# Context Aware Collaborative Computing Model for Natural Disaster Management Systems

Hamid Mcheick[1], Raef Mousheimish[2], Ali Masri[2], and Youssef Dergham[2]

[1] Department of Computer Science, University of Quebec at Chicoutimi (UQAC), 555 Boulevard de l'Université Chicoutimi, G7H2B1, Canada
hamid_mcheick@uqac.ca

[2] Department of Computer Science, Faculty of Sciences (I), Lebanese University, Rafic Hariri Campus, Hadath-Beirut, Lebanon
{mch.raef, alimasri1991, youssefdergham}@gmail.com

**Abstract.** Nowadays, natural disaster management is considered one of the critical issues, where many governments are spending a huge amount of money to master it. And to help these governmental bodies in managing this kind of situation, we used the concept of collaborative computing, to introduce an approach for mobiles to collaborate in order to act as helper agents for other ones with limited resources. Our approach is called the Disaster Pool. And in this paper we highlighted the importance of collaborative computing, have a quick look on previous work, and discuss our approach and the implemented code.

**Keywords:** Collaborative computing, context-aware applications.

## 1    Introduction

Looking at our reality, we can easily notice that every person now is counting on the mobile devices to accomplish many tasks in his life, ranging from simple basic tasks, and up to complex and business ones. The continuously evolving capabilities of mobile devices, in all their aspects, like the computational power, speed of processing, the rapidity of interactions, and the list would go on… All these technological advancements affected the mobile world and the usage of its devices, so research and industrial communities are both working on reaching the maximum benefits from these devices, and exploit their capabilities and potentials in so many domains, like military, sport, business, and especially in the field of health. Where mobile devices and counting on the fact that they are always with their users, can be really helpful in monitoring and guiding users to preserve their safety and their health, and more importantly these devices can act as a life guards in case of a disaster to help and guide the users. And this is exactly what we tried to express in this article.

When we talk about health, we can't ignore the increasing usage of mobile devices, and we can easily see the wide spread of mobile applications, concerning fitness, monitors, and the disaster management applications. The last ones are the topic of interests now, because any person's first concern is his safety, and if we have a guide that is

always beside us, and we can blindly count on him, then we need to fear less about our lives, and not to mention that this last type of application is really important to highlight the concept of context-aware applications, where the device is aware of its environment and can benefit from the resources offered by the components of this environment. So after reading and surfing the current works on disaster management, we tried to leave our own mark in this field, and we merged in our approach the world of mobile and collaborative computing, and indeed we added the awareness and the smartness to our solution by using some of the context-aware techniques.

This paper is organized into five sections, we first show some previous works that are related to our approach, and then we discuss our motivation which carried us to write such a paper, we will then detail our approach with the implementation, and finally conclude with our conclusion and mention some of the future remarks.

## 2    Related Works

We do not intend to perform a complete review of what is context in this paper, interested reader might refer to *Brezillons*work [2]. The reader is referred to *Chen and Kotz*[4] and *Korkeaaho* [5] for a more detailed list on the projects and researches in the context-aware field.

Briefly, the definition of context is not satisfied in general. Many researchers have defined context by giving examples of contexts. Schilit divides context into three categories [11]: i) *Computing context*, such as network connectivity, communication costs, and communication bandwidth, and nearby resources such as printers, displays, and workstations. ii) *User context*, such as the user's profile, location, people nearby, even the current social situation. iii) *Physical context*, such as lighting, noise levels, traffic conditions, and temperature.

Dey *et al.* [10] defined context as any information that characterizes a situation of any entity. An entity is a person, place or object that is consider relevant to the interaction between a user and an application, including the user and applications themselves, and extension, the environment the user and applications are embedded in.

In *Chi-Sheng Shihet al.* [1], the authors tried to build a context-aware architecture to help users in disaster situations, by contacting some specific servers to get instructions and directions on how to act in such situations.

Some other examples on health care context awareness projects would include, the Vocera communication system [3]. Hospital of the future centre for Pervasive Health care, Denmark [6][7] where a context-aware prototype is proposed. And Context-aware mobile communication CICESE, Mexico [8].

## 3    Motivation

Clearly the growth of the smart-phone market is huge and is expected to continue growing, therefore, such devices have already being hailed as the next wave in computing.

Smartphones are predicted to become nearly ubiquitous and are thus a major step towards the vision of ubiquitous computing so often dreamed of. The combination of pervasive wireless networks and computational devices has created an era of mobile computing, the likes of which have never been seen before.

In large scale disasters, the network is a significant problem, and thus many of the constraints that we place on our work are centered on dealing with the loss of networking, while still enabling distributed computing when the network is available. We want to create novel distributed applications which take advantage of the distribution of the nodes in powerful ways.

Our approach suggests to make use of available resources and application almost in every smart-phone (Bluetooth, Wi-Fi, Hotspot, 3G etc.) and to collaborate the use of these resources between nodes (mobiles, cells, body sensor networks) in the disaster area (pool).

## 4 Disaster Pool Model

Our contribution is named **The Disaster Pool Model**. In the pool we could find smartphones and/or body and wearable Sensors. The smart phones interact with the cloud through 3G, Wi-Fi and Cells. Other sensors interact through Bluetooth, Wi-Fi-hotspots and Wi-Fi, trying to connect with the nearby available smart phones.

If a smart phone is suffering from a lower bandwidth, no signal or internet access, it could use other phone in the area to communicate with the cloud, or to execute a code and so on. Figure 1 shows an overview of our approach.
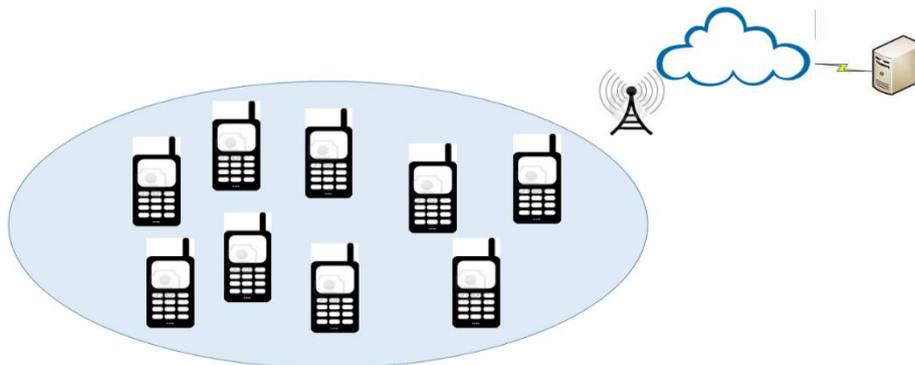


**Fig. 1.** A pool of collaborative mobile devices (smart phones and body sensors)

Counting on our design of the disaster pool approach, and more specifically the heavily usage of the collaborative computing concepts and the context-aware techniques, would turn so many useful fictional activities to become real, possible and feasible ones, we

will count some of these activities, Let's take two smart phone **A** and **B**, and one body sensor, that is, **C**, the three devices are classified by the server to be in the same pool:

- **A** suffers from the lack or the speed of mobile data connection, so it can use the data from the near smart phone **B** using the Wi-Fi hotspot. So the server framework would enable automatically the hotspot on **B** and provide the password to **A**.
- **A** suffers from the lack or the speed of mobile data connection, so it can connect to a near Wi-Fi, this could be possible because the server registers the Wi-Fi provider in the area, and authenticate **A** to let it use the connection reliably.
- **A** suffers from a dead battery, and **C** usually send its monitored data to the server through **A**, in this case the server turns on the Bluetooth on **B**, and then **C** could send its data about the human handler of **A** through **B**, to continue monitor his health conditions.
- **A** suffers from the lack of computational resources, so **A** could send his data to be executed on **B**, and then get the results back.

Talked about the applicability and some of the desirable outputs of our approach, we will step now to discuss an important construct of the whole application, that is the communication protocol, and we tried our best to make it as simple as possible. So two keywords were used.

The first one is **REGISTER** and it's sent from the mobile device to the server, so in this way the server could track the mobile devices, know their locations, classify them into pools, and so in case of a disaster, and if a mobile device asks for help, the server could propose a near registered mobile device to be used.

Secondly and after mentioning the help procedure, asking for it can be done using the second keyword, i.e. **HELP**, this keyword can be sent from a mobile device to another one, and then followed by the class that needs to be executed, after this the demander could get the response from the helper. Also the keyword **HELP** could be sent to the server and in this case the server returns an IP address for a helper that is near the demander. The following figure depicts more clearly the concept of our interactions:
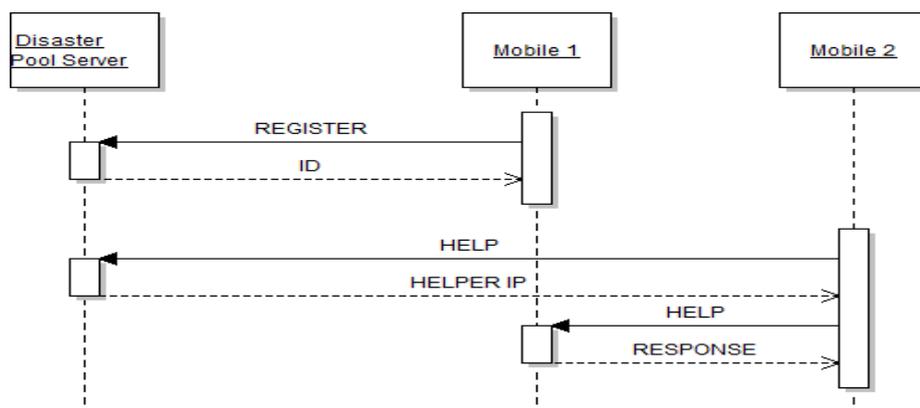


**Fig. 2.** The commands of the protocol depicted in a sequence diagram

# 5       Implementation

To put some life in our approach, we chose to implement the way of communication between the mobile phones together. Two programs where implemented using the JAVA programming language and the eclipse development environment with eclipse ADT plugin. The first one represents the disaster pool, i.e. the server, and the second one is an android mobile application.

Our implementation simply describes the fact that a mobile has a limited amount of resources in case of emergency as in disasters, and needs some other mobile to carry out the communication between it and the server or to execute the code for it. The demanding mobile is called the **demander**, the mobile used for resource sharing is called the **helper** and finally we have the old traditional **server**.

First of all, if a disaster occurs and the demander was unable to communicate with the server, it will connect to a lookup-server that can be found nearby, - A solution is that these servers could be provided by the government for this use - , then the demander asks the lookup server for a helper that is able to handle some kind of computations specified by the properties sent, and the lookup server by its way will return the desired helper. Another approach is to communicate directly with another mobile via a Wi-Fi hotspot, since at disaster time, mobiles will be alerted to open Wi-Fi hotspots for others with low resources to connect with. After selecting the helper, the demander sends it some data (files, script, parameters, etc.) to be executed, and this code communicates with the server or gets processed locally to get some data to be used. Figure 3 shows clearly our system design:
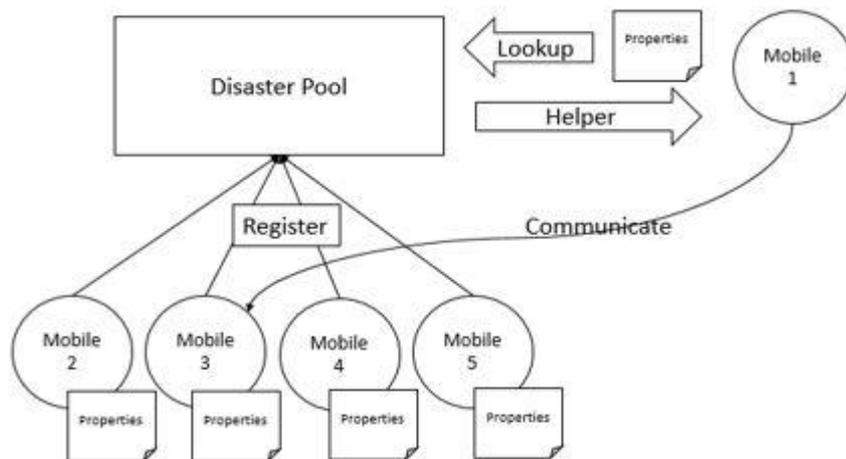


**Fig. 3.**The overall system design

Concerning the Disaster Pool, this java application runs as a server that keeps listening for registering clients, or for demanders that requests help. All the registered clients

will be saved in a collection 'Pool' so they can become helping candidates later. When a demander requests a help, the disaster pool will seek for a helper with good resources and pass back its address. The code of the service is a little bit long, because we took into account some of the multi-threading and the authentication issues, but in the following listing, we shows how the servers works on accepting the connections from clients and creating a thread for each client.

```java
public void runServer() {
    System.out.println("Server is running...");
    while (true) {
      try {
        newClientThread(server.accept(), clients, id++);
        System.out.println("New client attached");
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
  }

  public static void main(String[] args) {
    DisasterPooldPool = null;
    try {
      System.out.println("Creating pool...");
      dPool = new DisasterPool(1234);
      System.out.println("Pool created");
      System.out.println("Running server...");
      dPool.runServer();
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
```

**Listing. 1. a piece of the java code of the server**

Now crossing towards the mobile application, and as we stated earlier, it's an android application that has two main jobs, the helper and the demander. When the mobile phone runs it will connect to the disaster pool and registers itself to be a helping candidate, and it will run as a server for demanders who will ask it for help. The help could be requested by using the collaboration between the two parties, like the demander would send a .class file to the helper followed by some additional parameters indicating the method's name with its arguments if needed. The usage scenarios of this application are unbounded and the advantages are very good since the .class file sent is very small. But one disadvantage is the privacy, in which it may not be important in case of emergency. Where people usually, care less about their private data when they are in an emergency situation, and care more about saving their lives.

The code of the mobile application is quiet complex, this is because of the android added files, like the manifest file, the layout one, and we took into account, in our coding, the steps to create a background application that doesn't affect the resources of the android device, from all the different perspectives, like the battery, the CPU, the RAM, and so on… Thus the impact of our application on the android device was kept to minimum. Almost all the Java code for this mobile application, is important, where it's divided into separate tasks and each one of these tasks, handles an important job. The following listing for example, illustrates the helper task, where the helper is programmed, and in the background, to wait for a connection from another peer, process the help request and respond with the appropriate results.

```
classHelperTask extends AsyncTask<Void, Void, Void> {

    @Override
    protected Void doInBackground(Void... arg0) {
      while (true) {
        output.println("REQUEST_HELPER");
        etLog.append("Requesting help...\n");
        String helperIP = input.nextLine();
        try {
          Socket socket = new Socket(helperIP, 1234);
          Scanner helperIn = new Scanner(socket.get-
InputStream());
          // PrintWriterhelperOut = new
          // PrintWriter(socket.getOutputStream(), true);
          etLog.append("Message from helper: " + help-
erIn.nextLine());
          helperIn.close();
          if (socket != null) {
            socket.close();
          }
          break;
        } catch (Exception e) {
          etLog.append("Error connecting with helper\nRe-
trying...\n");
        }
      }
      return null;
    }
  }
```

**Listing. 2. The helper task android code**

## 6　　　Analysis & Benefits

After implementing and trying the work we have done, we came across an analysis phase, where we clearly and practically defined the effectiveness of our solution, its light impact on the resources, its reliability and the benefits from using such approach.

Experimenting and concluding depending on two independent experiments. The experiments goals is to depict the effectiveness and the reliability of our application and the communication protocol. So two android mobile phones were registered successfully at the server, which is a java code, executed at a remote laptop. In the first experiment we tried to ask the server for help, and the server replies with the IP address of the other android device, and then a peer-to-peer connection took place between the two near devices, the demander of the help sends the HELP command to the helper device, and then the class that needs to be executed, in this way preserving the power and the resources of the demander. We didn't specify a class, because it could be anything, it could be processed locally or it could access the internet, make calls to the server and so… In the second experiment, we detected and contacted directly the helper device, and send it the class to be executed. And indeed in the two cases, we got the expected results from the operation.

Using this approach, we counted primary on the collaboration between the android devices, and so utilizing the collaborative computing concepts, and also we used the two architectural design of the network, that is, the client/server and the P2P paradigm, and by exploring the environment to detect nearby devices, we apply some context-aware techniques. The scenario could be summarized with the following figures, that shows how a demander, asks for help, and getting it. The figures illustrate the three main procedures, i.e. fetching for a helper, asking for help, and getting the response.

Figure 4 shows the running server after the two smartphone were registered to it:
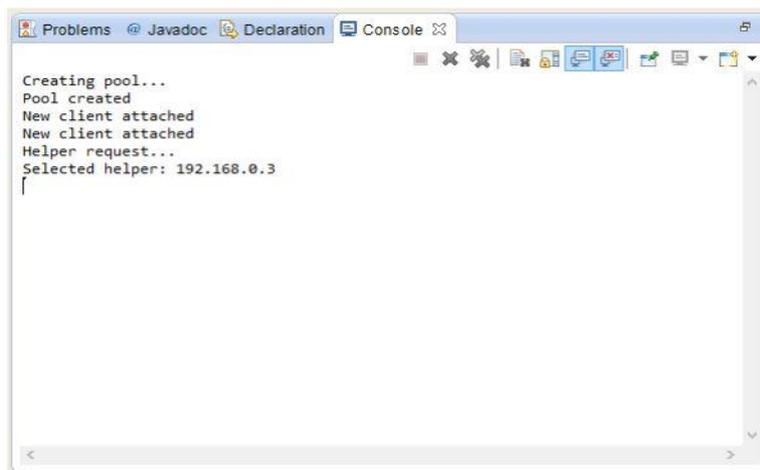


```
Creating pool...
Pool created
New client attached
New client attached
Helper request...
Selected helper: 192.168.0.3
```
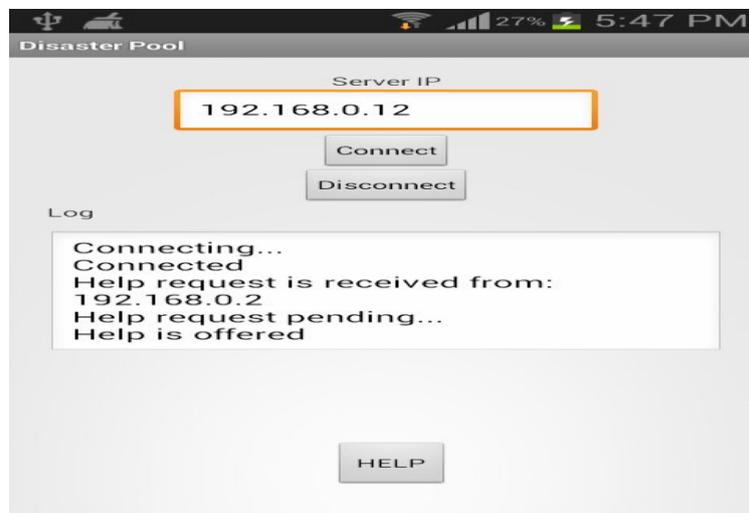
**Fig. 4.** The disaster pool application

Figure 5 shows the demander while asking the server to locate an available helper, then sends the help request to the located smart phone, and finally get back the result.



**Fig. 5.** The demander role

Figure 6 shows the helper after receiving a help request and returning the result.



**Fig. 6.** The helper role

In addition to this work, it's worth to mention that using our approach, which is heavily counting on the hotspot solution, can preserve the life of the battery, and prolong the activeness of the mobile device. This figure, got from a previous study [9], can clear the idea:
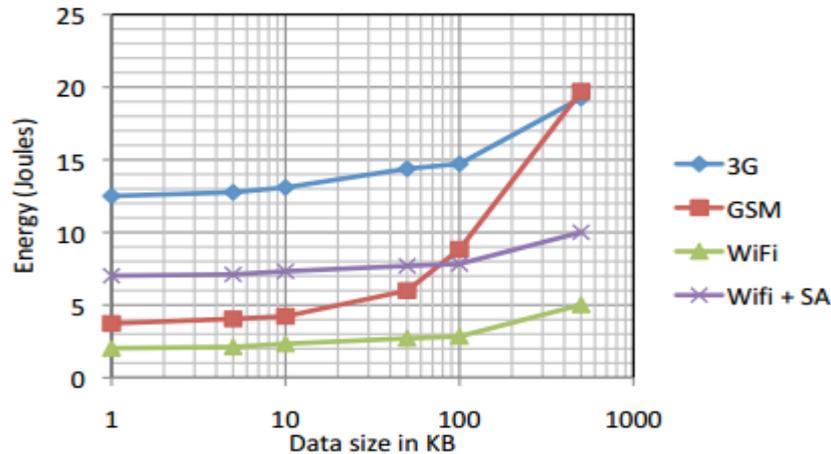


**Fig. 7.** The energy consumption

## 7      Conclusion

In any job in the world, if people collaborate they will reach higher and more significant results. The same idea happens in emergencies. When people are in danger, the only thing matters is how to save these people and how each individual may interact to bypass this situation with minimal losses. These human being characteristics were adopted by collaborative computing, where systems interact together to perform some computations and so on. In this paper we have shown our approach that benefits from collaborative computing characteristics to build a giant human-mobile network that cope to help people in emergency situations. We introduced an implementation of a system that does this task.

Our future work is to add privacy characteristics to this implementation and integrate it with more systems that can push its capabilities to higher limits, to finally create a complete framework that is capable to handle and to process the different types of the aforementioned activities in the proposed approach section.

## Acknowledgement

# References

1. Chi-Sheng Shih, Trang-Khon Trieu. Shadow Phone: Context Aware Device Replication for Disaster Management, RACS-ACM, Montreal, Canada, 2013.
2. P. Brezillon, Context in problem solving: a survey. Knowl. Eng. Rev. 14(1999) 134.
3. V. Stanford, Beam me up, Dr. McCoy, IEEE Pervasive Computing Mag. 2 (3)(2003) 1318.
4. G. Chen, D. Kotz, A Survey of Context-Aware Mobile Computing Research, Dartmouth Computer Science Technical Report TR2000-381, Hanover, 2000.
5. M. Korkea-aho. Context-Aware Applications Survey, 2000, http://www.hut.fi/mkorkeaa/doc/context-aware.html
6. J. Bardram, Applications of context-aware computing in hospital work examples and design principles, in: Proceedings of SAC, Cyprus, March 1417,2004
7. J. Bardram, Hospitals of the future ubiquitous computing support for medical work, in: Hospitals Workshop Ubihealth 2003, 2003
8. M. Munoz, M. Rodriguez, J. Favela, A. Martinez-Garcia, V. Gonzalez, Context-aware mobile communication in hospitals, IEEE Comput. 36 (9)(2003) 3846.
9. N. Balasubramanian, A. Balasubramanian, A. Venkataramani, Energy Consumption in Mobile Phones : A Measurement Study and Implications for Networks Applications, in Proceedings of 9th ACM SIGCOMM conference, New York, USA, 2009
10. A.K. Dey, D. Salber, & G.D. Abowd, (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction Journal 16(2-4), (pp. 97-166).
11. B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, pages 85-90, Santa Cruz, California, December 1994. IEEE Computer Society Press.