EAI Endorsed Transactions
on Collaborative Computing Research Article ICST.ORG

Emergency Response using Ephemeral Social Communities across Online Social Networks

Youna Jung[1,*], Renato Figueiredo[2] and José A. B. Fortes[3]

[1] Virginia Military Institute, 426 Mallory Hall, Virginia Military Institute, Lexington, VA 24450
[2] University of Florida, 336 Larsen Hall, University of Florida, Gainesville, FL 32611
[3] University of Florida, 339A Larsen Hall, University of Florida, Gainesville, FL 32611

Abstract

In an emergency situation, receiving prompt and organized help from nearby people is of critical importance. The growing use of online social networks (OSNs) in emergency situations is a clear indication of the natural applicability of online social networking technologies to emergency responses. Despite this intense interest, a number of fundamental limitations still exist, such as lack of conceptual models and limitations on group organization and cooperation. To address existing limitations, we propose Whistle$^+$ – a cooperation framework for OSN users which can 1) dynamically organize an emergency community with nearby eligible users who are distributed in heterogeneous online social networks, and 2) guarantee secure communication, unrestricted cooperation and resource sharing by leveraging the SocialVPN virtual network and the Jitsi communicator. To test the feasibility and applicability of Whistle$^+$, we present a prototype implementation and demonstrate its applicability to an example use case.

Keywords: Emergency cooperation, Ephemeral social community, online social network, cooperation framework, temporary virtual private network, privacy protection.

Received on 24 February 2015, accepted on 04 July 2015, published on 17 December 2015

doi: 10.4108/eai.17-12-2015.150805

1. Introduction

In general, the success of emergency response depends on the promptness of information dissemination and emergency cooperation. When people face an emergency admitting no delay, such as a child missing or a medical emergency, prompt and organized help from nearby people is critical. For example, in the case of natural disasters, such as earthquakes 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, and the location of life jackets or fire extinguishers. In the case of a child missing or a hit-and-run accident, 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 successfully respond to an emergency, it is required to meet four core requirements: 1) Rapid dissemination of emergency information, 2) Real-time discovery of people in need of help and voluntary helpers based on their contexts such as location or ability, 3) On-demand organization of an emergency community with essential members, and 4) 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, currently the largest OSN, has 1.4 billion users worldwide, i.e. 11% of people on Earth [1]. Second, OSNs provide access to diverse user contexts, including location and ability contexts. OSN users spontaneously maintain their personal information up to date and disclose it to others, in order to establish new relationships and maintain existing relationships. With proper permission granted by

[*]Corresponding author. Email: younajung@gmail.com

users, an application can easily obtain user contexts through the OSN's APIs. Third, OSNs enable people to easily share information with their friends or followers

In recent 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 platform for gathering people and enabling cooperation amongst them. For example, 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] through Twitter. 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, existing cooperation through OSNs is still in its infancy due to the lack of key mechanisms: maintaining users and user contexts that are distributed across heterogeneous online social networks, searching eligible users among very large numbers of users at request time, forming a well-organized group, orchestrating cooperation between members in order to effectively achieve a goal, and supporting intuitive and rich cooperation methods while 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 some of the limitations described above, Jung et al. [9] proposed a Facebook application called SeCON employing the role-based community model, the situation-based cooperation model, and the community-centric property based access control model (CPBAC). However, some issues still remain unsolved, such as lack of context models to represent user contexts and limitations of sharable resources, and cooperation methods.

To address these remaining issues, in this paper, we propose Whistle$^+$, a cooperation framework that is able to obtain and maintain user contexts, find out necessary people who can give essential help, create an emergency community on demand, and enable members to directly cooperate with each other in a peer-to-peer fashion independently from OSNs. To do so, Whistle$^+$ has its own context models, as well as existing models that were previously proposed in SeCON [9], and mechanisms to maintain user contexts obtained from multiple OSNs and search users based on user contexts. Furthermore, Whistle$^+$ leverages Jitsi [10] and SocialVPN [11] to guarantee unrestricted and secure network communication for cooperation. Figure 1 shows us an example implementation of Whistle$^+$ that is connected to three online social networks: Facebook, Twitter, and LinkedIn. Existing social users who are using Facebook, Twitter, or LinkedIn can sign up for Whistle$^+$ using existing credentials and add their other accounts, optionally, so that Whistle$^+$ retrieves the user's contexts that are stored in other OSNs. Note that Whistle$^+$ does not request a user's passcode – only the username. When an emergency occurs, Whistle$^+$ organizes an emergency community consisting of distressed people and helpers regardless of which OSNs they are using at execution time, and then establishes VPN connections between their personal devices.

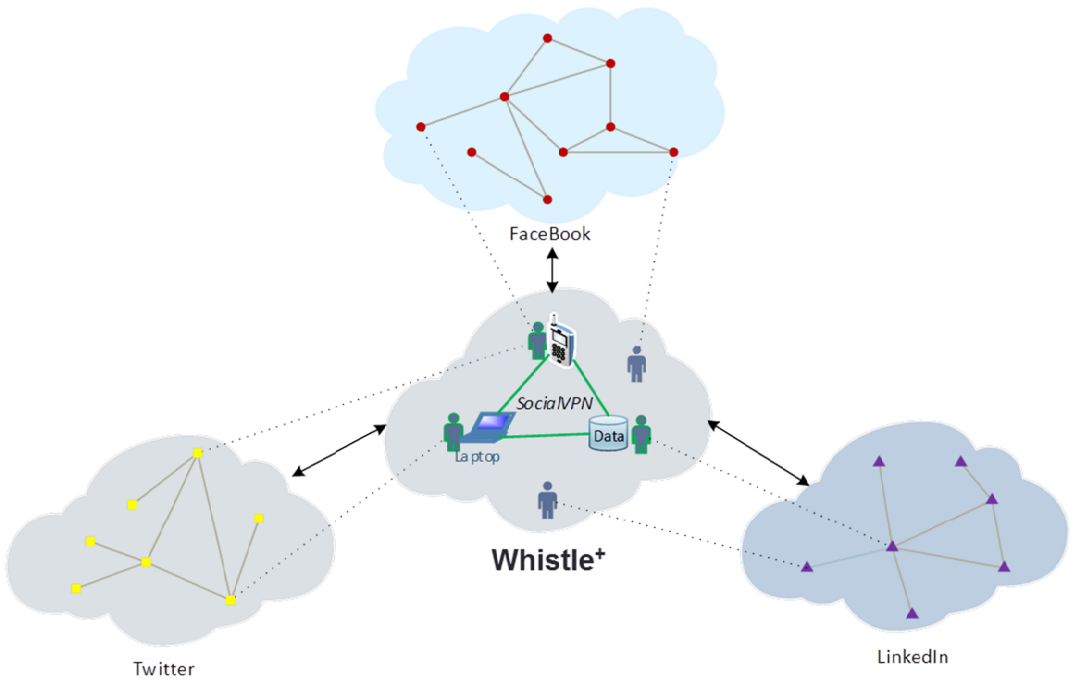


Figure 1. Overview of Whistle$^+$

The rest of the paper is organized as follows. In Section 2, we introduce two motivating examples, and in Section 3 we overview preliminary work. In Section 4, we propose Whistle$^+$, a cooperation framework for OSN users, and describe its major contributions; in particular, the ontology-based models for the *ability* and *location* contexts, the context management method, and the dynamic creation of emergency communities across OSNs. In Section 5, we describe the architecture and cooperation flow of Whistle$^+$ and demonstrate a prototype implementation based on an example (finding a lost child). In Section 6, we discuss related work and then present conclusions and future work in Section 7.

2. Motivating Examples

To shed light on limitations of existing cooperation among OSN users and the innovation of Whistle$^+$, in this section, we describe two motivating examples: *Finding a lost child* and *Emergency rescue*. Both show emergency situations that require prompt, well-organized, and secure cooperation among nearby available people. When a user requests an emergency cooperation, Whistle$^+$ can dynamically organize a community with suitable users who are able to help the user immediately by examining user contexts obtained from the users and multiple OSNs in real time.

- *Finding a lost child* – Alice lost her daughter at a toy store in a crowded shopping mall. To ask for help, she calls 911 and explains the situation. After receiving information about the lost girl (for example, how she looks like and what she is wearing), policemen scatter and search for her. Meanwhile, Alice asks people if they saw her daughter. In such traditional way, it is however difficult to effectively disseminate the information in a timely fashion and gather voluntary helpers. Furthermore, the identity of the lost girl may be misused if broadcasted indiscriminately.
- *Emergency rescue* – A hurricane hits a coastal region. Bob, who lives in the disaster area, drives to the closest shelter, but it is already destroyed by the hurricane. He finds another shelter and narrowly escapes from the hurricane's damage. He wants to let nearby people know that the closest shelter is currently unavailable, but there is no way to find out people in a disaster area and share local information with them. Meanwhile, Carol is injured while escaping the disaster area. Although there is a doctor who can immediately help her in the next block, there is no way for her to discover this information and ask him or her for help.

3. 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 using 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:

- *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 of user contexts. Furthermore, absence of semantic correlation between context values is another issue. Some contexts, such as location or occupation, 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 both standard terms and context models to represent semantics of contexts.
  Challenge: We need appropriate models to represent user contexts and efficiently search users based on contexts.
  Contribution: To address the challenge of context modelling, we first propose two ontology-based context models: the location ontology, using toponyms which have become 'de facto' standards in the Internet (in particular, they are used by Google); and the ability ontology, which uses the standard occupational classification of the Bureau of Labor Statistics in the United States. In addition to the context models, we propose schemes to manage conflicting contexts from different OSNs and update contexts with reduced user intervention. Especially for location contexts, we also propose a two-step user searching algorithm for efficient location-based user search.
- *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.

- *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 from services provided by OSNs.

4. Cooperation Framework for OSN users

Whistle$^+$ is a cooperation framework that maintains users and user contexts, promptly forms emergency communities by contacting users who are distributed across OSNs, dynamically establishes virtual private networks for communities, and supports cooperation among community members. The cooperation services of Whistle$^+$ pass through two phases: the bootstrapping phase and the operation phase, as shown in Figure 2.

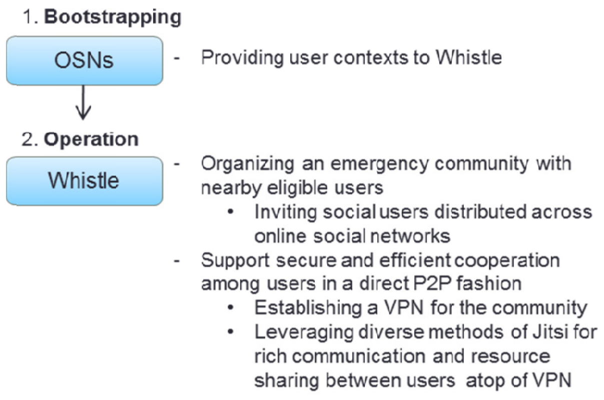

1. Bootstrapping
OSNs
- Providing user contexts to Whistle
2. Operation
Whistle
- Organizing an emergency community with nearby eligible users
• Inviting social users distributed across online social networks
- Support secure and efficient cooperation among users in a direct P2P fashion
• Establishing a VPN for the community
• Leveraging diverse methods of Jitsi for rich communication and resource sharing between users atop of VPN

Figure 2. Two Phases of Community Service in Whistle$^+$

In the first phase, online social networks provide contexts of users who register with the Whistle$^+$ service – which is a service external to the OSNs. This step allows Whistle$^+$ to build a user pool with abundant contexts with minimal user intervention. When an emergency arises, the operation phase commences; Whistle$^+$ dynamically organizes communities of OSN users to deal with emergencies.

The use of Whistle$^+$ in emergency situations could be beneficial to both people in need and administrators of an emergency management agency. Whistle$^+$ helps people to better assess an emergency situation by enabling them to share local information in real time -- for example, "The shelter in the section 9 was destroyed.", or "The road collapsed on Magnolia Street between 5th to 14th" -- so that they can effectively cope with changing emergency situations. Furthermore, Whistle$^+$ can also assist government agencies in collecting and analysing emergency-related data, as it allows people to share emergency data and store these data in their personal devices. As a result, each user becomes a witness of an emergency and a data provider.

4.1. Creation and maintenance of a user pool

Whistle$^+$ has four major functionalities: 1) Creation and maintenance of a user pool, 2) On-demand organization of ephemeral emergency communities, 3) Setting up a temporary SocialVPN virtual network connecting community members, and 4) Rich communication and resource sharing through Jitsi. In this section, we describe how to obtain and maintain user contexts and two ontology-based context models in detail.

1) Creation and maintenance of a user pool

Whistle$^+$ creates 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, *ability* context, and *location* context. Among various user contexts, Whistle$^+$ first focuses on *location* and *ability* contexts since, in most cases, emergency communities are likely formed with nearby people who have abilities that are required to cope with an emergency. In the following, we elaborate on what user contexts are, and how to obtain, represent, and update them:

(i) *Types of user context*: Whistle$^+$ represents a user with four types of contexts: *Identity*, *Ability*, *Location*, and *Reputation*. The *Identity* contexts, describing who a user is, include basic user information, such as *name*, *gender*, *age*, *birthday*, *email*, *phone number*, *relationship status*, and *religion*. The *Ability* contexts, consisting of *job* and *skill* contexts, describe what a user can do. The *Location* contexts include location-related information, such as *current city* and *places checked in*, and inform the system about the places where a user is likely to be at request time. The *Reputation* context represented by a reputation value classifies how reliable a user is. The value of reputation contexts are systematically calculated by using a trust model

(System-generated context), while the values of other contexts are created by users (User-generated context).

(ii) *Consent-based semi-automatic context acquisition*: Whistle+ obtains user contexts from two different sources: OSNs and users. A user can register in Whistle+ by using an existing OSN account. After signing up, the user can add more OSN accounts if he/she has accounts for other OSNs so that Whistle+ is able to fetch the user's contexts from multiple OSNs 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, including: *name*, *age*, *bio*, *birthday*, *email*, *gender*, *languages*, *relationship status*, *current location*, *education*, and *work*. If essential contexts like *location* are null, Whistle+ asks a user to directly enter contexts in the registration step (User-provided context). If a context conflicts with another context that is received from different OSNs, Whistle+ lets the user choose one or enter new context value. If the user skips the selection step, then Whistle+ selects the most recent context.

When a user registers, Whistle+ adds a new user to its user pool. In the pool, each user is represented as a set of contexts. Especially for the *location* and *ability* contexts, Whistle+ uses ontology-based context models which are specified in Sections 4.1.2 and 4.1.3 below. For the remaining contexts, we use a key-value context model [13, 14] which allows exact matching retrieval only. The context models for other contexts are left for future work.

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. Currently, Facebook and Twitter support the notification service but others do not provide it. In the prototype of Whistle+, we create accounts in Facebook and Twitter to use the notification service. 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). In addition, Twitter notifies followers of changes in tweet location only. For the other OSNs such as LinkedIn and Google+, alternatively, Whistle+ can periodically fetch contexts of users and update contexts if changed.

2) 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 6, 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 expressive 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 3.

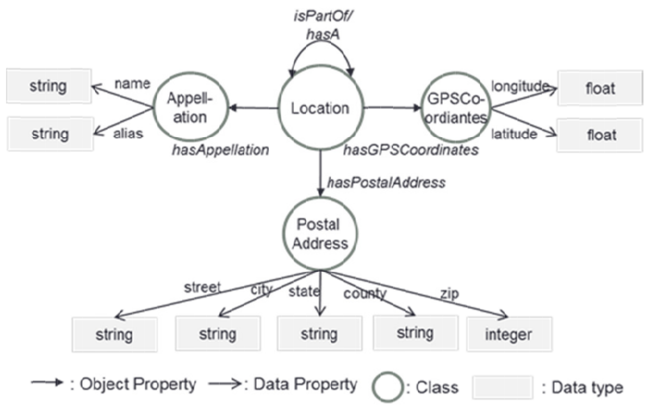


Figure 3. The Whistle+ Location Ontology

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.

As an example, we present a location representing the *ToysRus* store in the *Oaks Mall* in *Gainesville*, *Florida* in Figure 4. 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.

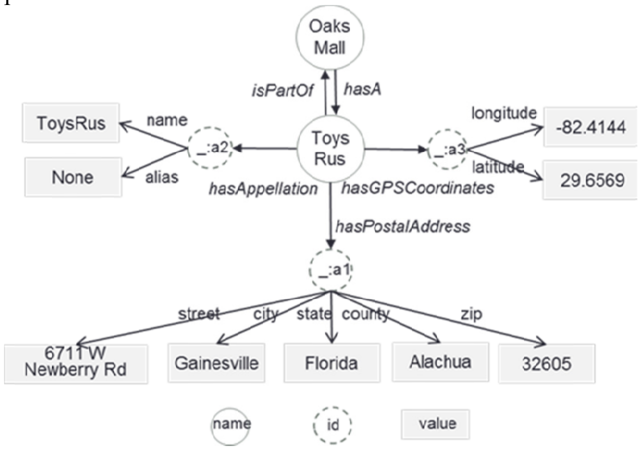


Figure 4. An example location for *ToysRus* in the *Oaks Mall* in Gainesville, Florida

Table 1. 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].

3) Whistle$^+$ ability ontology

The Whistle$^+$ ability ontology represents a user's ability as structured information of his/her job and corresponding skills as shown in Figure 5. The *Job* class has two data properties: *jobID* and *jobName*. The *jobName* data property represents standard names of jobs, while the *jobID* data property represents identification numbers of occupations. The *Skill* class has two data properties: *skillName* and *skillID*. The *hasParent* object property represents hierarchical relationships between jobs. The *requires* object property, an inverse property of the *requiredBy*, indicates relationships between jobs and skills. A job may require one or more skills in order to perform the job and a skill may be required by one or more jobs.

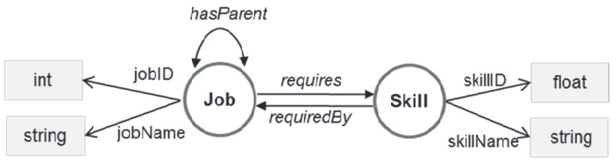


Figure 5. Whistle$^+$ Ability Ontology

To create instances of the Whistle$^+$ ability ontology with standard terms, we use the most recent version of occupational classification published by the Bureau of Labor Statistics of the United States in 2010 [18]. The classification data contains unique classification codes, standard occupation names, and detailed descriptions for each occupation. To extract necessary information from the data, we implemented a parser by employing APIs of the Stanford parser [19] as shown in Figure 6.

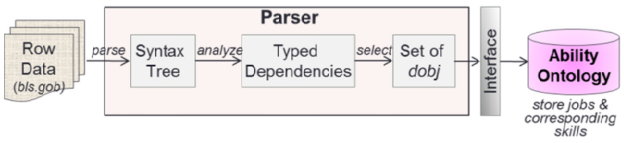


Figure 6. The Creation Process of Whistle$^+$ Ability Ontology Instances

The Stanford parser [19] is a well-known natural language parser that works out the grammatical structure of sentences which groups of words go together and which word are the subject and/or object of a verb. Using the Stanford parser APIs, Whistle$^+$ parses the occupational classification data, builds syntax trees for each job descriptions, analyse typed dependencies that represent grammatical relationships in a sentence, and selects *dobj* relationships that represent dependencies between a verb and its direct object. If an object in a *dobj* relationship selected is a compound noun or has modifiers, Whistle$^+$ complements the *dobj* relationship by adding its associated words. The set of the complemented *dobj* relationships of an occupation becomes the skill set that is required for the job. For example, consider the raw data for Podiatrist: "*29-1081 Podiatrists- Diagnose and treat diseases and deformities of the human foot.*" From the data, Whistle$^+$ extracts the two skills that are required for a podiatrist as follows: *diagnose diseases of human foot* and *diagnose deformities of human foot*. Through the interface of the ability ontology, Whistle$^+$ sends information about jobs (*jobID*, *jobName*, and *hasParent* relationships) and skills (*skillID*, *skillName*, and the *requires* and *requiredby* relationships). Note that probabilistic parsers like the Stanford parser still can make mistakes [19]. To increase the accuracy of information, we thus need to verify the ability ontology instances at a later stage.

4.2. On-demand organization of ephemeral emergency communities

Whistle$^+$ dynamically creates an ephemeral social community with eligible users nearby at request time, such that a user in danger can receive prompt help. To do so, it searches necessary users based on contexts, establishes a temporary virtual private network between co-operators, and launches a Jitsi communicator to enable them to cooperate with each other in a peer-to-peer fashion.

1) Types of Community

Before discussing communities and cooperation services of Whistle$^+$, it is worth defining the types of human communities and narrowing down the scope of Whistle$^+$ services. We define the types of communities using four criteria: *Why*, *How*, *Who*, and *When*. The criterion *Why* represents a community's goal and dynamics. It specifies if a goal is pre-defined or unexpectedly created. The *How* represents the preparation of a cooperation process. If a cooperative process for a goal is pre-defined, a community follows the process at execution time. Otherwise, a cooperation process is dynamically designed when a community is created. The criterion *Who* describes the way to assign co-operators to a community. We can assign users before a community is created, or dynamically select members depending on user contexts at request time. The criterion *When* indicates the predictability of community creation. A community can be created at a designated time, or be dynamically created depending on user demands.

Based on four criteria, we classify communities into seven types as shown in Figure 7. A type 1 community is created at a designated time, with fixed users who follow a pre-defined process to achieve a pre-defined goal, while a type 2 community is dynamically created for a known goal with fixed users and fixed cooperation. For instance, assume a meeting community with fixed members who are geographically distributed. If the community is for a regular meeting, then it is a type 1 community. If the community is for a special meeting occasionally held, then the community is a type 2 community. The type 3 and type 4 communities dynamically select co-operators who are able to effectively obtain known goals by following a designed process. The type 3 communities are created at an expected time, whereas type 4 communities are created on demand. For examples, a community for a regular fire drill that trains people in a particular building to follow existing exit procedure can be a type3 community. However, an emergency transportation community that consists of nearby paramedics and doctors who are available at request time is a type 4 community. In the type 5 and 6 communities, cooperation processes for known goals are dynamically designed at request time depending on environmental situations and abilities of co-operators who are dynamically selected. The type 7 community is the most dynamic scenario, and deals with unexpected requests by designing a cooperation process and gathering the most suitable co-operators in real time. Among diverse types of communities, Whistle$^+$ only focuses on the type4 communities in this stage.

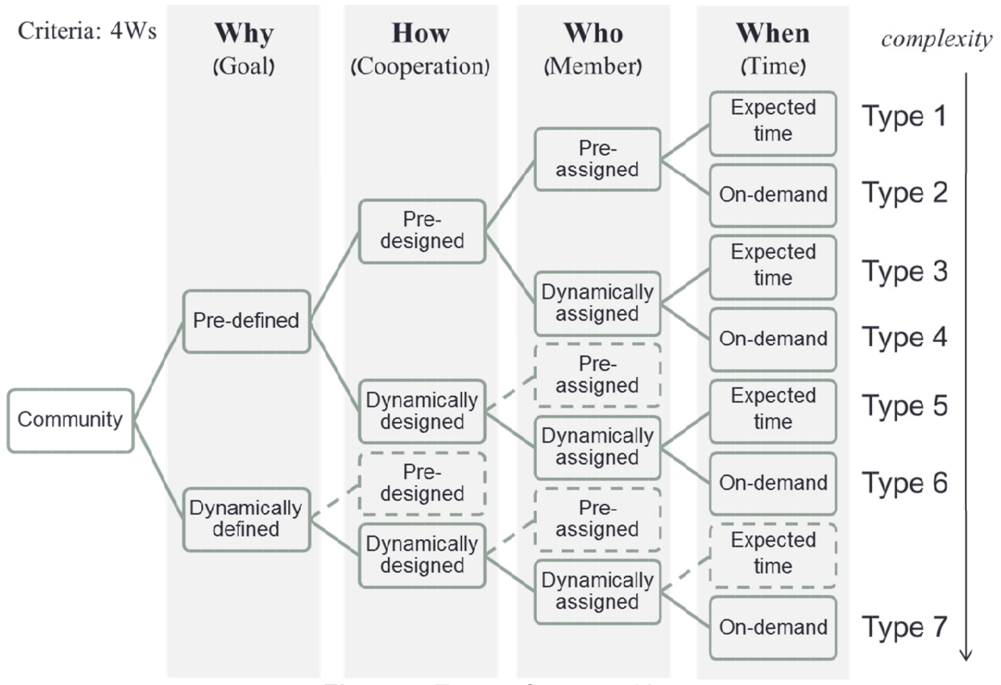


Figure 7. Types of communities

2) Context-based user search

To find out who are the most suitable users at request time, Whistle$^+$ conducts context-based user search. In this section, we describe eligibility rules and user search based on two essential contexts, *location* and *ability*.

(i) *Eligibility rule*: For each community role, an eligibility rule represented as a logical expression should be specified. A rule indicates conditions related to user contexts, which a user should satisfy in order to take a corresponding community role.

(ii) *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 2). 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 2). 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* [20]. The algorithm of location-based user search is specified in Table 2. By conducting the two-step user search, Whistle$^+$ can significantly reduce the number of users who need to be checked for real-time locations (because 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.

3) 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.

Table 2. Algorithm for location-based user search

```
1. $l_t$ = name of target location
2. $mem_{min}$ = minium 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
          excluded in existing $nl$.
    b. for each user $(u \nexists pcand)$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 $candidate$
```

4.3. Secure and unrestricted cooperation independent of OSNs

1) 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. streaming video/audio from a camera) 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 to share external resources that are stored in their personal devices or cloud, regardless of resource types and OSNs that members use. To accomplish this, Whistle$^+$ leverages the social virtual private network (SocialVPN) [11], a virtual networking system 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.

2) 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 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.

5. Implementation of Whistle$^+$

5.1. Architecture

Whistle$^+$ consists of a centralized server, applications for each OSN (Whistle$^+$ App), and a number of clients that are connected through the Internet. The Whistle$^+$ server and the Whistle$^+$ Apps are always connected and interact with each other. Whistle$^+$ clients are dynamically connected and disconnected through SocialVPN. The overall architecture of Whistle$^+$ is illustrated in Figure 8.

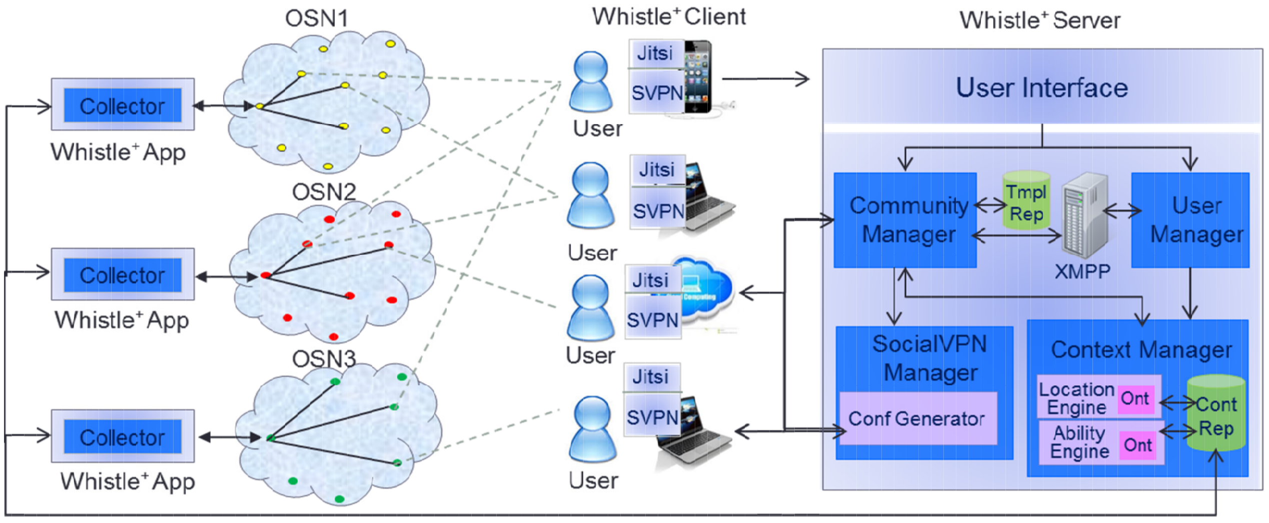


Figure 8. Overall Architecture of Whistle$^+$ consisting of a centralized server (Whistle$^+$ Server), three OSN applications (Whistle$^+$ App), and a number of clients (Whistle$^+$ Client). The full names of acronyms are as follows: The Community Template Repository (Tmpl Rep), The SocialVPN Configuration Generator (Conf Generator), The Context Repository (Cont Rep), Ontology (Ont), The SocialVPN Controller (SVPN).

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 standard instant messaging protocol with open-source implementations, 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 rules. With a user's OSN accounts received from the *User Manager*, the *Context Manager* fetches various user contexts from multiple OSNs that the user is interacting with. 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*, and stores them in the *Location Database*. The *Context Manager* periodically receives updated user contexts from *Whistle$^+$ Apps* 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*. In case of the ability contexts, the *Ability Engine* compares job and affiliation information fetched from OSNs to the standard occupation names that are listed in the *Whistle$^+$ ability ontology*. If a matched name exists, the job and a corresponding skill set are stored in the *Context Repository* as the user's contexts. Otherwise, the *Ability Engine* asks the user to choose a correct job among several jobs whose names are partially matched.
- *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.

2) Whistle$^+$ Application

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, a Whistle$^+$ App needs to examine all user contexts periodically. To do so, it first checks 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 9.

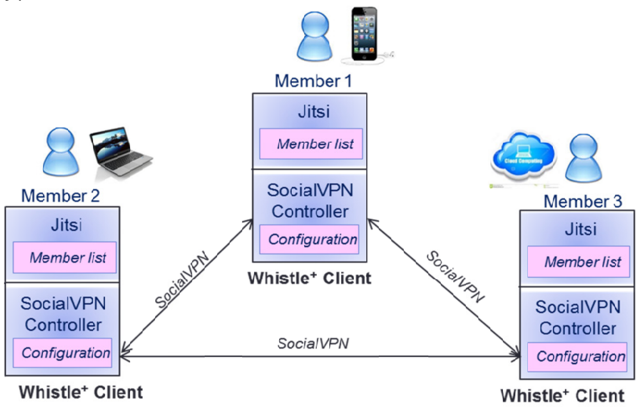


Figure 9. An example diagram of cooperating clients

- *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.

5.2. 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$^+$.

(i) *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).

(ii) *Request for an emergency community* – To ask for help, a user sends a request to Whistle$^+$ with required information, such as a selected community template, a

target location, and optional user-defined eligibility rules and preferences on helpers.

(iii) *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 4.2.ii 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 via an OSN that a user is using at that time with information about an emergency community, and then finalizes a list of members, while the *SocialVPN Manager* creates configuration files for members.

(iv) *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.

(v) *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.

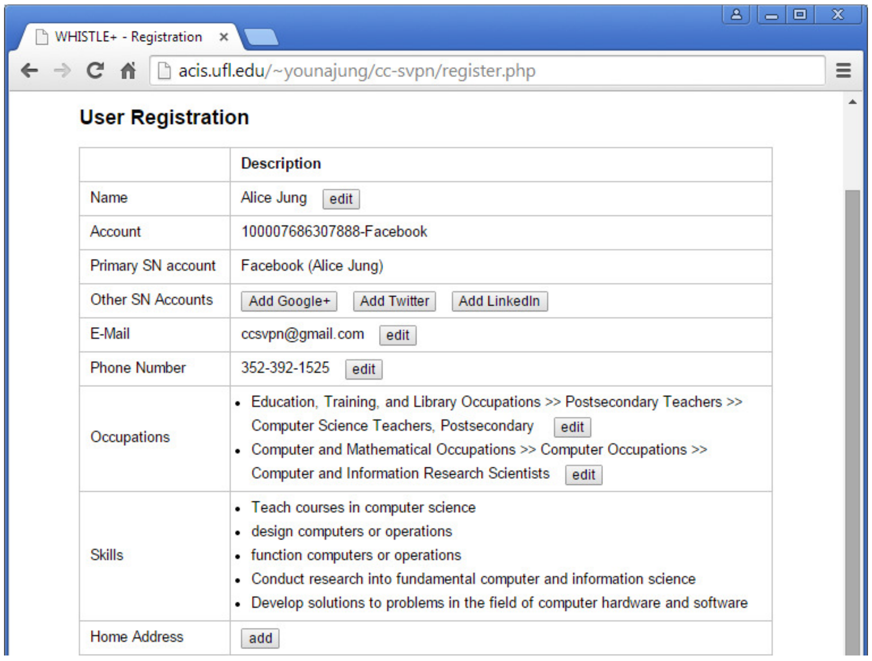
5.3. Prototype Implementation

We implemented a prototype of Whistle$^+$ to verify its feasibility and applicability. The 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. The prototype client is implemented as a software package including the Jitsi software version 2.4 and the SocialVPN software version 15.01.0.

To demonstrate the prototype, we reuse the example scenario of 'Finding a lost child' in Section 2. Consider a scenario involving Alice, a Whistle$^+$ user. She registered in Whistle$^+$ using her Facebook account and added her other OSN accounts (Twitter and LinkedIn) as shown in Figure 10, so that Whistle$^+$ can fetch her contexts distributed across multiple OSNs.

While shopping at a toy store in a shopping center, she suddenly realizes that she lost her daughter. To ask for immediate help from nearby people, she accesses 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 11.

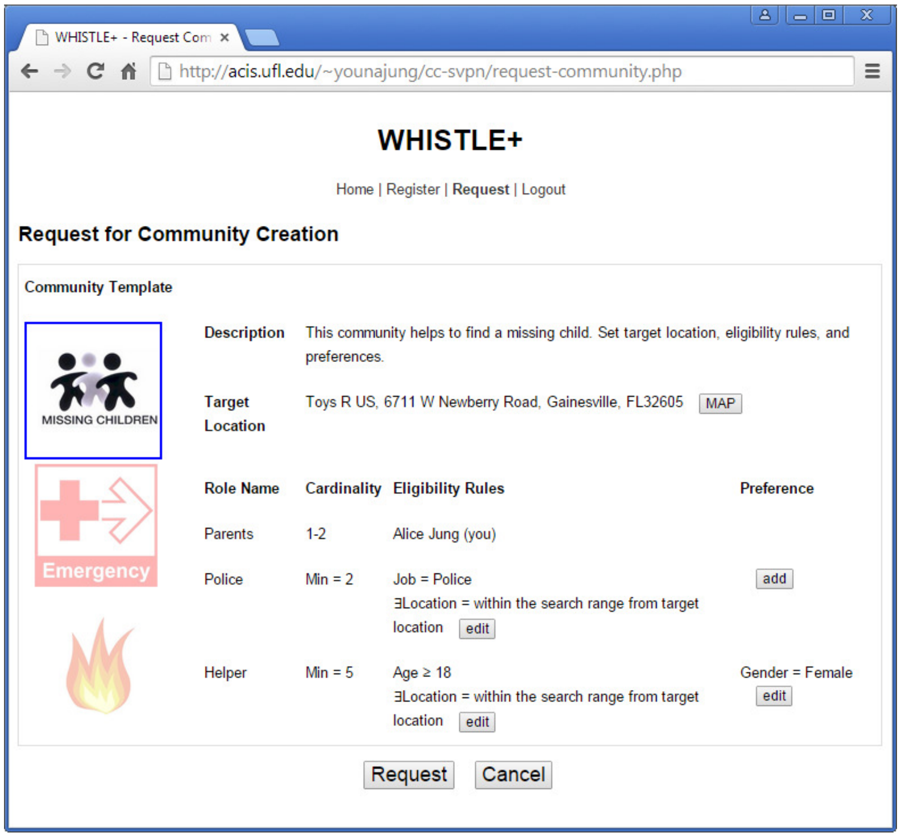
With information provided by Alice, Whistle$^+$ creates an emergency community with nearby eligible users and then members start cooperation through Jitsi atop of their community SocialVPN. Alice sends to community members the lost girl's identification and photos that are stored in her smart phone. 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 12. 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.



User Registration

	Description
Name	Alice Jung edit
Account	100007686307888-Facebook
Primary SN account	Facebook (Alice Jung)
Other SN Accounts	Add Google+ Add Twitter Add LinkedIn
E-Mail	ccsvpn@gmail.com edit
Phone Number	352-392-1525 edit
Occupations	• Education, Training, and Library Occupations >> Postsecondary Teachers >> Computer Science Teachers, Postsecondary edit • Computer and Mathematical Occupations >> Computer Occupations >> Computer and Information Research Scientists edit
Skills	• Teach courses in computer science • design computers or operations • function computers or operations • Conduct research into fundamental computer and information science • Develop solutions to problems in the field of computer hardware and software
Home Address	add

Figure 10. A community request made by Alice to find a missing daughter lost in the 'Toys R Us' store



WHISTLE+ - Request Com

http://acis.ufl.edu/~younajung/cc-svpn/request-community.php

WHISTLE+

Home | Register | **Request** | Logout

Request for Community Creation

Community Template

MISSING CHILDREN

Emergency

Description	This community helps to find a missing child. Set target location, eligibility rules, and preferences.
Target Location	Toys R US, 6711 W Newberry Road, Gainesville, FL32605 MAP

Role Name	**Cardinality**	**Eligibility Rules**	**Preference**
Parents	1-2	Alice Jung (you)	
Police	Min = 2	Job = Police ∃Location = within the search range from target location edit	add
Helper	Min = 5	Age ≥ 18 ∃Location = within the search range from target location edit	Gender = Female edit

Request Cancel

Figure 11. A community request made by Alice to find a missing daughter lost in the ‘Toys R Us’ store

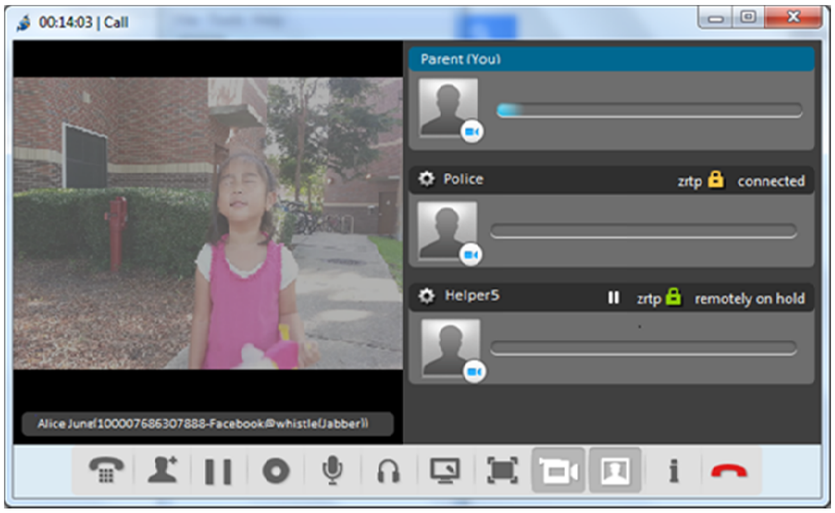


Figure 12. Whistle^{+} Clients connected by SocialVPN

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.

6. Related Work

6.1. Location Models

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 [21], the Augmented World Model (AWM) [22] 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) [23] but also symbolic information like room number and detailed relationship information such as inside, overlaps, includes, excludes and closest. The AWM is specified using its own modeling language, the Augmented World Modeling Language (AWML) [24], and queried using the Augmented World Querying Language (AWQL). As stated in the term of 'World Model', this model aims to model all types of location information from physical objects to services in detail, while Whistle$^+$ requires only specific types of location information.

In the Location Representation Model of RAUM (RAUM-LRM) [25], 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 [26] 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 compared 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) [23] 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 [27] 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 [28] 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.

6.2. Location-based Social Networks

Location-based Social Networks (LBSNs) aim to enable social users to share location-embedded information with friends and also make new friends who are recommended based on similarities in locations in the physical world as well as their location-tagged media content [29]. At present, there are a variety of LBSNs; for example, NeerbyFeed [30], Facebook Places [31], and Sonar [32]. According to the study of TNS in 2012 [33], mobiles users are increasingly using location-based services. Around a quarter uses it to find restaurants and entertainment venues (26%) and one in five is using it to find their friends nearby (22%).

Unlike existing LBSNs that are mainly focusing on sharing of geo-tagged information and recommending nearby friends, Whistle$^+$ focuses on dynamic organization of local emergency communities. Thus, existing LBSNs are insufficient for emergency response due to lack of real-time organization and cooperation management mechanisms.

7. Conclusion

Although many researchers pointed out the great potential benefits of using OSNs to enhance human cooperation – and many actual cases have supported 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 an emergency community on demand and supports secure and unrestricted cooperation among users.
- We proposed two ontology-based context models that represent location context and ability contexts with standard terms and structured relationships. For practical use, we also propose maintenance mechanisms for those contexts.
- 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.
- 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$^+$.

Acknowledgements.
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 and Found Pets. https://twitter.com/findlostpets
[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/ documenta tion/geocoding/
[18] Standard Occupational Classification, Bureau of Labor Statistics, United States, 2010, *http://www.bls.gov/soc/ 2010/soc_alph.htm*
[19] The Stanford Parser, http://nlp.stanford.edu/software/lex-parser.shtml
[20] R. W. Sinnott, "Virtues of the Haversine," Sky and Telescope, vol. 68 issue. 2, pp. 158, 1984
[21] 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.
[22] 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.
[23] Geography Markup Language (GML), http://www.opengeospatial.org/ standards/gml
[24] 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.
[25] 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.
[26] I. Satoh, "A location model for pervasive computing environments," Third IEEE International Conference on Pervasive Computing and Communications (PerCom'05), pp. 215-224, 2005.
[27] W3C Geospatial Ontologies, http://www.w3.org/2005/ Incubator/geo/ XGR-geo-ont/, October 2007.
[28] GeoNames Ontology, http://www.geonames.org/ontology/ ontology_ v3.1.rdf
[29] Yu Zheng and Xiaofang Zhou, "Location-based social networks," Computing with Spatial Trajectories, Yu Zheng and Xiaofang Zhou (Eds), Springer, ISBN 978-1-4614-1629-6, 2011.

[30] NeerbyFeed, http://www.nearbyfeed.com/
[31] Facebook Places, https://www.facebook.com/ places/.
[32] Sonar, https://play.google.com/store/apps/details?id=me.sona r.android&hl=en
[33] TNS, "Two thirds of world's mobile users signal they want to be found", http://www.tnsglobal. com/press-release/two-thirds-world´s-mobile-users-signal-they-want-be-found, April 2012.