Friends Radar: Towards a private P2P location sharing platform

Rene Mayrhofer, Clemens Holzmann, and Romana Koprivec

Upper Austrian University of Applied Sciences, Campus Hagenberg
Softwarepark 11, A-4232 Hagenberg, Austria
{rene.mayrhofer,clemens.holzmann}@fh-hagenberg.at

Abstract. In this paper, we propose a new approach to live location sharing among a group of friends that does not rely on a central server as a hub for exchanging locations messages, but is done in a peer-to-peer manner by pushing *selective location updates* via XMPP messages. That is, only those contacts specifically authorized by users will be notified of location changes via push messages, but no third-party service is involved in processing these locations. We describe an initial implementation called *Friends Radar*, and discuss the associated trade-offs between privacy and resource consumption.

1 Motivation

One of the most often asked initial question upon establishing a telephone call from or to a mobile phone is "Where are you?". As communication becomes increasingly untethered and more dynamic, the mobility aspect transforms from a hindrance to a central element. That is, location becomes the subject of communication (communication about location) to support social interactions such as micro scheduling ("I am at the main entrance, where shall we meet?") or more dynamic time frames ("I will be there in 2 minutes"). It is therefore unsurprising that – driven by broad availability of smart phones with always-on connectivity and GPS – a multitude of services for live, global location sharing have been developed in the past few years to better support or even replace textual or voice communication about location.

Google Latitude is probably the most prominent live location sharing service, but Loopt, Facebook Places, Foursquare and many others[1] work with the same principle of sending location updates to central servers and then distributing these updates to lists of contacts from there. That is, all mobile clients log in to this server, send regular updates of their own location as determined by the built-in GPS receiver, and receive push notifications or poll for the locations of other mobile devices. The server manages user accounts, lists of "friend" accounts who may query one's location, and enforces access control based on these lists. And

[1] A non-comprehensive list can e.g. be found at http://bdnooz.com/lbsn-location-based-social-networking-links/.

as Wagner et al. [1] have shown, people are cautious about when, with whom and at which granularity they share location information in social networks.

Although the approach protects users' location traces from other, non-"friend" users of the same system, it places implicit and irrevocable trust on the server operators and the specific server implementation. From a privacy point of view, this should be unacceptable, as it gives complete control over highly private data, namely the mobile phone users' complete location traces, to unknown operators with whom no legally binding contract will typically be created. And even if the users use pseudonyms, they may be identified just by their location traces as as has been shown recently by Krumm [2]. However, due to lack of alternatives – and, often, lack of awareness –, many users currently accept this potential intrusion into their privacy for increased convenience.

We therefore propose a platform to enable live location sharing without relying on any central servers that all users are forced to trust with their private location data. This requires a decentralized but globally scalable architecture for message transmission, management of "friends" lists on the mobile devices themselves, and peer-to-peer (P2P) location updates initiated by the respective sender. In this paper, we make the following contributions:

- We present *Friends Radar*, a live location sharing application that builds upon the globally available, distributed XMPP[2] messaging architecture, and has so far been implemented for the Google Android (see screenshot in Figure 2) and Samsung bada platforms (section 3). Friends Radar allows live location updates to be shared via extended XMPP status messages among arbitrary groups of "friends" as defined by group membership in the XMPP roster (section 3.1). The locations of friends can be visualized in different views, including an augmented reality (AR) view with friend positions overlaid on a live camera image (section 3.2).
- By exploiting the fully decentralized nature of the global XMPP infrastructure, there is no single point at which these highly sensitive messages may be logged or intercepted. The privacy of Friends Radar therefore depends on the number of servers used. If every user has their own XMPP server, perfect privacy protection (the optimum for the use case of intentionally sharing location data) is achieved (section 4.1).
- We contrast the P2P to central server approaches and present an initial, quantitative analysis of the involved overhead of the privacy-friendly XMPP approach in terms of the number and size of messages on clients and servers (section 4.2).

2 Related Work

In contrast to our recently proposed privacy-sensitive, spatial group messaging platform Air-Writing [3], the Friends Radar system does not rely on a central server infrastructure with privacy guarantees designed into the communication

[2] http://www.xmpp.org

protocol, but on direct peer-to-peer messaging. This is also in contrast to e.g. the mix zones model [4], which requires new and currently not widely deployed infrastructure components to mix location queries. In Friends Radar, we only build upon already existing infrastructure (namely public XMPP servers) and therefore support immediate deployment of our approach for practical use cases. The VIS system [5] also relies on existing infrastructure in the form of virtual machine hosting to improve client privacy while our approach is based only on mobile clients without additional (virtual) servers. A peer-to-peer protocol specifically designed for social networks is PeerSoN[3], which is still subject of ongoing research and not yet available for download.

The ContextContacts application[4] built on top of the ContextPhone platform [6] comes closest to our approach, but focuses on a broader sharing of contextual data without providing the ease-of-use and added features of specialized location sharing applications, and its current implementation is limited to Symbian S60 phones. Virtual Compass [7] is also highly relevant to Friends Radar, as the authors propose a method to derive in-door localization maps based on different radio links such as Bluetooth and WiFi; it is an orthogonal extension to our current use of GPS to support in-door scenarios and we plan to include this localization technique in future versions of Friends Radar.

3 The *Friends Radar* system

Friends Radar is a client application running on mobile phones with current implementations for Android and bada and an iPhone version under development. Communication between multiple clients is done via XMPP messages with an extension for transmitting structured location data. We first discuss the specific protocol and then provide a brief description of the current user interface.

3.1 Protocol

The Extensible Messaging and Presence Protocol (XMPP) is an open, extensible, real-time communication protocol that can be used to build a wide range of applications. Currently, it is mainly used for instant messaging and voice-over-IP (VoIP) connection set-up, e.g. by social networking platforms including Google Talk and Facebook, collaborative services like Google Wave, and geo-presence systems like Nokia Ovi Contacts. It evolved out of the early XML streaming technology developed by the Jabber Open Source community and is now considered the most flexible – and the only officially standardized – protocol for exchanging real-time structured data. The protocol itself is extensible and can be used to transport arbitrary XML elements for structured data.

One of the main advantages is that XMPP uses a decentralized structure with the possibility of public, private, or federated servers. Although public servers

[3] http://peerson.net/
[4] http://www.cs.helsinki.fi/group/context/

with free registration are available, every group can easily set up their own server and therefore remain in control of their instant messaging data. Standard client-to-server and server-to-server connection security based on TLS and X.509 certificates is already widely deployed and the default for many servers.

XMPP servers are responsible for session management, routing (forwarding) messages, and contact list storage. That is, a list of "friends" along with their categorization in groups is stored in the so-called *roster* on the server. In Friends Radar, only one group is used to hold all friends to which the user wants to send their location updates. Users can add and remove their contacts to/from the "follow" group within the friends view. Clients connect to servers using their *JabberID* (JID, e.g. `username@ jabberserver.com/home`), which is composed of a user name part, the domain, and an optional *resource identifier* (forming a full JID) that can be used to distinguish multiple concurrent clients using the same account (the account without the resource id is also called the bare JID). When a client sends a message to one of the contacts registered at a different domain, then the client connects to its home server, which then connects directly to the respective server responsible for the contact without intermediate hops.

All client-server communications is handled via XML streams. The basic unit of communication used in XMPP is called a *stanza* with three possible types: message, presence, and iq (Info/Query). Each kind of stanza is routed differently by servers and handled differently by clients. The `<presence>` stanza is used for contact on-line notification, status changes and contact exchange in the form of a specialized publish-subscribe method. Contacts may request subscription to presence and, if authorized by the account holder, will automatically receive notifications about the account status.

In Friends Radar, standard chat messages can be sent to contacts in any of the views. However, the main use case in Friends Radar is to exchange the users' geographical locations. XMPP does not implement exchanging location as a core feature, but defines an XMPP protocol extension for transmitting the current geographical or physical location of an entity [8]. This includes a `<geoloc/>` element that is qualified with the `http://jabber.org/protocol/geoloc` namespace and which supports various child elements for describing location information of the contact with different data types (altitude, building, country, floor, longitude, latitude, postal code, region, room, etc.). For the purposes of defining user location we use just 3 attributes: longitude, latitude and altitude. The proposed way to exchange location information is to use publish-subscribe (PUBSUB) or the subset specified in the Personal Eventing Protocol (PEP). Because PUBSUB and PEP are also just extensions to XMPP and not all public servers support them (for example gmail.com does not), we instead attach a location extension to standard presence stanzas for interoperability with most XMPP server implementations (Fig. 1 shows an example). In each implementation, the respective platform-specific location listener automatically distributes presence stanzas with an attached location update when the location of the user changes. Users can control the frequency of these location updates by specifying a minimal time interval between two updates.

```
<presence id="TIMW6-9" to="user@jabberservice.com">
   <geoloc xmlns="http://jabber.org/protocol/geoloc">
     <lat>48.422005</lat>
     <lon>14.084095</lon>
   </geoloc>
</presence>
```

Fig. 1: Example presence stanza with XEP-0080 user location extension as used in Friends Radar

3.2 User Interface

The user interface of Friends Radar provides multiple views to the locations of "friends". The main view is an augmented reality (AR) view with the friends' positions overlaid on a live camera image as shown in Figure 2, which provides the users with an awareness of the direction of their friends with respect to their line of sight. In addition, the positions of friends are shown in a radar view similar to that of the InfoRadar [9] system on the upper right corner of the AR view. There are two more views, namely a map which displays the positions of friends as well as a simple list, and they can all be selected via buttons on the top of the screen. A distinct feature of all three views is the appearance of a menu upon clicking on a user's portrait, which allows to establish a phone call, send a text message or an email, start a chat or view the profile details, which can also be seen in the Figure 2. Separate setting windows allow to make privacy settings (e.g. to define who can see the own position), edit the radius within which friends are shown or set the rate of position updates, among others.

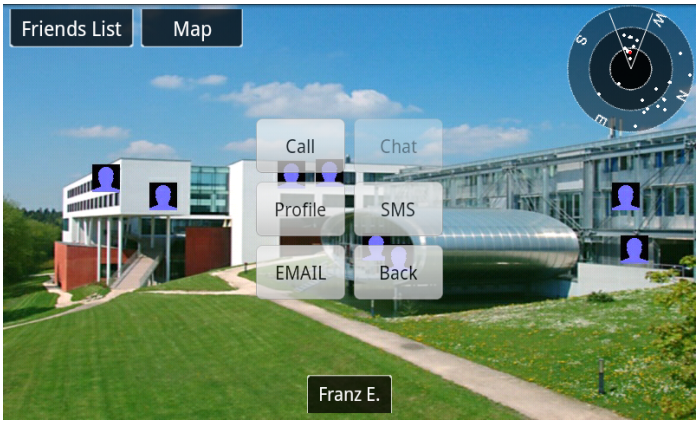

Fig. 2: Screenshot of the *Friends Radar* augmented reality view

4 Evaluation

Evaluation of Friends Radar as a system needs to cover three separate aspects: offered privacy guarantees, performance concerning network load (memory and CPU usage negligible on current smart phones), and usability. In this paper, we focus on the network aspect and therefore on privacy and performance analysis, while a future paper will more thoroughly analyze usability aspects of the different views.

4.1 Privacy

A general weakness of all friends finder type applications is that all "friends" are able to record and subsequently distribute the complete location traces they receive. An analysis of different privacy guarantees can therefore only assume other kinds of privacy threats involving third parties and explicitly exclude all contacts marked as friends by assuming them as trusted.

In all centralized approaches with caching and filtering of messages on a central server (cluster), there is an additional privacy threat that is more significant than the assumed-to-be-trusted group of friends: the server operators (which includes the administrators, company management, law enforcement, and other government bodies) have full access to *all* location traces of all users. An intentional or unintentional breach of privacy can therefore lead to large-scale profiling and detailed tracking of users, exposing their inherently private movements to unknown third parties.

Friends Radar reduces this threat significantly by eliminating this single point where all location traces are collected. The privacy guarantee offered by this decentralized approach scales directly with the number of XMPP servers (or with the percentage of direct peer-to-peer messages in contrast to using the XMPP server infrastructure, which is subject to future research). If every user runs their own XMPP server, which communicates directly and spontaneously with all other XMPP servers, we achieve the best possible privacy protection for this type of application: only the location traces of users who already marked potential attackers as a "friend" can be monitored (attackers can exploit this threat with social engineering, and this is therefore a common flaw of all social network based sharing approaches). On the other side of the spectrum, when all users connect to the same, central XMPP server (such as the services offered by Google, Facebook, or Nokia Ovi), the respective operators can again monitor and record all traces, making Friends Radar equivalent to a centralized model from a privacy point of view.

A more practical level of privacy can be achieved by running XMPP servers for coherent groups of users such as employees of a company, university students, activists in an NGO, or family members. Within a closed group, it is more likely that the server operators can be trusted not to violate user privacy. We therefore conclude that the level of privacy achieved by Friends Radar can be scaled according to user preferences by simply running private XMPP servers (which are easily set up and do not demand significant server resources).

4.2 Performance

To analyze the impact of decentralized messaging in terms of the number and size of messages in the worst-case scenario, we assume a fully-connected network of friends (i.e., each member of this group is a "friend" of all other members) of size n, in which each of the participants distributes their location with an average update frequency f (e.g. 3 updates/minute means that the current location is distributed every 20 s). We further assume the average message size to be s and be roughly constant (as it only contains the current location in terms of longitude and latitude, cf. section 3.1 for details).

In the best-case scenario (i.e., with only few selective location updates being distributed instead of a broadcast of each location to all members of the respective group), we assume an average number m of clients that offer "interesting" location updates from the point of view of each of the local clients. Filtering interesting updates can be done manually (each user selects who they want to track) or automatically (e.g. based on a maximum destination of friends, their current activities, or common goals). Although this number can vary significantly for each of the n clients in the friends network, we define m to be the mean of all interesting (or "tracked") friends from the point of view of each client. In the worst-case scenario, $m = n$. In the best-case scenario, m does not depend on n and is assumed to be asymptotically constant. This assumption is based on the consideration that each user will typically be interested in only a limited number of friends' location updates, independently of how many friends they have in their list.

We can now easily derive that, when using a centralized location server which handles all the location updates in terms of caching, filtering, and forwarding the update messages, each client will send $f \cdot s$ message bytes/s to the server (or, more typically, server cluster). The server will then distribute the received updates of the other group members to each client, adding an additional $(n-1) \cdot f \cdot s$ message bytes/s for each client. Therefore, each client sends or receives $n \cdot f \cdot s$ bytes/s. In a best-case analysis with centralized filtering of location updates, each client consequently sends or receives $(m+1) \cdot f \cdot s$ bytes/s, which can be a considerable improvement when compared to the worst case.

In the Friends Radar approach, there is no centralized location server that could perform the caching and filtering of messages. For this analysis, we also assume the optimal case in terms of privacy to better highlight the penalty in terms of message overhead. That is, we assume each user to run their own XMPP server, which communicate among each other[5]. Therefore, each combination of Friends Radar client and XMPP server must generate $(n-1) \cdot f \cdot s$ message bytes/s and send them directly to the respective recipients, causing each client to receive $(n-1) \cdot f \cdot s$ message bytes/s in turn. Overall, each client/server combination sends or receives $2(n-1) \cdot f \cdot s$ message bytes/s. However, our use of XMPP status messages (cf. section 3.1) allows each client to define its status message with a specific list of recipients of this status update and let the

[5] This assumption is unrealistic in practice, but allows us to give an upper bound on the message overhead.

Table 1: Summary of message overheads for worst-case (no filtering) and best-case (constant average number of friends each client is interested in)

	centralized location server	Friends Radar
worst-case per client	$n \cdot f \cdot s$	$2(n-1) \cdot f \cdot s$
worst-case all messages	$n^2 \cdot f \cdot s$	$n \cdot f \cdot s$ (P2P) $2n \cdot (n-1) \cdot f \cdot s$ (hub)
best-case per client	$(m+1) \cdot f \cdot s$	$2m \cdot f \cdot s$
best-case all messages	$n \cdot (m+1) \cdot f \cdot s$ $\Rightarrow O(n \cdot m)$	$2n \cdot m \cdot f \cdot s$ (hub) $\Rightarrow O(n \cdot m)$ (hub)

XMPP server generate and distribute the $n-1$ status update messages to each of the subscribers. The overhead for each client is therefore significantly smaller and is close to $f \cdot s$ message bytes/s with the overhead of attaching a recipient list of m XMPP user ids (n user ids in the worst-case scenario) to each status message. Because the decentralized model does not support centralized filtering of update messages, each client needs to decide locally to which other clients updates should be sent (assuming m for this analysis) and which updates to process (also assuming m in the best case when friend updates are co-ordinated and clients only send updates to those that will actually process them).

Considering the amount of messages in the whole network instead of a client-only point of view, the centralized case simply takes n times as many messages, resulting in a network load of $n^2 \cdot f \cdot s$ bytes/s. In the decentralized case, we need to distinguish two variations: in a fully connected peer-to-peer network (e.g. with XMPP link-local messages), we have only $n \cdot f \cdot s$ bytes/s, as each client generates one message for each update; in the XMPP infrastructure case, with XMPP servers (hubs) forwarding the messages, $2n \cdot (n-1) \cdot f \cdot s$ bytes/s are generated. For the best-case scenario, the centralized model consequently generates $n \cdot (m+1) \cdot f \cdot s$ bytes/s total messages, which is of order $O(n \cdot m)$ or $O(n)$ when m is assumed to be constant with regards to n. In the decentralized model, each client sends updates to and receives updates from all clients they are interested in, which is therefore of the same order $O(n \cdot m)$. However, taking into account the distinction between XMPP clients and XMPP servers when using status messages instead of push messages, the involved overhead can be reduced to approach the centralized case. Table 1 summarizes these analytical results.

5 Discussion and Future Outlook

In its current version, Friends Radar already supports private sharing of location among friends without relying on the trustworthiness of centralized services. It has been implemented for Android and as a prototype for Samsung bada and is currently available in a beta version from the Google Market. Next steps will include porting the Friends Radar application to other mobile platforms such as iPhone and Windows Phone 7 to make it available to a larger user base

and a more systematic analysis of the different views and their advantages and disadvantages in various use cases. On the protocol side, we intend to implement the option of using XMPP link-local messages [10] based on multicast DNS and DNS-SD. In terms of localization, we will evaluate different options to make Friends Radar usable when GPS is not available or not suitable, e.g. in large in-doors environments.

Acknowledgments

The Friends Radar application code on Android and bada was developed as part of a semester project at Upper Austria University of Applied Sciences by Georg David Bauer, Matthias Braun, Daniel Fuhry, Michael Jakubec, Romana Koprivec, and Andreas Weitlaner. We are grateful to an anonymous reviewer for pointing out a flaw in our decentralized best-case performance analysis.

References

1. Wagner, D., Lopez, M., Doria, A., Pavlyshak, I., Kostakos, V., Oakley, I., Spiliotopoulos, T.: Hide and seek: location sharing practices with social media. In: Proc. MobileHCI'10, ACM (2010) 55–58
2. Krumm, J.: Inference attacks on location tracks. In: Proc. Pervasive 2007: 5th International Conference on Pervasive Computing, Springer (2007) 127–143
3. Mayrhofer, R., Sommer, A., Saral, S.: Air-Writing: A platform for scalable, privacy-preserving, spatial group messaging. In: Proc. iiWAS2010, ACM (2010) 181–189
4. Beresford, A.R., Stajano, F.: Location privacy in pervasive computing. IEEE Pervasive Computing **2**(1) (2003) 46–55
5. Cáceres, R., Shakimov, A., Cox, L., Lim, H., Varshavsky, A.: Virtual individual servers as privacy-preserving proxies for mobile devices. In: Proc. MobiHeld 2009
6. Raento, M., Oulasvirta, A., Petit, R., Toivonen, H.: ContextPhone: A prototyping platform for context-aware mobile applications. IEEE Pervasive Computing (2005)
7. Banerjee, N., Agarwal, S., Bahl, P., Chandra, R., Wolman, A., Corner, M.: Virtual compass: Relative positioning to sense mobile social interactions. In: Proc. Pervasive 2010. Springer-Verlag (2010) 1–21
8. Hildebrand, J., Saint-Andre, P.: XEP-0080: User location. http://xmpp.org/extensions/xep-0080.html (September 2009) XMPP extensions standards track, *draft*.
9. Rantanen, M., Oulasvirta, A., Blom, J., Tiitta, S., Mäntylä, M.: Inforadar: group and public messaging in the mobile context. In: Proc. NordCHI 2004, ACM
10. Saint-Andre, P.: XEP-0174: Serverless messaging. http://xmpp.org/extensions/xep-0174.html (November 2008) XMPP extensions standards track, *final*.