

Development of Mobile Intercloud Applications

Yuxuan Deng and Henry C. B. Chan

Department of Computing
The Hong Kong Polytechnic University
Hong Kong

Abstract — With the advent of cloud computing and mobile computing, many smart computing applications for mobile data management can be developed. In this paper, we present a development framework and tool for developing innovative mobile Intercloud applications. By means of mobile Intercloud, data can be transferred and processed through the collaboration of clouds as controlled by a mobile terminal. To facilitate the development of mobile Intercloud applications, we have developed a package of App Inventor blocks. Working with the InterCloud Communications Protocol, these graphical programming blocks allow various mobile Intercloud applications to be developed effectively and efficiently. Three innovative applications are presented to illustrate the benefits of mobile Intercloud applications. An evaluation and performance analysis were conducted for an automatic file transfer application. The evaluation and results provide valuable insights into the design of smart computing applications in general and mobile Intercloud applications in particular.

Keywords — *cloud computing, mobile Intercloud, mobile computing*

I. INTRODUCTION

Mobile cloud computing provides an effective way to manage data and computing resources for mobile terminals over the Internet. While current work mostly focuses on computational offloading, another important mobile cloud computing service is cloud-based mobile data service, which makes various smart computing applications possible. In recent years, there has been various related work on mobile data service over clouds. For example, with the advent of mobile cloud computing, cloud-based mobile databases can be set up more effectively. [1] and [2] presents a service framework for mobile cloud databases. To facilitate mobile data management, another important issue is to set up data centers cost-effectively, and [3] presents a method to address this problem. With the development of many clouds, it is also of interest to study how clouds can cooperate to provide more effective mobile data services [4]. In fact, this paper also seeks to contribute to this direction. From an efficiency perspective, it is also important to enhance the data throughput of mobile data service. For instance, [5] presents a multi-path prefetching algorithm for this purpose. Obviously, security is of great concern in a cloud-based mobile data service. In this aspect, [6] studies secure data storage over multi-clouds, [7] presents a mechanism to enhance data integrity and [8] proposes a secure data distribution method. Last but not least, energy-efficiency and fault-tolerance are also important issues in a cloud-based mobile data service [9]. Contributing to this important

research area and complementing previous work, this paper studies the development of mobile Intercloud applications for data management. Based on the IEEE P2302 architecture and our previous work on the InterCloud Communications Protocol (ICCP), we developed App Inventor graphical programming blocks for creating mobile Intercloud applications. We present three innovative use cases or smart computing applications, and evaluate the implementation of an automatic file transfer application. By using graphical programming blocks, various mobile Intercloud applications can be developed effectively and efficiently.

The remaining sections of this paper are outlined as follows. Section II presents an overview of the Intercloud system and protocol. Section III explains the graphical programming blocks for developing mobile Intercloud applications based on MIT App Inventor. Section IV illustrates three mobile Intercloud applications. As an example, Section V evaluates one of the applications – an automatic file transfer application. Finally, Section VI concludes this paper.

II. INTERCLOUD SYSTEM AND PROTOCOL

In this paper, we adopt the Intercloud system based on the IEEE P2302 Intercloud architecture. There are three main components, namely the Intercloud root for providing directory and certification service, Intercloud gateways for enabling Intercloud communications, and Intercloud exchanges for facilitating communications among Intercloud gateways and the Intercloud root. Each cloud is connected to an Intercloud gateway. Based on our previous work [10], the Intercloud gateways can communicate with one another using the InterCloud Communications Protocol (ICCP) with XML-based messages. Based on [11], the following request is a simple example of transferring Object1 from 'cloud1.iccp.hk' to 'cloud2.iccp.hk'.

```
<Request Version="1.0" ID="123456789">
  <GeneralInformation From="cloud1.iccp.hk"
    To="cloud2.iccp.hk" Date="2018-01-12"
    Time="16:36:00"/>
  <RequestInformation Service="ObjectStorage"
    Command="PutObject">
    <ObjectName>Object1</ObjectName>
    <TransferMethod>Embedded</TransferMethod>
  </RequestInformation>
  <AdditionalInformation>... </AdditionalInformation>
</Request>
```

In this example, cloud ‘cloud1.iccp.hk’ requests cloud ‘cloud2.iccp.hk’ to receive data via embedded transfer and store it by overwriting the previous versions. Such data are presented in the Base64 encoding and are named, ‘movie1’. Once ‘cloud2.iccp.hk’ receives the message request, it will download the object from ‘cloud1.iccp.hk’ first. After processing, it will send a response back to ‘cloud1.iccp.hk’. The example of the response message is shown below.

```
<Response Version="1.1" ID="987654321">
  <GeneralInformation From=" cloud2.iccp.hk" To="
cloud1.iccp.hk" Date="2018-01-12" Time="16:37:03" />
  <ResponseInformation Service="ObjectStorage"
Command="ConfirmationForPut">
    <ObjectName> Object1</ObjectName>
  </ResponseInformation>
  <AdditionalInformation>... </AdditionalInformation>
</Response>
```

An Intercloud gateway can provide an application programming interface (API) (in Java) for writing Intercloud computing applications [11]. Essentially, the API can be used to invoke required ICCP messages for the Intercloud operation (e.g., to obtain or input data). In this paper, we develop App Inventor blocks to facilitate the development of mobile Intercloud applications using the API.

III. PROGRAMMING BLOCKS

This section discusses the functions of the fundamental programming blocks of a mobile Intercloud system written by App Inventor. The programming blocks are divided into three groups: configuration-related blocks, operation-related blocks and security-related blocks. Configuration-related blocks are responsible for storing basic information from the gateways and clouds. Operation-related blocks perform cloud-based processing and operation. Security-related blocks are for security or cryptographic operations, such as encryption, decryption and digital signature.

The configuration-related blocks consist of functions and events. There are four configurations: cloud, location, automated transfer files, and backup files. Configurations should be used before saving. For example, if we want to use cloud-related information, we should first use the “ConfigureCloud” block to store the required configuration, including the cloud domain name, cloud type, bucket, access key and secret key. The second group is the operation-related blocks containing “TransferFile”, “List”, “Download”, “Upload” and “Delete”. “TransferFile” is used to transfer files/data between two clouds. The other four blocks are used between a mobile terminal and a cloud. “List” returns file name, file size and modified time of the user files in the selected cloud. “Download” moves the chosen files from the cloud to the mobile terminal’s local file system. “Upload” uploads the selected files from the mobile terminal to the cloud. “Delete” deletes files on the selected cloud. The last group is security-related blocks, including encryption, decryption, and signature generation and verification functions. Tables I-II show some examples of the key programming blocks.

TABLE I. CONFIGURATION-RELATED PROGRAMMING BLOCKS

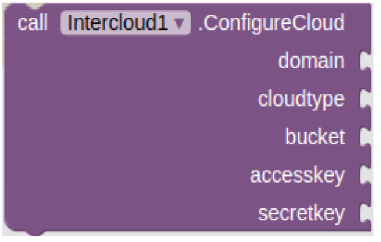
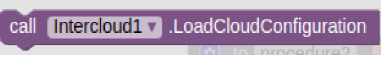
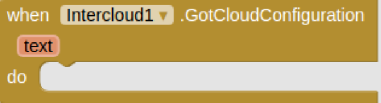
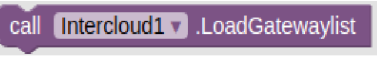
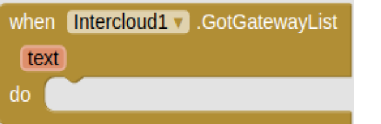
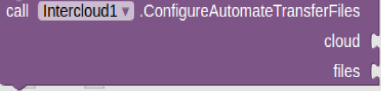
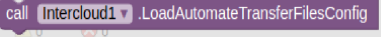
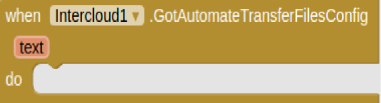
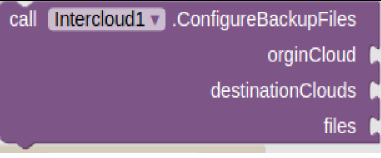
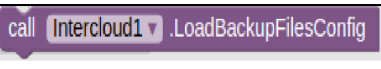
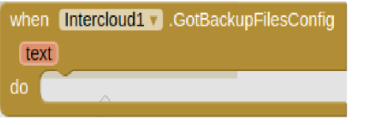
	Save different cloud configurations.
	Load cloud configuration.
	Invoke actions after loading the cloud configuration.
	Load the gateway’s list.
	Invoke actions after loading the gateway’s list.
	Save names of files that need to be transferred automatically.
	Load names of files that need to be transferred automatically.
	Invoke actions after loading the file names of automated transfers.
	Save names of files that need to maintain a backup.
	Load names of files that need to maintain a backup.
	Invoke actions after loading the names of files that need to maintain a backup.

TABLE II. OPERATION-RELATED PROGRAMMING BLOCKS

call Intercloud1 .TransferFile origin destinations files	Transfer files from origin cloud to multiple destination clouds.
call Intercloud1 .List cloud	Get file list from the cloud.
call Intercloud1 .Download cloud files	Download multiple files from the cloud.
call Intercloud1 .Upload cloud files	Upload multiple files to the cloud.
call Intercloud1 .Delete cloud files	Delete multiple files from the cloud.

IV. MOBILE INTERCLOUD APPLICATIONS

In this section, for evaluation purposes, we use the aforementioned App Inventor-based programming blocks to develop three useful and interesting mobile Intercloud applications. By using the programming blocks, various mobile Intercloud applications can easily be developed.

A. Application 1 – Automatic File Transfer

As shown in Fig. 1, this application allows a user to transfer files from one cloud to another cloud based on certain user-defined policies (Step 1), such as location-based policies. Initially, a user should predefine policies for automated file transfer, such as when the file transfer should occur (e.g., based on GPS information) and what files should be transferred. For example, when the user travels from HK to UK (Step 2), the GPS sensor of the mobile terminal detects the location change (i.e., the associated cloud should be changed) (Step 3). Then the app will notify the gateway in HK to send predefined files to the destination cloud in UK. After the files are transferred to UK (Step 4), the user can download the files from the UK cloud instead of the HK cloud (Steps 5.1 and 5.2). However, he or she can also request and obtain files from the HK cloud as well (Steps 6.1 and 6.2). This means that the user can access frequently used files from the nearby cloud and other files from the origin cloud. Once the person travels back to HK (Step 7) as indicated by the GPS sensor (Step 8), the files will be transmitted from UK back to HK (Step 9). The app invokes the TransferFile block to copy frequently used files from one cloud associated with the previous location to another cloud associated with the current location. In this scenario, it is more cost-effective to read/write files located closer to the user because of less access delay.

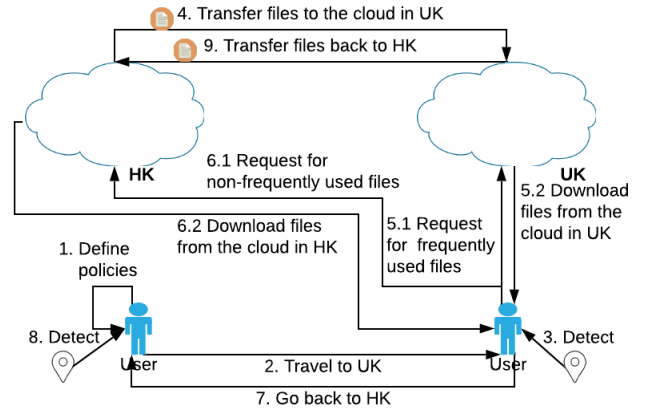


Fig. 1. Basic operation of automated file transfer

B. Application 2 - File Backup Manager

For both personal and business reasons, we generate many files and data, which should be backed up effectively and efficiently. Therefore, it is desirable to develop a File Backup Manager to automatically back up files in different clouds (see Fig. 2). In this case, a user should have at least two clouds (e.g., one private cloud and one public cloud). Initially, the user can make the backup plan (Step 1), for example, backing up at 2 a.m. daily, at 5 a.m. weekly on Saturday or at 4 a.m. monthly on the second Friday of the month, etc. He/she can also decide the scope of the backup files. For example, the files can be specific files that are changed within three days or whose size is less than a pre-defined threshold. When these conditions are met (Steps 2.1 – Step 2.n), the mobile app will trigger the backup process. The user can also initiate the backup process manually (Step 3.1). Once the user selects a file in one cloud and clicks the backup button, a copy of the file will automatically be sent to the other cloud (Step 3.2). The PutObject command in ICCP is used for this operation. Before transferring, the app should check to determine whether there is enough storage space in the target cloud. Additionally, the app provides a version control function to facilitate better file management. According to the user's instructions, if a file is modified after being transferred to another cloud, the new backup file will be renamed using the old name plus a version number, to avoid rewriting the original backup file. This means that a list of backup files with version numbers can be maintained.

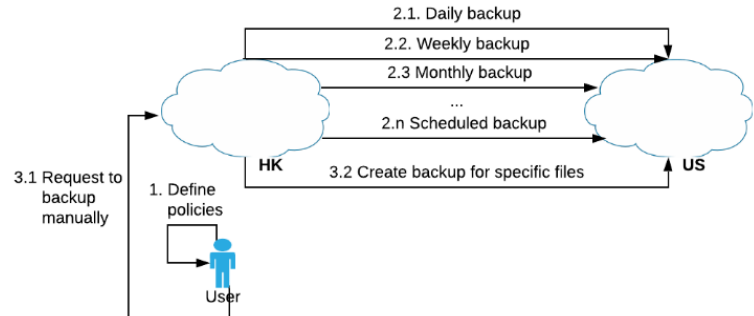


Fig. 2. Basic operation of file backup manager

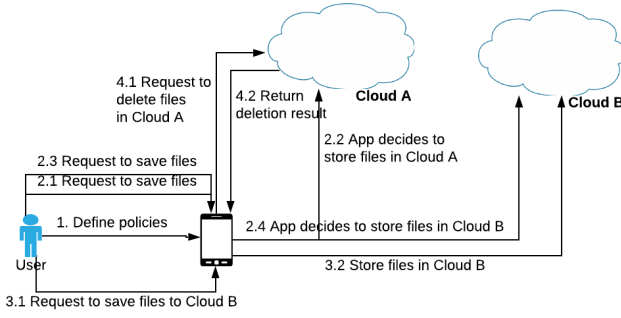


Fig. 3. Basic operation of cloud object manager

C. Application 3 - Cloud Object Manager

Nowadays, we usually have more than one cloud to store files, hence it is important to manage files among different clouds. As cloud resources are limited (e.g., a cloud provider may only provide a certain amount of free space), we should try to make the best use of available storage. It is not easy for a human to handle storage management manually. So, it is necessary to develop a mobile app for Intercloud storage management (see Fig. 3). First of all (Step 1), the user should set required parameters for each cloud such as the total space, cloud domain name, access key, secret key, bucket, and cloud type. Then each time when the data changes, the remaining space will be adjusted accordingly. For instance, when the user wants to save a file to a specific cloud, the remaining space will be computed. Conversely, if the user wants to delete a file from a cloud, the remaining space will also be determined (Steps 4.1 – 4.2). As the app knows the available space of the clouds, it can recommend the best cloud to store files so that the overall space can be maximized (Steps 2.1 – 2.4).

Apart from the above application scenario, a user can also store files to the clouds at his/her discretion (Steps 3.1 – 3.2). By using the mobile Intercloud system, he/she can manage files/data among different clouds through a single interface. There is a cloud list to show all clouds the user can use. Once clicking the cloud name, all objects in the cloud should be listed on the screen, which uses the “List” block to retrieve all file names from the cloud. After selecting each object, user can delete it using the “Delete” block. Furthermore, the user can upload or download files from or to the selected cloud. “Upload” and “Download” blocks are used in these scenarios accordingly. Moreover, the user can put a file from one cloud to another by using the “TransferFile” block. The user can also protect against the accidental deletion of files. For instance, if a file is deleted accidentally, the manager can restore it from another cloud if there is a backup copy in another cloud.

V. IMPLEMENTATION AND EVALUATION

In this section, as an example, we implement and evaluate Application 1 based on the Intercloud graphical programming blocks and other App Inventor blocks. Other applications can be implemented in a similar manner. Application 1 has three main modules: cloud list, cloud location and file transfer. For the cloud list module, it shows all user clouds and allows the user to add new clouds or modify existing ones. Through the cloud location module, the user can link a cloud with its server location for the

automatic file transfer application. The file transfer module is for transmitting files from other clouds to the current one.

To support the aforementioned modules, the corresponding blocks are divided into three categories, which are page-initiation-related, cloud-related, and file-transfer-related. The blocks in the page-initiation-related category are responsible for page initiation and user interface control on different tabs. When a screen is initiated, the LoadPage function is invoked to initiate the ListViews and to read different configuration files. Once the files are read successfully, the related call-back functions are executed to set the global variables and to load the ListViews. The setTab function is used to control changes in the user interface on the Cloud List tab and Configuration tab. It also initiates the location sensor. When the location change is detected, two global variables, CurrentLatitude and CurrentLongitude, are set to the corresponding values. If the location change triggers a file transfer, the corresponding file(s) will be transferred to the current cloud. That means the file transfer is automatic, based on user-defined policies.

The blocks in the cloud-list-related category handle the operations of cloud list, mapping of cloud-files and cloud configuration. ShowCloudConfig function is used to control the visibility of different panels and is invoked after clicking the following buttons: Add, Cancel and Save. For the ButtonSaveCloud click event, the ConfigureCloud function is executed to save the configuration of a cloud, which stores domain, cloud type, bucket, access key and secret key. For the AfterPicking event of ListViewCloud, it shows different files of the selected cloud. The AfterPicking event of ListViewLocation is used to obtain the domain information, which is passed with the current latitude and longitude to the SaveLocation function. The SaveLocation then calls the ConfigureLocation function to save the cloud-latitude-longitude mapping.

The third block category is the core function for file transfer. We shall discuss this core function in detail. There are three blocks in this category: Distance, FileTransfer and Transfer respectively. The Distance block (see Fig. 4) calculates the distance between two points. Based on the given latitude and longitude, this block calculates the distance according to the GPS coordinates using the formula in [12].

The FileTransfer block transfers files when the user moves to another area/region as detected by the mobile terminal’s GPS sensor. The location/area information is stored in the CloudLocation block using the following format: cloud domain name, latitude and longitude. The domainLoc block is used to store the list of the concerned clouds. Essentially, by using the aforementioned Distance calculation block, the distance to each concerned cloud is calculated based on the current GPS location. Hence the nearest cloud or current cloud can be determined. Finally, the Transfer block is invoked using the domain name of the current cloud.

The Transfer block is for transferring frequently used files (i.e., as defined by the user) from other clouds to the current cloud. The frequently used file information is stored in the CloudFileList block with the domain name of a cloud and the corresponding file names. Based on the CloudFileList’s information, the files are transferred

according to the current cloud using the TransferFile function of the Intercloud programming block. Finally, a notification is shown.

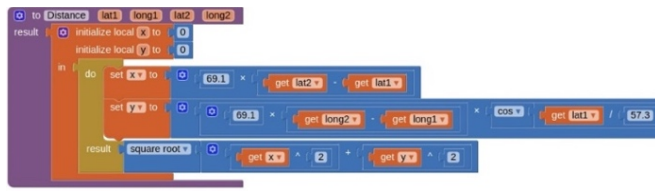


Fig. 4. Distance calculation function

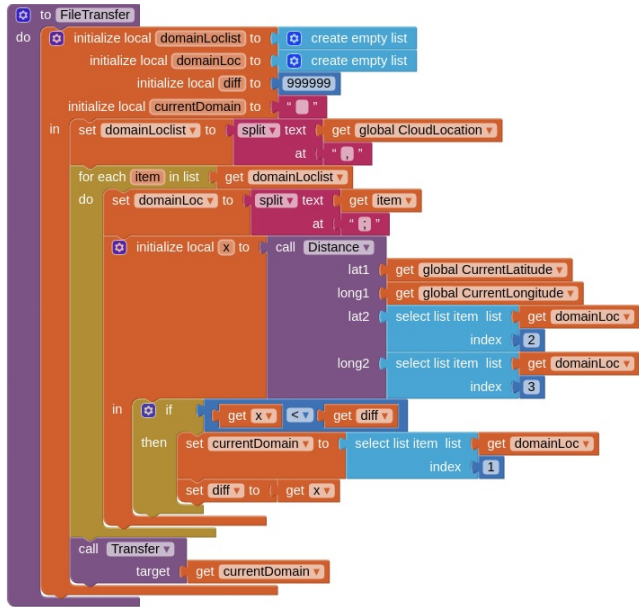


Fig. 5. File transfer function

To evaluate the performance of the file transfer application, we conducted experiments to transfer files with different sizes: 1MB, 10MB, 25MB, 60MB, 100MB, and 150 MB. The transfer time comprises three main components: the download time from the origin cloud, the transfer time from the origin cloud to the destination cloud and the upload time to the destination cloud. The origin cloud was Amazon S3 and the corresponding gateway was installed on a server located in the U.S. on the Google Cloud Platform, while the destination cloud was MinIO (i.e., private cloud), installed with a gateway at a data center in Hong Kong.

The following figures (Figs. 7-9) show the experimental results (download time, transfer time, upload time and total time). It can be seen that the transfer time accounts for the largest percentage of the total time. For instance, for the 1 MB file, 61% of the total time was used for transferring the file. When the file sizes are 10 MB and 100 MB, the percentages become 73% and 78%, respectively. From the results, it can also be seen that when the file size is increased from 1 MB to 10 MB (i.e., 10 times), the total time is increased by only 2.4 times. Furthermore, when the file size is increased from 10 MB to 100 MB (i.e., 10 times), the total time is increased by 3.9 times.

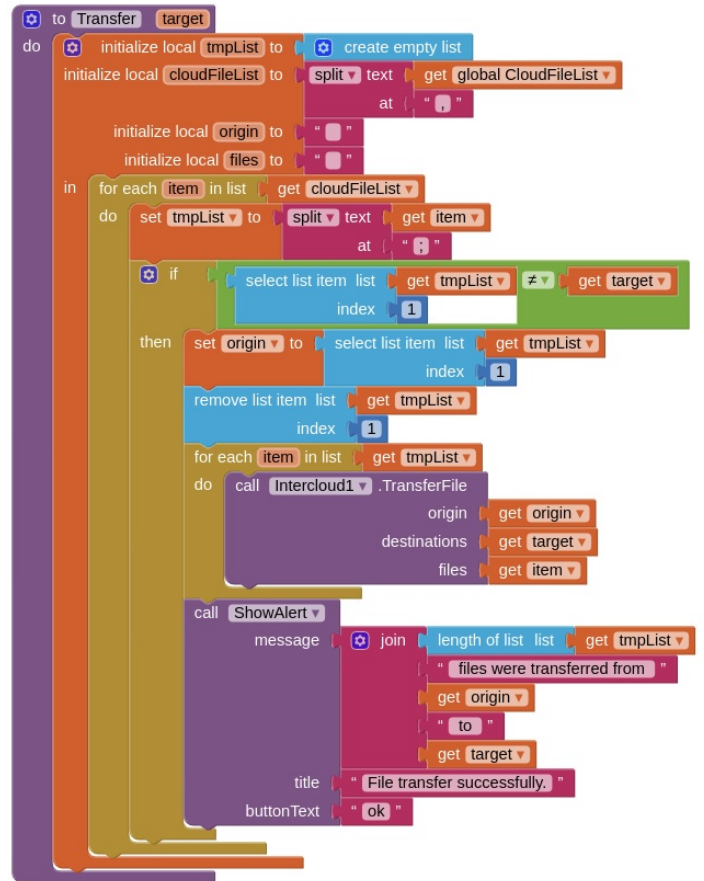


Fig. 6. Transfer function

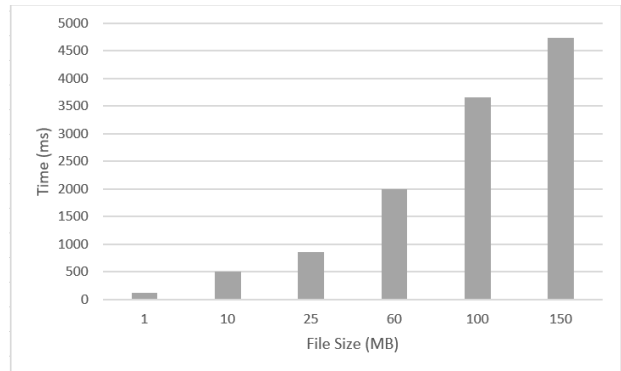


Fig. 7. Upload time

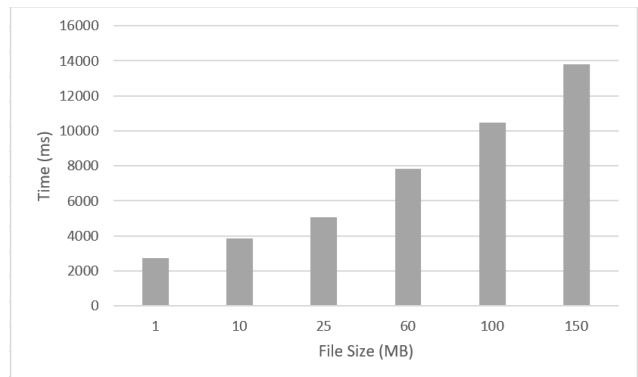


Fig. 8. Download time

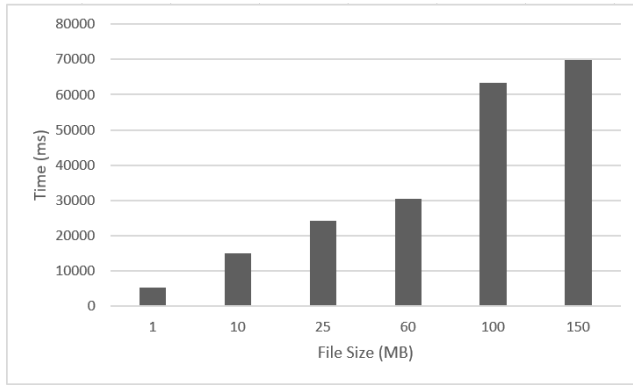


Fig. 9. Transfer time

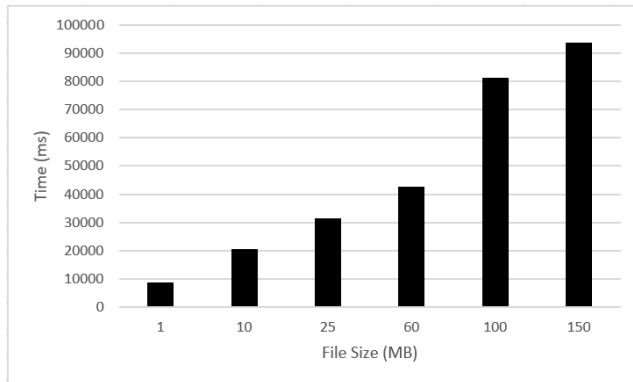


Fig. 10. Total time

VI. CONCLUSION AND FUTURE WORK

In conclusion, with the aim of providing smart computing applications, we have developed a package of App Inventor programming blocks to facilitate the development of mobile Intercloud applications for various data management purposes. We have presented three innovative application scenarios that can be developed by means of the package of App Inventor blocks. We have conducted an evaluation for the automatic file transfer application using the package, including performance analysis. The results provide useful insights for the development of smart computing applications in general and mobile Intercloud applications in particular. The package also provides a general framework for further research and development work (e.g., further blocks can be developed in conjunction with ICCP).

ACKNOWLEDGMENT

This work is supported by The Hong Kong Polytechnic University under account G-YBAJ. We would like to thank Alex Ho for developing the security-related blocks.

REFERENCES

- [1] S. Li and J. Gao, "Moving from Mobile Databases to Mobile Cloud Data Services," in *Proc. of the 2015 3rd IEEE Int. Conf. on Mobile Cloud Computing, Services, and Engineering*, 30 Mar - 3 Apr 2015, San Francisco, CA, USA.
- [2] L. Lei, S. Sengupta, T. Pattanaik, J. Gao, "MCloudDB: A Mobile Cloud Database Service Framework," in *Proc. of the 2015 3rd IEEE Int. Conf. on Mobile Cloud Computing, Services, and Engineering*, 30 March-3 April 2015, San Francisco, CA, USA.
- [3] C. Jaiswal and V. Kumar, "IGOD – Identifying Geolocation of Cloud Datacenter Hosting Mobile User's Data," in *Proc. of the 2015 16th IEEE Int. Conf. on Mobile Data Management*, 15-18 Jun 2015, Pittsburgh, PA, USA.
- [4] A. Kertesz, "Interoperating Cloud Services for Enhanced Data Management," in *Proc. of the 2014 IEEE 4th Int. Conf. on Big Data and Cloud Computing*, 3-5 Dec 2014, Sydney, NSW, Australia.
- [5] L. Han, H. Huang, C. Xie, "Multi-path Data Prefetching in Mobile Cloud Storage," in *Proc. of the 2014 Int. Conf. on Cloud Computing and Big Data*, 12-14 Nov. 2014, Wuhan, China.
- [6] R. D. Pietro, M. Scarpa, M. Giacobbe, F. Oriti, "WiP: ARIANNA: A Mobile Secure Storage Approach in Multi-cloud Environment," in *Proc. of the 2018 IEEE Int. Conf. on Smart Computing (SMARTCOMP)*, 18-20 Jun 2018, Taormina, Italy.
- [7] L. Krithikashree ; S. Manisha ; M Sujithra, "Audit Cloud: Ensuring Data Integrity for Mobile Devices in Cloud Storage," in *Proc. of the 2018 9th Int. Conf. on Computing, Communication and Networking Technologies (ICCCNT)*, 10-12 July 2018, Bangalore, India.
- [8] J. Zhang, Z. Zhang, H. Guo, "Towards Secure Data Distribution Systems in Mobile Cloud Computing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 11, pp. 3222 – 35, 2017.
- [9] C. Chen, M. Won, R. Stoleru, G. G. Xie, "Energy-Efficient Fault-Tolerant Data Storage and Processing in Mobile Cloud," *IEEE Transactions on Cloud Computing*, vol. 3, no. 1, pp. 28 – 41, 2015.
- [10] C. T. Yu, H. C. B. Chan and D. W. K. Kwong, "Discovering resources in an intercloud environment, in *Proc. of the 2017 IEEE Global Communications Conference (GLOBECOM)*, 4-8 December 2017, Singapore.
- [11] Intercloud website - <http://iccp.cf/wordpress/>.
- [12] A. Jimenez-Meza, J. Arámburo-Lizárraga and E. de la Fuente, "Framework for Estimating Travel Time, Distance, Speed, and Street Segment Level Of Service (LOS), based on GPS Data", *Procedia Technology*, vol. 7, pp. 61-70, 2013.