AppaaS: Provisioning of Context-aware Mobile Applications as a Service

Khalid Elgazzar, Ali Ejaz, Hossam S. Hassanein
School of Computing
Queen's University, Canada
{elgazzar, aliejaz, hossam}@cs.queensu.ca

***Abstract*—The global mobile application market is booming and business giants, viz. Google and Apple, have acknowledged the huge expansion in their respective application market. There is a demand though for a system that can elevate the momentum of context-aware mobile applications, where application behavior can be customized according to various context information. This paper proposes *AppaaS*, a context-aware system that provides mobile applications *as a service*. AppaaS handles various context information to provision the best relevant mobile application to such a context. AppaaS also supports state preservation, where application-specific data is stored for future references. Our prototype demonstrates an agile system performance with respect to finding relevant applications to a specific context and controlling the applications functions according the users requirements and access privileges.**

I. Introduction

Recent years have witnessed a lot of momentum for mobile applications that can adapt their behavior according to context changes. Adaptation to various context changes reshapes the application behavior to support personalized services, which then become more appealing to mobile users. Such context includes location information, user profile, user ratings, device profile, and time.

Currently, there are more than 500,000 applications available on *Google Play* [1] for Android-based devices and 585,000 applications available on *Apple Store* [2] for Apple devices. Many of these applications are available from various businesses to benefit their customers and offer a differential experience. There are a number of open questions related to context awareness. From a user perspective, how would a user know that there is an application relevant to his current location, and how can a user decide on which application to use from the many available choices. From a business perspective, how can businesses control access to their applications while maintaining a high level of mobility flexibility to their employees. Our proposed system targets these questions.

We propose a context-aware system for mobile application dissemination and control, which provisions mobile applications as a service (AppaaS) on demand with the appropriate access constraints. AppaaS system enables business entities to have full control over their mobile applications in terms of distribution, access rights, and management.

II. Motivating scenarios

To better understand the functionalities and the need for such a system to exist, consider the following scenarios.

- *Finding the Appropriate Application*: Suppose that Adam walks into a store with his smartphone and is looking for an application by which he can browse through different offers, product catalogues, or find information relative to a specific product of interest. AppaaS makes this scenario possible. As soon as Adam enters the store's physical space, the store-specific application gets downloaded onto Adam's smartphone, with his consent. As soon as Adam leaves the space, the application preserves its current state (browsed items, shopping cart, etc.) and gets uninstalled to free up the mobile resources it is holding. If Adam comes back to the same store (or another related store that shares the same application), the application launches back with the last preserved state that is relevant to Adam.
- *Controlling Application Behavior*: Now Adam is at his workplace. Adam holds a position that gives him access to confidential data. As soon as Adam enters the workplace premises, AppaaS installs or activates the functionality of the enterprise application on his mobile device that allows him access to such data. Towards this end, it is possible that an enterprise restricts access to such critical functions at coffee rooms or break lounges. Furthermore, enterprises might restrict their business-related mobile applications (or certain functionality) to specific physical places (such as enterprise premises, bidder's conference) with variant access privileges according to the user/employee position or role at the enterprise.

III. Background & Related Work

Mobile services that are capable of changing their behavior according to context changes are of particular interest to the provisioning of personalized behaviors [3]. To achieve the desired functionality, such context-aware services typically exploit a variant combination of several context information, including location information, user profile, user satisfaction indicators, device features and capabilities, and time [4].

Location-based services in ubiquitous computing environments is a rich domain of research [5]–[8], in which services make use of location information in particular to better provide relevant ubiquitous services. The application of location-based services is popular in many domains. Such as navigation and traveler services [9, 10], shopping and entertainment services (ex. finding nearby pharmacy), emergency situations (ex. nearest medical facility), information services [11], and many others [12]. Such services require access to the user's location, which compromises the user privacy even if several location cloaking techniques are applied [13]. To tackle this challenge, Amoli et al. [13] propose a protocol to preserve privacy in location-based services while satisfying the requirement of accurate location use. Similarly, Puttaswamy et at. [14] propose an approach to encrypt location data.

The ultimate objective of AppaaS is to relieve users from having to search and install relevant applications to their current situation, especially with the huge size of the application market. Sharing our objective, Quah et al. [15] propose a mobile application distribution system that retrieves relevant application based on location information. The system uses the location as the main diver to download mobile applications. Which App? [16] also address the same objective in a different way. It proposes a mobile application recommender system that monitors the social interaction and behavior of users to recommend relevant applications accordingly. Toye et al. [17] employ personal information stored smartphone to customize the behavior of site-specific application to best fit a particular user. AppaaS shares the same interests of finding relevant applications to a particular context. However AppaaS offers more advantages in managing and controlling the behavior of such applications according to users' access privileges and time constraints. In addition, AppaaS offers the opportunity to preserve the application state that is relevant to users for their future benefit.

A. Context Management

Exploiting context information in developing mobile applications opens up new opportunities for a smart generation of applications that dynamically adapt to the user's context.

- Location: The location of a mobile user influences what applications the user may run on his/her mobile device. A user within a certain location might not be aware of relevant application(s) to run in order to get the best service a location might provide.
- User Profile: Different users may have different access rights to perhaps the same applications. For example, within an enterprise different users may have different roles and various job responsibilities, which influence their privilege to access certain information or enterprise-related confidential data. Furthermore, same users could have a variant level of access privilege to such data according to their location or current time.
- Device Profile: AppaaS exploits the device profile in order to identify the appropriate version of a relevant application that fits the device platform.
- User ratings: Web 2.0 and open environments have enabled users to leave feedback and share their real experience in consuming services. User ratings reflect the user perceived quality of service.

B. State Preservation

State preservation means that applications can save their latest status and user-specific data for future access. Current mobile platforms do not natively support state preservation at any level. However individual applications can manage their own state at different levels depending on the objective from state preservation. For example, applications may use checkpointing techniques [18, 19] to suspend and resume their execution for migration purposes. Generally speaking, an application-independent state preservation of user-specific data remains challenging. To this end, our current implementation of AppaaS assumes that applications provide two proprietary APIs for the sake of state preservation. One API saves the application current state, where all user-specific data is saved in an XML file format. The other API uploads an application state from an XML file when the application launches.

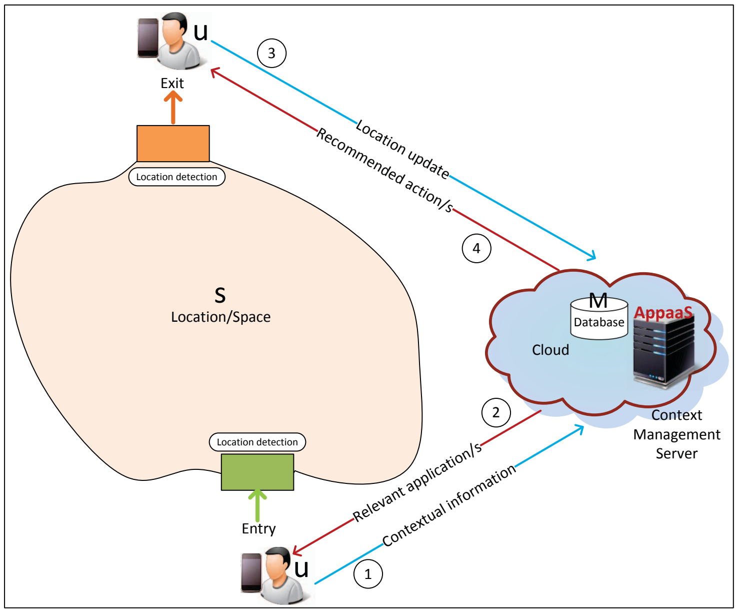
IV. AppaaS: System Architecture

AppaaS aims to cater the right mobile application with the right access regulations according to the current context, including the location, user profile, mobile devices, and time. Figure 1 shows an overview of AppaaS system architecture. The architecture encompasses three main entities, mobile user, location, and AppaaS server. We assume that the user has a smartphone that is capable of running mobile applications, the location is associated with particular mobile applications to better provide a specific service, and the AppaaS server is hosted on the cloud and is continuously available.

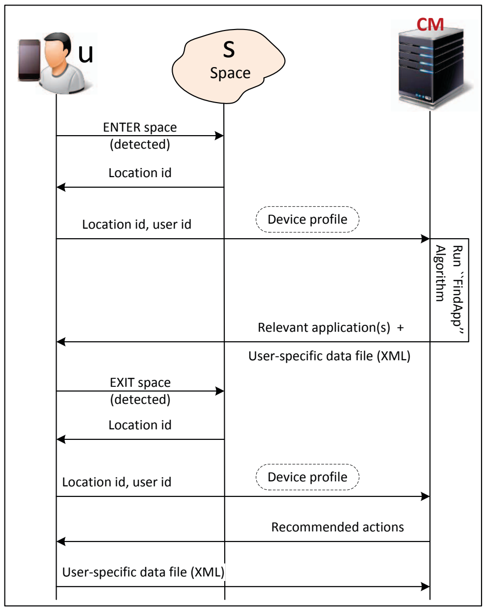
AppaaS maintains a database that stores information about each of the system main entities, namely, users (U), spaces (S), i.e. physical locations, and mobile applications (M). A user $u \in U$ has a set of credentials R_u and an application $m \in M$ has a set of access right A_m. A user u is granted access to an application m if $R_u \overset{satisfy}{\rightarrow} A_m$. Variations of access rights could be granted to a user according to how the user's credentials satisfy an application's access constraints. Users are identified by their user ID (`user_id`), while applications are identified by a system-generated (`app_id`). AppaaS sets the user ID to be the user's email address. Similarly, spaces (or locations) are identified by a numeric location ID (`location_id`) representing a physical location $s \in S$.

Each space $s \in S$ is associated with one or more mobile applications $m \in M$. Once a registered user enters a designated space that is associated with mobile application(s), the user's smartphone detects the location and sends the location information to AppaaS context manager for manipulation. AppaaS collects the device profile during communication sessions and retrieves respective context (such as user profile) from the database. AppaaS then checks these various context information and dispatches the appropriate application to the user (if any found). If the user has previously used this application, it starts with the latest state that the user had left the application with when it was suspended or removed last time.

Upon leaving the designated space, the user's smartphone reports that the user is currently outside the space. The AppaaS server reassesses the current context and sends proper actions



(a) Spatial view of AppaaS system architecture illustrating different composing entities



(b) Logical interactions between various system entities

Fig. 1: A high level system Architecture and interactions between different system entities of AppaaS

to follow. These actions could be uninstalling or inactivating certain applications as shown in Figure 1b.

V. Implementation Details

We have developed a prototype for AppaaS. The prototype has two parts, one part resides on the mobile side in a form of mobile application that represents the system user interface. This part is implemented using the latest Android SDK [20]. The second part resides on the AppaaS server side, which in our case hosted on the cloud. The server is responsible for handling context information and provisioning relevant mobile applications (if any found) along with recommended actions.

A. User Interface

The registration is straightforward since it needs only a few information, namely ``name'', ``email address'', and ``password''. The ``email address'' is the user ID which is unique in the system and it is used to identify the user related information throughout the system functions. Figure 2 exhibits the flow of the main system processes and how they exchange information between each other.

Once the user is successfully authenticated, AppaaS main process daemon is initiated. When the user enters an AppaaS-registered location, AppaaS sends the location information, user ID, and device profile to AppaaS server for processing. In our prototype, we identify the location using RFID tags. We deploy two RFID tags, one at the space entrance and the other one at the space exit. The user taps the mobile application to scan the location ID using the Near Field Communication (NFC) technology, at both sides and report that to the AppaaS *Context Manager*. *NFC scanner* reads NFC Data Exchange Format (NDEF) messages from RFID tags. We store the location information on tags in plain text. The entrance tag holds "ENTER:location_id" while the exit tag holds "EXIT:location_id", where the location_id represents the location ID.

AppaaS *Context Manager* manipulates the context information and replies with recommended actions. In the case where an application is found relevant to the processed context, the AppaaS *Downloader* downloads and installs the respective application onto the user's smartphone along with the proper access privileges or function constraints. The *Context Manager* applies Algorithm 1 to find relevant application to the current user's context. The *"match"* function applies Equation 1 to match the user credentials with the applications access requirements.

$$match(R_u, A_m) = \frac{\sum_{i,j} F(r_{u_i}, a_{m_j})}{i * j} \tag{1}$$

where $r_u \in R_u$ denotes a single user credential and $a_m \in A_m$, denotes a single applications access constraints, i and j are the number of credentials and access constraints, respectively. $F(r_u, a_m)$ represents how much r_u satisfies a_m and it produces values $\{0,1\}$. A value of 0 means that if the respective a_m is a constraint at the application level, the application will not be available for the user and if a_m is a constraint at the function level, this functionality will be disabled for the user. Otherwise access is granted. Each relevant application has two scores, relevancy score ($match_m$) and rating score ($rating_m$). For simplicity, we consider the two scores are equally important.

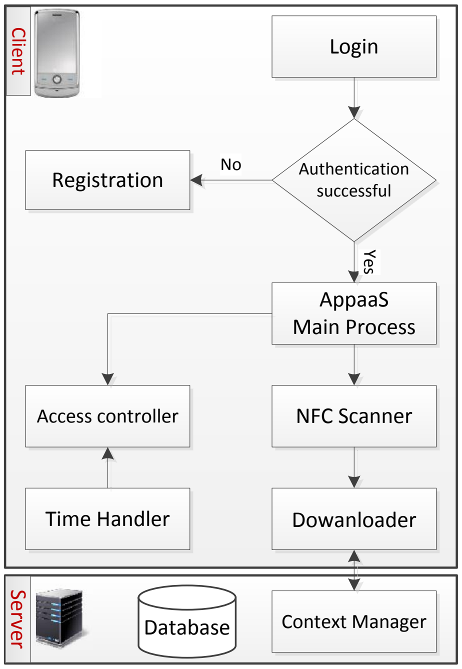


Fig. 2: AppaaS components and process flow

Algorithm 1: Find relevant application(s) to a user's context

```
Input: user_id, location_id, device_profile,time
Output: download url of relevant application
Function FindApp(user_id,location_id,device_profile,time)
Initialize Apps=[] //App collector
if location_id has application(s)($M$) then
    foreach $m \in M$ do
        Initialize $match_m$=0 //application relevancy score
        Initialize $rating_m$=0 //application rating score
        // retrieve application access constraints $A_m$
        if $m$ has no absolute time constraints then
            if $m$ has open-access then
                // m gets max. score
                $match_m$ =1
            end
            else
                // retrieve user profile and credentials $R_u$
                Initialize $match_m$=0
                foreach $r_u in R_u$ and $a_m \in A_m$ do
                    // check how much $r_u$ satisfy $a_m$
                    $match_m$ = $match_m$ + $F(r_u, a_m)$
                end
            end
        end
        $m = 0.5 * match_m + 0.5 * rating_m$
        // add $m$ to relevant Apps
        Apps=Apps + $m$
    end
end
if Apps is not null then
    //choose the application with the highest score
end
return url //if null means no applicaitons
```

B. AppaaS Server-Side Components

AppaaS system has two components that run on the server: *Database Server* and *Context Manager*. The latter handles context information sent by the user, while the former maintains information about various system entities, users, applications, and a list of registered locations. The database is implemented using MySQL Database Server [21]. It maintains various tables that hold information about entities and their relations.

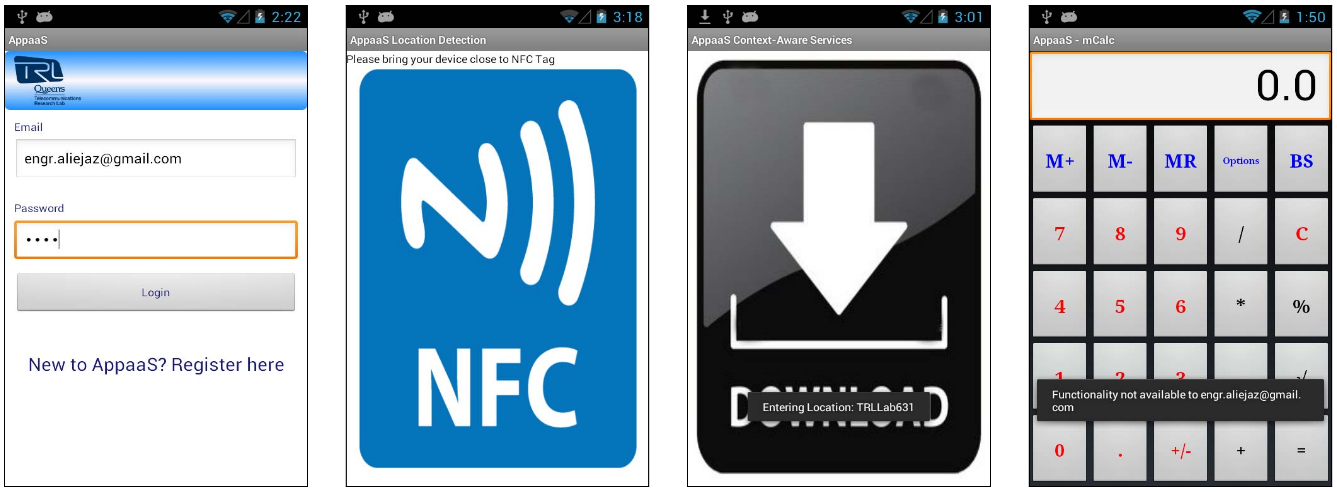
VI. Experiments & Discussions

The prototype aims to highlight three aspects of AppaaS functionalities. The first aspect is how AppaaS finds mobile applications which are relevant to a specific context information. The second aspect demonstrates the ability of AppaaS to customize the application behavior in order to fit certain access rights constraints. The third aspect shows how AppaaS preserves the application state relevant to a user for future reference.

To demonstrate the former functionality, we have defined some locations and associated these locations with specific mobile applications. The mobile application packages (.apk) are stored on the AppaaS server with each package associated with a *url* for download purposes. The *ID* of one of these registered locations is stored on two RFID tags, one represents the location entrance while the other one holds the NDEF message that denotes the location's exit. To start with the system, a user logs into the AppaaS as shown in Figure 3a. The user then taps the RFID tag that represents the location entrance as shown in Figure 3b. The user's context and the location information are sent to the context manager for processing. The context manager finds the relevant application that fits this context and sends the download *url* back to the user's device. The *Downloader* at the user's device downloads and installs the application as exhibited in Figure 3c.

AppaaS can also apply access limitations to certain functionalities according to the user's access privileges. To demonstrate this functionality, we have developed *mCalc*, a proof-of-concept mobile application that shows how AppaaS supports control over access. *mCalc* implements the basic functionality of a calculator, Figure 3d. We have set restrictions on some functionalities to be accessible only by particular users. Therefore, if an unauthorized user tries to perform these restricted functions, an access denial message appears as illustrated in Figure 3d.

To test the state preservation aspect, we have set the application *mCalc* to be accessible by a specific user during two specific time periods, 10:00-11:00 AM and 11:15-11:30 AM. This user logs into the system during the first time period and performs calculations. When the first period ends, AppaaS stops mCalc and initiates the state preservation procedure. The generated state file is sent to the context manager, where it can be stored. Then, AppaaS invokes the uninstall procedure of *mCalc*. The same user logs into the system again during the second period of time, when the application downloads onto the user's smartphone, the context server finds that there is a previous state preserved for the user for this application. The context manager sends this state with application download *url*. The user interface shows a message while the application is launching to choose whether to resume the application with the latest previous state or to initiate a clean start.


AppaaS

Email: engr.aliejaz@gmail.com

Password: ••••

New to AppaaS? Register here

AppaaS Location Detection

Please bring your device close to NFC Tag

AppaaS Context-Aware Services

Entering Location: TRLLab631

AppaaS - mCalc

0.0

Functionality not available to engr.aliejaz@gmail.com

(a) A user login screen.

(b) A user scans the location identification tags using NFC technology.

(c) A relevant application downloads to a user's smartphone.

(d) mCalc shows access denial message for restricted functionalities.

Fig. 3: Various AppaaS screenshots that show different system activities.

Acknowledgment

This research is funded by a grant from the Ontario Ministry of Economic Development and Innovation under the Ontario Research Fund-Research Excellence (ORF-RE) program. The authors would like to thank Dr. Kashif Ali for his valuable input.

VII. Conclusion

This paper presents *AppaaS*, a system for provisioning context-aware mobile applications *as a service*. AppaaS uses various context information including location, user information, user ratings, device profile, and time to provide the appropriate mobile application(s) to such a context. AppaaS also supports access control over application functionality according to users' privileges and access rights. We have developed a prototype to highlight three features, specifically, finding the appropriate application to a certain context, controlling access over an application's functions, and preserving the application's state that is relevant to a particular user for future reference.

References

[1] Google Play Store, https://play.google.com/store?hl=en.

[2] Apple Store, http://store.apple.com/ca.

[3] F. Toutain, A. Bouabdallah, R. Zemek, and C. Daloz, "Interpersonal context-aware communication services," *Communications Magazine, IEEE*, vol. 49, no. 1, pp. 68 –74, 2011.

[4] K. Elgazzar, H. Hassanein, and P. Martin, "Effective web service discovery in mobile environments," in *P2MNETS, The 36th IEEE Conference onLocal Computer Networks (LCN)*, pp. 697–705, October 2011.

[5] M. F. Mokbel and J. J. Levandoski, "Toward context and preference-aware location-based services," in *The 8th ACM International Workshop on Data Engineering for Wireless and Mobile Access*, pp. 25–32, 2009.

[6] I. A. Junglas and R. T. Watson, "Location-based services," *Communications of ACM*, vol. 51, no. 3, pp. 65–69, 2008.

[7] L. Chatterjee, S. Mukherjee, and M. Chattopadhyay, "A personalized mobile application using location based service," in *Advances in Computer Science and Education Applications*, vol. 202, pp. 413–419, 2011.

[8] C. Ahn and Y. Nah, "Design of location-based web service framework for context-aware applications in ubiquitous environments," in *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, 2010.

[9] W. Husain and L. Dih, "A framework of a personalized location-based traveler recommendation system in mobile application," *Journal of Multimedia and Ubiquitous Eng.*, vol. 7, no. 3, pp. 11 – 18, 2012.

[10] X. Shi, T. Sun, Y. Shen, K. Li, and W. Qu, "Tour-guide: Providing location-based tourist information on mobile phones," in *The IEEE 10th International Conference on Computer and Information Technology*, pp. 2397 – 401, 2010.

[11] P. Stuedi, I. Mohomed, and D. Terry, "Wherestore: Location-based data storage for mobile devices interacting with the cloud," in *The 1st ACM Workshop on Mobile Cloud Computing and Services: Social Networks and Beyond*, 2010.

[12] C.-Y. Chow, M. F. Mokbel, and X. Liu, "Spatial cloaking for anonymous location-based services in mobile peer-to-peer environments," *Geoinformatica*, vol. 15, no. 2, pp. 351–380, 2011.

[13] A. Amoli, M. Kharrazi, and R. Jalili, "2ploc: Preserving privacy in location-based services," in *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pp. 707 –712, 2010.

[14] K. P. N. Puttaswamy and B. Y. Zhao, "Preserving privacy in location-based mobile social applications," in *The 11th Workshop on Mobile Computing Systems and Applications*, pp. 1 – 6, 2010.

[15] J. T. S. Quah and L. R. Lim, "Location based application distribution for android mobile devices," in *The IASTED International Conference on Wireless Communications*, pp. 118 – 123, 2011.

[16] E. Costa-Montenegro, A. Barraga andns Marti andnez, M. Rey-Lo andpez, F. Mikic-Fonte, and A. Peleteiro-Ramallo, "Which app? a recommender system of applications in markets by monitoring users' interaction," in *2011 IEEE International Conference on Consumer Electronics*, pp. 353 –354, 2011.

[17] E. Toye, R. Sharp, A. Madhavapeddy, and D. Scott, "Using smart phones to access site-specific services," *IEEE Pervasive Computing*, vol. 4, no. 2, pp. 60 – 66, 2005.

[18] R. Tuli and P. Kumar, "Analysis of recent checkpointing techniques for mobile computing systems," *International Journal of Computer Science and Engineering Survey*, vol. 2, no. 3, pp. 133–141, 2011.

[19] S. Biswas and S. Neogy, "A mobility-based checkpointing protocol for mobile computing system," *International Journal of Computer Science & Information Technology*, vol. 2, no. 1, pp. 135–151, 2010.

[20] Android SDK, http://developer.android.com/sdk/index.html.

[21] MySQL Database Server, http://www.mysql.com/.