Dynamic Identification of Participatory Mobile Health Communities

Isam Mashhour Aljawarneh[1], Paolo Bellavista[1], Carlos Roberto De Rolt[2], and Luca Foschini[1(✉)]

[1] University of Bologna, Bologna, Italy
{isam.aljawarneh3,paolo.bellavista,luca.foschini}@unibo.it
[2] Universidade do Estado de Santa Catarina, Florianópolis, Brazil
rolt@udesc.br

Abstract. Today's spread of chronic diseases and the need to control infectious diseases outbreaks have raised the demand for integrated information systems that can support patients while moving anywhere and anytime. This has been promoted by recent evolution in telecommunication technologies, together with an exponential increase in using sensor-enabled mobile devices on a daily basis. The construction of Mobile Health Communities (MHC) supported by Mobile CrowdSensing (MCS) is essential for mobile healthcare emergency scenarios. In a previous work, we have introduced the COLLEGA middleware, which integrates modules for supporting mobile health scenarios and the formation of MHCs through MCS. In this paper, we extend the COLLEGA middleware to address the need in real time scenarios to handle data arriving continuously in streams from MHC's members. In particular, this paper describes the novel COLLEGA support for managing the real-time formation of MHCs. Experimental results are also provided that show the effectiveness of our identification solution.

Keywords: Mobile healthcare · Middleware · Crowdsourcing · Mobile Health Communities · Community detection

1 Introduction

Today's proliferation of mobile technologies including networks, devices and software systems has caused the widespread creation of mobile healthcare services and systems. Mobile telemedicine, mobile patient's health status monitoring, location-based medical services and emergency response are just few examples, thus constituting an opportunity to provide anytime and anywhere healthcare services in a timely and location-based manner.

In the domain of health care, a challenging problem is to inject novel methods for health care assistance based merely on extemporaneous interactions and collaborations. Due to the unsuitability of traditional virtual health networks in handling today's anytime and anywhere healthcare scenarios, it is essential to activate novel forms of healthcare communities, the so-called Mobile Health Communities (MHC). As an example scenario, MHC can be created by forming a community that consists of passing-by

A. Longo et al. (Eds.): IISSC 2017/CN4IoT 2017, LNICST 189, pp. 208–217, 2018.
https://doi.org/10.1007/978-3-319-67636-4_22

volunteers and professionals, who are willing to provide instantaneous assistance in case of emergency, and also patients who are geographically co-located. Participants may include health staff, friends, neighbors, and passing-by people who can provide instant aid or contribute to patient's rescue. As a simplified scenario, patients with illnesses like cardiopathy and asthma, who have the potential to alert anytime and anywhere, would wear sensor-enabled devices equipped with some kind of network connectivity support (wireless, Bluetooth, LTE, ..) in order to be able to send notifications in case of emergency. On the other side, passing-by volunteers, who are in a relatively nearby distance with that patient, and for whom specific constraints apply, would be notified to provide suitable assistance to that patient. Participants can either be trained enough to give instantaneous medical care, or act as information collectors of relevant data to provide to official medical staff upon arrival.

In the last decade, the use of Mobile CrowdSensing (MCS) has increased vastly, promoting a participatory approach for the management of MHCs. MCS is a community sensing method that collects information using people's sensor-equipped devices [1]. The purpose of those sensing technologies is not only sending emergency alerts, but also feeding specialized database servers with information necessary for disease's diagnosis and management. For instance, MCS's users can feed the system with information related to air pollution's measurements or allergens in some locations of a city, which, in turns, can alert patients with asthma to avoid those areas. Also, MCS support can be useful in proposing sensing activities to users belonging to MHCs based on their location, like taking photos or examining the availability of defibrillators, pharmacies and medical facilities within a city.

However, despite its increased adoption, many technological challenges hinder a proper implementation of participatory MHCs. Wireless-enabled monitoring, control health indicators, location discovery systems for identifying patient's location at the time where help is emergent, and appropriate crowdsensing platforms for supporting participatory mobile healthcare, to name but a few.

Specifically, today it is essential to find solutions that can dynamically form appropriate MHCs nearly on-the-fly, and upon arrival of an emergency event, which will offer a basement for a dynamic interaction among participants, which in turns, facilitates quick decisions suitable for those emergency situations. We define the main requirements that should be met to be able to speed up the decision making process. However, current methods fall short in achieving these requirements and that calls for a novel contribution to accomplish dynamic health's scenario-specific requirements.

Recently, a novel approach for analyzing large streams of data has emerged, the Apache Spark platform, which will be referred to as Spark for short hereafter. Spark has many features, including fast processing of streaming data, and parallel dynamic analysis [2]. These unique features and many others make Spark an excellent candidate for addressing the requirements of our specific mobile healthcare scenarios.

However, our main focus in this paper is not Spark as a core. Alternatively, we are interested in a genuine platform that has been built on the top of Spark, specifically for graph processing, the so-called GraphX. Its introducers claim that it acts faster than traditional programming paradigms in terms of performance and an ability to easily integrate with other platforms, which may need to act on constructed graphs for more

specific analysis [3]. Spark through an implementation for GraphX facilitates a construction of MHC's dynamically. Compared to traditional programming paradigms, it is the best current choice available for forming MHCs for many reasons. First, it enables near real-time formation of MHCs by applying functionalities of its GraphX. Second, it easily enables the system to apply intelligent algorithms on the constructed MHCs, then to find best rescue plan depending on a specific situation of a patient in the presence of emergency. For example, this includes, but not limited to, discovery of a most suitable passing-by participant, discovery of a best hospital that is nearby a location of a patient, and which hosts medical staff and contains equipment appropriate for a specific case of a patient.

As far as we know, currently there are no specific integrated platforms capable of achieving all of the requirements for dynamic mobile healthcare. In this paper, we extend our middleware architecture, called COLLEGA (COLLaborative Emergency Group Assistant) [4] with an innovative MHC formation support. Based on the analysis of requirements for dynamic MHC creation, we have developed and integrated a Spark's-based community detection support for MHCs.

The rest of the paper is divided into the following sections. First, we provide some background of participatory MCS and its application in the construction of MHCs. This is followed by detailed explanations of the COLLEGA middleware and our community detection support based on Spark. In last sections, we present experimental results and conclude the paper reporting future development directions.

2 Background

The high-quality performance of participatory healthcare together with the minimized associated costs have motivated an extended research in the field. Different related aspects have been researched, including wireless area networks, monitoring systems, virtual health community management and construction. In the following subsections, we focus on crowdsensing and community detection in mobile healthcare scenarios.

2.1 Crowdsensing for Mobile Healthcare

The widespread of wireless networks technologies and sensor-enabled devices encourages the emergence of healthcare scenarios, where patients are sending continuous timely and location-based health indicators while moving. As a consequence, novel forms of communities referred to as MHCs should be activated. In addition, the unprecedented smartphone sensing and communication capabilities facilitate healthcare delivery, diagnosis and control of diseases that potentially may disseminate quickly. The emergence of MCS, a promising large-scale data sensing paradigm plays a vital role for the realization of the new participatory healthcare.

In fact, it became obvious that MCS significantly contributes to the construction of MHCs, which is specifically efficient in such scenarios where an urgent first-aid is needed in cases like a sudden asthma attack for a patient equipped with a sensor-enabled mobile device. To put it in another way, MCS is capable of simplifying the formation

of MHCs, which consist of potential passing-by participants suitable for providing instant first-aid. Those users are probably volunteers who use MCS systems and involve voluntarily to MCS participation campaigns.

In addition to provide a base for forming MHCs, MCS exploitation can be extended to attract new volunteers, and strengthen social relationships among volunteers for MHCs that can last. To be more specific, volunteers can actively contribute by feeding the database system with updated data regarding disease's diagnosis and management while moving around in a city, thus enforcing the realization of participatory healthcare. All these features make MCS-based studies a promising and interesting research direction, to the extent that experts are now utilizing it for monitoring infectious disease outbreaks.

Despite the fact that tremendous research efforts have been put in the field of mobile healthcare [5–8], many challenges need to be addressed in a more efficient manner. Many MCS platforms have been proposed in the literature. However, none of them has been encapsulated with a mobile healthcare system. To this end, we introduced COLLEGA, which is basically a middleware that merges MCS with latest community detection paradigms for near real-time and dynamic formation of MHCs, and also continuously injecting updated data suitable for resolving emergencies.

2.2 Community Detection Methods for Mobile Healthcare

Community detection algorithms are prevalent in many real life scenarios. In the last decade or so, they have tremendously been used in various applications like, social network analysis [9], and detecting suspicious events in telecommunication networks [10]. However, there is still a lack of application for relevant algorithms in community detection for forming MHCs.

Detecting communities in mobile healthcare scenarios is non-trivial due to expensive computations required for realizing hidden relationships. This leads us to define a set of requirements under which a system should operate. First, system should be able to construct communities within seconds due to the fact that patients might be in an emergency situation, which in turns means that a quick action should be taken. Second, data arrives continuously in streams and needs instantaneous processing. Third, caching previously generated communities is essential to speedup future processing. Fourth, the form of generated communities should be suitable for an effortless application of data mining algorithms, in order to discover best potential solutions.

Traditional applications of algorithms for community detection have been in action for decades [11, 12]. However, as far as we know, none of those applications have considered collectively the four above mentioned requirements.

Moreover, the running time of those algorithms is typically not linear; therefore, their application in a batch-processing fashion is not suitable for constructing MHCs. Recent trends have tried to reduce running time by implementing those algorithms to run in a multithreaded fashion. However, none of them have considered the scenario where data arrives continuously on the form of streams and needs immediate processing; likewise, as far as we know no speedup strategies like caching have been applied so far in participatory healthcare scenarios.

In order to be able to detect communities, it is a prerequisite to represent actors and relationships of a given scenario as graphs. A graph is a data structure that consists of nodes and edges, where each node models an actor in the scenario, and each edge models a connection between two nodes [13]. However, computations required for detecting communities using the graph paradigm is costly. As a consequence, and because of the need for an enhanced graph computation tools for improving performance, it was essential to introduce new graph-parallel systems, like Pregel [14]. However, these systems did not implicitly consider a streamlined integration with data mining. This, in its turns, led to the introduction of a novel platform called GraphX that is encapsulated with Spark. GraphX simplifies graph computations by expressing graphs using the Spark big data parallel processing framework [3].

Since its introduction, GraphX has been applied to many scenarios including social network analysis [15]. One benefit of GraphX is its adaptability and a precious ability to be executed in a parallel fashion using the core building block of Apache Spark, Resilient Distributed Dataset (RDD), which is a read-only collection of objects distributed on a group of hosts to obtain horizontal scalability and fault tolerance [2]. However, to the best of our knowledge, these methods have never been applied to scenarios specifically related to mobile healthcare that present unique requirements.

Having such information in hand, traditional algorithms for community detection can be implemented with an ease using GraphX. As an example, label propagation algorithm (LPA) is a well-known algorithm in terms of performance [11]. It has been implemented in GraphX using the Pregel framework and can be embedded in any application for community detection. As better explained in the following, for our solution we selected LPA because its running time in a batch-processing scenario have been proved to be near real time, and we wanted to investigate its behavior when implemented using GraphX and in a mobile healthcare scenario.

One more rationale for the selection for GraphX is the ability to apply machine learning algorithms in a straightforward fashion. In fact, on top of the Spark core it is available the MLib that encapsulated many machine learning algorithms. That allows to easily apply machine learning algorithms on a graph constructed using GraphX.

Finally, let us remark that in our scenarios mobile health data will arrive as continuous streams, which means that the discovery of communities and potential participants must be done in a near real time fashion. The ability to extend the static implementation of a community detection algorithm using Spark's Streaming is a major benefit and is expected to perform with a high performance.

3 The COLLEGA Middleware

COLLEGA constitutes a middleware which provides participatory emergency support for moving patients with chronic diseases. It constitutes a comprehensive lifecycle for an emergency scenario, from danger alarm to the construction of MHCs, and to the formation and execution of emergency action plans. Indeed, in COLLEGA mobile emergency scenarios are composed of mobile monitored patients and co-located passing-by participants. Simply put, COLLEGA pre-assumes that mobile patients either

have wireless sensors attached to their clothes or implanted under skin for observing vital body health indicators and constructing a Wireless Body Area Network (WBAN). To this end, Patient's smart devices send alarms in case of emergency while moving. On the other side, COLLEGA enables potential passing-by participants to receive those alarms, with information including the type of emergency and helpful information to provide first-aid.

COLLEGA exploits our previous experience in the ParticipAct middleware [16] and the Proteus access control model described in [17]. In addition, new mobile healthcare specific functions are added here as better detailed in the following.

3.1 COLLEGA Architecture

Our COLLEGA architecture constitutes its core modules and their associated workflow as shown in Fig. 1. Each module is devoted to realize a specific function of the mobile healthcare emergency management lifecycle.

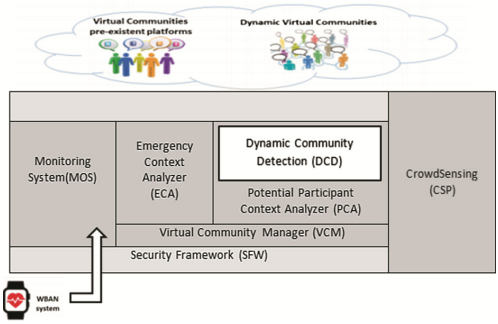

Fig. 1. COLLEGA middleware modules.

The Monitoring System (MOS) is a module that is interfaced with a patient's WBAN for collecting and analyzing sensor's data, thereby sending emergency alerts and coordinating communications with the patient. In addition, MOS is also responsible for suggesting first-aids to patient in relative to off-the-shelf control plans.

The Emergency Context Analyzer (ECA) is responsible for combining data received from MOS module with patient's medical history, probably received from a remote database, then compares them with similar cases stored in a distributed knowledge base, to analyze and classify the severity degree of the emergency situation. Thereafter, ECA selects the most suitable control plan, which contains tolerance limit's values for each event's severity degree. It also chooses the most appropriate actions to be performed for a specific level of severity, the equipment required to accomplish every action and a set of skills that a potential participant should have.

The CrowdSensing module (CSP) encapsulates a variety of mobile crowdsensing-related functionalities. To be more specific, CSP is responsible for discovering user's status, such as user walking and running. Also, it detects user's location by geolocalization functions. Furthermore, it uses geofences to detect user proximity in relative to

a specific geographical point. Furthermore, it assigns tasks to users while entering given geofences.

The Participant Context Analyzer (PCA) dynamically selects potential participants from an established MHC. PCA is also responsible for dynamically detecting communities through the Dynamic Community Detection (DCD) sub-module. DCD receives appropriate data from CSP, joins them with corresponding data in the knowledge base, and then applies a community detection algorithm to construct MHCs. To this end, PCA is also responsible to choose the most appropriate participant depending on several factors, including the control plan to be performed.

The Virtual Community Manager (VCM) distributes tasks to the identified participants, gets their acceptance, provides a set of instructions, and asks security support permission to access patient's medical data. Also, VCM gathers data throughout the whole cycle of an emergency situation in order to update the user's medical history and knowledge base.

The core task of the Security Framework (SFW) is to allow the patient to set access control policies, and thereby enforce mechanisms for accessing patient's personal data. Also, SWF assures user's health data confidentiality during the transmission to participant's devices.

In order to realize the modules of the COLLEGA middleware, it is essential to implement and deploy related software packages at client and server sides. In this paper, we focus on the aspect of MHC construction and detection using Spark GraphX. According to our main requirements detailed in Sect. 2.1, we have designed the DCD sub-module and integrate it with the PCA module, aiming to meet those requirements as detailed in the following sub-section.

3.2 COLLEGA Community Detection

We have identified four extreme requirements in MHCs for which any detection algorithm should obey in order to be considered efficient. We believe that those requirements best fit the context of our MHC's formation scenarios. We here list them and rationale their selection.

First, MHCs should be constructed in a near real-time fashion. This means it should only take few seconds to complete the whole process from sending the alert by patient's device, to forming communities, to finding the potential participant who can provide an appropriate first-aid. Second, the algorithm should be able to receive data in streams. The purpose of this constraint is that in our health care scenarios data continuously arrives in streams. This, in turn, means that while participants and patients are moving, they keep sending information and signals to the server which hosts the processing algorithm. Third, the algorithm should be extensible in a way that makes it adaptively update the network community structures based on its cached structures instead of repeating the whole process from scratch. The last requirement is that communities detected using the algorithm should be available in a form that streamline the application of data mining and machine learning algorithms on the constructed graphs [3]. This is because, upon completion of MHCs construction, it is essential to identify the most potential assistance plan, including the identification of the most appropriate participant,

finding the nearest hospital that contains medical equipment's relevant to assist that patient, and probably finding the best way to move the patient from current location to the identified hospital. This requires the application of data mining and machine learning techniques and obtaining results within minimal timeframes.

The responsibility of the DCD module is to discover potential nearby participants to provide first-aid. The module comprises three sub-modules deployed on a distributed system. The first sub-module, deployed on patient device, is responsible to generate an alert signal in case that an emergency was detected by the MOS module. The alert is implemented in a formatted packet such that it encapsulates information like an identifier of the health problem type and GPS coordinates of current patient location. The second sub-module is deployed on a remote-server running Spark and implements a community detection algorithm (e.g. LPA). The server-side module will receive the alert, analyze its components, form MHCs and apply data mining and machine learning algorithms on them to discover the best rescue plan. The third sub-module is deployed on the participant's device, and is always listening to alerts. In fact, listening mode is adaptable in accordance with participant's preferences. For example, participant may prefer to receive alerts through messaging services, in order to avoid exhausting the battery and mobile device's resources. After completion of the analysis, the server sends to the potential participant necessary information to commence with the first-aid.

4 Experimental Results

In this section, we present results obtained from applying a community detection algorithm using Spark's GraphX. Static LPA has been tested using two implementations. The first one is a conventional (vanilla) implementation of LPA that uses traditional programming modules that have been implemented using Java 1.8 version libraries. The second one, instead, is purely based on Spark's GraphX. We have generated synthetic data that model the relationships between participants and patients. IN particular, generated data consisted of tables, each table contains two columns, and each column contains a set of nodes. This means that each row in the table models the relationship on a patient-participant or a participant-participant base.

Our testing environment consists of one machine hosting Linux Ubuntu 64-bit operating system, with four Intel Core 2.20 GHz processors and 4 GB RAM. In order for the test to be fair with respect to the vanilla LPA concentrated implementation, we have decided to test on a single machine, taking into consideration that the conventional libraries cannot run in a parallel and distributed platform. The goal of our tests is to evaluate the scalability of the two LPA implementations, namely, the evolution of the running time as we increase the network size.

As depicted in Fig. 2, GraphX-based LPA implementation outperforms vanilla LPA implementation in terms of running time, even though both implementations act in a near linear fashion.

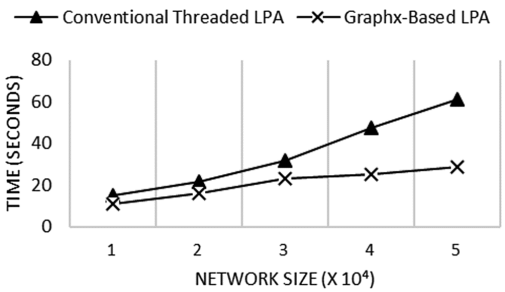

Fig. 2. Comparison between conventional and Spark Graphx implementations of LPA.

We also believe that applying cache mechanisms to the GraphX-based LPA will definitely reduce the running time, which means that formation of MHCs can depends effectively on caching for improving the overall performance and enforcing the near-real time objective. The fact that any application that is constructed using Spark libraries can be executed in a parallel fashion encouraged us to adopt GraphX. The running time shown in the figure will definitely decrease dramatically when executing the algorithm in a cluster or a cloud.

This simplified scenario exhibits the efficiency of adopting Spark and GraphX, for community detection in mobile healthcare scenarios. Add to this the fact that it is a streamlined process to apply a machine learning algorithm on the obtained result in order to identify the most potential participant in case of emergency. For instance, algorithms can be applied to discover the shortest path to the most suitable hospital.

5 Conclusions and Future Work

In this paper, we have introduced the solid COLLEGA middleware and presented the novel function to dynamically discovery potential participants for providing first-aid to co-located patients. Further, we have incorporated a recent technology for the construction of MHCs with COLLEGA. To validate this incorporation, we have tested the middleware with a synthetic data and compared the performance of MHCs construction with a conventional paradigm.

Boosted by obtained results, we are currently exploring other ongoing work directions. First, despite the fact that the incorporated method for MHCs outperforms its predecessors, it is worth noticing that only synthetic testing scenarios have been considered; hence, we plan to test with benchmark data in order to strengthen the theory. In addition, data mining and machine learning algorithms will be applied on the constructed MHCs.

Acknowledgments. This research was supported by the program CAPES- Pesquisador Visitante Especial - 3º Cronograma - Chamadas de Projetos nº 09/2014 and by the Sacher project (no. J32I16000120009) funded by the POR-FESR 2014-20 through CIRI.

References

1. Ganti, R.K., Ye, F., Lei, H.: Mobile crowdsensing: current state and future challenges. IEEE Commun. Magaz. **49**, 32–39 (2011)
2. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: Presented at the Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, Boston, MA (2010)
3. Xin, R.S., Gonzalez, J.E., Franklin, M.J., Stoica, I.: GraphX: a resilient distributed graph system on Spark. In: Presented at the First International Workshop on Graph Data Management Experiences and Systems, New York (2013)
4. Rolt, C.R.D., Montanari, R., Brocardo, M.L., Foschini, L., Dias, J.D.S.: COLLEGA middleware for the management of participatory mobile health communities. In: 2016 IEEE Symposium on Computers and Communication (ISCC), pp. 999–1005 (2016)
5. Alali, H., Salim, J.: Virtual communities of practice success model to support knowledge sharing behaviour in healthcare sector. Procedia Technol. **11**, 176–183 (2013)
6. Christo El, M.: Mobile virtual communities in healthcare the chronic disease management case. In: Sabah, M., Jinan, F. (eds.) Ubiquitous Health and Medical Informatics: The Ubiquity 2.0 Trend and Beyond, pp. 258–274. IGI Global, Hershey (2010)
7. Chorbev, I., Sotirovska, M., Mihajlov, D.: Virtual communities for diabetes chronic disease healthcare. Int. J. Telemed. Appl. **2011**, 11 (2011)
8. Morr, C.E.: Mobile virtual communities in healthcare: self-managed care on the move. In: Presented at the Third IASTED International Conference on Telehealth, Montreal, Quebec, Canada (2007)
9. Zhao, Z., Feng, S., Wang, Q., Huang, J.Z., Williams, G.J., Fan, J.: Topic oriented community detection through social objects and link analysis in social networks. Knowl. Based Syst. **26**(164–173), 2 (2012)
10. Akoglu, L., Tong, H., Koutra, D.: Graph based anomaly detection and description: a survey. Data Mining Knowl. Disc. **29**, 626–688 (2015)
11. Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. Phys. Rev. E **76**(3 Pt 2), 036106 (2007)
12. Yang, J., McAuley, J., Leskovec, J.: Community detection in networks with node attributes. In: 2013 IEEE 13th International Conference on Data Mining, pp. 1151–1156 (2013)
13. Lei, T., Huan, L.: Community Detection and Mining in Social Media. Morgan & Claypool, San Rafael (2010)
14. Malewicz, G., Austern, M.H., Bik, A.J.C., Dehnert, J.C., Horn, I., Leiser, N., et al.: Pregel: a system for large-scale graph processing. In: Presented at the Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, Indianapolis, Indiana, USA (2010)
15. Lan, S., He, G., Yu, D.: Relationship analysis of network virtual identity based on spark. In: 2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), pp. 64–68 (2016)
16. Cardone, G., Cirri, A., Corradi, A., Foschini, L.: The participact mobile crowd sensing living lab: the testbed for smart cities. IEEE Commun. Magaz. **52**, 78–85 (2014)
17. Toninelli, A., Montanari, R., Kagal, L., Lassila, O.: Proteus: a semantic context-aware adaptive policy model. In: Eighth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2007), pp. 129–140 (2007)