, but this is not a problem because Alice has configured in the LSAM that they share her location information so when one receives an update, the other can as well update its location information for Alice accordingly. This way, Teo and her other Google Latitude friends can access Alice's location information as before.

1) Discussion and architecture: The need arised in this use case is how to coordinate and authorize the sharing of Alice's location information between several hosts assumming that all the hosts are going to access this information indistinctly and that an update of Alice's location can be done at any host at any moment. We are going to focus the discussion on

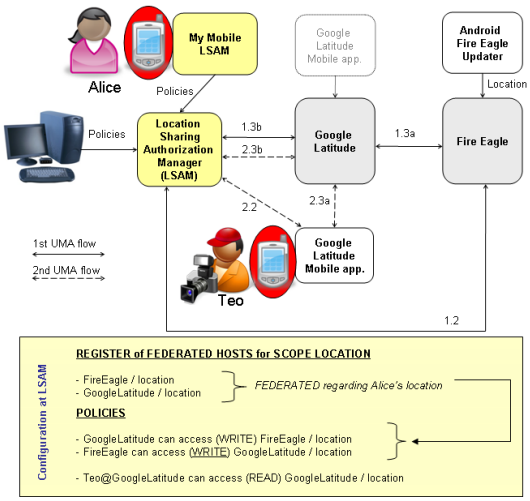

Fig. 3. Illustration of the user-managed sharing between multiple hosts use case: Entities and their interactions for the PUSH mode

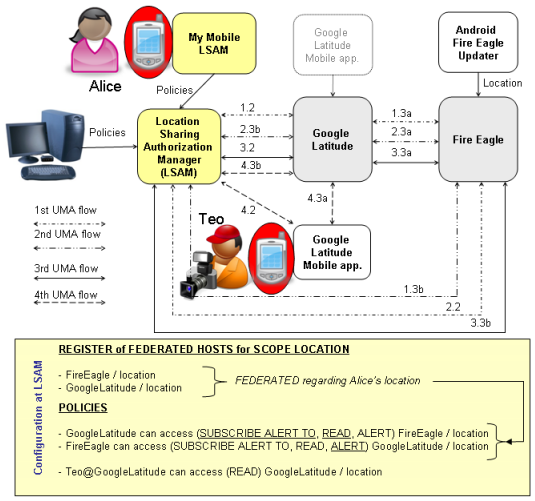

Fig. 4. Illustration of the user-managed sharing between multiple hosts use case: Entities and their interactions for the PULL mode

the specific example where Alice (Authorizing User) uses the Android Fire Eagle Updater to update her location information at Fire Eagle and Teo (Requesting Party) requests Alice's location using the Google Latitude Mobile Application at his handset (Requester). Furthermore, the example focuses on sharing location information between two Hosts (Google Latitude and Fire Eagle), but they could be more.

To solve this requirement, we propose that the LSAM gives Alice the option to configure a *federation of hosts* for some of her resources, in this use case, her current location. This would mean that the federated hosts are going to share these resources between them and they are going to cooperate in order that these resources, updated dynamically at any of them, are synchronized. We assume that the hosts have been introduced to the LSAM and that they have registered their scopes for Alice's location (step 1 of the UMA 1.0 Core Protocol). Then, when Alice accesses the LSAM to configure the policies for her location at these host, she may choose to federate these scopes with others regarding the same information at different hosts. Also, when Alice desires to defederate a host, the LSAM should notify the rest of federated hosts. At the moment, the UMA framework does not consider this possibility, so it should be extended to address this proposal if convenient.

Once the hosts have been federated for some scope, their behaviour in order to propagate the location information between them can be shaped in different ways, once one of them receives an update (in the specific example, Fire Eagle is the primary host that receives the update of Alice's location). We have identified three options: push mode, pull mode and delegation mode.

The suggested interactions between the different entities and the configuration in the LSAM for the *push mode* are shown in Figure 3. In this mode, Fire Eagle acting as a Requester, tries to update the location information for Alice at Google Latitude, who acts as Host. In order to do so, Fire Eagle retrieves an access token from the LSAM to write Alice's location at Google Latitude (step 1.2 in Figure 3). Then, Fire Eagle sends the request for writing Alice's location and the access token to Google Latitude (step 1.3a), who validates the token by sending it to the LSAM (step 1.3b) and enforces the decision of the LSAM to authorize the location update. Finally, Teo interacts with Google Latitude and the LSAM as in the previous use case (steps 2.2, 2.3a and 2.3b in Figure 3).

Of course, the push mode needs that federated hosts implement locally some mechanism that allows them to know which other hosts have to be updated regarding this specific resource and, once a new location information has been received, launch the process for updating it at the rest of the federated hosts. Also, it has to be taken into account that, although it seems a simple solution, the primary host (the first who receives the information) increases its load more than the other hosts in the federation.

The suggested interactions between the different entities and the configuration in the LSAM for the *pull mode* are shown in Figure 4. In this case, we suggest that the primary host, Fire Eagle in our example, is in charge of sending alerts to the rest of hosts in the federation after the reception of an update. Then, the rest of the hosts in the federation, after receiving the alert and if they need to update the information, retrieve it from the primary host.

The information that Alice's location has changed is sensitive by itself (Alice is no more where she was), so it may be convenient to provide access control mechanisms for the alerts. A possible solution is to provide this access control at the LSAM by letting "subscribe to alert" and "alert" be additional actions that are configured in the sharing policies when hosts are federated. Of course, all federated hosts should implement an alert service and keep track of the rest of hosts subscribed to it.

The subscription to the alerts should be done only once. In Figure 4 it is shown how Google Latitude subscribes to Fire Eagle's alert service regarding Alice's location (steps 1.2,

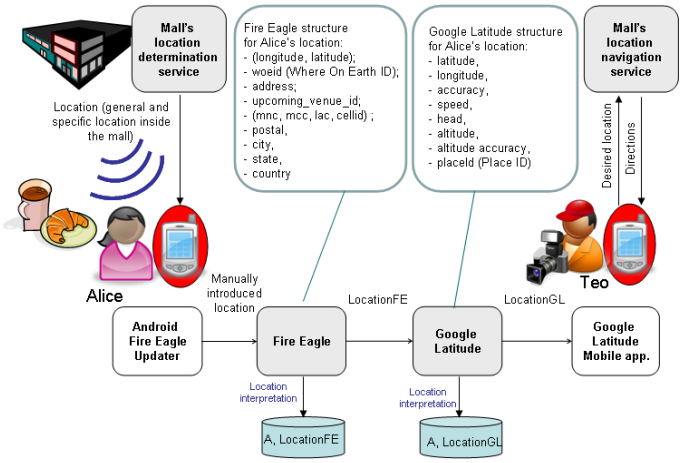

Fig. 5. Illustration of the interoperable hierarchical structure for location information use case: Current representation of location information in each location-based web service.

1.3a and 1.3b). In this interaction, Google Latitude would act as Requester and Fire Eagle as the Host.

Alerts should be sent each time an update is received. The sending of the alerts could also be authorized at LSAM if it provides some advantage. Authorizing each alert by using UMA is clearly inefficient but, on the other side, it provides a more robust assurance regarding that only the authorized parties ae notified by the authorized parties about a change of Alice's location. This interaction is shown in steps 2.2, 2.3a and 2.3b in Figure 4, where Google Latitude acts as the Host and Fire Eagle as the Requester.

Although, authorizing each alert may be not convenient in most settings, another option is to take advantage of the duration attributes of the OAuth access tokens and the OAuth refreshing mechanism to reach some kind of compromise for efficiency and security. This alternative can be also applied to the push mode for authorizing the primary host to update Alice's location information at the rest of the federated hosts.

As in the push mode, finally, Teo interacts with Google Latitude and the LSAM (steps 4.2, 4.3a and 4.3b in Figure 4).

A third option, the *delegation mode*, would be to consider multi-layered delegation of the authorization in the line of some of the proposals made within the OAuth WG [26]. This mode is interesting because it would allow a setting where it is not mandatory that every host in the federation stores or accesses the location information in order to make it available to third parties, as the idea is that they act as proxies regarding authorization only, but that they do not actually retrieve and send the requested location. The proposal in [26], however, has not evolved much beyond the original idea due lack of implementation experience and they need still more discussion to reach a consensus. The interest and efficiency of other alternative mechanisms should also be researched, for example, a chain of federated hosts using the push mode.

C. Interoperable hierarchical structure for location

Alice is having a rest from work and decides to enter a nearby shopping mall (commercial center) to have a coffee. While she is at the coffee shop, she receives an alert from Google Latitude telling her that Teo is also within the surrounding area. She decides to use the specific mall's location determination service (based on the free WiFi network deployed inside the mall) to retrieve at her handset the specific location of the coffee shop where she has stopped. She wants to make this information available to Teo to let him get concrete directions inside the mall to reach the coffee shop using the mall's location navigation service. To do so, she introduces the retrieved location information manually in the Android Fire Eagle Updater [4]. She wants that this information propagates till Teo through her federation of location-based services, as she has configured at the LSAM.

1) Discussion and architecture: The first problem that arises in this use case (see Figure 5) is that the set of location-based web services need an interoperable language to exchange the location information about their users. In Figure 5 the current structures used by each service to represent location information are shown. The location provided by the mall's location determination service can be of any form, but let us assume that it takes the form of the address of the mall and an internal identifier for the coffee shop's local. When Alice updates her location information at Fire Eagle, Fire Eagle will do an interpretation of this location information in order to store it using its structure for representing location information. The address of Alice will be updated with the mall's address but the information about the coffee shop will be probably lost. Again, when Google Latitude tries to retrieve Alice's location information from Fire Eagle (assumming the pull mode is used), it expects a pair of latitude and longitude coordinates. A geocoding process has to be done in order to get the coordinates from the address, either by Fire Eagle or by Google Latitude. At the end, in this specific example, most of the location information, having possibly suffered some transformation in the process, would propagate through Fire Eagle and Google Latitude till it reaches Teo, except for the specific location of the coffee shop inside the mall (which is in fact the more interesting data for Teo in order to get the directions inside it).

The problem is caused because the location of some entity respect to the Earth can be described following different perspectives and current location-based services do not provide the location information for an entity in the same way.

The most common systems currently used to identify or describe the geographical location of some feature are the coordinate reference systems (e.g., WGS84 latitude and longitude), geographical grid systems, geographical names (areas, regions, localities, cities, towns, or other topograhical features of public interest), administrative units and addresses (which

[4]This software has already the option of introducing a location manually. We imagine that the location can also be retrieved from a local file to avoid Alice typing it.

include a geographical name). Humans beings are more happy to work with semantic locations instead of with raw coordinates (generally unmeaningful for the common public).

The OGC Geography Markup Language (GML) Encoding Standard [27] describes an encoding specification for geodata in XML that enables the storage, transport, processing, and transformation of geographic information. It is widely used in Geographic Information Systems but, usually, it is not used by location-based services for representing the location of entities. Instead, these services use their own and simpler languages and structures for representing the different areas (e.g., a circle, a sector of a circle, an intersection of three circles, and other shapes) that are rendered by current positioning systems for the location of an entity. In these cases, it is usual to approximate the position of the entity to a pair of coordinates following not normalized methods.

Furthemore, there do not exist common databases and directories (gazzetters) for geographical names, addresses or Point of Interest. Currently there is work in progress regarding the development of infrastructures for spatial information (e.g., the European INSPIRE work [28]), which include the elaboration of consensuated databases for geographical names and addresses, but there will pass sometime till they are adopted by commercial services.

In order that the proposed federation of hosts works, there is the need that they share a normalized way to interpretate the location information given by current positioning systems, a common representation language (such as GML or a profile of it) to represent the location information of entities and that they use some common infraestructure for spatial information (e.g., common gazzetters which can be referenced).

A solution for interpreting in a normalized way the various shapes of location provided by different location determination services could be to use a common grid structure (such as the one proposed in [29]) as reference in order to interpret the location without loosing information.

In our example, the specific location of the coffee shop could be identified using one of the cells of the grid defined in [29] (which can easily provide granularities of 1 meter). These would allow the location information to propagate through the chain of federated hosts without loosing accuracy and correctness. Each host could use then this information to interpret it respect their own databases if needed. In the example, the mall's location determination system would produce the same information as before (address and local identifier) and the normalized location respect the grid for the coffee shop local. Fire Eagle would try to interpret the data received from Alice in the best available way without loosing information. As no direct match is possible and Fire Eagle would be unable to interpret the coffee shop identifier, Fire Eagle would store the address and the received normalized location. This is the information that Google Latitude would receive, who can interpret the address and the normalized location to a pair of coordinates, but still conserve the normalized location and trasmit it to Teo. This way, Teo can provide an accurate and correct location description of the location of the coffee shop he wants to visit to the mall's location navigation service.

General scope	Specific scope	
	Spatial system of reference	Granularity
Location /	EuropeanGrid /	1000 (1 km.)
Location /	AddressXYZ /	city
Location /	AddressXYZ /	street
Location /	WGS84Coordinates /	1 (1 m.)

Fig. 6. Illustration of how scopes could be configured at the LSAM for mapping different granularities if an interoperable hierarchical structure for location information can be used.

Till now, we have assumed in this use case that all the services are allowed to access the best available location information for Alice, in whatever granularity it comes. This does not reflect the desired situation, because Alice may have interest in authorizing the access to her location information with different granularities for different services or for different requesting parties.

Having a hierachical structure to represent location information would ease this task from the user point of view, as it would be possible to define policies such as "Provide my location information (whatever the spatial layer used to represent it) with a maximum granularity of 1 kilometer". Using a hierarchical structure for describing location information, similar to part of the structure used in Yahoo's GeoPlanet [5], and a hierachical grid as reference, such as the one proposed in [29], enables this feature. If the location of the entity can be indentified by the name of a town, and the geographical representation of this town fits inside the specified granularity (using the grid as reference), access to this information (town name) would be provided. The same can be made for other types of location information description. If it does not fit, for example, the user is located at a street that is 200 meters long and has configured a maximum granularity of 1 kilometer for this service, some degradation process using the grid and the hierachical structure for the used spatial layer (in this case, addresses) should be done in order to find the most appropriate resolution and description for the location information. In the example, a parent of the specific street where the user is located could be provided, if possible, or, if not possible, the normalized location using the grid as representation.

Of course, in our use case, if the location information has to propagate through the federation of hosts, its accuracy may be reduced, but not its correcteness if the reference grid is used.

Having a hierachical mechanism to represent location information, is also useful for mapping different granularities as different scopes in the LSAM in an elegant way. Some examples for Alice's location scopes representing different granularities can be seen in Figure 6.

Considering a grid for representing location information would also allow to incorporate more easily some spatial cloaking scheme (see, for example, [30]) to further protect

[5] http://developer.yahoo.com/geo/geoplanet/guide/concepts.html#hierarchy

the privacy of users.

IV. Summary and open issues

In this paper we have analyzed some of the problems for providing an user-managed system for sharing the user's location information in the Future Internet and we have proposed a set of architectural mechanisms that allow to address these issues. The three presented use cases and the provided discussion illustrate the complexity of providing such user-managed sharing location information system. Still, we have already identified a set of open issues that should also be addressed.

For example, location information has a temporal dimension that should not be forgotten when representing and processing spatial information. Location information for the past and the future is already used in some location-based web services (Google Latitude Location History feature stores past location information for a user and Dopplr service is an example of service that offers future location information).

It is foreseen that policies that depend on the location of the Authorizing User and the Requester/Requesting Party are used. If the policy is dependent on the position of the Requesting Party, they are now considered as Claims in the UMA framework, but specific issues for location claims should be still studied. If policies depend also on the Authorizing User's location (e.g., "Share my location with my friend only if we are nearby (within some specific distance)", then, in order to evaluate the policy, the LSAM should have access to Alice's location, which means that the LSAM may have to act as a Requester. It should be studied how this kind of situations can be incorporated in the UMA model.

A further issue that should also be investigated is how to retrieve the location information of an entity independently of the location service (Host) that can provide it. That is, make the location information globally referenceable for any party and for any location aware service. That would need first making entities globally referenceable (for example, using OpenID mechanisms or similar).

Acknowledgment

The authors would like to thank Eve Maler, chair of UMA WG, and all colleagues for our fruitful weekly discussions. Furthermore we would like to thank Google for their support specifying an architecture for user-managed location sharing. A reference implementation based on Android and Chrome will come soon.

References

[1] (Last accessed on Oct. 2010) Google latitude. [Online]. Available: http://www.google.com/latitude

[2] (Last accessed on Oct. 2010) Fire eagle. [Online]. Available: http://fireeagle.yahoo.net/

[3] (Last accessed on Oct. 2010) Dopplr. [Online]. Available: http://www.dopplr.com/

[4] A. Cavoukian, "Privacy in the clouds," *Identity in the Information Society*, vol. 1, pp. 89–108, 2008.

[5] M. Duckham and L. Kulik, "Location privacy and location-aware computing," in *Dynamic & Mobile GIS: Investigating Change in Space and Time*. Boca Raton, FL USA: CRC Press, 2006.

[6] "Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications)," European Union, Jul. 2002.

[7] "OECD guidelines on the protection of privacy and transborder flows of personal data," Organisation for Economic Co-Operation and Development (OECD), 2001.

[8] C. Ardagna, M. Cremonini, S. D. C. di Vimercati, and P. Samarati, "Access control in location-based services," in *Privacy in Location-Based Applications*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2009, vol. 5599.

[9] J. Krumm, "A survey of computational location privacy," *Personal Ubiquitous Comput.*, vol. 13, no. 6, pp. 391–399, 2009.

[10] E. Hammer-Lahav, D. Recordon, and D. Hardt. (2010) The OAuth 2.0 Protocol (draft-ietf-oauth-v2-10). Internet Draft. [Online]. Available: http://tools.ietf.org/id/draft-ietf-oauth-v2-10.txt

[11] (Last accessed on Oct. 2010) Facebook. [Online]. Available: http://www.facebook.com/

[12] (Last accessed on Oct. 2010) Twitter. [Online]. Available: http://twitter.com/

[13] (2009, last accessed on Oct. 2010) My Google Latitude location in Twitter and FireEagle via Ipoki. [Online]. Available: http://martinrstone.wordpress.com/2009/08/14/my-google-latitude-location-in-twitter-and-fireeagle-via-ipoki/

[14] (Last accessed on Oct. 2010) Kantara Initiative. [Online]. Available: http://kantarainitiative.org/

[15] (Last accessed on Oct. 2010) User-Managed Access Working Group. [Online]. Available: http://kantarainitiative.org/confluence/display/uma/Home

[16] (Last accessed on Oct. 2010) UMA Requirements. [Online]. Available: http://kantarainitiative.org/confluence/display/uma/UMA+Requirements

[17] (Last accessed on Oct. 2010) UMA Scenarios and Use Cases. [Online]. Available: http://kantarainitiative.org/confluence/display/uma/UMA+Scenarios+and+Use+Cases

[18] (Last accessed on Oct. 2010) UMA 1.0 Core Protocol. [Online]. Available: http://kantarainitiative.org/confluence/display/uma/UMA+1.0+Core+Protocol

[19] C. Scholz, M. Machulak, and E. Maler. (2010) OAuth Dynamic Client Registration. Internet Draft. [Online]. Available: http://www.ietf.org/id/draft-oauth-dyn-reg-v1-00.txt

[20] ——. (Last accessed on 1.10.2010) User-Managed Access (UMA) Scope Registration Protocol (draft-uma-core-v1-00.txt). [Online]. Available: http://mrtopf.clprojects.net/uma/draft-uma-scope-registration.txt

[21] (Last accessed on Oct. 2010) Claims 2.0. [Online]. Available: http://kantarainitiative.org/confluence/display/uma/Claims+2.0

[22] (Last accessed on Oct. 2010) Legal Considerations in UMA Authorization. [Online]. Available: http://kantarainitiative.org/confluence/display/uma/Legal+Considerations+in+UMA+Authorization

[23] (10 August 2010) OASIS eXtensible Access Control Markup Language (XACML) Version 3.0. Committee Specification 01. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf

[24] (Last accessed on Oct. 2010) Protocol Flow. [Online]. Available: http://kantarainitiative.org/confluence/display/uma/Protocol+Flow

[25] E. Maler. (Last accessed on Oct. 2010) Controlling data usage with user-managed access (uma). Position paper at the W3C Workshop on Privacy and data usage control, 04/05 October 2010, Cambridge (MA). [Online]. Available: http://www.w3.org/2010/policy-ws/papers/18-Maler-Paypal.pdf

[26] B. Vrancken and Z. Zeltsan. (2010) Using OAuth for Recursive Delegation. Internet Draft. [Online]. Available: http://tools.ietf.org/html/draft-vrancken-oauth-redelegation-01

[27] *OpenGIS Geography Markup Language (GML) Encoding Standard - OGC 07-036*, Open Geospatial Consortium, Inc. Std., 2007.

[28] (Last accessed on Oct. 2010) Inspire. [Online]. Available: http://inspire.jrc.ec.europa.eu/

[29] A. Wirthmann, A. Annoni, L. Bernard, and J. Nowak., "Proposal for a european grid coding system," in *European Reference Grids*, vol. EUR 21494 EN, 2005. [Online]. Available: URLhttp://www.ec-gis.org/sdi/publist/pdfs/annoni2005eurgrids.pdf

[30] M. L. Damiani, E. Bertino, and C. Silvestri, "The PROBE Framework for the Personalized Cloaking of Private Locations," *Transactions on Data Privacy*, vol. 3, p. 123148, 2010.