Graph Search Centered Solution for Contact Information Retrieval

A value-added alternative for built-in People application of Android devices

Arumapperuma A. A. G. C. K.
Department of Computer Science and Engineering
University of Moratuwa
Sri Lanka
charith.11@cse.mrt.ac.lk

***Abstract*—Mobile Computing industry is a rapidly growing software engineering field where mobile application market accounts for a huge share. There are numerous software solutions introduced to mobile devices over the past decade. But, industry has not been able to improve one of the fundamental features of mobile phones, the "Address Book", to provide complete user satisfaction. This paper introduces a graph based contact information storage and searching solution as a value-added contact information management system. Core concept of this solution is modelling mobile phone user contact details as a graph and traversing the graph to find people. The solution represents people in graph vertices while storing "is a contact of" relationships in directed edges of the graph. The searching algorithm is developed by implementing a depth limited breadth first search while ensuring privacy of contact information. The outcome of this solution is a cloud based hybrid mobile application that can be used to replace the default "People" application of Android devices while ensuring the portability of the system to other mobile, desktop and web platforms.**

Keywords—mobile computing; graph search; social graph; Android; mobile application; address book

I. INTRODUCTION

Mobile Computing is one of the most dynamic industries of the world. It has grown in an exponential rate over the past decade. Statistics show that mobile phone subscribers by the end of year 2014 was 6.95 billion which is more than double the value in 2007. It is 119.9% of population in developed countries and 91.1% in developing countries. The mobile broadband subscription statistics are showing more exponential growth over the past decade [1]. These statistics undoubtedly indicate the global shift towards mobile technologies; mobile devices such as mobile phones and tablets are getting closer and closer to being a human need.

Growth in mobile device usage has introduced a new means for business and personal interactions. It in return introduced many opportunities for software industry to build mobile solutions. But, in general, mobile devices contain low processing power, low memory and low battery power which causes mobile applications to be limited in their functionality. In a computer science perspective, these are a few problems that were addressed very lightly due to the enormous performance that traditional computer systems provide. Many computer science solutions were introduced over the past years for these problems. And today, mobile devices are almost identically powerful as traditional computers while mobile applications are powered by various computer science solutions that perform better even on low performing hardware.

Even though today's mobile devices are rich in features, where you can browse internet, shop, handle money, etc., initially their primary task was personal long distance communication. Today, "call" has become yet another feature of phones. Similar to "call" feature, there are other means of communication such as "SMS", "MMS", etc. The common feature for all these means of communication is the "Contacts List" or "Address Book".

The mobile phone has evolved rapidly over time. Means of communication are also evolving in a similar rate. Yet, the common element of all these means, "Address Book" has not been improved in a similar manner. We can see almost identical functionalities in "Address Book" from the latest flagship smartphone as well as from a mobile phone from year 2000.

The lack of functionality in "Address Book" can be identified in various real life situations. One of the most encountered problems is the unavailability to perform contact information search beyond locally stored data. For instance, consider when there is a need to communicate with a person via call or SMS. If there is no contact information present in the phone's local contacts storage, it is impossible to continue communication without manually collecting necessary contact information from a 3rd party. Also, recognizing an unknown caller is yet another common issue in today's mobile phones.

The solution presented in this paper addresses the first problem described above through an Android application. The system can easily address the second problem described above even though it is not implemented in the system. The solution is developed in a way that it can be easily implemented into other mobile phone platforms, web and desktop applications.

II. RELATED WORK AND TECHNOLOGIES

The solution for contact information retrieval is build using crowdsourcing, cloud computing technologies, mobile

computing technologies and graph based data representation and searching. These research areas are extremely popular and active nowadays due to the importance of those to the vast mobile computing industry.

A. Crowdsourcing

In software development, crowdsourcing can be considered as the most efficient method for collecting large amount of data. Wikipedia (https://en.wikipedia.org/), a free encyclopedia written collaboratively by the people who use it, is one of the most popular systems that uses crowdsourcing for data gathering. But, crowdsourced data cannot be used directly since it lacks validity of data. Thus, various data mining techniques are used to yield useful information from crowdsourced data [2].

B. Cloud Computing

Cloud computing is yet another rapidly growing computer science study which has proven excellent outcomes in developed products and its user acceptance. Similar to mobile applications, cloud applications are becoming more popular among community since they provide on demand services. Today, almost all the desktop applications (programs) are available as cloud applications or similar cloud applications are developed to accomplish the same task. For example, Microsoft Corporation provides Microsoft Word, a graphical word processing program as a desktop application as well as a cloud application (https://office.live.com/start/Word.aspx), and Google has developed a similar cloud application, Google Docs (https://docs.google.com), to accomplish the same task.

Cloud Computing can be categorized into three fundamental models [3],

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Infrastructure as a Service (IaaS)

Considering the nature of this solution, PaaS model, which provides fundamental building blocks to develop cloud applications and systems can be used to develop the solution due to the features and flexibility that PaaS systems provide.

There are various PaaS service providers available today including Google App Engine, Microsoft Azure and Amazon Web Services (AWS) [4]. The services they provide vary in features and pricing. For instance, Google App Engine provides the most essential features free of charge while providing advanced features as add-ons with pay per usage plans [5].

C. Mobile Computing

Mobile Computing has become one of the largest industries with the invention of smartphone. Today, Apple Inc., Samsung Electronics, Microsoft Corporation are holding the larger market share of mobile devices with their powerful hardware and software. Android, iOS and Windows Mobile operating systems are extremely competitive with their software to provide best user experience. The Android market is the largest of all due to its flexibility [6]. Android application development is an extremely vigorous topic in mobile computing.

There are three types of mobile applications [7],

- Native Apps
- Web Apps
- Hybrid Apps

Native apps are completely run on the device as a standalone program. Web apps are truly websites which users can access by using device web browser. Hybrid apps are a combination of above types, where it provides services through a web based system to be accessed by the mobile application.

The solution described here is a hybrid application, where system logic is distributed on both web server and mobile device. Necessary computational power, memory and communication bandwidth are considered when distributing the logic, so that the user experiences a flawless system with minimum data connectivity usage.

D. Graph Based Data Representaion and Searching

Graph study is a vast field in computer science. Graphs play an important role in social network analysis, data mining, image segmentation, clustering, image capturing, networking, etc. Graph based data representation is massively used to manage large structured data repositories [8]. As an example, Facebook (https://www.facebook.com/), an online social networking service, uses graph based data structure to store all the data available in the system like people, posts, pages, groups, etc. [9].

Graphs consist of two basic elements, vertices and edges. Various combinations of these elements can generate different types of graphs like trees, directed graphs, complete graphs, bipartite graphs, etc. Social Graphs are undirected graphs at its very basic level, where people are represented by vertices, and connections between people are represented by edges [10]. Such graphs can be further improved to represent more meaningful information by adding various properties to edges like adding a weight to represent connection strengths between people, or adding directional property to edge to represent connection type (is friend of, is parent of, etc.).

Even though graphs are logically viewed easily with connected set of points, its implementation is considerably complex. There are two graph representations are used in general. [11]

- Adjacency Matrix
- Adjacency List

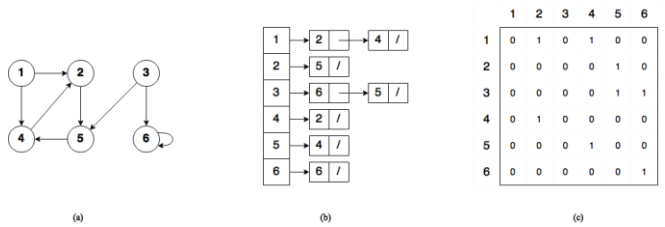

Fig. 1. Representation of graphs. (a) Directed graph. (b) Adjacency List representation. (c) Adjacency Matrix representation.

Adjacency Matrix uses a $n \times n$ matrix, where n = |V| of graph G = (V, E). Adjacency matrix representation is preferred when the graph is dense, i.e. the number of edges |E| is close to the number of vertices squared, $|V|^2$, or if one must be able to quickly look up if there is an edge connecting two vertices.

Adjacency List uses only the amount of storage that is necessary to store edges. Thus, adjacency lists are generally preferred to represent sparse graphs which contains relatively few edges.

Graph searching is also a huge computer science problem where many approaches have been taken to find a fast and memory efficient algorithm. The most basic algorithms available are Depth First Search (DFS) and Breadth First Search (BFS). More complex graphs can be traversed using advanced algorithms like A*, Hill climbing, etc.

III. DESIGN AND IMPLEMENTATION

A. System Design

The developed contacts manager application (Address Book) is primarily a hybrid mobile application. Its main functionality is divided into two sections as client logic and backend logic. Client logic contains functionalities to collect and display data, and handle user interactions. Backend logic contains functionalities to handle user request such as add, delete, change privacy (update) and search. Each section is designed using layered architecture. The system uses cloud endpoints to communicate through a REST layer.

The main information stored in the system are contact details which are represented by a directed graph. The contacts

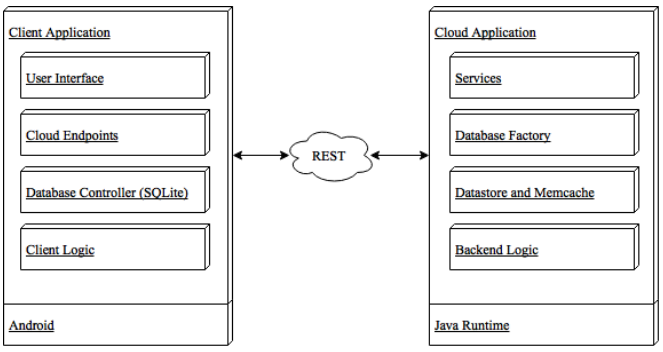

Fig. 2. System Architecture. System is divided into two sections as client application (client logic) and cloud application (backend logic). Each section is further layered into different layers to improve maintainability.

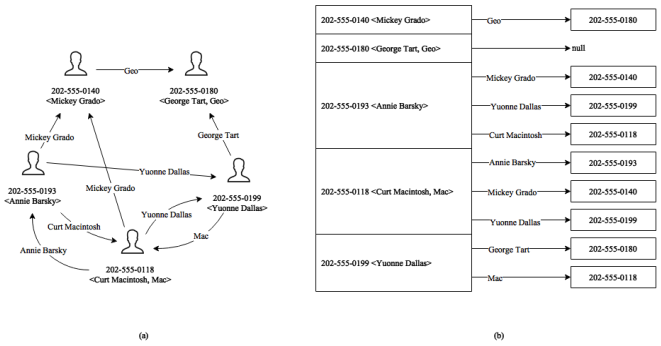

Fig. 3. Graph model of the system. (a) Contacts Graph; each vertex represents a person and each edge represents availability of a connection between two people. (b) Representation of contacts graph as a "Adjacency List" in the system. The system uses a different approach than the regular "Adjacency List" representation where, rather than using a list of "Linked Lists", the system uses a list of "Maps" for efficient indexing.

graph contains people as vertices, and connection between two people as edges. Vertices are flexible entities that provides developer to store any information about the person represented by the vertex. Edges are restricted to hold two properties. Edge direction represents "is a contact of" relationship between two vertices and "connection privacy" edge property holds information about how the connection between two people should be visible to others in the system.

B. Graph Representation

The graph is represented as an adjacency list. The information stored in the system do not pose a complete graph. It is possible to see complete sub-graphs in the system when it represents a small group of known people (e.g. immediate family members), but the overall graph contains a large amount of pairs of vertices that are not connected, i.e. the people represented by such vertices do not know each other or has no records of contact information on their mobile device. In this scenario where the graph can be identified as a sparse graph, adjacency lists are preferred for representing the graph in the system due to the use of least amount of memory.

The adjacency list representation of this system is different to the general representation. General adjacency list uses a list of linked lists as shown in Fig. 1 (b). But, this system uses a list of maps instead of linked list. The map is better suited in this scenario since it provides faster indexing of the graph. The key of each map represents the name in which the contact holder refers to the second party. I.e. if person A has person B stored in his phone contact list as "b" regardless of what B's real name is, the edge connecting A and B will be directed towards B and the edge is labeled as "b". This method provides the system the ability to efficiently find a name out of all the immediate relative nodes of a node.

C. Graph Search

The searching algorithm used in this system is based on breadth first search (BFS). The search is limited to two levels in depth. The searching algorithm will first search all the neighbor vertices of the user who is performing the search. If

there are no vertices found, the search will continue to the next level, where all neighbors of immediate neighbors of the user is searched. Searching algorithm is stopped at two levels due to the complexity, low performance and privacy.

Consider the example in Fig. 3 (a) and the scenario where Annie Barsky wants to search George Tart. Annie has three people in her contact list, Mickey, Younne and Curt. Each of these first level contacts has different people in their contact lists. Annie has two paths to find George, through Mickey and through Younne. But, Mickey has stored Goerge as Goe in his contact list. So Annie cannot see Goerge through Mickey if she search for "George". But, she can see Goerge through Younne. In case Annie searched for Goerge by "Goe" she will see him though Mickey but not through Younne. Finally, she will receive all the details about Goerge Tart since there is at least one path available for Annie to find Goerge.

D. Privacy

Contact details are extremely sensitive information. Meanwhile this application provides a computer science solution for mobile computing industry, it poses the threat to expose sensitive contact details to undesired parties.

The system is designed to overcome this issue. The solution is implemented by including an extra property to graph edges, privacy, where it act as a global override for all search queries. When an edge is marked as private, graph search cannot go through the edge. This property will be set by the user for his contact list when he synchronize it with the system. So, the system provides complete control to the user to ensure privacy of his friends, family, etc. Also system has included with a functionality for user to set his privacy level.

In the example given in the previous section where Annie Barsky wants to search George Tart, consider Younne has marked Goerge as private in his contact list. In case when Annie searches by "Goerge", the path through Mickey is hidden since the search query is different and the path though Younne is hidden since it has been blocked by Younne.

But, with this extra privacy setting, a new issue arises since the system cannot deliver requested results even when the details are available. To overcome this and to give an immediate communication path to users to contact required people, a notification system is introduced to the system. A user (P_1) can search for a person (P_2), if P_2 is available but hidden to P_1, P_1 will not be presented with contact information of P_2 but will get the opportunity to request contact details through the system. The system will notify P_2 about the request through a system notification and an SMS. The notification will include information about P_1, so that P_2 can contact P_1. This allows users to use the system without any privacy concerns since it provides them to control their own privacy with maximum flexibility.

The system also incorporates a complete transaction log, which records all the actions carried out by users, including, adding, deleting and updating contact information of their contacts. Also, the system log records all the search queries to track any illegitimate or immoral activities, to improve system privacy and support legal concerns.

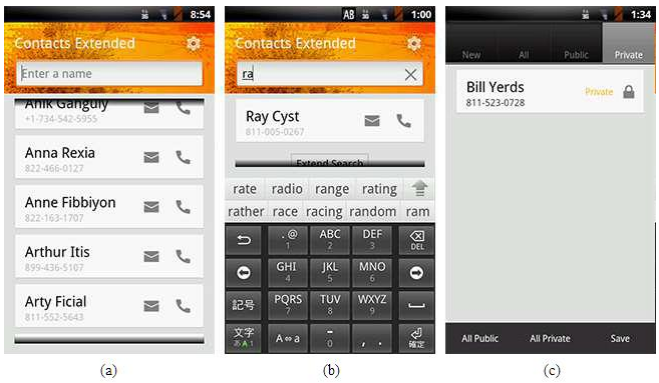

Fig. 4. Mobile application user interface. (a) Application home. Displays contacts stored in mobile phone's local storage. (b) Dynamically filterred contacts list. (c) Privacy management screen.

E. SMS Notification System

The SMS based notification system facilitates users to reach others with minimum delay. There can be situations where a system user (P_1) cannot connect to the system due to unavailability of an internet connection or a device issue. In such situations, and when another user (P_2) requests contact details of P_1, the SMS based notification system guarantees on time delivery of the search request to P_1. P_1 has the flexibility to contact P_2 manually using the contact details of P_2 which is sent through the SMS notification.

Also, there can be situations where the contact details available in the system belongs to people who do not use the system. Such situations can occur when a system user uploads contact details of the people available in his address book, and those people are not using the mobile application (due to the unawareness of the system, unavailability of a smartphone, etc.). Such people must be aware of the information gathered by the system about them. This feature is discussed under future work.

F. Mobile Application

The mobile application of the system is an Android application that uses cloud application's web services to handle user actions and queries. Mobile application contains functionalities such as, read phone's contact list, send synchronize requests, read user inputs, send search requests, and manage local database to store application configuration information.

The mobile application is built using up-to-date Android standards and features to provide best user experience. The primary user interface is inspired by latest Google cards layout [12]. Dynamic lists, tabular pages and toggle buttons are heavily used in the application while minimizing menus and textual inputs to allow fast user interaction.

An SQLite database is used to store application's internal data, including synchronized users, new users, privacy settings and user account information and settings.

G. Cloud Application

The cloud application of the system is developed using Google App Engine, a Platform as a Service (PaaS) [13]. Google App Engine provides various features to build cloud mobile applications. Portability to various mobile platforms and programming languages are some of the major features of Google App Engine (GAE).

The system uses GAE Datastore to store graph data and Memcache service to store graph as an adjacency list to improve performance. Data are stored as Java Database Objects (JDO) to ensure that the system can be maintained and extended easily [14].

IV. Results and Future Work

Developed Android application successfully executes its intended task accurately. But, the application has shown some performance issues that can be fixed with minor implementation changes.

This solution is a gateway for contact information management applications to identify user requirements and adapt. Developing a social graph using contact information present in all mobile phones will make every person in the world available as a digital entity. The uses of such a data repository is enormous.

The system has shown the potential of graph data structures. It can be adapted to represent any relationship among infinitely large amount of entities.

The mobile application has shown various future improvements that can be applied to improve usage and performance of the system. The main improvement that needs to be done is implementing the application into other mobile platforms. Since users act as both information providers and consumers, it is essential to reach a large user base to ensure validity and availability of information. Developing the system to other mobile platforms will be helpful to reach this goal.

Another mandatory improvement is handling new contact information. An SMS verification system could help in this scenario by notifying the people who are not using the mobile application when a new entry for his phone number is added to the system. Until such people allow (by following the given instructions in the SMS) their information to be visible in the system, the information will be set as private. Search queries will not return contact details of them directly which will in return result sending an SMS notification to that person about the search query as mentioned previously.

Even without the above mentioned functionality the current system holds basic yet functional privacy handling mechanism. But, in order to release the mobile application as a marketable product, a better and highly secure privacy mechanism must be implemented.

The authentication mechanism used in the current system is developed only for testing purposes. A secure authentication system must be implemented prior to the release of the product. Google App Engine provides supports to OAuth 2.0 and it provides support to authenticate with Google id [15]. Such features can be used to implement the authentication system.

As an additional feature, system can be easily updated to identify unknown callers. Even though there are some applications available to accomplish this task, a complete "Address Book" application has not been introduced yet.

The final outcome of the product would be a value-added contact management application for mobile, desktop and web users that allows them to easily interact with people they know in physical world. The final graph of this system will be a phone number based global social graph where every member represents one or more clusters of the whole system, where each cluster represent a real life social group.

References

[1] International Telecommunication Union. "Key ICT indicators for developed and developing countries and the world (totals and penetration rates)." Available: http://www.itu.int/en/ITU-D/Statistics/Documents/statistics/2015/ITU_Key_2005-2015_ICT_data.xls

[2] G. Barbier, R. Zafarani, H. Gao, G. Fung, and H. Liu, "Maximizing benefits from crowdsourced data," *Computational and Mathematical Organization Theory*, vol. 18, pp. 257-279, 2012.

[3] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," *Special Publication 800-145*, National Institute of Standards and Technology, Gaithersburg, Maryland, 2011.

[4] C. Burns, "10 most powerful PaaS companies," *Network World*, 04-Jun-2014. [Online]. Available: http://www.networkworld.com/article/2288002/cloud-computing/10-most-powerful-paas-companies.html. [Accessed: 25-Jul-2015].

[5] "App Engine Pricing," *Google Developers*. [Online]. Available: https://cloud.google.com/appengine/pricing. [Accessed: 25-Jul-2015].

[6] "IDC: Smartphone OS Market Share," *www.idc.com*. [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp. [Accessed: 25-Jul-2015].

[7] S. N. Ali, "Types of Apps - Develop Application for Buisiness," *www.socialhunt.net*. [Online]. Available: http://www.socialhunt.net/blog/types-of-mobile-app/. [Accessed: 25-Jul-2015].

[8] J. A. Bondy and U. Murthy, *Graph Theory with Applications*, New York: Elsevier, 1976.

[9] "Under the Hood: The natural language interface of Graph Search," *Facebook Engineering*. [Online]. Available: https://www.facebook.com/notes/facebook-engineering/under-the-hood-the-natural-language-interface-of-graph-search/10151432733048920. [Accessed: 25-Jul-2015].

[10] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The Anatomy of the Facebook Social Graph," *arXiv:1111.4503 [physics]*, Nov. 2011.

[11] S. G. Shirinivas, S. Vetrivel, and N. M. Elango, "Applications of Graph Theory in Computer Science – An Overview," *International Journal of Engineering Science and Technology*, vol. 2(9), pp. 4610-4621, 2010.

[12] "Cards - Components," *Google design guidelines*. [Online]. Available: https://www.google.com/design/spec/components/cards.html#. [Accessed: 10-Sep-2015].

[13] "App Engine - Run your applications on a fully-managed Platform-as-a-Service (PaaS) using built-in services," *Google Cloud Platform*. [Online]. Available: https://cloud.google.com/appengine/. [Accessed: 10-Sep-2015].

[14] "Java Data Objects (JDO)," *Oracle.com*. [Online]. Available: http://www.oracle.com/technetwork/java/index-jsp-135919.html. [Accessed: 10- Sep- 2015].

[15] "Using OAuth 2.0 to Access Google APIs," *Google Developers*. [Online]. Available: https://developers.google.com/identity/protocols/OAuth2. [Accessed: 10-Sep-2015