Dipartimento di Informatica e
Scienze dell'Informazione

Social Networks as a Support for Event Notification Services

G. Ciaccio, A. Pastorino, C. Ragnini, M. Ribaudo

DISI, Università di Genova, Italy

{ciaccio, ribaudo}@disi.unige.it

{antonio.pastorino, chiara.ragnini}@gmail.com

Abstract

Event notification based on the publish-subscribe paradigm provides a powerful abstraction for interactions based on a demand-supply pattern. As such, they might become the software backbone for new kinds of end-user services, especially tailored to the needs of people living in and around urban areas. In this report we propose the integration of a publish-subscribe system into popular social networks so that the matching of demand and supply concerning various aspects of individual and social needs can take advantage of the existing infrastructures and user communities characterizing the social networks themselves.

Keywords: Publish-subscribe, social networks, event notification, middleware

1   Introduction and motivation

The municipality of the city of Genoa (Italy) has recently started a number of different initiatives in cooperation with the local University, aimed at deploying new online services and refurbishing existing ones, while at the same time pushing for a pervasive diffusion of these services over the whole urban area. All these initiatives are collectively referred to as the *Genova Città Digitale* project.

Within the *Genova Città Digitale* project, the municipality is indeed fostering the diversity and social-awareness of the new interaction media provided by social networks and Web 2.0 technologies[1], in synergy with more traditional interaction channels based on web servers.

The focus of this report is on online services of use for people living a urban area, either permanent resident or short-term tourists. These services are not to be intended as necessarily related to the Public Administration (PA) of the municipality, although they may be possibly interfaced with and operated by it.

Let us start with an example. A key advantage of urban areas is their rich and diverse offer in terms of cultural or political events, shopping possibilities, and social activities in general. One (if not the main) parameter for evaluating the quality in an offer of cultural or social events is *diversity*. A rich offer is one that provides a large number of events, spanning a possibly wide range of people interests. Due to this very reason, the cultural and social offer in a urban area is often made of *niche events*. A niche event may be characterized by a narrow target or by a small budget organizer, and often by a combination of both. In the former case, the number of possibly interested people is small (e.g. poetry readings), thus traditional advertising systems are inappropriate for forwarding the information to the target. In the latter case, however, an event may have a potentially larger target but the organizational entity may not afford advertising it (think of the not-for-profit segment, or the numerous summertime events organized by small communities in the suburbs). Both situations prevent an otherwise valuable cultural offer from being effectively made known to urban residents and tourists.

To sum up, there exist a social and cultural "abundance" that is neglected because of a difficulty for organizers to inform people about the existence of events and services they are willing to provide. Such a difficulty in disseminating information is incurred not only by small private entities, but also by relevant public bodies acting in the urban area in support of citizens.

In our vision, this "selective advertising" problem is an instance of a general problem of matching demand and supply in an environment where the demand has a long-tail distribution over the supply spectrum.

To help solving this problem (actually an entire range of problems, all related to a demand-supply scheme) we propose the deployment of a *publish-subscribe* event notification infrastructure, to share user generated content in the form of advertisements. However, we do not intend to propose yet another separated web-based service offered to users upon registration. We instead propose to integrate the publish-subscribe functionality into existing

---

[1]The municipality has already opened different communication channels on the most popular social networks, see for instance `http://www.youtube.com/user/GenoaMunicipality`, `http://www.myspace.com/genova`, `http://www.facebook.com/comune.di.genova`.

and successful social networks (like Facebook and MySpace), already counting plenty of users, thus obtaining an innovative medium (and perhaps an unusual architecture, as we shall see) for service delivery.

It is necessary to point out that the envisaged system provides a kind and level of service that no social network currently support natively. It is indeed true that any social network already offers the possibility to "share" information with "friends", or invite "friends" to join "events" or "causes", ultimately spreading information across the social network. But this kind of dissemination requires manual intervention by the user in order for the information to be forwarded to others, with an expectedly huge impact on the diffusion speed. Moreover, it is the user herself that ultimately decides what information to propagate to whom, and, most often, she propagates everything to everyone or nothing to nobody, depending on her personal opinions or just current mood, with an expectedly very low selectivity and reliability (or, more formally, precision and recall).

This work reports on our experience in the framework sketched above, and presents the general ideas of a prototype system, embedded in a social network, whose main goal is to match demand against supply and proactively notify users in the context of cultural and social offers typical of a urban area.

The balance of this report is the following: Section 2 briefly introduces the building block of publish-subscribe systems and Section 3 sketches the high level architecture of the proposed system. Section 4 introduces the current prototype, and Section 5 highlights the points still open to investigation. Finally, Section 6 shows some screenshots of the running application within the context of Facebook.

2   Building blocks

The publish-subscribe paradigm is not a novelty in the field of middlewares for distributed systems. The first papers on this subject date back to a decade ago [1, 2, 4]. Publish-subscribe is a paradigm for interaction among entities in a distributed system. It is based on selective dissemination of messages called *events*, supplied by *publishers* and demanded by *subscribers*, subject to filtering criteria autonomously established by each subscriber. Relevant events are *notified* to each subscriber by way of a network of *brokers*, which know the filtering criteria and thus can route events conveniently. Event filtering can be more or less sophisticated, ranging from simple *thematic channels* to a more flexible *event tagging* to a true analysis of the message content. More recently, the physical (either actual or planned) position of the subscriber has been recognized as an important criterion for event filtering, giving rise to *location-aware* publish-subscribe services [6, 3, 5, 8].

A location-aware publish-subscribe service is the fundamental building block for selectively reaching people based on their current or planned position and expressed preferences and needs. But there are other important ingredients for an ubiquitous demand-supply facility to operate, namely:

- An *identification system* and a *reputation system*, both necessary for assessing the identity and reliability of the entity advertising an event or a service or goods.

- A pervasive wireless network, allowing services to be delivered through latest-generation handheld devices (smartphones and netbooks).

- A possibly transparent way for the handheld device to assess its current location (by way of GPS, GSM cell ID, Bluetooth, . . .), so that the user is allowed to perform location-sensitive requests in an easy way.

A location-aware publish-subscribe infrastructure might be of great help as a substrate for innovative online services. Let us consider again the problem of niche event advertising. Any user, either resident or just tourist, might set up or adjust her own event filter ahead of time, specifying location-related constraints (in addition to other constraints, e.g. time span or topics), and subsequently get advertised about interesting opportunities offered in the area of interest by other users or groups. A reputation system would enforce a degree of reliability, allowing users to assess the expected quality of the advertised events given the reputation of the organizer (as deduced by

user feedback concerning past events from the same organizer). Support to mobile users could bring added value to the service, enhancing its usefulness for nomadic users.

3   Requirements, challenges, and overall organization

The main challenge is to implement the envisaged publish-subscribe service within an already existing infrastructure that the average user has already registered with and is acquainted to, rather than forcing the user to subscribe to yet another web server. Social networking platforms are an obvious candidate. Following this approach, the social network becomes an interface (perhaps the preferred one) to the event notification service.

A subsequent challenge is then to find the most convenient way to exploit the mechanisms and primitives made available by the selected social networking platform, in order to obtain event dissemination without investing too much in a separate server infrastructure. Clearly, not all social networks are expected to be equally profitable in this respect.

An orthogonal challenge is to let the publish-subscribe system interoperate with pre-existing web servers, that follow a more traditional design yet are still operational. In our case there is an "old times" web site called the Genoa municipality Agenda[2]; it is reasonable to require that the most relevant and trusted contributions from the publish-subscribe system shall appear on the Agenda web servers upon approval by the administrator whereas all new entries fed in the Agenda database also feed the publish-subscribe system unconditionally.

The emerging overall picture is an information system to be fed by a number of distinct input channels, namely:

- the social network, operated by a vast community of users and not bound to any institution;

- the back-end of a web server operated by the PA; and

- other feeding systems, by way of suitable REST interfaces to the repository.

Moreover, the system has a number of distinct output channels feeding the community, namely:

- the social network itself, where the advertisements are selectively forwarded to interested users in a proactive fashion, possibly on handheld devices;

- the front-end of the web server operated by the PA, with traditional mechanisms like browsing and searching, and publishing only approved advertisements; and

- other broadcasting systems, e.g. content aggregators, by exporting the database content according to standard output formats (i.e., JSON, XML, RSS, Icalc).

Figure 1 describes the high level structure of the envisaged system. Pale grey blocks represent components of the pre-existing "Web 1.0" system maintained by the PA. White blocks are the new components, namely the publish-subscribe service with its brokers that are interfaced with the social network and interoperate with the older system.

Users may be either publishers of new events or subscribers interested in being selectively notified, or both. A PA moderator is in charge of marking some of the incoming events as "suitable for the PA web pages"; on the other hand, one or more PA operators may insert advertisements via the backend interface of the older system, according to the procedure currently adopted by the municipality (as of today, only PA operators are allowed to insert advertisements; all organizers of cultural events must send an email to the PA operator and ask for event insertion).

The white block labelled *content aggregator* represents any third-party entity interested in seeking and aggregating the information content collected in the repository; the content must of course be exported through open

---

[2]http://www.agenda-eventi.comune.genova.it/.
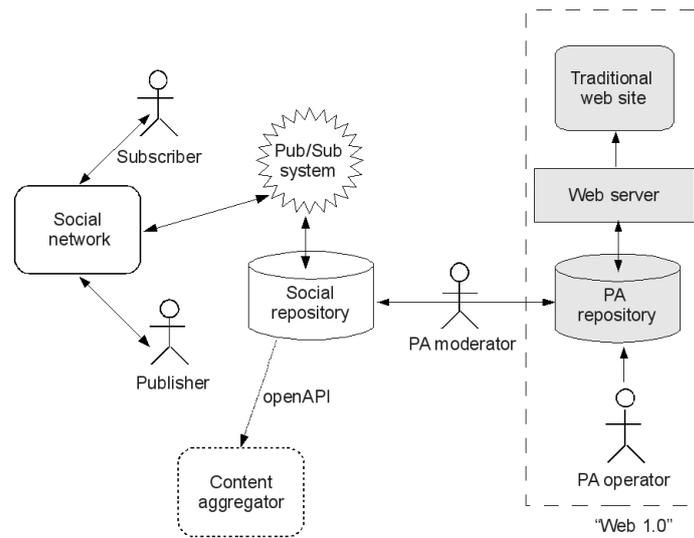
Figure 1. High level architecture.

APIs according to standard formats (e.g. JSON, XML, RSS, Icalc). The aggregator may even post this content back in the social networks, by exploiting advertising features of the social network themselves (for instance, Facebook events) that allow the information to further disseminate (although with poor performance and selectivity, as pointed out at the end of Section 1).

As a final remark, it is worth noting that this event notification infrastructure may well be leveraged by more than one social network at a time, sharing the same repository of events and broker structure. It is thus possible to propagate advertisements from one social network to another, with the broker structure behaving like a sort of "bridge" across distinct social networks. This form of cross-platform interoperability increases the size and diversity of the overall community of users, and a wider and more diverse community increases the usefulness of the notification service for each single participant.

4 The prototype

Event Forecaster is the name of the current prototype of a system that implements most of the functionalities sketched above. Currently, the publish-subscribe system has a single broker and a centralized storage where all outstanding advertisements are kept.

The broker is accessed through a web interface implemented in PHP, that allows one to insert events, set up own filters, edit or cancel a previously inserted event, and display all outstanding events. This part of the system leverages mostly standard technologies, mashed up with the Google Maps API for helping the user to associate the event with a geographic location. This part of the system, however, lacks proactivity: it cannot dispatch event notifications because there is no autonomous client that could be proactively contacted by the broker.

The proactive side of the system, namely, the event notification feature, relies on specific mechanisms offered by the social networking platforms. Most social networks (including MySpace and Facebook) allow user profiles to host small frames called *boxes*, that may contain code (HTML or Javascript or dialects of these) injected from external servers. The code injection is made possible through primitives offered by the social networking programming interface. Using these injection primitives, the broker can selectively inject event notifications inside the box of each user. Each box will look to the user as a sliding list of notifications that get updated on a regular basis.

4

The two sides of the system described so far look like belonging to different worlds. Indeed the publish and subscribe functionalities reside on a web server, whereas the event notification is performed within the social network (more exactly, within user boxes hosted by the social network servers). We need to connect these two sides, because of three reasons: (1) users must be provided with a single point of access to the publish-subscribe services as a whole; (2) the broker must link each event filter in its repository with the corresponding user identifier in the social network, in order to dispatch event notifications to the correct boxes; and, (3) the user must be allowed to explicitly authorize the broker to inject code into her own box.

The key feature that allows to unify everything consists of the possibility, provided by most social networks (including MySpace and Facebook), to allow for an external application (running on a server external to the social network) to be "adopted" by a user. A user adopting an external application will be able to access that application's web interface from within the social network, while simultaneously allowing the application itself to access her own user profile, update boxes and read personal data. In response to the adoption by the user, the social networking platform sends a message to the external application server, labelled by the user identifier in the social network; this identifier is caught by the external server and exploited as a form of user registration, and as the link between event filters (maintained at the external server) and notification boxes (managed by the social network servers).

In conclusion, a user willing to take advantage of our publish-subscribe facility has only to log in the social network and perform a one-time adoption in favour of the Event Forecaster application; from then on, she will be able to operate the Event Forecaster from within the social network without installing any software nor adding yet another URL to her browser's bookmarks.

As a final remark, it is worth noting that the Event Forecaster works regardless of the user having a lot of "friends" on the social network or not. Even a user with no "friends" at all can adopt the application, submit events, and get notifications for event advertised by other, unknown users.

5 Work in progress

The current prototype uses the social network as little more than a mere interface to the publish-subscribe system. Yet, we think this is a step in the right direction, as a new service is offered within a platform most users have already registered with.

However, in addition to being a mere interface, a social network might possibly act as selective *event disseminator* on its own. We are investigating on the extent to which the social network could act as a network of brokers and disseminate events autonomoulsy, rather than relying on an external broker infrastructure that has an additional cost. The publish-subscribe system would then take the general shape of a peer-to-peer system whose nodes are the individual profiles in the social network running some kind of Javascript code, with the goal of automating the "word of mouth" dissemination that is so effective in the field of information filtering [7] and in the human society in general.

The use of a social network offers the possibility of inferring user interests via profiling. Inferred interests might work as a complement to user-decided event filters (with the approval of the end user), and help delivering notifications in an even more selective and effective way.

Security is an often overlooked aspect of distributed systems, that is of main concern in our project. Security means, for instance, preventing spam (by blocking multiple insertions of the same advertisement, or blocking advertisements that are "similar" to an already inserted one and thus do not add useful information), avoiding anonymous insertion of advertisements (so as to discourage abuses), but also the very challenging goal of preserving user privacy (subscribing to given topics may indeed reveal information that the user might not want to disclose).

There are other issues that should be further investigated to allow an easier integration of the proposed idea within an "ecosystem" of other services. As an example, we plan to develop a REST-like public APIs for exporting

the user-generated advertisements, so as other services may be built following a mash-up approach.

Finally, it is quite clear that the proposed idea (an event notification service embedded within a social network) is not limited to advertising cultural events, but could rather be the basis for supporting a potentially very wide range of application domains, all sharing a common goal, namely, matching demand and supply in a social environment.

The interested reader may find the Event Forecaster at the following URLs:

- within Facebook:
  `http://apps.facebook.com/eventforecaster/` (user must be logged into Facebook and authorized by the developers)

- within MySpace:
  `http://profile.myspace.com/Modules/Applications/Pages/`
  `Canvas.aspx?appId=134122&appvers=dev` (user must be logged into MySpace and authorized by the developers)

6 Screenshots from the prototype

The Event Forecaster provides two distinct and non-overlapping user interfaces, namely:

- the **notification box**, appearing as a small box in the user profile within the social network. This box contains a sliding list of all notifications received by the user according to that user's event filter;

- the **management interface**, which can be reached (a) by clicking on the icon or link associated to the Event Forecaster (any external application, once adopted by a user, appears under that user profile as an icon or a link, depending on the specific social network), or (b) by clicking on one of the entries displayed in the notification box.

While the notification box only serves to subscribers for receiving event notifications, the management interface offers a number of different functionalities to both subscribers and publishers of events.

6.1 The notification box

Figure 2 shows a notification box of the Event Forecaster, framed within a Facebook user profile. Recall that the notification box is the place where a user receives notifications of events of interest (namely, events that match against the event filter of that user). The notification box get updated regularly with no intervention by the user. A vertical sliding bar allows the user to browse the entries in the list; each entry is a hyperlink pointing to a description of the corresponding event. Note that Facebook is just our running example; an analogous notification box has been implemented on MySpace as well.
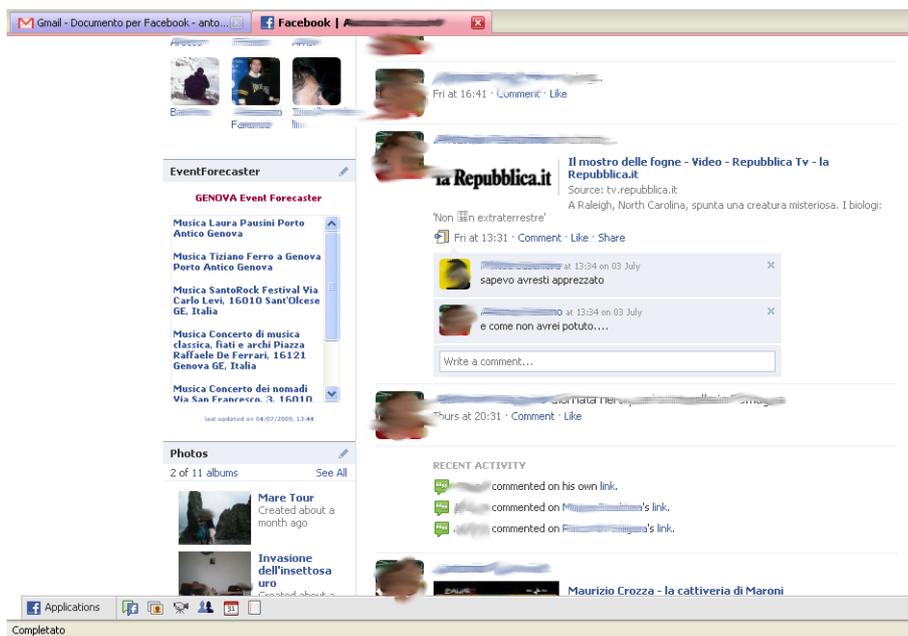
Figure 2. The notification box (middle of left column) in a Facebook user profile. Some notifications and the shift bar are shown.

6.2 The management interface

When accessing the management interface of the Event Forecaster (by clicking on either an entry in the notification box, or the icon or link associated to the application within the user profile), the user is shown a page containing a Google map with several pinpoints, each associated with an event of interest for the user (Figure 3) [3]. Pinpoints have different shapes, which recall the various thematic cathegories of the events they represent.

Figure 3. The home page of the Event Forecaster, offering the management interface.

---

[3]The map shows events occuring in the next 10 days, but the number of days can be easily changed.

The page includes a top banner with several links, corresponding to the *management functions* of the Event Forecaster. These management functions are the following:

1. **Tutti gli eventi**[4] (all events): displays all outstanding events with no filtering at all;

2. **Inserisci nuovo evento** (advertise a new event): allows a *registered* user to advertise a new event, so that interested subscribers get notified of it in their respective notification boxes;

3. **I miei eventi** (my advertisements): displays all outstanding events the user has advertised on her own so far, and allows the user herself to modify or cancel the corresponding advertisements;

4. **Imposta filtri** (set my filter): allows any user to set up a personal event filter, so as to get notified of only those events of personal interest;

5. **Mappa degli eventi** (map of events): displays events of interest for the user (hence filtered according to the user preferences), georeferenced on a Google map.

Let us now examine each of the functions in turn.

---

[4]The interface is currently in Italian; the multilingual version is left as a future extension.

1. Tutti gli eventi   By clicking on the link *Tutti gli eventi* (all events) the list of all outstanding events is displayed (Figure 4). All events are listed, regardless of the user filter.

Figure 4. List of all outstanding events, with no filtering.

By clicking on one of the listed events, the user is offered a detailed page providing a full description of the selected event (Figure 5). Events of the same type, occurring in the same area during the same period are also shown in the map by corresponding pinpoints.

Figure 5. Full description of an event.

2. Inserisci nuovo evento Each user, as a publisher in our publish-subscribe system, is allowed to advertise new events upon registration with the system (for safety reasons). The registered user can advertise events by using the input form of Figure 6, which becomes available following the link *Inserisci un evento* (advertise a new event) of the management interface.

The advertiser must specify name, place, time scheduling, cost, and other information concerning the event being posted. She has also to specify a thematic category (i.e., art, cinema, environment, science, sport, etc.) and kind (competition, exposition, protest, etc.), along with a number of optional free tags. All this information might in principle be useful for event matching towards each subscriber's filter; however, the current prototype implements only one filtering criterion, based on thematic category.

Figure 6. Advertising a new event.

3. I miei eventi By following the link *I miei eventi* (my advertisements), the user is provided with a list of the advertisements she had posted so far that are still outstanding. Figure 7 shows an instance of such a list. Since the user is the "owner" of those advertisements, here she is given the opportunity for editing or withdrawing each of the listed advertisements. A substantial change to an existing advertisement may trigger a re-notification of the event to interested subscribers; this is performed transparently by the underlying event notification service. This is a crucial point of the application that needs to be further investigated to avoid notifications spamming. A malicious user, in fact, could submit, withdraw, re-submit the same event again and again to continuously activate re-notifications. Mechanisms to recognize and avoid such an unfair behaviour are under investigation.

Figure 7. List of own advertisements, with editing and cancellation options.

4. Imposta filtri Each user, as a subscriber of the event notification service, is allowed to set up a personal event filter so as to be subsequently advertised by events of interest rather than spammed with every sort of advertisements. User filters can be set in the web page shown in Figure 8, which can be reached by following the link *Imposta filtri* (set my filter).

As already pointed out, the current prototype only allows filtering by thematic category, but this limitation can easily drop. The left box displays all the available categories, the right box diplays the categories currently selected by the user. The user filter can be changed at any time by the owner, and a new filter becomes effective immediately.
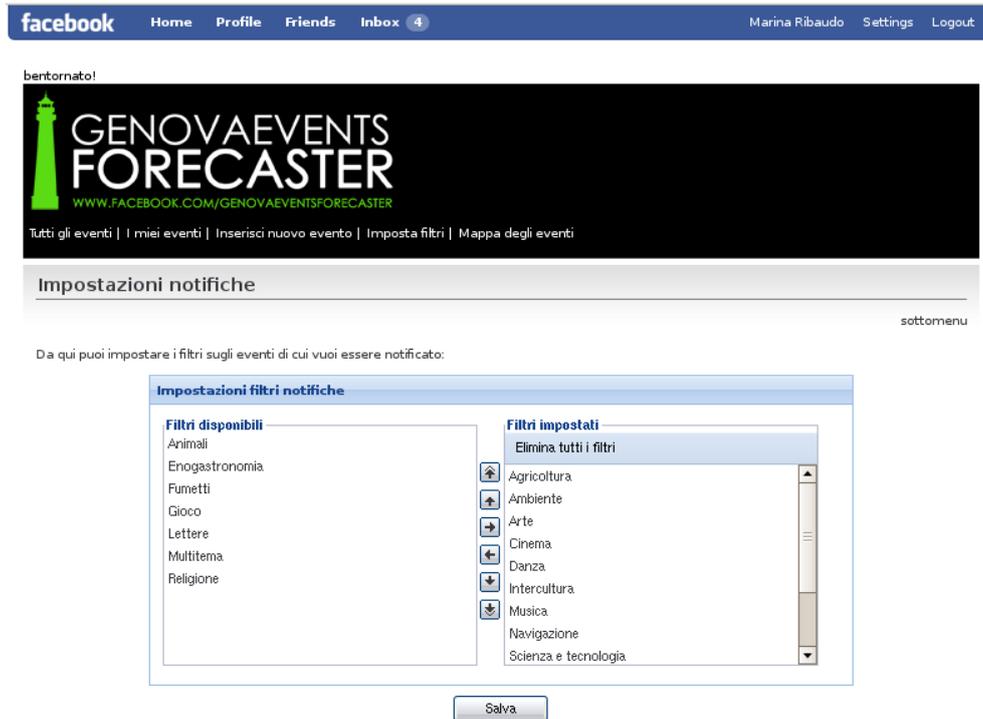

Figure 8. Adjusting own event filter for getting notified selectively.

5. Mappa degli eventi By following the link *Mappa degli eventi* (map of events), the user is offered a Google map where events of interest are represented as pinpoints. Pinpoints of various kinds represent the various thematic categories. By clicking on a pinpoint, the user is provided with a short description of the selected event (Figure 9). A further click on the short description leads to the detailed description of the event (see Figure 5).
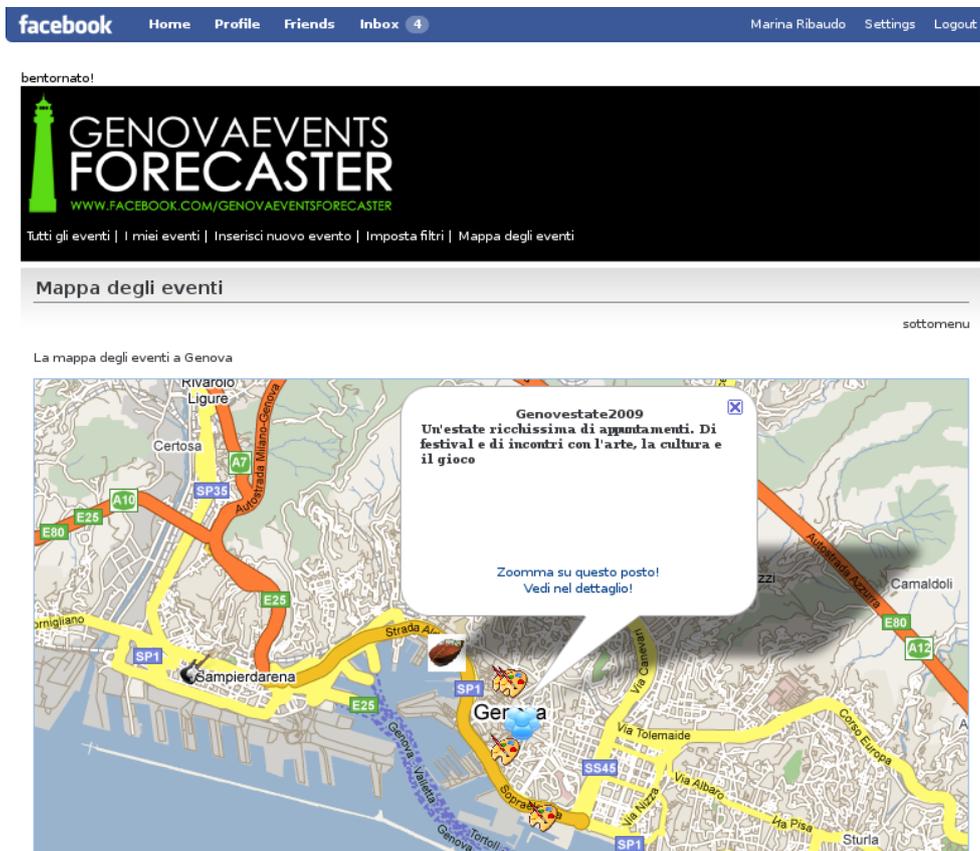

Figure 9. Maps with events of interest for the user.

References

[1] K. Birman. The process group approach to reliable distributed computing. *CACM*, 36(12):36–53, 1993.

[2] A. Carzaniga, D. Rosenblum, and A. Wolf. Design and evaluation of a wide-area event notification service. *ACM Trans. on Comput. Syst.*, 19(3):332–383, 2001.

[3] G. Cugola and J. E. M. de Cote. On introducing location awareness in publish-subscribe middleware. In *Proc. of the Int.l workshop on Distributed Event-Based Systems (ICDCSW/DEBS'05)*, June 2005.

[4] P. T. Eugster, P. A. Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[5] L. Fiege, F. Gartner, O. Kasten, and A. Zeidler. Supporting mobility in content-based publish/subscribe middleware. In *Proc. of the ACM/IFIP/USENIX International Middleware Conference*, June 2003.

[6] R. Meier and V. Cahill. Steam: Event-based middleware for wireless ad hoc networks. In *Proc. of the Int'l workshop on Distributed Event-Based Systems (ICDCSW/DEBS'02)*, June 2002.

[7] U. Shardanand and P. Maes. Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Proc. of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.

[8] T. Sivaharan, G. Blair, and G. Coulson. GREEN: A Configurable and Re-configurable Publish-Subscribe Middleware for Pervasive Computing. In *Proc. of International Symposium on Distributed Objects and Applications (DOA)*, Oct. 2005.