The Case for a Location Metasystem

Nick Doty
School of Information, UC Berkeley
npdoty@ischool.berkeley.edu

ABSTRACT
Microsoft has proposed an identity metasystem to standardize identity services and the principles behind them. A location metasystem can support interoperation between location services, protect users' privacy and handle issues of granularity. The simple OAuth protocol may be a good model for working towards a location metasystem.

Categories and Subject Descriptors
H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *Web-based services.*

General Terms
Security, Human Factors, Standardization, Theory.

Keywords
Location, metasystem, identity, privacy, granularity.

1. INTRODUCTION
As more services take advantage of location information in order to provide contextually useful features to users, the need grows for consistency and flexibility in accessing that location information. Although individual providers are beginning to provide such frameworks (Apple's CoreLocation and Yahoo!'s Fire Eagle, for example), development of location services will ultimately benefit from a location metasystem.

Otherwise, the danger exists that developers will tend toward ad hoc solutions or informal standards in creating their services. Though tempting, this approach can unnecessarily compromise user privacy and duplicate effort in handling granularity.

This paper will describe what a location metasystem would look like and outline principles that location systems should follow to protect user privacy and resolve granularity issues. Finally, it will propose a way towards this metasystem using a simple analogy to the OAuth effort.

2. WHAT IS A METASYSTEM?
A metasystem is a system of systems. There may be multiple location systems, in fact, they're likely to proliferate: providers of location data (like Apple's iPhone), consumers of location data (any of the many multiplying location-aware services) and brokers of location data (like Fire Eagle). A metasystem is a set of principles and protocols – not a single physical or executable entity – for the interoperation of the many providers and consumers of location data.

One analog is the identity metasystem proposed by Microsoft in Kim Cameron's paper "The Laws of Identity" [1]. By outlining an architecture for how various identity systems can interoperate, the identity metasystem promises to allow all identity providers, relying parties (identity consumers) and subjects (users) of authenticated web services to work together using existing systems. Component systems may be developed by anyone; no single party has control.[6]

3. PRIVACY
In describing Microsoft's proposal for an identity metasystem, Cameron lists seven laws of identity, "fundamental principles to which any [...] identity architecture must conform". Many of those principles apply to location as well as identity. This is far from coincidental: in many ways, our location is a part of our identity and we feel hesitant to reveal it.

Location systems must only reveal a user's location information with the user's consent. We see this today in the dialogue box that pops up on your iPhone and the privacy settings in Fire Eagle. Providers already recognize that location information is personal information and that users feel betrayed if personal information is revealed without consent. (Compare to Cameron's "User Control and Consent".)

Furthermore, *location systems should reveal only the location information necessary.* Users should be able to reveal their city or neighborhood to a restaurant-finding system without revealing their precise location to the commercial vendor that they may not entirely trust. (Compare to Cameron's "Minimal Disclosure for a Constrained Use".)

This principle is not yet recognized by all location providers. For example, Apple's CoreLocation always reports the user's exact latitude and longitude, even when an application may have only required the city and even when the user might have preferred to reveal less about their location. Yahoo!'s Fire Eagle, on the other hand, provides a plethora of options: users can specify any of eight different levels of precision (no information, exact location, zip code, neighborhood, city, county, state, country) for each application that has access to their information.

The principle of minimal disclosure effectively limits revealing location information to a "need-to-know" basis. By requesting only the detail of information they require (rather than always requesting precise location), location consumers mitigate the risk to their users in the event of a security breach [1].

4. GRANULARITY
One challenge for developers of location-aware services is handling the granularity of location information. Users can update their location in any number of ways: they might type their current street address into a web form, automatically transmit GPS

http://doi.acm.org/10.1145/1507136.1507140

co-ordinates from their smartphone, or speak the name of their neighborhood over the phone.

Some developers handle this multiplicity by controlling the input directly. For example, a mapping site can ask for the street address in a structured web form on its own website and Google Maps can accept multiple formats in its generic search box (lat/long, street address, landmark names). But most location-aware service developers cannot dictate the granularity of location information they receive from users as more and more often the location-aware service gets the location automatically from a location provider, with little interaction from the user. And few developers want to duplicate Google's algorithms for evaluating the granularity of a location from unstructured text.

Relying on a particular system's APIs may fulfill a particular service's needs. For example, an application running on an iPhone obtains lat/long co-ordinates that are aggregated from a mix of GPS, cellular and WiFi inputs. But web services need not limit their users to a single software platform or a single way of obtaining location.

Furthermore, without a way of specifying granularity of location information, it is impossible to meaningfully fulfill the minimal disclosure principle from the previous section. By exclusively using exact co-ordinates, CoreLocation cannot adjust reporting based on a user's privacy concerns. A concept of granularity is not just handy for integrating different location services (though it certainly is that), but is also essential for protecting user privacy.

Location providers should specify all the levels of granularity of location information they are able to provide, and location consumers should specify all the levels of location information they are able to consume. This way, location providers and location consumers can be easily mixed and matched, neither requiring a particular location system, nor sacrificing the user's privacy.

Fire Eagle supports this principle by providing an API for location consumers to specify the level of granularity in their requests. This is a good start. But in order to realize the integration of different location systems, standards for these levels of location granularity will need to be decided upon and these hierarchies of location may not be simple. What qualifies as a neighborhood? (Some addresses may be thought of inside more than one.) And what about concepts of location not tied to precise geography (like "at home" or "on the bus")? [6]

Once these standards are defined, third parties may handle conversion between levels of granularity. Google's Geocoding service [4] and Urban Mapping's Neighborhood API [9] are already on the market.

5. WHY A METASYSTEM?

It might be argued that there's no need for an elaborate metasystem when a single location broker can fulfill these requirements for us. Yahoo!'s Fire Eagle does a commendable job of enabling (to some extent) all three proposed principles: consent, minimal disclosure and granularity specification.

But we shouldn't feel any more comfortable relying on a single broker and holder of location information for the World Wide Web than we were relying on Microsoft's Passport as a single holder of authentication and identity for the Web. Exclusively using Yahoo!'s implementation creates a single point of failure, trusts a corporate entity with a massive amount of personal information (and as a result makes Yahoo!'s servers a large and dangerous target) and doesn't allow for competition and differentiation of features.

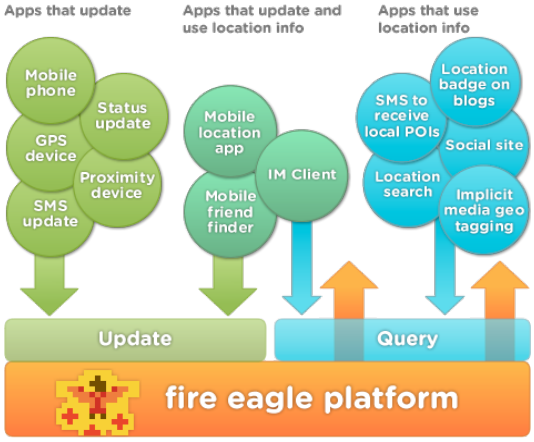

Figure 1 The Fire Eagle platform is a store and broker of location information. By holding this central position, Fire Eagle is able to keep control of permissions and handle formatting and granularity. [5]

On the other hand, an open set of protocols would let location service developers mix and match providers or brokers of information, take advantage of features from different vendors and switch between providers when one is shut down or temporarily unavailable.

6. PROPOSAL (BY ANALOGY): OLoc

The question remains how to implement a location metasystem. I won't try to duplicate here the important work being done by the W3C Geolocation Working Group (on standards for client-side interfaces for releasing location data to web applications[3]) or the IETF GeoPriv Working Group (on authorization requirements and a location format which includes privacy[2]). Ideally, a standardized set of protocols, file formats and privacy rules would be agreed upon by all the major players (Yahoo!, Google, Microsoft, Apple, et al.).

But such political resolutions are unlikely to happen quickly and may not be necessary. Although the Microsoft-described identity metasystem is deeper and more featureful, the OAuth project goes a long way towards effecting the metasystem's goals. Our location metasystem may be achieved in the same way.

6.1 What is OAuth and how does it work?

OAuth is an open protocol for delegating authentication.[8] Identity consumers can forward users to a service provider to gain access to some (but not all) pieces of data held by that service provider. This eliminates the insecure practice of users' providing their passwords for one service provider to a third-party consumer, while still enabling user data to be used in more than one place. The common analogy is giving a special "valet key" that will let the valet park your car, but won't let him unlock your glove compartment.

OAuth is an open standard and aspires to be used by many different web service consumers and providers. As such, it fulfills the laws of identity discussed above (consent and minimal disclosure) in a decentralized way.

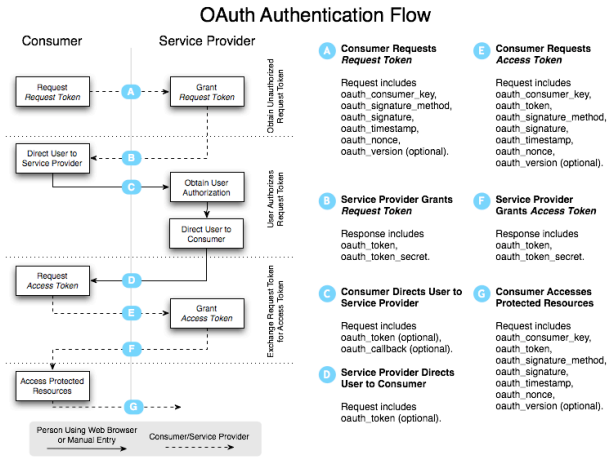

Figure 2 OAuth Authentication Flow. [7] Note that user authorization is completed entirely in the context of the Service Provider.

6.2 OLoc, an OAuth equivalent

I propose that similar progress towards the goal of a location metasystem could be made with an OAuth equivalent we might call OLoc. Such a protocol need not and should not specify how the location consumer or location provider are constructed. Instead, the OLoc protocol will specify how any identity consumer and identity provider can communicate such that the principles of consent, minimal disclosure and granularity are all satisfied.

Identity consumers should call into the service provider with an identifier for the user, the consumer's identity and a granularity level (potentially several) requested.

```
requestLocationAuth <user, consumer, granularity[]>
```

The location provider asks the user whether he is willing to reveal his location to that consumer and at what level of granularity. This consent might be obtained by redirection to a webpage (as in Fire Eagle) or a client-side dialog (as on the iPhone). If and only if access is granted, the location provider responds to the consumer.

```
requestGrantedCallback<user, granularity, token>
```

The response contains a token that the consumer can use to access location information for that user and specifies the granularity of location information that the user has allowed – knowing this in advance, the consumer can choose when to perform a query (a location-tracking service will request data much more often if it has address-level access rather than country-level access).

Making use of the token it received, the consumer can then request location data.

```
requestLocation<user, granularity[], token>
```

The consumer specifies the particular granularity it needs since the consumer may, in keeping with the "need-to-know" principle of minimal disclosure, only be requesting the state even though it has permission to access address-level information. Optionally, the consumer may specify an ordered list of granularities that it can accept: flexible consumers will make the best use of the location data that the user is willing to reveal and providers will know not to even send imprecise location to a strict consumer that won't make use of it.

```
returnDataCallback<user, granularity, location>
```

Finally, the provider returns the location data to the consumer, specifying what its granularity is so that the consumer can intelligently and gracefully make use of (or ignore) the data.

OLoc is not inherently a server-side or client-side protocol: the same basic structure (requesting authentication, granting authentication, requesting data, returning data) applies whether the provider and consumer are on the same device (as in the iPhone) or widely separated (as in Fire Eagle). The same parameters for authentication and granularity apply as well (although in a client-side case, it may not be necessary to specify a particular user).

This is only a brief outline of an OAuth-style protocol for location, which would necessarily be more complex if implemented. But this sketch shows how simple a protocol can fulfill the three proposed principles and be interoperable with component systems. Combined with a system for automated discovery of location systems (and their granularities), an OLoc protocol could lead to rapid development of interoperable, zero-configuration, location-aware web services.

7. CONCLUSION

These principles of consent, minimal disclosure and granularity can address problems with location services for both users and developers and speed both development and adoption of the location-aware Web.

How to build the protocols and standards of the metasystem remains an open question. Defining an unobtrusive user interface for consent and minimal disclosure will be one challenge; defining a flexible but useful standard for granularity will be another. But a simple, open protocol for requesting location information of a particular granularity may bring us towards our end.

8. REFERENCES

[1] Cameron, Kim. "The Laws of Identity". http://msdn.microsoft.com/en-us/library/ms996456.aspx. May 2005.

[2] "Geographic Location/Privacy (geopriv)". http://www.ietf.org/html.charters/geopriv-charter.html

[3] "Geolocation Working Group Charter". http://www.w3.org/2008/geolocation/charter/

[4] "Google Maps API – Geocoding". http://code.google.com/apis/maps/documentation/services.html#Geocoding

[5] "Introduction to Fire Eagle". http://fireeagle.yahoo.net/developer/documentation

[6] "Microsoft's Vision for an Identity Metasystem". http://msdn.microsoft.com/en-us/library/ms996422.aspx. May 2005.

[7] "OAuth Core 1.0". http://oauth.net/core/1.0/. Diagram by Todd Sieling.

[8] "OAuth: Introduction". http://oauth.net/about

[9] "Urbanware: Neighborhood Database". http://urbanmapping.com/urbanware/neighborhood-database

[10] Wilde, Erik and Martin Kofahl. "The Locative Web", First International Workshop on Location and the Web, Beijing, China, April 2008. http://dret.net/netdret/docs/wilde-locweb2008-locative-web.pdf