Cooperative Computing for Mobile Platforms

Jing Chen[*], Jian-Hong Liu, Tin-Yen Lin[#],

Institute of Computer and Communication Engineering
Department of Electrical Engineering, National Cheng Kung University
1 University Road, Tainan City 70101
Taiwan, R.O.C.
e-mail: jchen@mail.ncku.edu.tw[*], {liuken, q36001169[#]}@rtpc06.ee.ncku.edu.tw

***Abstract*— While mobile platforms with increasing computing power nowadays have become popular, applications running on mobile platforms, however, still suffer from the limitations of resource availability and architecture variety imposed by mobile platforms. Offloading mobile application to a virtual machine deployed on a server or cloud computing environment is effective only for the cases of running stand-alone applications and is not able to achieve cooperative computing across platform boundary. In attempting to address the issue, this paper considers cross-platform Inter-Process Communication (IPC) to be an essential capability towards achieving cooperative computing at application level and, as a demonstrative example, expands the IPC mechanism of Android system to be the foundation of building a collaborative and cooperative working environment. This expanded IPC mechanism is called XBinder. The main contribution of this work is providing a way for mobile applications to cooperate with local or remote services without developing complicate network transmission mechanism. Mobile applications are able to effectively and efficiently communicate with services which execute either on local node or remote node.**

Keywords- Cooperative Computing; Mobile Computing; IPC; Resource sharing; Remote Service.

I. INTRODUCTION

With advances in hardware and software technologies, consumer embedded system products, in particular mobile platforms, are everywhere around our daily life. It has been envisioned that the next generation of computing systems would be mobile while embedded, in a virtually unbounded number, and dynamically connected [6]. This is getting true especially for mobile platforms such as smart phones and pad platforms which not only have become very popular embedded system products but also serve as personal mobile multifunctional platforms. It is also observed that desktop applications are being moved to run on mobile platforms. For example, accessing Internet services by using browser software or apps running on mobile platforms is now very common. However, a long existent issue is that most mobile platforms impose limitation on available resources such as computing capability, memory, storage, power supply (due to battery life), etc. In general, when application on mobile platforms is running in stand-alone manner, there would be some restrictions limiting its benefit or advantage.

Mobile applications which are executed on cloud server can overcome the limitations mentioned above [9]. Building a virtual mobile platform in cloud computing environment is another solution for resource limitation [12]. When a mobile application is offloaded to run on a server deployed on cloud side, the server pushes the screen display of execution results directly to the user side [18][21]. The disadvantage is that in general a lot of image data transmission is required. Another effective approach is developing, in a case-by-case manner, mobile application with specific constructs [8]. However, it usually needs to establish some protocols between server and mobile applications for the purposes of exchanging data or cooperating. This approach appears comparatively not only quite complicated but also generally error-prone. In general, resources can be shared and processes can be cooperative. When the capability of cooperative computing, which is one type of distributed computing model and in which resources are shared among processes running on different connected platforms (called nodes), is taken into consideration, the approaches mentioned above do not fit.

In this paper we consider that, in supporting cooperative computing at application level, cross-platform Inter-Process Communication (IPC) is essential, and present XBinder as an example of cross-platform IPC mechanism in attempting to address the issue of cooperative computing on networked mobile platforms. Fig. 1 illustrates an application scenario of XBinder, in which a number of services are shared among applications running on different networked platforms.

XBinder expands the IPC mechanism of Android system to help set up the foundation of building a collaborative and cooperative working environment. Its development shows the following desirable features:

- Remote process communication: A local process can communicate and cooperate with a remote process.
- Easy application development with remote objects: Remote services can be built without developing new complicate network transmission mechanism.
- Peer-to-peer communication: Every mobile platform with XBinder installed is a peer node. A peer node is able to directly transmit network packet to other ones, which may provide and apply remote service.
- Multiple concurrent connections: XBinder supports concurrent operations for multiple connected nodes. The IPC of a connection will not be interfered when there are other connections working concurrently.

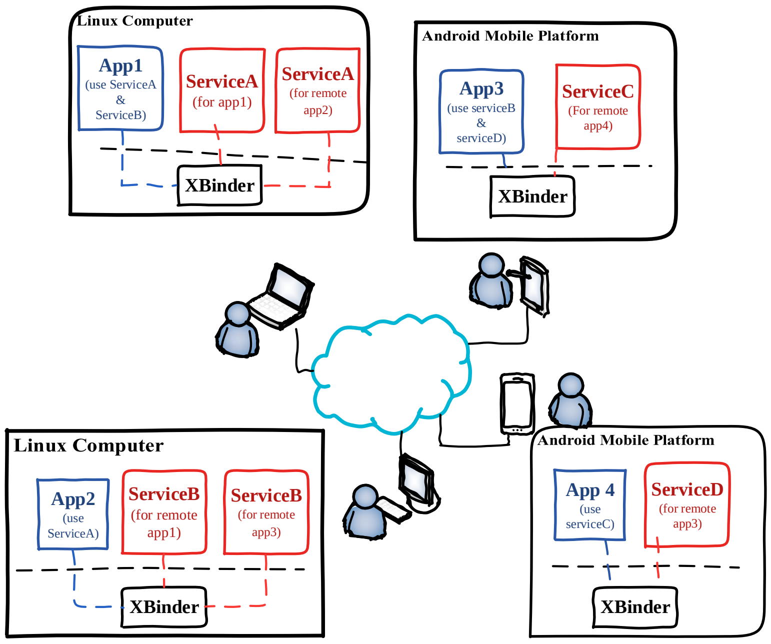

Figure 1. Cooperative computing with XBinder

The rest of this paper is organized as follows. Section II gives a brief discussion on related works. Section III presents the development of XBinder which serves as an example for demonstrating the fundamental capability to help achieve co-operative computing. Section IV describes the evaluation of XBinder. Finally, Section V concludes this paper.

II. RELATED WORKS

This section briefly discusses some works related to solving the issues of resource limitation and cooperative computing on embedded systems or mobile platforms.

Android as a Server Platform (AASP) [13] proposed to deploy Android applications on servers which work in cloud computing environment. The approach effectively addressed the issues, as described above, of resource constraint, power consumption and battery life through achieving a server of Android system. However, AASP did not take into account the operation requirements of data sharing, nor cooperation in either client-server or distributed computing styles.

Reference [17] presented Distributed IPC using Virtual Device Driver in Monolithic Kernel (DIPC) to realize an IPC mechanism for distributed computing systems and achieved communication among processes in different systems. DIPC was implemented in kernel space and showed advantages of high priority and reducing extra copying or movement in data sharing. The disadvantage, nevertheless, comes from its operating in kernel space because it can use only kernel level function library. In addition, security becomes another issue.

Borcea [6] and Iftode [11] both presented a distributed computing model for programming large networks which are composed of embedded systems and implemented prototypes for two applications on sensor networks. Their distributed computing model and the proposed architecture showed the main features of cooperative computing. In their works, co-operative computing applications are composed of migratory execution units (including both code and data), called Smart Messages, working together to accomplish a distributed task. Their model showed generality and worked based on the IPC through message passing. The work described in this paper is mainly motivated by their contributions.

Android system is developed based on Linux operating system [20]. It has its own IPC mechanism, called Android Binder, which demonstrates many desirable features such as stability, efficiency, security, and good resource management [3][19]. However, Android Binder adopts an object-oriented communication approach which put burdens on application developers by requiring case-by-case defined details. Never-theless, this is considered quite suitable for remote process communication [10] and Android has shown its potential of being popular in the world of mobile platforms. Therefore, it is adopted in this work.

III. THE DEVELOPMENT OF XBINDER

The development of XBinder adopts the IPC mechanism of Android system as its base. The reasons are mainly that many software components of Android system, including the Binder driver and its associated software components, are open source software and freely available, and that Android has a large, and still growing, share in the arena of mobile platforms. XBinder has as its components XBinder Manager and XBinder Driver. Its architecture is depicted in Fig. 2 in which the dotted squares indicate the components which are modified from Android Binder framework [7], and the other parts are the components that are developed in this work.

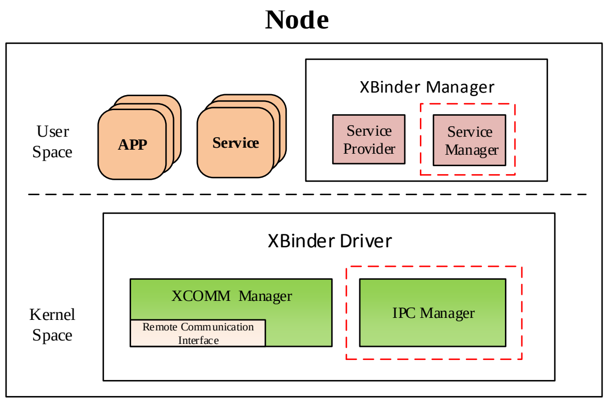

Figure 2. XBinder Architecture

A. *XBinder Driver*

- The IPC Manager manages all the communication between processes, such as applications and services, including allocating message space for processes, and helps transfer messages. Its functionality is achieved by modifying the mechanisms in Android for message delivery, process management, and space allocation.
- XCOMM Manager is responsible for setting up node-to-node connections and maintaining the connections. In addition to delivering services for those active connections, it handles the messages received from remote nodes.
- The Remote Communication Interface, which is part of XCOMM manager, allows the XCOMM Manager to establish connections with XBinder of other nodes and exchange messages through the connections.

XCOMM Manager is the core component in handling remote services. Its architecture is shown in Fig. 3. For each established connection, XCOMM Manager creates a Remote Request Handler and a corresponding Remote Node Object. There are two types of established connections: connecting the local node as requested by a remote node, and connecting a remote node requested by the local node. Remote Request Handler receives through Remote Communication Interface a message and processes that message. The Remote Request Handler puts the processed message into the corresponding Remote Node Object to be dispatched, by IPC Manager, to the destination process.

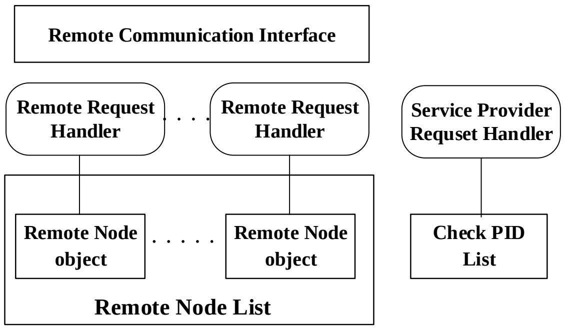

Figure 3. The Architecture of XCOMM

B. *XBinder Manager*

- The Service Manager manages all the services that are either executing on local node or provided by remote nodes. Its functions include acquiring, adding, and deleting services in order to serve processes.
- The Service Provider is responsible for servicing user demands by setting up and closing connection with remote nodes, allocating and releasing services.

Service Provider works internally with XCOMM Manger and IPC Manager in XBinder to achieve providing remote services. The main steps are depicted in Fig. 4. When user issues a request to access a remote service, Service Provider passes the connect command with the IP address of the remote node to local XCOMM Manager which attempts to establish the connection and returns the status of connection to Service Provider. At the time when the connection from a remote note is accepted, XCOMM Manager informs Service Provider in order to start the requested service by creating all needed processes to run the service and passes the identifiers of all these processes to XCOMM Manager for recording purpose. The IPC Manager is designed such that, for a launched service, there is no difference in serving requests from local node or remote nodes. An advantage of this is that existing services need not be modified in order to fit the operations with XBinder. When a connection of utilizing a service is to be closed, the local XCOMM Manager informs Service Provider to terminate all the processes associated with that service.

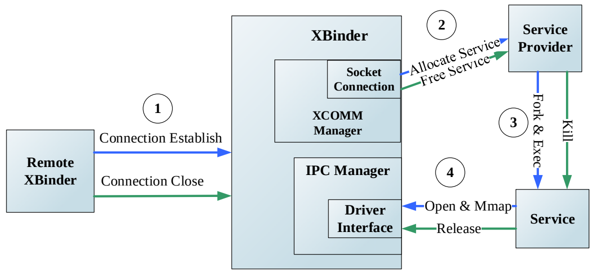

Figure 4. The steps in providing remote services

XBinder is implemented using C programming language with GNU library functions so that not only the required network operations can be realized using sockets without difficulty but also the implementation can be installed to work in both Linux and Android systems. The modification done to the components in Android is accomplished by using Android software development kit [1][2][4][5].

IV. Evaluation

The evaluation of XBinder, including its functionality and performance, was conducted for the purpose of serving the proof of concept. A working environment of four nodes was set up, as shown in Fig. 5, in which two Wii sticks were connected to a desktop computer and serve as the controllers of user input devices [15]. In order to reduce the interference from possible yet unpredictable variation in the quality of network communication, wired Ethernet was adopted. The configuration of this environment is listed in Table I.

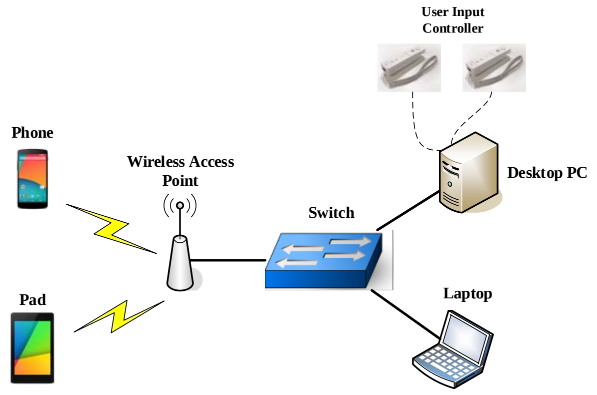

Figure 5. The environment for evaluating Xbinder

TABLE I. THE CONFIGURATION OF THE EVALUATION ENVIRONMENT

Node	Desktop PC	Laptop PC	Nexus 7	Nexus 5
CPU	Intel Core 2 Quad (2.5 GHz)	Intel Core i5-540M (2.5 GHz)	Nvidia Tegra 3 (1.3 GHz)	Qualcomm Snapdragon (1.5GHz)
Memory	4 GB	4 GB	1 GB	2 GB
O/S	Ubuntu 12.04	Ubuntu 11.10	Android 4.4	Android 4.4
Network	Wired	Wired	Wi-Fi	Wi-Fi
Devices	Wii stick	N/A	▪ Accelerator ▪ GPS	▪ Accelerator ▪ Vibrator ▪ GPS

In testing the functionality of XBinder, Fig. 6 shows the case in which multiple nodes were concurrently using remote services and Fig. 7 demonstrates the operations in the test. In Fig. 7, while the smart phone (Nexus 5) and the Smart Pad (Nexus7) were accessing concurrently the sensor service provided by the right Wii stick which was connected to the desktop PC [15], the Smart Pad is providing a remote service to the laptop computer [16]. The test conducted in this case verified the concurrent operations of multiple nodes (users), the IPC with Linux and Android systems, and the connection functions in peer-to-peer style.

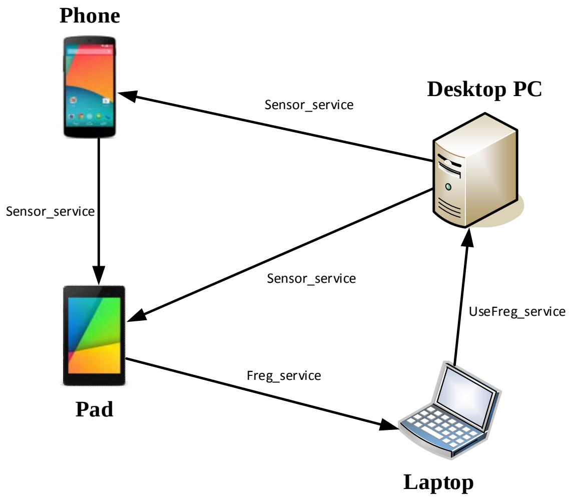

Figure 6. Multiple nodes concurrently access remote service

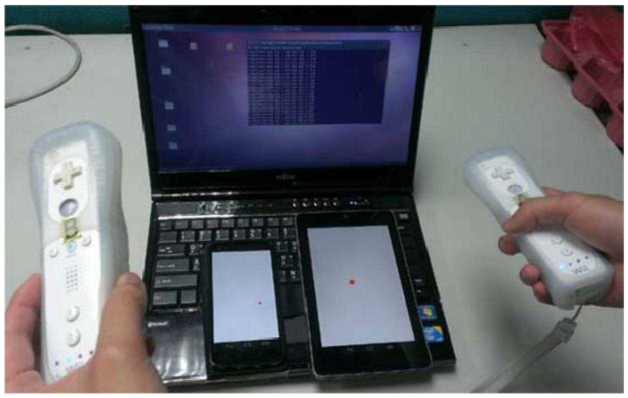

Figure 7. Applications concurrently access a remote service

The performance evaluation of this implementation was done by comparing the time of transmitting data blocks. Java RMI [14] was selected as the metric of comparison for the cases that involve remote communication, while the original Android Binder was the metric for the cases of local service. The transmission time measured starts from a data request is issued to the requested data is received, and for each selected data block size, 100 transmissions were separately measured. To avoid interference in measuring the transmission time, no file operations were involved. Table II lists the average time of transmitting data in different sizes, while Fig. 8 depicts the comparison of XBinder versus Java RMI. It can be observed that XBinder maintains similar performance compared with Android Binder and performs better than Java RMI in most test cases. When the data size grows larger, Java RMI shows better performance than XBinder. The reason is that XBinder needs two more copy operations during its operation in order to copy the data to the assigned address space.

TABLE II. COMPARING XBINDER WITH JAVA RMI

Data Size	Remote Communication		Local Communication	
	XBinder	*Java RMI*	*XBinder*	*Android Binder*
16 B	255	797	20	18
1 KB	555	995	21	19
64 KB	6095	6250	146	144
256 KB	23404	23132	531	516
1 MB	92788	90388	3669	3663
2 MB	185066	179943	N/A	N/A

Time Unit: micro-second (μs)

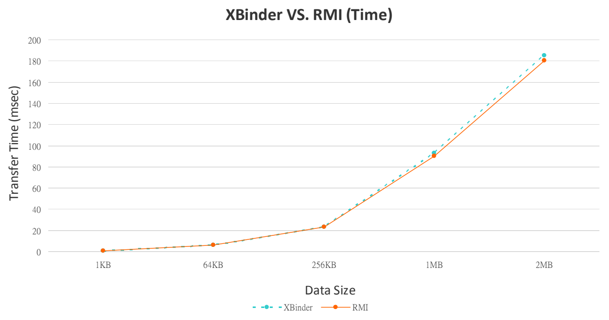

Figure 8. Performance comparison of XBinder with Java RMI

V. CONCLUSION AND FUTURE WORK

Mobile platforms usually run applications with limitation on available resources. While sharing can be one solution for the issue of limited resources, cooperative computing across platform boundary is very beneficial for mobile platforms. In this paper, we consider cross-platform IPC to be the essential capability in achieving cooperative computing at application level and, as one demonstrative example, present XBinder to provide Android-based mobile platforms the functionality of IPC over network connections.

The design and implementation of XBinder are based on Android Binder which is the default IPC mechanism for processes running in Android-based platforms. XBinder extends the Binder Driver of Android to support network communication mechanism in order to form a multi-node working environment in which each node is an Android-based mobile platform. In addition, each node with XBinder installed can simultaneously function assuming both the roles of "client" and "server". The results of evaluating its implementation showed that a process can communicate with other process running on remote node over network connection effectively and efficiently.

An additional benefit of XBinder is that XBinder can be installed easily into any Linux-based platform to support cross-system IPC between Linux and Android systems. This is because Linux is the basis of Android and XBinder is built in the kernel space of Linux. Therefore, XBinder can run in Linux or Android system to provide a high-level abstraction of IPC mechanism and help programmers develop remote cooperative applications and services with ease.

ACKNOWLEDGMENT

This work is partially sponsored by Ministry of Science and Technology (MOST) of Republic of China (ROC) under the contract MOST105-3011-E-006-001.

REFERENCES

[1] Android, "Android Interface Definition Language (AIDL)", https://developer.android.com/guide/components/aidl.html. [retrieved: July, 2016]

[2] Android Developers, https://developer.android.com, accessed on 2016-07-22.

[3] A. Gargenta, "Deep Dive into Android IPC/Binder Framework", Android Builders Summit, 2013. Available from: https://events.linuxfoundation.org/images/stories/slides/abs2013_gargentas.pdf.

[4] Android Studio, "Control the Emulator from Command Line", https://developer.android.com/studio/run/emulator-commandline.html, accessed on 2016-07-20.

[5] Android Studio, "Android Studio: The Official IDE for Android", https://developer.android.com/studio/index.html. [retrieved: July, 2016]

[6] C. Borcea, D. Iyer, P. Kang, A. Saxena, and L. Iftode, "Cooperative Computing for Distributed Embedded Systems", Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS 2002), July 2002, pp. 227-236, doi: 10.1109/ICDCS.2002.1022260.

[7] D. K. Hackborn, "OpenBinder Documentation Version 1.0", http://www.angryredplanet.com/~hackbod/openbinder/docs/html/index.html. [retrieved: August, 2016]

[8] E. Kim, K. Yun, and J. Choi, "RSP: A Remote OSGi Service Sharing Scheme", Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing in conjunction with the UIC'09 and ATC'09 conferences, July 2009, pp. 318-323, E-ISBN: 978-0-7695-3737-5, Print ISBN: 978-1-4244-4902-6 doi: 10.1109/UIC-ATC.2009.79.

[9] K. Kumar and Y. Lu, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?", IEEE Computer, Vol. 43, Issue 4, pp. 51-56, Apr. 2010, doi: 10.1109/MC.2010.98.

[10] K. Nakao and Y. Nakamoto, "Toward Remote Service Invocation in Android", Proc. of the 2012 9th International Conference on Ubiquitous Intelligence and Computing and the 9th International Conference on Autonomic and Trusted Computing (UIC-ATC'12), Sept. 2012, pp. 612-617, ISBN: 978-1-4673-3084-8, doi: 10.1109/UIC-ATC.2012.22.

[11] L. Iftode, C. Borcea, and P. Kang, "Cooperative Computing in Sensor Networks", in Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems, M. Ilyas and I. Mahgoub, Eds. Boca Raton: CRC Press, pp. 26-1--26-19, 2005, ISBN: 0849319684.

[12] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing", IEEE Pervasive Computing, Vol. 8, Issue 4, pp. 14-23, Oct. 2009, doi: 10.1109/MPRV.2009.82.

[13] M. Toyama, S. Kurumatani, J. Heo, K. Terada, and E. Y. Chen, "Android as a Server Platform", the 8th Annual IEEE Consumer Communications and Network Conference, Jan. 2011, pp. 1181-1185, ISBN: 978-1-4244-8789-9.

[14] Oracle Inc., "An Overview of RMI Applications", https://docs.oracle.com/javase/tutorial/rmi/overview.html. [retrieved: August, 2016]

[15] P. L. Wang, "The Design and Implementation of a Unified Hardware Abstraction Layer for Lunix Operating System", Master Thesis, National Cheng Kung University, Taiwan, ROC, 2014.

[16] R. Stones and N. Matthew, "Beginning Linux Programming", Wrox – Wiley Publishing Inc., 4th edition, 2007, ISBN: 0470147628.

[17] S. Bagchi, "Distributed IPC using Virtual Device Driver in Monolithic Kernel", The 18th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2012), Aug. 2012, pp. 51-57, ISSN: 2325-1271, E-ISBN: 978-0-7695-4824-1, Print ISBN: 978-1-4673-3017-6.

[18] S. Ghorpade, N. Chavan, A. Gokhale, and D. Sapkal, "A Framework for Executing Android Applications on the Cloud", The 2nd International Conference on Advances in Computing, Communication and Informatics (ICACCI 2013), Aug. 2013, pp. 230-235, ISBN: 978-1-4799-2432-5.

[19] T. Schreiber, "Android Binder: Android Interprocess Communication", Seminar-thesis, Buhr University, 2011. [Online]. Available from: https://www.nds.rub.de/media/attachments/files/2012/03/binder.pdf.

[20] Wikipedia, "Android (operating system)", https://en.wikipedia.org/wiki/Android_(operating_system). [retrieved: July, 2016]

[21] Wikipedia, "Thin_client", http://en.wikipedia.org/wiki/Thin_client. [retrieved: July, 2016]