# Location-based Timely Cooperation over Social Private Network

Youna Jung, Renato Figueiredo, and José Fortes
Advanced Computing and Information Systems (ACIS) Laboratory
Department of Electrical and Computer Engineering, University of Florida
Gainesville, Florida, United States
{younajung, renato, fortes}@acis.ufl.edu

*Abstract*— **The increasing use of online social networks (OSNs) in emergency situations shows us a promising future of human cooperation through OSNs. Despite this intense interest, a number of fundamental limitations still exist, such as lack of appropriate conceptual models and limitations on cooperation methods and shareable resources. To address existing limitations, we propose Whistle – a cooperation framework for OSN users which can dynamically organize an emergency community with nearby users and guarantee unrestricted cooperation and resource sharing by leveraging the Jitsi communicator and the SocialVPN. To test the feasibility and applicability of Whistle, we present an implemented prototype and demonstrate its applicability to an example use case.**

***Keywords- location-based cooperation; online social network; cooperation framework; temporary virtual private network; provacy protection***

## I. INTRODUCTION

When people face an emergency admitting no delay, such as a child missing, a hit-and-run, or a medical emergency, help from nearby people is important – but must be prompt and organized to be effective. For example, in the case of fire or natural disasters such as earthquake or flooding, it would be required to quickly inform people in the disaster area about an emergency situation and give them vital information such as evacuation routes, place of life jackets or fire extinguishers. In the case of a child missing or a hit-and-run, quick and wide dissemination of information to nearby people about a lost child or a car that left an accident scene would be helpful in finding the child or car. In a medical emergency, immediate help from nearby people and/or medical experts is key to save a patient's life. To address these issues, emergency response systems need to satisfy three core requirements: 1) Real-time location-based discovery of nearby helpers and persons in need of help, 2) On-demand organization of an emergency community with essential members, and 3) Efficient and secure cooperation methods.

Emerging online social networks (OSNs) have great potential to meet those requirements. First, they have a large number of users geographically distributed. For example, Facebook is the largest OSN and has 1.4 billion of users worldwide, i.e. 11% of people on Earth [1]. This motivates our focus on Facebook as a first step in this paper – Whistle is generalizable to other social networks, and a Facebook-based prototype implementation demonstrates the feasibility of the approach for a large, representative OSN. Second, OSNs provide access to user contexts, including location contexts. OSN users spontaneously disclose and update their personal information to establish and maintain social relationships. With proper permission granted by users, an application can easily obtain user contexts through the OSN's APIs.

In past years, many real use cases have proven the potential of OSNs as an infrastructure for human cooperation. OSNs have played an important role in emergency situations, not only as an alternative media that collects and spreads useful information, but also as a basis for gathering people and enabling cooperative communication amongst them. Through Twitter, people have found their lost pets by broadcasting information [2, 3] as well as coped with a medical emergency by quickly contacting paramedics or medical doctors [4]. In the case of a natural disaster, such as Hurricane Irene or the 2011 tsunami in Japan, people actively shared news about the disaster and communicated with their family and friends through OSNs, while much infrastructure was destroyed [5, 6].

However, cooperation using OSNs is still in its infancy due to the lack of key mechanisms: searching eligible users among very large numbers of users at request time, forming a well-organized group, orchestrating cooperation between members, supporting intuitive and rich cooperation methods, and protecting user privacy. Such limitations lower efficiency, reduce the scope of applicable domains, and cause people to hesitate to ask and/or give help through OSNs [8]. To address these issues, Jung et al. [9] proposed the role-based community model, the situation-based cooperation model, and the community-centric property based access control model (CPBAC). However, some problems, such as lack of context model to represent user contexts and limitation of sharable resources and cooperation methods, still remain unsolved.

In this paper, we propose a cooperation framework called Whistle that has its own location model and management mechanism for location contexts, in addition to previously proposed models in [9]. Furthermore, Whistle leverages Jitsi [10] and SocialVPN [11] to guarantee unrestricted and independent cooperation. In this framework, the cooperation service consists of two phases: the bootstrapping phase and the operation phase, as shown in Figure 1. In the first phase, Facebook provides contexts of users who register with the Whistle service – which is a service external to the OSN. This step allows Whistle to build a user pool with contexts. When an

COLLABORATECOM 2014, October 22-25, Miami, United States

DOI 10.4108/icst.collaboratecom.2014.257321

emergency arises, the operation phase commences; Whistle dynamically organizes a community with eligible users nearby, and enables users to cooperate with each other using diverse cooperation methods of Jitsi atop of a virtual private network that is dynamically established for the community.

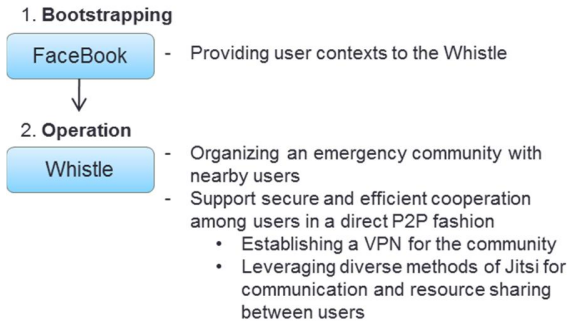


Figure 1. Two phases of a cooperation process using Whistle

The rest of the paper is organized as follows. In Section II, we overview preliminary work and identify its limitations. In Section III, we propose Whistle, a cooperation framework for OSN users; in particular, we propose the ontology-based location model of Whistle and user context management scheme in details. In Section IV, we demonstrate a Facebook-based prototype of Whistle with an example use case scenario (finding a lost child). In Section V, we discuss related work. We present conclusions and future work in Section VI.

## II. PRELIMINARY WORK AND CONTRIBUTIONS OF THIS PAPER

Before proceeding, we introduce our preliminary work, the SeCON App. The SeCON App [9] is a Facebook application that supports secure cooperation among Facebook users by employing three conceptual models: the role-based community model, the situation-based cooperation model, and the community-centric property based access control (CPBAC) model. When a request is received, it finds eligible users by exactly matching user contexts with eligibility conditions without context models, and then creates a community by assigning users to a certain community role based on a community model. Once a community is created, the App creates a community page in Facebook and orchestrates cooperation according to the situation-based cooperation model. During cooperation, all information and resources must be shared through Facebook. Thus, sharable resources need to be uploaded to the community page and all communications between members need to take place within the page. Although the SeCON App has improved efficiency of human cooperation through OSNs, some limiting issues still remain unsolved:

*1) Lack of context model:* The SeCON App stores user contexts received from Facebook in flat tables without using standard terms and structural relationships among context values. Since Facebook does not define standard terms and restricts formats and styles for users' profile data, the SeCON App may have many different context values having the same meaning (e.g. 'University of Florida', 'UF', and 'U. of Florida'). In case of conducting exact matching, such naïve context handling leads to incomplete search on user contexts. Furthermore, absence of semantic correlation between context values is another issue. Some contexts, such as addresses or organizations, have their own structure. For example, there exists an *inclusion* relationship between the 'University of Florida' and 'Gainesville' because 'University of Florida' is located in the 'Gainesville' city. It is hence difficult to expect effective user search based on contexts without standard terms and context models to represent semantics of contexts.

*Challenge*: We need appropriate models to represent user contexts and search users based on contexts efficiently.

*Contribution*: To address the challenge of context modeling, we first propose an ontology-based location model that uses toponyms which have become 'de facto' standard in the Internet (in particular, they are used by Google), and a maintenance scheme to create and update location contexts with reduced user intervention. For efficient user search based on location contexts, we also propose a two-step user searching algorithm.

*2) Limitation of shareable resources:* The SeCON App enables members to temporarily share resources stored in Facebook; however, no outside resources can be shared. This limitation is a major weakness if an important resource is not stored in Facebook at cooperation time. It limits its practical use if it requires users to upload resources to Facebook in an emergency. Furthermore, Facebook allows only limited file types, such as image and video files, to be shared.

*Challenge*: Users need to be able to share resources regardless of resources types and locations.

*Contribution*: We propose a way to directly access users' external resources stored in personal devices (and cloud resources) in a peer-to-peer fashion by dynamically establishing a virtual private network that inter-connects community members in real time.

*3) OSN-dependent communication and resource sharing:* During cooperation, all communication and resource sharing must take place through an OSN. Although there are many other rich methods of user cooperation, such as file transfer, remote file access, audio/video conferencing, and multimedia streaming, there is no integrated way to use them through an OSN. Furthermore, there is no way to cooperate with social users logged in other OSNs.

*Challenge*: We need a communication and sharing method that is not restricted to a single OSN.

*Contribution*: Whistle leverages Jitsi to allow users distributed in different OSNs to communicate and share resources in diverse and rich ways, in a peer-to-peer fashion, independently of services provided by OSNs.

## III. COOPERATION FRAMEWORK FOR FACEBOOK USERS

Whistle is a cooperation framework implemented by a centralized trusted server, an OSN application, and clients at user devices. For better understanding of Whistle, we describe its functionalities, architecture, and cooperation flow in this section.

### A. Functionalities

*1) Creation and maintenance of a user pool with structured location contexts*

Whistle creates and maintains a user pool in which a user is represented as a set of user contexts. We define context as information that can be used to characterize the situation of a user, such as *gender* context, *age* context, and *current location* context. Among various user contexts, in this paper, Whistle focuses on location contexts to provide location-based services. A detailed explanation about how to obtain, represent, and update location contexts follows:

*a) Consent-based semi-automatic context acquisition*: Whistle obtains user contexts from two different sources: OSNs and users. It can automatically fetch a user's profile in an OSN with user consent (*OSN-provided context*). By invoking OSN's API (e.g. Facebook's *Graph* API [12]) through the HTTP GET method, Whistle gets a variety of user contexts: *name*, *age*, *bio*, *birthday*, *email*, *gender*, *languages*, *relationship* (*marriage*) *status*, *current location*, *education*, *work*, and etc. When location contexts (such as *current location*, *work*, and *education*) are null, Whistle asks a user to directly enter location contexts in the registration stage with the Whistle service (*User-provided context*).

Whistle adds a new user to its user pool, whenever a user registers in Whistle. In the pool, each user is represented as a set of contexts. Whistle uses an ontology-based location model to represent and search location contexts. For the remaining contexts, we use a key-value context model [13, 14] which allows exact matching retrieval only and the context models for other contexts are left for future work. The location contexts obtained from the location model are stored in the Whistle server.

To track changes in user contexts, Whistle itself has user accounts on the OSN and becomes a friend of users so that it is notified whenever users change their contexts in the OSN. In the prototype of Whistle, we create an account in Facebook. Note that currently Facebook allows a user to have a maximum of 5000 friends. For scalability, Whistle thus should have multiple accounts in Facebook (or negotiate with Facebook to lift the 5000 friend limit). Alternatively, Whistle can periodically fetch contexts of users and update contexts if changed.

*b) Whistle location ontology:* For effective representation and management of location contexts, we propose an ontology-based location model. Although there are many existing location models, as discussed in Section V, we develop a new location model because most existing models are too heavy for Whistle.

Among diverse context models, we choose an ontology-based model due to its expression power and the powerful techniques available for reasoning and validation [15]. The Whistle location ontology defined in OWL 2 [16] represents a location with three types of information: 1) Geometric information, represented by the GPS Coordinates class, 2) Appellation information, represented by the Appellation class, and 3) Administrative information, represented by the Postal Address class. The graphical representation of the Whistle location ontology is shown in Figure 2.

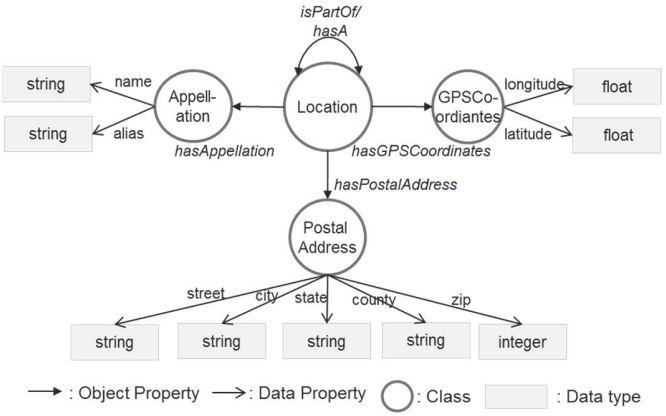


Figure 2. The Whistle Location Ontology

The *GPS Coordinates* class has two data properties: *longitude* and *latitude*. The *Appellation* class has two data properties: *name* and *alias*. The *name* data property represents a standard toponym, while the *alias* data property represents its alternative names. Whistle uses Google's toponyms as a 'de facto' standard. For example, Google uses 'University of Florida' as the standard toponym of the University of Florida. Accordingly, the 'University of Florida' is saved as the value of a *name* property, and 'UF' and 'U. of Florida' are saved as the value of an *alias* property in the Whistle location ontology. The *Postal Address* class has five data properties: *street*, *city*, *state*, *county*, and *zip* (note that the current prototype implementation assumes only addresses in the United States). An *inclusion* relationship between locations is expressed by the *hasA* object property, an inverse property of the *isPartOf* object property. The *inclusion* relationships are used to infer GPS coordinates of locations that are too fragmented, or unknown. For example, if Whistle does not know the precise GPS coordinates of an office in a building, it can assign the GPS coordinates of the building instead.

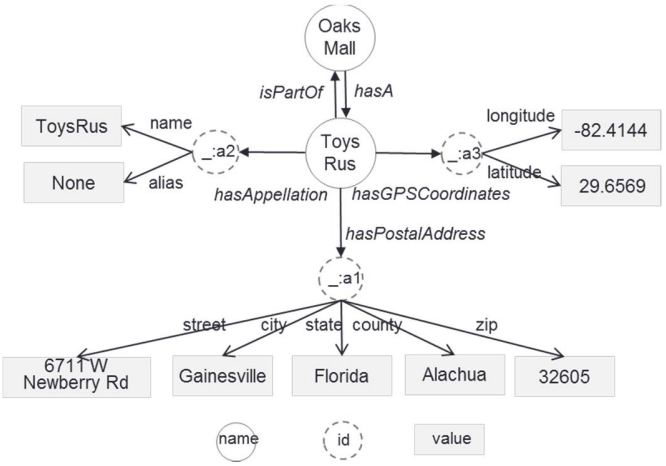


Figure 3. An example location for *ToysRus* in the *Oaks Mall* in Gainesvill, Florida

As an example, we present a location representing the *ToysRus* store in the *Oaks Mall* in *Gainesville*, *Florida* in Figure 3. The *ToysRus* location is a part of the *Oaks Mall* location and has three object properties represented as identifier: '_:a1', '_:a2', and '_:a3'. Its standard toponym captured by the *name* property is 'ToysRus' and its GPS coordinate is expressed by the *longitude* value '-82.4144' and the *latitude* value '29.6569'. Its administrative information is represented by corresponding postal address with the *street* property '6711 W Newberry Rd', the *city* property 'Gainesville', the *state* property 'Florida', the *county* property 'Alachua', and the *zip* property '32605'. The OWL 2 specification of the *ToysRus* location is presented in TABLE 1.

TABLE I. A FORMAL REPRESENTATION OF THE *TOYSRUS* LOCATION USING OWL 2 FUNCTIONAL SYNTAX STYLE

```
Declaration( NamedIndividual ( :OaksMall) )
Declaration( NamedIndividual ( :ToysRUs) )
ClassAssertion( :Location :OaksMall)
ClassAssertion( :Location :ToysRUs)
ObjectPropertyAssertion( :isPartOf :ToysRUs :OaksMall )
ObjectPropertyAssertion( :hasPostalAddress :ToysRUs _:a1 )
DataPropertyAssertion( :street _:a1"6711 W Newberry
                       Road"^^xsd:string )
DataPropertyAssertion( :city _:a1 "Gainesville"^^xsd:string )
DataPropertyAssertion( :county _:a1 "Alachua"^^xsd:string )
DataPropertyAssertion( :state _:a1 "Florida"^^xsd:string )
DataPropertyAssertion( :zip _:a1 "32605"^^xsd:integer)
ObjectPropertyAssertion( :hasAppellation :ToysRUs _:a2 )
DataPropertyAssertion( :name _:a2 "Toys R Us"^^xsd:string )
ObjectPropertyAssertion( :hasGPSCoordinates :ToysRUs _:a3 )
DataPropertyAssertion( :latitude _:a3 "29.6569"^^xsd:float )
DataPropertyAssertion( :longitude _:a3 "-82.4144"^^xsd:float )
```

Since OSNs and users give only Appellation information, Whistle needs to derive the corresponding Geometric and Administrative information. This can be implemented by using services such as the Google *geocode* API [17]. Whenever receiving location contexts, Whistle retrieves the corresponding address and GPS coordinate values of the location from the service, and then stores a complete location context. Once Whistle stores a user's location contexts, it monitors changes in user contexts and keeps the contexts up to date by establishing friendships between Whistle and users. For example, Whistle is able to gather updates on friends' contexts without extra effort by using Facebook's *Graph* API [12]. More details of the context update process are given in Section III.B.

*2) On-demand secure cooperation among nearby users*

Whistle dynamically creates a community for an emergency with only eligible users nearby at request time, such that a user in danger can receive help immediately. To do so, it performs user search based on contexts, establishes a temporary virtual private network between cooperators, and launches a Jitsi communicator to enable them to cooperate with each other securely.

*a) Two-step location based user search :* To find out the nearest users, Whistle performs a two-step search on *location* contexts, which includes the *static location search* step and the *dynamic location search* step. The *static location search* step is to find out potential candidates who are most likely to be close to a target location based on stored location contexts. To do this, Whistle first calculates the minimum number of required members ($mem_{min}$) by adding up roles' minimum cardinalities defined in corresponding community template and sets a radius ($r$) of a search range. It then picks out potential candidates who are associated with locations within the range (*static location search*, steps 7 and 8 in TABLE II). At this time, if the number of retrieved users is not enough, Whistle expands the search range until it finds sufficient candidates. Once it determines a set of potential candidates, it performs *dynamic location search* (step 9 in TABLE II). The goal of this step is to exclude unqualified candidates who are not close to the target location at execution time by checking their current location in real time. Whistle then finalizes a set of nearby candidates. At this time, the distance between two GPS points is calculated by *Haversine Formular* [18]. The algorithm of location-based user search is specified in TABLE II. By conducting the two-step user search, Whistle can significantly reduce the number of users who need to be checked for real-time locations (i.e. Whistle does not track users' locations). However, it cannot find nearby users whose location contexts stored in Whistle are not associated with the target location.

TABLE II. ALGORITHM FOR LOCATION-BASED USER SEARCH

1. $l_t$ = name of target location
2. $mem_{min}$ = minum number of required members
3. $r$ = radius of search range
4. $l_{db}$ = location database
5. $Pf_{db}$ = profile database
6. $nl_{min}$ = minimum number of nearby location
7. *for* each user $u$ in $Pf_{db}$
   a. if $\exists\ u_{locations} \equiv l_t$ then $pcand \leftarrow u$
8. *while* $sizeof\ (pcand) < mem_{min}$
   a. $nl = get\_nearby\_location(l_t,\ nl,\ nl_{min})$
      // This function returns nearby locations that that excluded in existing nl.
   b. for each user $(u \notin pcand)$in $Pf_{db}$ and for each location $l_n$ in $nl$
      i. if $\exists\ u_{locations} \equiv l_n$ then $pcand \leftarrow u$
      *ii.* if $sizeof\ (pcand) \geq mem_{min}$ then *break*
   c. increase $nl_{min}$
9. *for* each user $u$ in $pcand$
   a. $u_{\text{gps}}$ = GPS coordinates of $u$, $l_{t\,\text{gps}}$ = GPS coordinates of $l_t$
   b. $d$ = distance$(l_{t\,\text{gps}}, u_{gps})$
      // calculated by using the Haversine formula
   c. if $d \leq r$ then $candidate \leftarrow u$
10. return $candidate$

*b) On-demand organization of emergency community:* To organize a well-structured community, Whistle employs the role-based community model [9] in which a community is formed with eligible users who take one or more roles. After determining a set of candidates, Whistle sends an invitation to each candidate via preferred contact method and assigns one (or more) roles to available candidates who accept the invitation. If the number of available candidates is less than

the required minimum members, then it conducts the two-step location-based user search again with an expanded search range to secure more candidates.

*3) Setting up temporary SocialVPN connecting members*

Although the SeCON App enables users to receive community services from most suitable users regardless of previously established friendships, its cooperation method is totally dependent on an OSN, Facebook. The App does not allow cooperating with users in other OSNs. External resources that are not uploaded on Facebook and external services (e.g. camera streaming) cannot be used during cooperation, no matter how important they are. Even though a conversation or resource is private, there is no simple way to eliminate interference of OSNs in the middle. Furthermore, most OSNs limit sharable resources to only a few types (e.g. profile data, short messages, and photo/video files) and do not allow sharing of other types of resources such as word or pdf files.

To overcome this limitation, Whistle enables cooperating members to communicate with each other directly in a peer-to-peer fashion, and share external resources that are stored in their personal devices or cloud, regardless of resource types and OSNs that members use. Towards this, Whistle leverages the social virtual private network (SocialVPN) [11], a networking approach that aims at bridging the gap between social networking and overlay networking. It is able to automatically establish direct peer-to-peer Layer 3 network links between social friends, and then allows secure communication between them using PKI-based encryption. SocialVPN allows users to utilize TCP/IP legacy software (for example, Jitsi Communicator, SSH, VNC, and RDP for remote access, VLC and iTunes for media streaming, and NFS and SAMBA for remote file access). By establishing SocialVPN connections, members can be directly and securely connected, while using diverse existing software. Whistle establishes SocialVPN connections as soon as a community is created, and then removes them when the community is dissolved. Thus, the connectivity is ephemeral; the SocialVPN temporarily exists only during cooperation.

*4) Rich communication and resource sharing through Jitsi*

Whistle allows members to use more diverse and rich services for communication and resource sharing (not limited to OSN-provided services) by leveraging Jitsi Communicator, formerly known as 'SIP Communicator'. Jitsi [10] is an open source multimedia communicator that enables users to communicate with remote social friends via various methods such as text messaging, audio/video conferencing, file transfer and desktop streaming. It works on most major operating systems such as Windows, Mac OS, Linux, and other Unix-like systems; it recently started to support Android so that mobile users can also use Jitsi. By utilizing Jitsi atop SocialVPN, Whistle supports direct device-to-device communication and guarantees more effective and unconstrained cooperation compared to cooperation through an OSN.

### B. Architecture

Whistle consists of a centralized server, an OSN App, and a number of clients that are connected through the Internet. The Whistle server and the Whistle App are always connected and interact with each other. Whistle clients are dynamically connected and disconnected through SocialVPN. The architecture of Whistle is illustrated in Figure 4.

*1) Whistle Server*

The Whistle server is a trusted party that complies with laws and regulations relevant to privacy protection and consists of four major components: the *User Manager*, the *Context Manager*, the *Community Manager*, and the *SocialVPN Manager*.

- *User Manager* with *XMPP server* – The main task of this component is to handle user registration and maintain user accounts by interacting with the Whistle XMPP server. The eXtensible Messaging and Presence Protocol (XMPP) is an XML-based open source instant messaging protocol, and an XMPP server provides basic messaging, presence, and XML routing features. Whistle has its own XMPP server to establish and manage the (temporary) community membership information needed to bootstrap SocialVPN connections. The *User Manager* creates an XMPP account using a user's OSN account when the user registers and then shares the XMPP account with the *Community Manager* and the *Context Manager*.
- *Context Manager* – This component obtains and manages user contexts and, if requested, searches eligible users whose contexts satisfy eligibility conditions. With a user's OSN account information received from the *User Manager*, the *Context Manager* fetches various user contexts from an OSN. For location contexts, it invokes the Google *geocode* API with the place names retrieved from the OSN to get necessary information, generates complete locations according to the Whistle location ontology model, and stores them in the *Location Database*. The *Context Manager* periodically receives updated user contexts from the Whistle App and updates user contexts in the *Context Repository*. To find eligible users, it sends information about target location to the *Location Engine* so that the engine conducts the two-step search on location contexts. Subsequently, it receives necessary information about a target location and a community template from the *Community Manager*.
- *Community Manager* with *Template Repository* – This component aims to organize a community with most suitable users. When receiving a request through the user interface, it delivers required information specified in a corresponding template to the *Context Manager*. A community template includes information about necessary roles and user-role assignment rules specifying eligibility rules and cardinalities [9]. If the *Context Manager* returns a set of candidates, it checks availabilities of candidates and creates a member list with only available users.
- *SocialVPN Manager* – The goal of this component is to dynamically create SocialVPN configuration files for members so that member clients can automatically establish SocialVPN connections among them. To do so, it gets members' XMPP accounts from the *Community Manager*, generates virtual IP addresses for members, and then distributes the generated configurations to member clients.

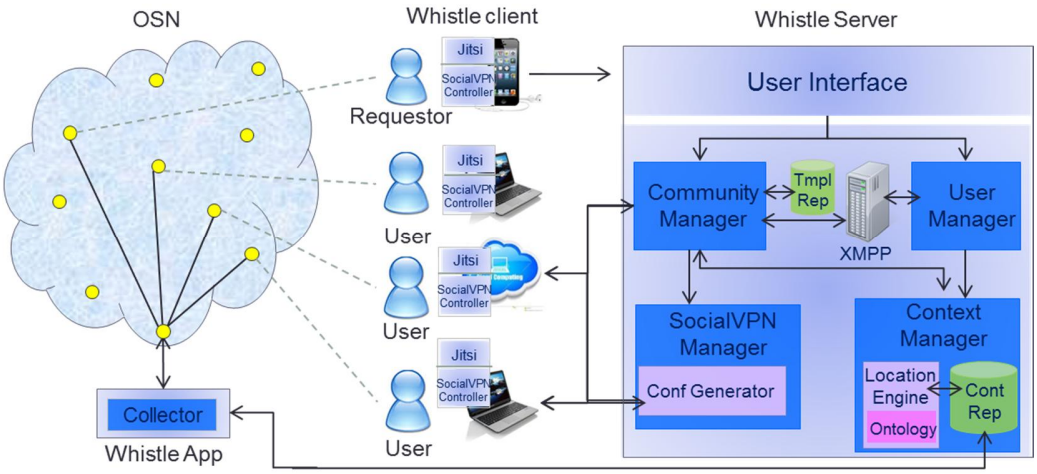


Figure 4. Overall Architecture of Whistle

*2) Whistle App*

The Whistle App acts like a bridge between the Whistle server and an OSN. It receives requests from the server and returns user contexts.

- *Collector* – This component fetches user contexts from an OSN by calling the OSN's API through HTTP GET requests. To get recent updates, the Whistle App needs to examine all user contexts periodically. To do so, it first brings update times of users. If an update is recently made, (i.e. made after last examination time), the *Collector* fetches full contexts of the user. The *Collector* then delivers a JSON object received from an OSN to the Whistle server.

*3) Whistle Client*

A Whistle client is composed of two components: the *SocialVPN Controller*, and the *Jitsi Communicator*. An example diagram of cooperating clients is shown in Figure 5.

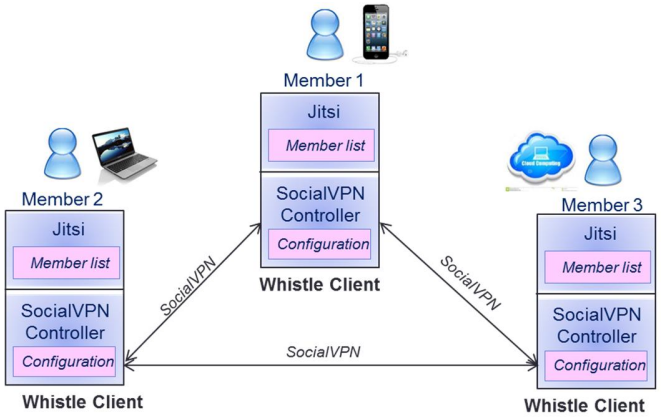


Figure 5. Whistle Clients connected by SocialVPN

- *SocialVPN Controller* – This controller takes responsibility of creating, maintaining, and removing SocialVPN links. According to a configuration file that the *SocialVPN Manager* sent, it establishes SocialVPN connections between members, maintains network condition, and removes connections when cooperation is terminated.
- *Jitsi Communicator* – Jitsi displays members and enables them to cooperate with each other through a rich set of Jitsi communication and sharing methods, such as text messaging, text/audio/video conferencing, and file transfer.

## C. Cooperation Flow

In this section, we describe the cooperation process from user registration to community dissolution from the perspective of a requestor who wants to receive a community service from Whistle.

*1) User registration* – A user registers in the Whistle server before taking or giving cooperative help through Whistle. A user can sign up with his/her OSN account and, if necessary, enter additional user contexts in the registration step. With user contexts, the *User Manager* creates the user's XMPP account and the *Context Manager* saves the contexts in the *Context Repository*. To complete a registration, the user must install a Whistle client software in his/her device(s).

*2) Request for a emergency community* – To ask for help, a user sends a request to the Whistle with required information, such as a selected community template, a target location, and optional user-defined eligibility rules and preferences on helpers.

*3) On-demand creation of an emergency community with eligible nearby members* – When receiving a request, the *Community Manager* retrieves a community template selected by the requestor from the *Template Repository* and asks the *Context Manager* to find out candidates who meet eligibility conditions. In candidate search, the primary criterion is users' locations. The *Context Manager* first performs the two-step location-based search as described in Section III and then, if required, filters out less-preferable candidates based on user-defined preference conditions. In turn, the *Community Manager* sends an invitation to each candidate with

information about an emergency community, and then finalizes a list of members, while the *SocialVPN Manager* creates configuration files for members.

*4) Secure and unrestricted cooperation among members* – As soon as a member client receives a configuration file and a member list, it establishes VPN connections and runs Jitsi. All conversations and resource sharing through Jitsi securely take place within the SocialVPN.

*5) Community dissolution* – When a community's goal is achieved, a leader of a community notifies members of the end of cooperation, and in turn each client removes all SocialVPN connections.

## IV. IMPLEMENTATION

We implemented a Facebook-based prototype of Whistle to verify its feasibility and applicability. A prototype server is developed on Ubuntu version 12.0.4 and has an Apache web server version 2.2.22 and an Ejabberd XMPP server version 2.1.10. Its web interfaces and components are implemented in PHP, AJAX, and Java script. A prototype client is implemented as a software package including the Jitsi software version 2.4 and the SocialVPN software version 14.01.1.

To demonstrate the prototype, we reuse the example scenario of 'Finding a lost child' in [9]. Let's assume Alice, a Whistle user, lost her daughter at a toy store in a shopping center. To ask for immediate help from nearby people, she accesses to the Whistle server, selects the 'Child Missing' template, enters a target location as 'toys R us' using Google map, and adds a preference of female helpers as shown in Figure 6.


Figure 6. A community request made by Alice to find a missing daughter lost in the 'Toys R Us' store.

With information provided by Alice, Whistle creates an emergency community with nearby eligible users and then members start cooperation through Jitsi atop of their community SocialVPN. Alice sends to community members the lost girl's identification and photos that are stored in her smart phone. If a member, e.g. an anonymous member with an alias 'Helper5', finds a girl who looks like the lost girl, he can make a video conference with Alice to make sure that the girl he found is the lost girl. Alice's Jitsi interface having a video conference with four members is shown in Figure 7. If the lost girl is found, a policeman, a leader of the community, announces the achievement to the Whistle server and members, and then the community is terminated. Compared to an approach that only relies on Facebook-exposed cooperation mechanisms, the Whistle approach enables richer and isolated interactions (from anyone outside the community). To accomplish the same task through Facebook, members would need to exchange their accounts of video conferencing software (such as Skype) and someone should initiate a conference call. A member who does not have an account for the software should create one for this cooperation. This process not only delays goal attainment, but also exposes members' accounts.

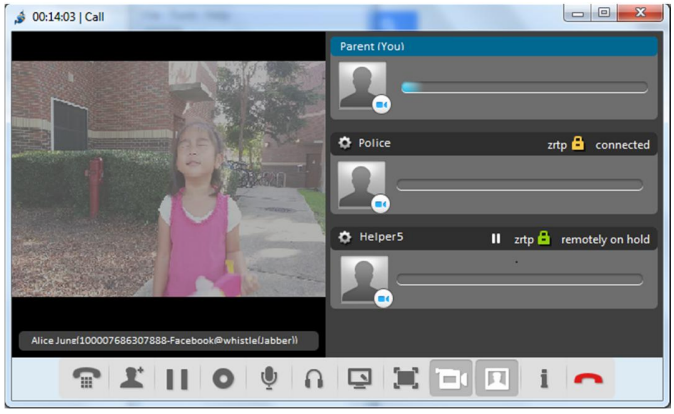


Figure 7. Whistle Clients connected by SocialVPN

## V. RELATED WORK

Most existing location models were developed to offer location-aware services whose ranges vary from a small-size specific space (such as a room) to city/country-size spaces. Regardless of scales, they all aim to model diverse types of objects and their spatial relationships in very fine-grained level.

In the NeXus platform [19], the Augmented World Model (AWM) [20] is used to describe location contexts of three types of objects: 1) *static* objects such as houses, streets, and offices, 2) *mobile* objects such as users, cars, and trains, and 3) *virtual* objects with which the real world is augmented. Each object is represented by not only geometry information specified in the Geographic Markup Language (GML) [21] but also symbolic information like room number and detailed relationship information such as *inside*, *overlaps*, *includes*, *excludes* and *closest*. The AWM is specified using own modeling language, the Augmented World Modeling Language (AWML) [22], and queried using the Augmented World Querying Language (AWQL). As stated in the term of 'World Model', this model

aims to model everything in detail while Whistle requires only specific types of location information.

In the Location Representation Model of RAUM (RAUM-LRM) [23], a location tree describes location information of associated objects. In a tree, symbolic information and inclusion relationships between objects are represented in intermediate levels, and geometric positions stated in three-dimensional Cartesian coordinates are represented at the leaf nodes. The RAUM system does not handle complicated relationships except inclusion because it only needs to know distances between objects to determine available objects within a specific spatial area. Similar to the RAUM-LRM, M-Spaces [22] also uses a tree-based location model, but supports distributed model management - while the AWM and the RAUM-LRM assume a centralized special data management. These location tree based models mostly focus on small-sized spaces and calculate three-dimensional distances, while Whistle deals with two-dimensional distance. In addition, the tree-based models are relatively less extensible compare to an ontology-based model.

Besides the application-specific models mentioned above, general-purpose location ontologies have been proposed. The Open Geospatial Consortium (OGC) has proposed the Geography Markup Language (GML) [21] as an XML grammar to describe geographical features of any kinds of objects including physical objects, users, and services. GML serves as not only a modeling language, but also as an open interchange format for geographic transactions on the Internet. To do so, it is capable of representing and integrating almost all forms of geographic information produced by different types of location sensors and devices. This ability is key to wide acceptability of GML, but, on the other hand, incurs a heavy overhead for location systems dealing with few types of location information like Whistle.

Inspired by GML, W3C proposed the Geospatial Ontologies [25] to provide a simple baseline of geospatial resource description for the web. Towards this, it updated the W3C GEO vocabulary and defines useful extensions and additions. The GeoNames Ontology [26] is a world-wide location model and a database containing over 10,000,000 geographical names. It includes location-related information such as latitude, longitude, elevation, population, administrative subdivision and postal codes, as well as coordinates information.

The above existing models aim at providing comprehensive location information in very detailed level to satisfy a variety of requirements of location-aware applications. Towards this, they deal with diverse objects ranging in size from buildings to small appliances and in type from physical objects to virtual services. To serve users more personalized and adaptive location-aware services, some models even include information about user preferences and services' characteristics while Whistle just focus on physical objects and consumes two-dimensional location data. Therefore, adoption of existing models may significantly increase the complexity and run-time overhead of Whistle without delivering tangible functional benefits.

## VI. CONCLUSIONS

Although many researchers pointed out great potential of using OSNs to enhance human cooperation – and many actual cases have proven the claim – existing OSN-mediated cooperation is still in an experimental stage because of lack of suitable models and restricted cooperation mechanisms. To address these issues, in this paper, we propose a cooperation framework allowing for more effective cooperation. The major contributions are as follows.

- We proposed a cooperation framework for social users, called Whistle, which organizes a location-based emergency community and supports secure and unrestricted cooperation among users.
- We proposed the Whistle location ontology that represents location contexts with standard toponym and structured relationships. For practical use, we also propose maintenance mechanisms for location contexts including creation and update.
- We proposed the two-step location-based user search algorithm to find out the nearest users.
- We proposed a secure and rich method for human cooperation by leverages Jitsi communicator and SocialVPN.

To provide complete community services through Whistle, the following work should be conducted in the future.

- Development of context models for different types of contexts such as *affiliation* or *skill*.
- Consideration of advanced cooperation model and access control model during cooperation in Whistle.
- Development of a trust model to evaluate users' reputation.
- Development of resiliency policies for Whistle.
- Development of diverse use cases.
- Comprehensive evaluation of implementation of Whistle.

## ACKNOWLEDGEMENT

This material is based upon work supported in part by the National Science Foundation under Grants No. 1339737, 1265341, and 1234983. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## REFERENCES

[1] Statistic Brain, "Social networking statistics," January 2014. http://www.statisticbrain.com/social-networking-statistics/

[2] Twitter. Lost dog found. https://twitter.com/TheLDFBand

[3] Twitter. Fidofinder, https://twitter.com/fidofinder

[4] Emory Healthcare. Can Twitter Help Save Lives? June 2011. http://www.emory.edu/EMORY_REPORT/stories/2011/06/campus_can_twitter_ help_ save_lives.html

[5] Fox News. People React to Irene on Facebook and Twitter, August 2011. http://www.myfoxtampabay.com/story/18031344/people-react-to-irene-on-facebook-and-twitter

[6] ABC News. Japan Earthquake and Tsunami: Social Media Spreads News, Raises Relief Funds, March 2011.

http://abcnews.go.com/Technology/japan-earthquake-tsunami-drive-social-media-dialogue/story?id=13117677

[7] Twitter. Volunteer on Twitter to Help with Hurricane Irene and Other Disasters, Aug 2011. http://hope140.org/blog/?p=209

[8] Y. Jung, M. Kim, J. BD Joshi, "Towards secure cooperation in online social networks," The 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), pp.80-88, Oct. 2012

[9] Y. Jung and J. BD Joshi, "CPBAC: Property-based access control model for secure cooperation in online social networks," Computers & Security 2013.

[10] Jitsi Communicator, https://jitsi.org/

[11] P. St Juste, D. Wolinsky, P. Oscar Boykin, M. Covington, and R. J. Figueiredo, "SocialVPN: Enabling wide-area collaboration with integrated social and overlay networks," Computer Networks, vol. 54, no. 12, pp. 1926-1938, 2012

[12] Facebook Graph API, https://developers.facebook.com/docs/graph-api/using-graph-api/v2.0

[13] T. Strang and C. Linnhoff-Popien, "A Context Modeling Survey", International Workshop on Advanced Context Modelling, Reasoning And Management at UbiComp, England UK, September 2004.

[14] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg , "A survey on context-aware systems", International Journal of Ad Hoc and Ubiquitous Computing, vol. 2 Issue. 4, pp. 263-277, January 2007.

[15] C. Bettini, B. Oliver, H. Karen, I. Jadwiga, N. Daniela, R. Anand, and R. Daniele, "A survey of context modelling and reasoning techniques," Pervasive and Mobile Computing vol. 6, no. 2, pp. 161-180, 2010

[16] W3C, "OWL 2 Web Ontology Language, " December 2012, http://www.w3.org/TR/owl2-overview/

[17] Google Geocode API, https://developers.google.com/maps/documentation/geocoding/

[18] R. W. Sinnott, "Virtues of the Haversine," Sky and Telescope, vol. 68 issue. 2, pp. 158, 1984

[19] F. Hohl, U. Kubach, A. Leonhardi, K. Rothermel, and M. Schwehm, "Next century challenges: Nexus—an open global infrastructure for spatial-aware applications," The 5th annual ACM/IEEE international conference on Mobile computing and networking, ACM, pp. 249-255, August 1999.

[20] D. Nicklas and M. Bernhard, "The nexus augmented world model: An extensible approach for mobile, spatially-aware applications," The 7th International Conference on Object-Oriented Information Systems, pp. 392-401, 2001.

[21] Geography Markup Language (GML), http://www.opengeospatial.org/standards/gml

[22] D. Nicklas and M. Bernhard, "On building location aware applications using an open platform based on the NEXUS Augmented World Model," Software and Systems Modeling vol. 3, no. 4, pp. 303-313, 2004.

[23] M. Beigl, T. Zimmer, and C. Decker, "A location model for communicating and processing of context," Personal and Ubiquitous Computing, vol. 6 no. 5-6, pp. 341-357, 2002.

[24] I. Satoh, "A location model for pervasive computing environments," Third IEEE International Conference on Pervasive Computing and Communications (PerCom'05), pp. 215-224, 2005.

[25] W3C Geospatial Ontologies, http://www.w3.org/2005/Incubator/geo/XGR-geo-ont/, October 2007.

[26] GeoNames Ontology, http://www.geonames.org/ontology/ontology_v3.1.rdf