Help Me: Opportunistic Smart Rescue Application and System

Osnat (Ossi) Mokryn  Dror Karmi  Akiva Elkayam  Tomer Teller
School of Computer Science
Tel Aviv Yaffo College, Israel
{ossi,drork,aahivaelk,tomertel}@mta.ac.il

Abstract—Communicating during disaster times is crucial for both survivors and rescue forces. While fast reaction is critical, communication infrastructures, wired and cellular, are often lost, and cannot be restored in a timely fashion. In this paper we present HelpMe, a self-learning opportunistic ad-hoc system, which enables smartphone-based ad-hoc communications at disaster times over Wi-Fi. HelpMe smartphone peers communicate using a sophisticated mechanism that performs a transparent on-the-fly classification and matching of requests to peers in the formed opportunistic ad-hoc network. Matching is further leveraged for a smart forwarding, enabling the request to reach the best matching user in the vicinity.

Our system enables best peers matching across an opportunistic ad-hoc network on a hop-by-hop basis, in a timely and power conservative manner. Location coordinates are sent with each request. The client is built on top of the Haggle middleware, leveraging its neighbor discovery and interest-based forwarding. The HelpMe client is fully implemented as an iPhone application on top of the Haggle middleware. The HelpMe system consists also of a HelpMe cloud-based server, used only when communication is available before and after the crisis. The server is used for profiling users and creating personalized apps for the users. When communication is restored, it can be leveraged for collecting information for missing persons services. The server is implemented as a web service. We tested the system using several iPhone / iPad clients communicating over Wi-Fi and showed that our settings not only enable a best match, but also enable willing users to become hub nodes in the formed opportunistic network. The system is self-adjusting and supports on the fly settings modifications.

Index Terms—Opportunistic Networks, Ad-Hoc Applications, Self Learning Applications, Disaster recovery.

I. INTRODUCTION

When disasters strike, traditional communication and cellular access points may no longer be available, denying communication when it is most needed. Recent disasters such as the Indian Ocean and Japanese tsunamis, or the Haitian earthquake have caused unprecedented damage and massive loss of life, and unveiled the necessity of immediate ad-hoc communication solutions for rescue forces and personnel. The research community has invested in the construction of management systems for communications at rescue areas, designing effective systems that provide timely access to comprehensive, relevant, and reliable information that is critical to humanitarian operations, as well as operations management [1]–[5].

While coordination and operations management is of the highest importance, there is a need for readily available smartphone-based opportunistic communication. An ad-hoc application that can be operated at a keystroke, and communicate with neighboring smartphones over Wi-Fi is crucial at the first hours until organized rescue forces arrive. Such communication does not only enable survivors to convey needs, but can also facilitate the organization of spontaneous, self-organized rescue operations. At later times, the system can be useful for communicating with survivors in need and for exploiting unofficial personnel for help and rescue missions.

In this paper we present HelpMe, a system solution for aid requests in crisis situations, when Internet connection is not available, between smartphone users situated in close proximity. HelpMes primary information stream is based on a request-reply exchange between different users. Each user has a user profile, and a personalized application, constructed by the server. The user profile is the avenue by which questions and answers are routed between smartphones.

In order to achieve successful routing of the request to a suitable user, the request itself is classified and matched. The request is then forwarded to the best matching neighbor who in turn performs local classification according to its local profile and tries to match again. If a better match is found, the request is forwarded again. This forwarding mechanism enables the system to extend the reach of each user well beyond her local immediate vicinity. The system trade-offs matching and distance to reduce delays, taking into account the need in fast response in such situations. Local adjustable settings enable users to determine the amount of information they are willing to receive. For example, unskilled users might be willing to become hub points if they have access to power. Adjusting their settings enables them to become forwarding hubs in the formed opportunistic network. On the other hand, on-the-go rescue experts that happen to be in the neighborhood when disaster strikes may set their app to accept only rescue related queries, to preserve battery life.

We have designed a classification system, which consists of an extended dictionary, containing categorized questions for each emergency call. Our server algorithm derives a partial dictionary, adapted to each user according to her user profile. This result in accurate classification and consequently faster response times. Both the skill set and the partial dictionary are used in the crisis ad-hoc situation for classifying the user-typed contents. The classified content is then forwarded to possible matching peers in the formed opportunistic ad-hoc network.
network, to allow for the best possible aid. At the receiving end requests are re-classified using the local dictionary, and if a better matching peer is found the request is forwarded along without notifying the user.

Location coordinates are calculated automatically and attached to each request, and all communications are stored locally at each device. When rescue forces arrive and communication is restored, post-operation processing at the server can relay the location information to missing persons services and initiatives.

The HelpMe client is fully implemented as an iPhone application in c/ c++/ objective-c, and contains over 13,000 code lines, compiled as a single iPhone/ iPad/ mac app. The Haggle middleware [6], [7] was leveraged for neighbor discovery and content-based forwarding. Haggle was tailored to expose further functionality and its iPhone drivers adjusted. The web service is written in the .NET framework, and has about 4,000 code lines. We then experimented to validate the effect of the local settings on both the routing and the amount of information routed through each device. We show that users may become hubs, while other users minimize usage to preserve power.

II. RELATED WORK

A. Information Sharing

Recent tremendous success of social networks has revealed a true social need for online communication and the almost unlimited potential of sharing information. The information gathered in the frames of the different social networks can be optimized to adhere to users’ needs and requests with the use of different programs such as Aardvark [8]. Aardvark satisfies users’ needs by finding the right person to answer and trust is based on intimacy. One may look at the Aardvark mechanism as solving the problem of finding a minimal coverage within a social network that gives a high enough probability of obtaining an answer, while the HelpMe system looks for a minimal coverage over a small opportunistic network.

To obtain the right answer, a question is first classified according to its content. In [9] the authors show that it is important to classify questions with respect to their answer types. To build a very detailed question taxonomy, [10] classified 18,000 questions by their answer type. [11] showed that by enriching parse trees semantically, based on answer types, they obtain a more accurate answers. [12] suggested the use of web-based search engines, such as Google and Yahoo to train classifiers. The HelpMe web classifier is trained in a similar way, and the process is explained further in IV.

In [13], a Task-Adaptive Information Distribution method is presented for rescue forces, according to a defined workflow. The information is gathered at a central location, and all the relevant information is routed to the relevant task forces upon arriving. Data classification is based on Naive Bayes, and routing is planned in a centralized manner according to the different disciplines.

B. Haggle Middleware

The HelpMe uses Haggle [6], [7], as a middleware that enables seamless network connectivity in dynamic mobile environments. Haggle uses a data-centric network architecture, and an interest-based forwarding. An application can subscribe its interests to data objects according to specified attributes or attribute values, and will receive data objects that match these interests. Haggle handles all communication aspects, including neighbor discovery, thus separating the application from the network.

Haggle is a chosen platform for a variety of ad hoc middleware and applications, e.g., [14]-[17]. MobiClique [16] is a mobile social software. When mobile users meet opportunistically, and if the two user profiles share some pre-defined relationship, the users are alerted and can choose to have an exchange, and possibly become friends.

In a recent paper [17] Haggle is used as a middleware of an application that triages victims in a mass casualty incident, transmitting this information to the emergency Coordination Point from the emergency area. In this work, much like in ours, ad hoc opportunistic communication is exploited without relying on unstable communication infrastructure, nor on the deployment of a new infrastructure.

III. HELPME SYSTEM ARCHITECTURE

During the first hours when disaster strikes and before help arrives and organizes, survivors are on their own. However, it is very plausible that given means of communications, they will be able to help each other. Among the survivors might be medical professionals, police or fire personnel and volunteers that given the opportunity would be able to help. Our system is designed to enable users to communicate using their smartphones in a peer-to-peer manner even if all communications lines are down. Our system enables on-the-fly routing of requests to the best qualified person that is still within a reasonable distance from the user requesting help, while leveraging other users as peering points.

The HelpMe system overview is illustrated in Figure 1. The system consists of a centrally located web service, and a smart phone based application, currently implemented as an iPhone app. The app has two basic modes. When Internet connectivity is available the app operates in a client mode, in which it communicates with the server at registry and update sessions. Otherwise, a peer mode is used during disaster times in an ad-hoc fashion when no other communication means is available.

A. Prerequisite: App personalization

Each app contains a built-in profile, personalized dictionary used for text classification, and attributes describing the user. This information enables the stand-alone functionality of the app. Our algorithms at the server build a personalized dictionary per user. The personalized dictionary is then used by the app to classify outgoing and incoming requests and replies, and this classification is used for matching requests to repliers. Text classification requires a trained database, consisting of possible categorizing of words. The bigger the database, the
better is the classification. When classifying a text, each word is searched for in the database. Clearly, searching through a large database imposes significant memory and power requirements. Smart phones, as advanced as they become, have storage, memory and power limitations. Moreover, during ad-hoc operation power reservation is crucial. To adhere to these requirements, The server downloads only a partial database to each user app, tailored for the user based on her profile. A database tailored for general users contains the most common words in each category, while a database tailored for skilled professionals is adjusted and more tuned to the main categories and sub categories in the user’s profile. While partial databases are crucial for a light weight smart phone app, the negative impact is that text cannot always be classified correctly, or classified at all.

Users are classified according to their skills and professions. A general, unskilled (i.e., have no medical, rescue or police related skills) user is assigned a default user profile. The default user profile includes default attributes and a basic general dictionary. Skilled and professional users, however, have more complicated profiles and dictionaries are built according to their skills. For example, a user who is trained for CPR is classified as medical personnel with a high rank in CPR. The corresponding generated dictionary is then biased to CPR related terms. A physician is given a more detailed medical profile, with high ranks attached to aspects that lie within her areas of expertise. The system creates a medically biased dictionary according to the indicated skills.

The personalized app, including describing attributes and a personalized dictionary is generated by the server and downloaded to the smartphone.

B. Disaster ad-hoc operation

Upon activation, the app enters a neighbor discovery phase, during which initial information is exchanged with the neighbors. At the end of this short phase, the app is aware of all other users within its vicinity, and their attributes. The entire life cycle of a HelpMe client / peer is depicted in Figure 2.

The user profile is the avenue by which requests and replies are routed between smartphones. When a user types a query or a request, the app performs an on-the-fly classification using the local partial dictionary that was downloaded during the registration phase. The request is given attributes based on local classification. The attributes are used in a matching algorithm and the request is forwarded to the best matching neighbor.

The matching algorithm takes into account the following parameters per each neighbor:

1) Neighbor’s categories and her ranks at each categories. Ranks are calculated online during operation for active users based on their responsiveness. Additionally, users may give a thumbs-up, thumbs-down rating to the aid they have received from the user, and affect her rank in the specified category.

2) Neighbor’s availability settings. Availability settings determine the amount of information the user is currently willing to receive. The default mode is set to normal operation. In this mode matching is based solely on the categories and ranks. Three additional availability modes exist, however. In a receive-all mode, users can become forwarding hubs for requests. Alternatively, users can reduce the amount of requests they receive to match only specific categories in their profile.

At the receiving end, the request is classified again using the receiver’s local dictionary, and local attributes are computed for it. Based on these the request is either displayed to the receiving user, or forwarded to a better matching neighbor. The process can repeat itself several times. The matching algorithm takes into account the number of times the request has been forwarded and forwarding stops when the improvement in the matching compared with the aggregated delay are not sufficient. The motivation behind this scheme is to allow for
a better routing of requests, such that requests are more likely to arrive to the best matching user around in a timely fashion.

The HelpMe peer app leverages Haggle engine as a middleware for neighbor discovery and data exchange. During Haggle’s neighbor discovery process, peers exchange their interests. This information is usually kept at Haggle’s internal database. We enhanced Haggle code to expose this information to the application. HelpMe app is thus aware of its neighboring ad hoc peers and their profile without exchanging information at the application layer. Requests and replies are exchanged over UDP between the Haggle middleware daemons as data objects, with interest attributes that correspond to the obtained classification.

C. WiFi considerations

During ad-hoc operation, peers that communicate directly are within WiFi transmission range of each other. Neighboring peers are in the same vicinity of each other, either outside or within a building. Requests are forwarded to a more destined peers by neighboring peers, and a request can travel several hops in the formed opportunistic network. [18] conducted real life experiments with WiFi signal transmission range due to attenuation within buildings and outdoors in the presence of obstacles for mesh networks. Hence, transmitters with different capabilities were placed within interference range from other transmitters. The received power of a transmitted signal was modeled using the log- distance path loss model. Namely, at distance \( d \) (meters) from the source the received power (in \( dBm \)) decays as a function of the log of the distance. Their results confirm the model. They further show that even with 802.11b, the signal is received (-80dBm to -90dBm) within 50 meters of the source. In an outdoor scenario the signal was very strong (-70dBm) at 50 meters, indicating that real range can vary up to 80 meters and more. Hence, an opportunistic ad-hoc HelpMe network of only five hops can span more than 250 meters, in an urban environment.

IV. Creating Personalized Apps

A. Preprocess: generating the user profile

![General Category Tree](image)

Figure 3. General Category Tree

Upon registering, the HelpMe service will prompt the user to indicate its professional skills (e.g. Fireman, Doctor, etc...). The HelpMe service will then create a category tree based on the user profession. A user that did not indicate any skills will be given a general category tree, as shown in Figure 3. However, a skilled user will obtain a weighted category tree with additional weights given to categories within its skills. On top of that, the HelpMe service will create a tailored weighted dictionary based on the users category tree. This tailored dictionary is substantially smaller than the servers dictionary and is comprised of a basic set of words per each category and additional words that are more likely to appear in requests within the users expertise and skills. The words are extracted from the servers full dictionary, and are given with their likely category, and the corresponding weight. The weight is determined according to the words frequency in the training text for this category.

For example, the following is a small fraction of a partial dictionary downloaded to the smartphone application:

- out => Fire and Rescue Service [2%]
- need => Emergency medical service [6%]
- Fire and Rescue Service [6%]
- blood => Emergency medical service [2%]
- ladder => Fire and Rescue Service [1%]
- danger => Fire and Rescue Service [1%]
- help => Fire and Rescue Service [4%]
- Emergency medical service [9%]

HelpMe’s novel approach of using partial tailored dictionaries at the smartphone is aimed to save space and power. However, to calculate the level of correct classification achieved by a partial dictionary we designed an experiment to evaluate the validity of our assumption - that partial dictionaries enable a reasonable classification. We used experimental data for question classifications distributed by [19]. The questions in the distribution list are already classified. A partial dictionary is built based on the user’s weighted category list. Dictionaries of different sizes were created for a general user, and the data was classified with them. The results were compared to the results obtained by a web based full dictionary. Figure 4 shows that partial dictionaries achieve more than reasonable classification results, compared to the original classification of the questions. Surprisingly, even a partial dictionary as small as 20% the size of the original dictionary has managed to classify correctly more than 65% of the questions.

Clearly, 65% an be perceived as unreasonable. However, classification can always use the general root category all. Also, queries are classified also at the receiving end, based on the receiver’s tailored dictionary. Hence, the chance of a repeated non-classification is smaller.

V. Ad Hoc App Standalone Operation

HelpMe app is activated when other communication means are not available. Hence, classification is performed locally at each smartphone, based on the local dictionary installed.

A. Ad-hoc classification

A user typed content is parsed into words and stripped off commoner morphological and inflectional endings of words using the Porter Stemming Algorithm [20]. Each word contributes to the classification of the sentence. Each dictionary word \( W \) is associated with a list of numerical scores correlated
The list of categories holds only user relevant categories for each word $W$ s.t. non relevant categories do not affect the classification. The classification is done based on the union of all the word scores $W_{0..n}$ in the message

$$\bigcup_{i=0}^{n} \{P|W_i\}$$  \hspace{1cm} (1)

Subscribing on interests: given a list of $n$ user categories $\{c_1, c_2, ..., c_n\}$, and a weight $w_i$ for each category $c_i$ s.t. $i \in [1..n]$. The weighted user categories array is then sorted in a descending weights order:

$$(c_1, w_1), (c_2, w_2), \ldots, (c_n, w_n) | i < j \implies w_i > w_j$$  \hspace{1cm} (2)

The top level categories are then subscribed on Haggle interests, with values that correspond to the categories’ weights.

### B. Matching based opportunistic routing

Based on a local profile each query is classified to create a weighted category list, or interest list. The matching algorithm then calculates the best neighbor to receive this query, based on a weighted interest list exchanged at the discovery phase and availability settings. The classifying and matching process is repeated at the receiving end, and the query is either forwarded further or ends its journey and is displayed to the receiving user. HelpMe employs an additional filtering level on top of Haggle. Requests are routed to the most skilled professional, even if several hops away.

The matching algorithm matches a query to a neighbor, by comparing the neighbors’ categories and their ranks to the weighted category list generated by the classifier, in accordance with local availability settings.

1) **Ranking algorithm:** A user improves her rank in one of the categories when answering a relevant query. The intuition behind this is that an active user who helps and responds is preferable over a non active one. Each user receives a ranking for every category in which she is active. The ranking system is based on a bottom-up model where depending on the amount of activity and quality of reply/service given, the user will be given credit that is higher than, lower than or equal to the credit of other users. A rank of a user in a category determines her level of activity and relevance. A HelpMe peer with two matching neighboring peers for a request will always prefer the highly ranked neighboring peer. A user with a higher rank in a category has shown to participate extensively and over time in activities in this category, received positive feedback ratings, and is therefore more likely to reply to the request and in a timely fashion. Additionally, users rate the answers / aid received using an thumbs-up/ thumbs-down scheme by which they may respond to the reply. In the personal category tree, the major categories are the nodes at the first layer of the tree from its root. Each node also carries a weight, which is the ranking of the user in this category, summed over its ranking in all relevant subcategories, if such exist. The categories are subscribed as weighted interests, and sent to neighbors at the discovery phase. The ranking is conducted through points, whereby the user gains more credibility in her user profile according to how high she is ranked for each category. The point system is scaled from low to high: the lowest score is gained by a request, the next up is by answering a request, and the highest is given to a user that received a rating feedback to her reply by the requester. Each score is per category or sub category. However, users’ score decay slowly over time, to accommodate both periods of inactivity, natural for rescue scenarios, as well as the possibility of reduced user’s responsiveness.

2) **Receiver’s availability settings:** HelpMe users can control the amount and type of information they wish to receive, by setting their availability to one of the following:

- **Accept All Requests:** To accept all requests regardless of their type, the matching function between the peer’s interests and every published request needs to always be satisfied. The peer then subscribes on the interest "type of the message = request” with value $T$. Users that set their sensitivity to all can becomes hubs who re-classify messages and forward to other users, possibly on different networks.
- **Accept Requests by General User Profile:** This is the normal Haggle based forwarding metric for propagating requests to peers.
- **Accept Requests by Specific User Profile:** The HelpMe app applies further filters that take into consideration specific categories in the user’s category tree, and their corresponding weights. This enables the receiving side to filter out queries within her categories and receive only specific requests.
- **Accept only Critical Emergency Requests:** Here, the user subscribes only to one special critical emergency interest.

Users can reduce the amount of information they receive (low availability) compared to a Haggle based system with no additional filtering. Interestingly, a HelpMe user that requests to receive all requests (high availability), will receive more messages than a corresponding Haggle user. An interesting implication of the availability is its use for forwarding and routing. A user with a high availability can become a hub for requests and re-route them to a better match that was not in the requester’s close vicinity.

3) **Matching rules:** Given a peer $k$ with $m$ subscribed interests, $I_k = \{(i_1, v_1), (i_2, v_2), \ldots, (i_m, v_m)\}$, where $i_j$ represents an interest $j$ in an interest vector $I$, and $v_j$ represents...
it’s value. If the sum of the values of a peer’s interests that are matched by categories in a request is higher than a given threshold, the request is sent to the peer. I.e., given a request \( R \) is classified to categories as follows:

\[
\mathcal{c}_R = \{c_1, c_2, \ldots, c_n\}
\]

The request \( R \) is matched to peer \( k \) if:

\[
\sum_{\mathcal{c}_s \in \mathcal{c}_R, \mathcal{s} \in I_N, \mathcal{c}_s \in \mathcal{I}_k} v_s \geq T
\]

Where \( T \) is a predefined threshold, and \( c \equiv s \) if the two name strings are the same. Our design takes into account the time from the initial transmission of the query, and matching is stopped if the elapsed time is bigger than a predefined \( \delta \) value.

C. Routing the Reply

The Reply has to be routed back to the peer that issued the request. To support that, HelpMe adds both source UserId and target UserId to both requests and responses. Although haggle has means to keep the identification of peers, HelpMe uses the unique ID given by the server during registration. The Source User Id is sent as one of the requests attributes. It does not affect the dissemination of the request, and thus should not have a weighting factor that interferes with the matching algorithm. The Target User Id is also sent as one of the reply’s attributes.

VI. IMPLEMENTATION AND EXPERIMENTS

The server was developed in the .NET Framework 4.0 over SQL Server 2008 R2, and consists of more than 4000 code lines. The iPhone app was developed in C++ and Objective-C using Apple’s Xcode environment, and GUI Interface builder. The HelpMe app without Haggle contains around 13,000 lines of code.

A. App implementation

The implementation of the HelpMe client consists of several modules that encapsulate the logic of the user profile, categories and dictionary, requests, replies and classifications. The helpMe engine interconnects the Haggle infrastructure and the iPhone GUI, storing all the relevant data both in a proprietary persistent storage and in transient memory data structures for fast access. The HelpMe client employs a dedicated module to support access to the HelpMe server.

1) Taming the beast: Haggle: Haggle is written in c++ and ported to several platforms including Windows mobile, Linux and iPhone/Mac. We integrated the core of our application with the Haggle infrastructure. To avoid jailbreak the iPhone the Haggle is used in user mode, and is integrated directly with the application, to form a single HelpMe Xcode project.

The main modifications to Haggle include:

1) Compilation issues with iPhone SDK, we had to modify several parts, including external libraries like SSL/Crypto that didn’t compile well on the new iPhone SDK.
2) Support missing functionalities for the latest iPhone platform, e.g., functions for the network discovery.
3) Expose internal data to the application.
4) Integrate the Haggle Kernel to run as an application thread.
5) iPhone environment changes: directory use as well as global defaults.

b) App functionality: Figure 5a and Figure 5b show the registration process in action. When a user downloads the HelpMe app she needs to first register with the server, and specify skills she may have. Upon processing the information the server creates her profile, a ranked list of categories and a tailored classifier partial dictionary. It creates a personalized app containing the above listed information, and downloads it to the user. Once the user has obtained the personalized app, she can operate it in a standalone ad-hoc mode. Once in a standalone mode, the app can communicate with other apps without any server support.

Figure 6a and Figure 6b show the Splash screen of the HelpMe app on the left, and the neighbors screen presenting nearby peers on the right, after the initial discover process.

B. Server side functionality

The server’s major components include

1) Database: An SQL database support for fast updates and efficient analysis of data.
2) A two tier classification engine: The Naive Bayes is implemented over the math module of the .NET Framework, and its results are updated using the SQL server. The search engine classification algorithm uses the Google and Bing search engines’ API.

3) The ranking and profiling logic is implemented over the Linq2SQL .NET framework.

4) Interfaces API: The server interfaces with the HelpMe smartphone app.

5) Web Service: A web server dedicated to tracking the activity of the system was built. The service was implemented over the .NET framework 4.0 web application platform. The service also contains a reports service.

C. The Effect of the availability settings on routing

The main advantage of the HelpMe system lies with its ability to form an opportunistic routing network. The request is routed to the best matching peer based on its content using classification, ranking and matching algorithms. To further measure and understand HelpMe’s availability feature and its affect on the routing, we have compiled a set of rescue related requests. The database consists of transcripts of emergency calls we gathered over extensive web searches, and assembled to one exemplary database.

The experiment was performed over a setting that included several Apple based devices such as iPhones and iPads. Four experiments were conducted, with several peers emulating people in a crisis situation. In our scenario, two of the peers represent skilled personnel: a fire person and a medic, and the rest of the peers represent victims that send requests hoping for help or advice. In all experiments the same requests were sent by the same peers, in a similar order of events, but each time we used different users’ availability settings. Figure 7 shows the results of the experiments held.

In the first experiment, all users use an accept all availability. The victims are initialized with a default dictionary. The personnel are initialized with general dictionary, but choose their specific skill. In this case, all requests were directed to all personnel and all victims, much like Haggie. In the second scenario, the victims set their availability to emergency only. As a result, they did not accept 95% of the requests. At the third experiment, the personnel set their availability to general user profile. The results show that 95% of the requests were delivered to either the fire or the medical personnel, according to the right classification. Specifically, the medic received 59% of the requests and the fire personnel got 36%.

To perform the 4th experiment, we held a preliminary stage, in which the personnel replied to requests and received ranks. The personnel needed to reply to the requests during the third experiment to receive ratings and improve their ranking. We set the medic to be a trauma doctor, and she answered 90% of the requests she received, 80% of which were trauma related. 60% of the ratings were positive, 20% negative, and 20% of the replies were not rated. We then repeated the experiment for the 4th time, this time the personnel set their availability to specific user profile. The trauma doctor received only 14% of the requests, that were trauma related, out of the 29% medical requests sent. 10% of the requests were not classified well, and were not delivered at all.

D. Server Statistics

Figure 8 presents the distribution of messages per categories over all users, as registered at the server. The distribution is presented as a percentage out of all registered messages. For lack of space, categories were numbered.

Figure 9 displays users’ activity distribution in a correlated scenario, as registered at the server. The scenario examined is the one described in Section VI-C. Users’ activities in this scenario are presented according to categories.

VII. CONCLUSIONS

We have built a smartphone-based platform for ad hoc communication between skilled personnel and the general public, as well as within the skilled personnel groups for disaster times when no other means of communication exist. The system targets the time before organized help arrives, and many people are in the vicinity of each other without means to communicate their needs. As some of these people might be
skilled rescue and medical professionals, the system enables them to receive requests for help or advice and respond.

HelpMe builds a content based opportunistic network, and requested are routed to the best matching peer to respond. Our system uses learning mechanisms adjusted to the smartphone environment to enable a content based routing. We show that partial classification dictionaries perform well and obtain accurate classifications. We implemented both client and server fully. We further show that by having an additional filtering mechanisms on top of Haggles matching algorithms we are able to further personalize the routing of requests, and enable willing peers to become distribution hub nodes in the formed ad-hoc network.

REFERENCES


