Pushing the Device-Oriented Information into a Cloud

Mr. Shadab Ahmed Nizami1, Mr. Pravin D. Soni2

1Department of Computer Science & Engineering
P.R Pote college of Engineering & Management, Sant Gadgebaba Amravati University, Amravati, India

2Asst. Prof of Computer Science & Engineering
P.R Pote college of Engineering & Management, Sant Gadgebaba Amravati University, Amravati, India

Abstract

In this paper, we propose a sharing user related information between devices can be achieved using cloud computing, users can access data and services anytime and anywhere. Some information is related to device such as Gmail or Evernote is called device-oriented information. We propose an information-sharing platform that enables easy sharing of device-oriented information we consider missed call history, location and battery information as typical device-oriented information. By cooperating with existing cloud systems, we realized the sharing of such kinds of information between devices. For a first trial, we proposed a platform and released a client application that works on Android. Our platform makes it easy to support new cloud services as output, and our Android application can set various kinds of trigger to push the information into the cloud.

Keywords: Platform, Device-Oriented Information, Cloud, Android.

1. INTRODUCTION

Mobile devices are spreading at a remarkable rate, and more people are starting to use several devices simultaneously as their own. For example, a user may carry around three devices: a cell phone, Android tablet, and laptop computer. In this case, sharing information between devices becomes important. Information such as mail or the schedule is related to the user and not to devices. Sharing such information between devices would be of great help to users.

At the same time, cloud computing [1] is also becoming popular. Through the development of cloud computing, service providers no longer need to worry about resource management. Resources are managed by cloud providers, and service providers can use resources depending on their demands. In addition, users can access data and services anytime and anywhere. This lets users share data more easily than before. Users can access the same data in the same way from any device. Cloud services include e-mail (e.g., Gmail), notebooks (e.g., Evernote), schedules (e.g., Google Calendar) and photos (e.g., Flickr), which can be accessed from any device. Users can easily share such information between devices, and there is no stress in switching their active devices depending on their situations. For example, when a user uses Google Calendar to manage his or her own schedule, he or she uses desktop computers at work, laptop computers during a business trip, and smart phones in transit. Location information (e.g., Google Latitude) is also becoming part of cloud service. Similar to foursquare, users can share location information to both themselves and their friends. Moreover, as in Android, the application list of a device is uploaded onto cloud services, and applications on the list can easily be installed in another device. This also helps users when one device breaks down since data from a cloud service can easily be recovered to another device.

However, some information such as missed calls and sensor information, referred to as device-oriented information, is still isolated in a particular device. However, since such information is related to a user, it should be shared between the user’s devices. When such sharing is realized, a user can be notified of a phone call to his or her home telephone or an SMS to his or her cell phone from any device the user has immediate contact with, and communication becomes smoother. Sharing sensor information between devices is also helpful. For example, PCs are usually not equipped with GPS sensors, but mobile phones usually are. If the location information of the mobile phone’s GPS sensor can be used at the user’s PC, this would help in the use of location-oriented services. We assumed that we can solve these situations by sharing this information through some cloud services.

An information-sharing platform that enables easy sharing of device-oriented information. To share information, we use existing cloud services. Through the cloud, information is shared among devices. Our platform provides users with an easy way to connect device-oriented information and cloud services. Users can select information that they want to share and services that they can easily refer to for information. We can set different triggers and conditions to upload information: for example, moving distance per hour, frequency of upload, message formatting, etc. We implemented the platform as an Android application, which we named Notitter. We can set the sharing information (input) and information sharing service (output) of the application. We assumed that typical input includes location information, battery information, and missed call history. We preset three types of cloud services as output: Gmail, Evernote, and Twitter. Information and sharing services are not limited to items above, and we can easily add more input and output to...
our platform. After Notitter is activated, it automatically uploads device-oriented information into cloud services, and information can be accessed from any of the user’s devices. This paper is organized as follows. In section 2, we present related works. In section 3, we present the architecture of the presented platform, and the implemented application. Finally, in section 4, we present our conclusions.

2. RELATED WORKS
Several solutions of information sharing have been proposed. Growl [2] is a local application that notifies users what happens in other applications and devices. This is activated in the user’s PC and smart phones. By using its API, every application that originally has no notification capability can automatically notify the user about application information. It can share information between devices if they are configured. For example, if a user receives an e-mail in his or her mobile phone, the application in the phone sends a notification to Growl, which shows a pop-up to notify the user that the phone received the e-mail. If Growl is configured in other devices inside LAN, the application will also notify the event to other devices. Since it is designed to notify application information to certain user’s devices, only devices that are inside LAN can access the information.

SECE (Sense Everything, Control Everything) [3] is a context-aware platform that lets users use context-aware services easily. SECE integrates information from sensors equipped to a PC and cloud services and enables easy organization of services. SECE is particularly suited as a platform for sensors without communication means of communication or API and helps to create services that integrate this information. Users can easily define their own service by SECE’s language, so they do not have to worry about each sensor’s programming. However SECE is designed to collect information of devices that are connected to a PC. Pushing information to other devices is not considered in SECE. Although users can access data from the PC, sharing of information between devices is difficult.

Context Watcher [4] is a context-aware platform that lets mobile phone users easily share personal data. It helps users share data with their friends and families by automatically providing context data to information services. It can also derive information from context data using other services including web services. However, it does not mention where data will be stored on. Photos are stored on Flickr, but it does not specify where other data is stored. This is a problem when accessing information from other devices. Since Context Watcher is a platform for sharing data with others, it does not mention users themselves observing their own data. To share device-oriented information of a user, we need to tie the devices of a single user together. This is also a platform that is specialized for mobile applications, and cooperation with other kinds of devices is not mentioned. These solutions share data only on particular devices or services. Our goal is to make any device that a user has be able to access to any information of the user. However, the solutions above let make users access information only in a particular way and restrict of free access to information. To solve this problem, we introduce the use of cloud services for sharing information. This makes it easier for users to access information in various ways. Access to cloud services is easy, and users are free from a need to specify each device. For example, users who want to notice device oriented information soon can use services they often use such as Twitter or Facebook. Users who want to have a life log can use storage services such as Evernote. Using a cloud service also frees users from managing data in a particular device. Storage, especially in mobile devices, is limited, and networks are still poor today. In contrast, cloud services have high network ability, and thus developers do not need to incur costs in managing them.

3. PROPOSED SYSTEM
A. System Outline
Our final goal was to complete Notitter as a platform to share all information and free users to use any device they have. To realize this, we hope many developers will help enhance functions of Notitter. It is necessary to make Notitter an easy
platinum to develop plug-ins. To realize this, we separated Notitter into three main parts: input, output, and controller. These are described later. The problems below needed to be addressed.

• Easy adding of services
A mechanism to make it easy to add information sharing services and shared information is needed because services in a cloud increase rapidly. Device oriented information also needs to be added more easily. Data that need to be shared differ with each service and user, and information may be data from a sensor or user context data. A definition on the interface between the controller and input/output is needed to solve these problems.

• Relation of Trigger and Information
Triggers and information sharing tend to be related, and we need to collect them. For example, the platform is first gets battery level and notifies the controller. The controller then checks the level; if it is lower than a threshold, the controller sends the level to output. We need to support easy development of the input and trigger so that developers can release services simultaneously.

• Enriching Information
We want to make variations in additional information. Adding information that is strongly related to the original information has already been realized. For example, add map can be added to location information from GPS. However, with the development of web services, there are more ways to use information. For example, location information can now be used to find addresses or landmarks nearby. Therefore, we need to make it easy for users to adapt to new services. Moreover, the relations between different types of information differ depending on the person. To resolve this problem, we let users connect information. System security depends on information sharing services that users use. When a user does not want to share information with others, he or she can use services that are limited to personal use. When a user wants to share information with his or her friends, he or her can use services with community.

B. System Implementation
In this study, we implemented a prototype of the platform on Android; we named the platform Notitter. Notitter consists of three main parts: input, output and controller. It is shown in Fig.2. Notitter first needs to set some parameters of the parts as shown below:

• Sharing Information (input)
A user can choose information to push to external services. Notitter currently supports three types of information: location information from GPS, battery power of the device, and missed call history.

• Controller
A user can set which information to share and the triggers for sharing information. There are currently two trigger settings: decline in battery power and missed call.

• Sharing Services (output)
A user can select services to share information. Notitter currently uses two services: Twitter and Evernote. We assumed that users refer to shared information from each external information service. Thus, this was not in the scope of this study. Notitter works as described in the steps below:
1) The user sets the settings above.
2) The user starts Notitter which works in the background.
3) Notitter automatically checks the input data at constant intervals.
4) If a trigger event is satisfied, Notitter share data with information sharing services.

In addition, if a user wants to check for additional information, Notitter adds information by using web services. Currently, we have implemented the adding of addresses. For example, when a user checks the location and additional information of an address, Notitter finds address from the latitude and longitude by using a location search web service and adds the address to sharing information.

Screen shots of Notitter are shown in Fig. 3. On the main page, each item that the user can select is shown. In the implemented system, we can choose the setting for sharing information, setting of cloud services, and activation of the system. The setting of each item is a layered structure. For example, when the battery is chosen, the input data and trigger level of the battery can be set. The settings of cloud services are implemented by using their APIs.

Fig. 4 shows the functions of Notitter. The user can select each item from the home screen. By checking “service,” Notitter activates in the background and pushes device oriented information to external services. “Tweet time” sets the update cycle of Notitter, in terms of minutes. “GPS,” “battery” and “missed call” are settings for sharing information. In “GPS”, the user can add address information in addition to the coordinate information from the GPS sensor. “Battery” is used as a trigger event, and the user can set the event level of the battery. “Missed call” is used as a trigger event. “Twitter” and “Evernote” are external services that are used as information sharing services. The user can register an account and set the use of services.

We present an example scenario of using Notitter. The user first sets the settings of Notitter on the smart phone.

---

Figure 3. Screen of Notitter

Figure 4. Functions of Notitter
Here, the user sets the battery level and location as information to be shared. For the battery level, the user sets two levels: emergency notifying level and battery charging level. The user also sets Twitter as an information sharing service. When the user is working with his or her laptop computer, he or she is not paying attention to the phone. In this situation, he will not notify if the phone has a low battery level. However, when the phone’s battery level becomes below the charging level, Notitter notifies the information to Twitter. Since the user is always paying attention on Twitter, he can be notified of the decrease in battery level from any device. This helps the user charge the phone’s battery. In addition, if the user loses his phone and cannot find it, Notitter will notify him or her of the location of the phone. When the battery level goes below the emergency level, Notitter notifies the user of where the phone is using the address of the lost device in addition to its coordinates.

4. CONCLUSION
In this paper, we propose a platform and application to share device-oriented information between devices. Three types of information were shared in cloud services, and two cloud services were selected as information sharing services. These let the user automatically share device-oriented information so that he or she can observe the same information on different devices. By using cloud services, users can now access information anytime and anywhere regardless of the situation of information source device. For future work, we need to adapt Notitter to various devices and share more information. Notitter was developed with Java and implemented on Android. For more integration of information between a user’s devices, adapting to various devices is needed. Notitter is still in the developmental phase, and plug-ins cannot be added. We need to develop an interface for the platform to provide easy development. We also need to make Notitter easy for users to connect information.

REFERENCES

AUTHORS:
Shadab Ahmad Nizami received B.E degree in Information Technology from Anuradha Engineering College Chikhli, Buldhana in 2011. And now pursuing M.E degree in Computer Science & Engineering from P.R Pote college of Engineering & Technology.

Mr. Pravin D. Soni received B.E in Computer Science & Engineering from Govt. College of Engineering Amravati in 2008. Worked as a software developer in Persistent System for 1 year. Currently working as a Assistant Prof. in P.R Pote(Patil) College of Engineering & Management.