International Journal of Hybrid Information Technology
Vol. 9, No.6 (2016), pp. 337-346
http://dx.doi.org/10.14257/ijhit.2016.9.6.30

Cloud-based Energy-Efficiency XMPP Protocol for Mobile Network

Ge Zhihui*, Cen Xiao, Wang Hailiang and Li Taoshen

School of Computer, Electronics and Information, Guangxi University, Guangxi Nanning, China
**corresponding author: gezhihui @foxmail.com*

Abstract

Using the Cloud-based Offloading heartbeat aggregation method can solve the smart mobile terminal bearing the burden of persistent connections. It effectively reduces the heartbeat storm. But with the popularity of MTC (Many-Task Computing), especially in the environment of MMTC (Mobile MTC), using XMPP to communicate cause high energy consumption problem at the same time. The traditional XMPP protocol format already cannot adapt to the high performance transmission under the new environment. This paper is based on the MMTC environment, proposed the new communication method of mobile terminal and cloud with XMPP protocol to achieve more energy efficiency.

***Keywords**: Cloud, XMPP, energy-efficiency, MMTC, migration aggregation*

1. Introduction

Advances in information and communications technology have increased the popularity of mobile devices. Starting with the popularization of the smartphone, people are connected to the Internet almost all of the time now. This ubiquitous connectivity makes our lives ever more convenient. The exponentially growing use of smart devices has facilitated the development of ubiquitous computing infrastructures for service provisioning[1]. Mobile devices liked to access various services from ubiquitous resources. The increasing demand of the smartphones for processing power, storage space and energy saving, and the continuous improvements of the telecommunication infrastructures for the provisioning of high performance services is leading the rapid adaption of the mobile cloud computing (MCC) domain[2].

The Extensible Messaging and Presence Protocol (XMPP) [3] is an open standard for XML-based communications in near real time. Small chunks of XML, called stanzas, are sent by clients to servers to be routed to other clients. When and how the routing occurs forms the basis for much of XMPP's functionality. XMPP provides a large range of services such as channel encryption, authentication, presence monitoring, unicast and multicast messaging, service discovery, capabilities advertisement, and federation.

Many research projects explore the rich possibilities of application level protocols like XMPP out of the traditional domain of instant messaging applications [4]. Kestrel [5] is a distributed computing framework for Many Task Computing (MTC) applications, based on XMPP. Later works extend this idea to XMPP-based service platforms and cloud computing. The i5Cloud [6] uses XMPP for inter-service communication in the cloud as well as communication between devices and the cloud service. Mobilis [7] defines a service platform where mobile clients may access XMPP-based services running as XMPP clients. These projects exploit the benefits of the XMPP protocol for collaboration and peer-to-peer XML messaging, they are not focused on energy efficiency and cloud services.

XMPP protocol does not define how to establish aggregation persistent connection. Using the XMPP connection mode, independent tasks directly establish persistent connection to the server. All tasks will build their own session connection when they need to use XMPP protocol. This is very energy consuming. XMPP protocol uses XML format for message

ISSN: 1738-9968 IJHIT

causes stick pack problem [8] more serious and result in performance degradation. the authors propose LLaF-E^2CXEP (Low Load and Fast Energy- Efficiency Cloud-based XMPP Extend Protocol) to solve the new challenge in mobile cloud-based offloading aggregation environment, increase aggregation efficiency and reduce smart mobile terminal energy consumption.

2.1. Aggregation Connection Establishment

XMPP protocol communication mode is point to point, which means that XMPP persistent connection is not shared with each other. When tasks' persistent connection need to be aggregated, they need to cooperate with each other to complete the connection establishment. There are two problems need to be solved. The first is that who and when to establish the aggregation persistent connection. The second is how tasks discover the aggregation persistent connection.

The aggregation persistent connection establishment needs a daemon. There must has initiator, because of tasks cannot do it alone. Daemon starts to run when smart mobile terminal is start. So the authors design a daemon called migaggd (migration aggregation daemon) to manage the whole life cycle of all persistent connections. When a new task T needs to use persistent connection starts, the establish process is as following:

Begin

Step 1: T broadcasts the message of finding migaggd.

Step 2: Migaggd daemon will establish a communication channel using IPC when catching the broadcast message from T.

Step 3: Migaggd checks the aggregation persistent connection, if the connection is not exist, it will establish one.

End

Modern smart mobile operating system such as Android, IOS, Windows Phone all support daemon running. In other platforms, migaggd can communication with tasks through IPC, just need to write a program for that platform. The experimental environment in this paper, the authors use Android operating system.

LLaF-E^2CXEP is compatible with the original XMPP protocol, using three phase handshake to ensure that the client and server supporting the new protocol format. In the first phase, client sends stream stanza to start, and a new xep:llaf-e^2cxep attribute is added in stream stanza. In the second phase, server will send back the message with xep-accept:llaf-e^2cxep attribute. In the third phase, when client received the attribute of xep-accept, it will start communication with the new format of LLaF-E^2CXEP.

2.2. Many-Task Port Routing

When all message packets of task are aggregated at the same communication channel, they need to be recognized. To ensure each task can get correct messages, message packets have to be routed correctly when they arrive at cloud and client. In XMPP protocol, when server keeps multiple connections, attribute id is used to identify messages. The id attribute also works in aggregation persistent connection, but in inefficiency way. In original method, the receiver receives the whole message at first, and extracts the attribute id. Then the message packet is routed to that corresponding task which id attribute refers to. The id attribute is a unique hash string. The drawback of using id attribute is that it will increase message packet length. And the id attribute is not required in message stanza and presence stanza.

To improve routing speed and reduce the length of message packet, This paper proposed Many-Task port routing MTPR method. Ports and tasks form a corresponding relationship, and a message start with port value. It no needs to parse the whole message, just read the head of message and routing the message by task's port value. In this way, it can improve the routing speed and CPU is used more efficiency.

The client needs to maintain a port mapping task table PMTT. Task port TP is integer. PMTT records task's information and TP. Client updates PMTT and cloud synchronization. When the first time to establish connection, client sends PMTT hash value. Cloud receives the PMTT hash value and compare with its own. If they are equal it means they don't need to modify. If they are not equal, cloud compresses the PMTT and sends to client. Client finds the difference between cloud's PMTT and its own, and sends back the different parts.

In order to improve routing speed and reduce the length of message packet, the authors proposed Many-Task Port Routing(MTPR) method. Ports and tasks form a corresponding relationship, and a message starts with port value. It needs not to parse the whole message, just read the head of message and routing the message by task's port value. In this way, the routing speed and CPU utilization is more efficiency.

The client needs to maintain a port mapping task table PMTT. PMTT records information of tasks and Task port (TP). When the first time to establish a connection, client sends out PMTT hash value. When cloud receives PMTT hash value, it will compare with its own. If they are equal it means they don't need to modify. If they are not equal, cloud compresses PMTT and sends it to client. Client finds the difference between cloud's PMTT and its own, and sends back the different parts.

2.3. Message Serialization

$$Pkt = \left\lceil \frac{L_{msg} + L_{last}}{L_{mss}} \right\rceil \tag{1}$$

Pkt represents the number of data packets which is splitted by TCP protocol. The length of message packets is L_{msg}. TCP protocol's MSS is L_{mss}. the whole data packet divided into *Pkt* packets in the TCP channel. The first packet is occupied by the length of L_{last}.

$$E_{tx} = \left\lceil \frac{\sum_{i=1}^{N} L_{msgi} + L_{last}}{L_{mss}} \right\rceil \cdot \frac{E_{pt}}{P} + N \cdot E_{smsg} \tag{2}$$

E_{tx} represents the energy of sending message for one time. E_{pt} represents the average energy to send each packet. *P* represents the probability of successful sending. E_{smsg} represents the average energy to serialize a message. *N* represents the number of messages. L_{msgi} represents the length of the *i* message.

$$E_{rx} = \left\lceil \frac{\sum_{i=1}^{N} L_{msgi} + L_{last}}{L_{mss}} \right\rceil \cdot E_{pr} + N \cdot E_{dmsg} \tag{3}$$

E_{rx} represents the energy of receiving message for one time. E_{pr} represents the average energy to receive each packet. E_{dmsg} represents the average energy to deserialize a message.

Formula (2) and (3) show that L_{msg}, E_{smsg} and E_{dmsg} have great affection on energy consumption. In different network environment, the other parameters are fixed. Energy efficiency can be achieved by changing the value of L_{msg}, E_{smsg} and E_{dmsg} .

LLaF-E^2CXEP uses length-known method. It informs receiver the length of the whole message. In this way, it can save CPU energy, and increase packets receiving speed.

It defines three message structures TAM, TTM and AUM, as shown in Figure 1. With the three message structures, it can express the whole information of XML, and it can aggregate messages more efficiency than XML, and routing faster. With the message structure, length and task_port using varints [9] encode method to encode integer. The attribute of attr_TaF using prefix-varints encoding method, type attribute is the prefix content. Three message structures nested each other by the order of TAM-TTM-AUM. TAM is the top of message structure. It encapsulates the aggregation messages. It has only one agg_length attribute. The agg_length attribute inform the whole aggregation message's length. The msg_content part include 0 or n TTM message structures. Each task package messages into TTM message structure. The tmsg_length attribute represents the length of task message. The task_port attribute represents task's port in PMTT. The migaggd only parses task_port then routing msg_content to task. The msg_content in TTM include 0 or n AUM message structures. AUM is the based message unit. It records message's attribute name and value, and uses AUM can express what XML can express by nested each other. The amsg_length attribute represents the length of AUM. The attr_TaF attribute represents two attribute, the first is type attribute, and the second is field attribute. There are two type attributes are defined, they are value type and set type. When AUM type is value type, msg_content include the value's bytes. When AUM type is set type, msg_content include 0 or n AUM message structures.

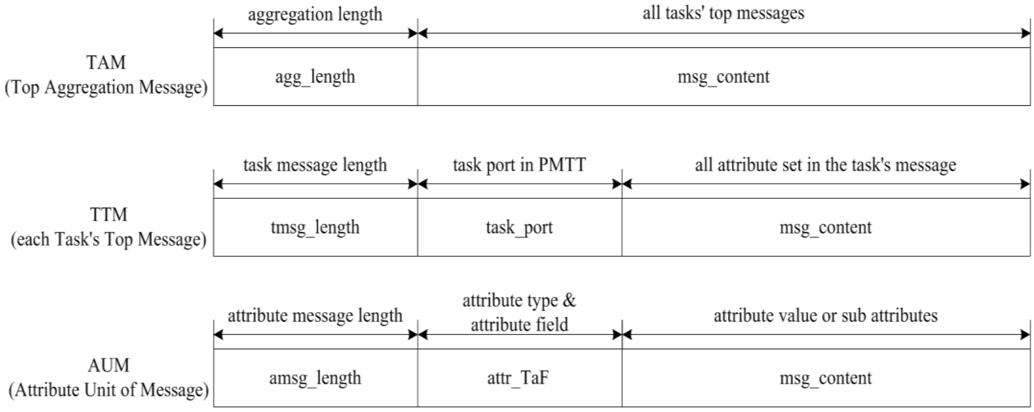


Figure 1. Message Structures Used To Serialization

The process of message serialization:

Begin

Step 1: The mitigated receives a message of TAM. If the data buffer size is as long as the agg_length attribute's value, jump to the step2. If the data buffer size is not match the agg_length attribute's value, it will stop and wait for more data.

Step 2: When TAM was received complete, it starts to parse the msg_content part of TAM. Parsing each TTM's task_port and following the PMTT to routing msg_content of TTM to each task.

Step 3: Parsing AUM structure after the task received the AUM, and building the data object.

End

The process of message deserialization:

Begin

Step 1: The migaggd catch the AUM send from Ti, search TP in PMTT, and build TTM message.

Step 2: Add TTM to TAM which wait for aggregate message and the send time is not arrive. Then update the agg_length attribute.

End

3. Experimental Results

The authors test the performance of LLaF-E^{2}CXEP in real environment. In real environment, the devices used as below: The test phone is ZTE U880, operating system is Android, the core chip is Marvell PXA910, the frequency of CPU is 806MHz, the architecture is Marvell ARMv5 CPU, the RAM is 512M. NS-3[10] be used to verify the energy efficiency.

The experiment compares XML, JSON, Protobuf to LLaF-E^{2}CXEP. The experiment tests the number of aggregation message from 1 to 30, and records L_{msg} which is the length of aggregation message. Message stanza is used. The experimental parameters are shown in Table 1.

Table 1. Experimental Parameters

attribute	value	attribute	value
from	13 byte	xml:lang	2 byte
to	10 byte	body	58 byte
id	8 byte	L_{mss}	1460 byte

As shown in Figure 2. It shows the different length of aggregation message with each encoding method. The longest message length is encoded by XML method. The result of JSON and XML is very close. Obviously, Protobuf reduce the length of the aggregation message, but LLaF-E^{2}CXEP is the smaller one. The expression of the same information, LLaF-E^{2}CXEP's length is only 64% of XML. XML and JSON as the most popular message format, it can be read easy by human. But it imports additional descriptor for human to read easily. LLaF-E^{2}CXEP takes encoding method like Protobuf, but it more suitable for XMPP protocol. Because of less type for LLaF-E^{2}CXEP, it can be smaller than Protobuf. The RFC of XMPP can be watched, that the attributes' value is string or set.

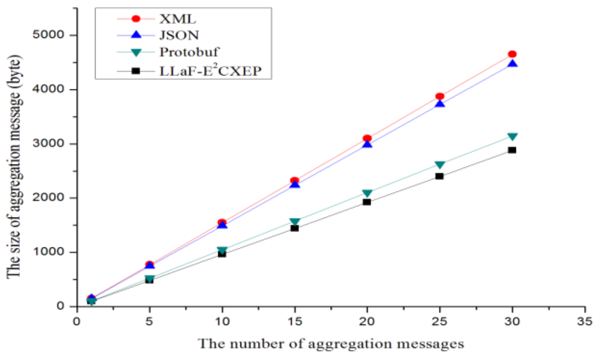


Figure 2. Length of Different Methods Serialize Aggregation Message

Different encoding methods lead to different number of packets. If MSS is 1460, the number of packets different methods serialize aggregation message was shown in Figure 3. The XML and JSON need more packets to send the whole messages when 10 or more messages are aggregated. When the number of aggregation message is 30, XML and JSON methods' packet number is twice of LLaF-E^{2}CXEP. The more packets send in TCP channel, the more resource is spent to solve the problem of TCP stick pack. LLaF-E^{2}CXEP reduces the number of packets. In ideal situation, the number of aggregation message is 30, LLaF-E^{2}CXEP only cost a half of XML or JSON energy consumption.

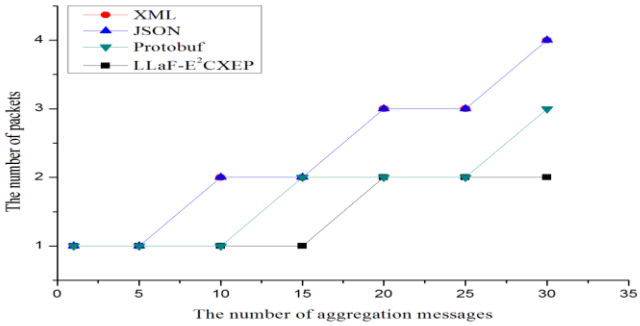


Figure 3. Number of Packets Different Methods Serialize Aggregation Message

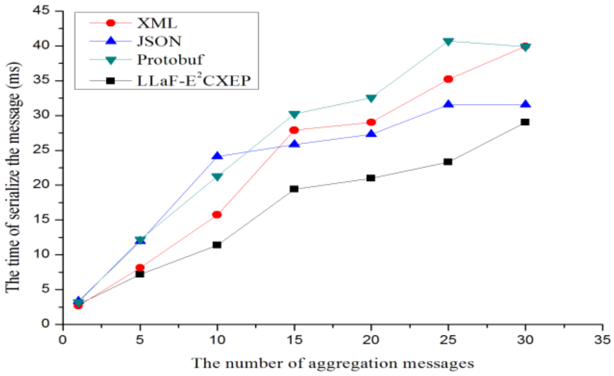


Figure 4. Time of Different Methods Serialize Aggregation Message

As shown in Figure 4. It is the time of different methods serialize aggregation message. As the number of aggregation message growth, LLaF-E^2CXEP is the shortest time. Protobuf takes more time to encoding, because of it was designed to data center. The message stanza structure is too simple, and it cannot show the advantage of Protobuf.

As shown in Figure 5. It is the time of different methods deserialize aggregation message. Obviously, Protocol spent the longest time to decoding the message. After parsed the process of the Protobuf experiment. It is found that Protobuf is not support well for mobile terminal. It cost high memory in Android operating system. When it takes memory over the VM heap, it causes garbage collection GC process. The system will be paused by GC. The LLaF-E^2CXEP uses the shortest time to decoding the aggregation message.

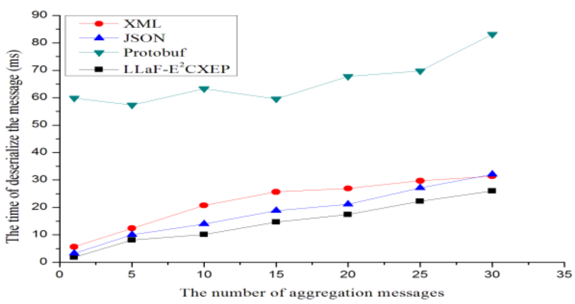


Figure 5. Time of Different Methods Deserialize Aggregation Message

Because of the lack of energy measurement equipment, and Android system's energy measurement tool cannot accurate description the energy consumption. Energy consumption was measured in NS-3 simulation environment. The simulation network is wireless environment with IEEE802.11b protocol. The experimental parameters are shown in Table 2.

Table 2. Experimental Parameters

attribute	value	attribute	value
Rx gain	-10 dBm	Tx current	17.4 mA
Tx gain	-1 dBm	interval	10s
distance	100 meters	total time	3600s

As shown in Figure 6, it is the energy consumption of different methods transmission aggregation message. The LLaF-E^{2}CXEP has good performance in energy consumption. From the figure, it can be observed its energy consumption proportional to L_{msg}. In simulation environment, it only can measure the energy of wireless module, it cannot records the CPU energy consumption.

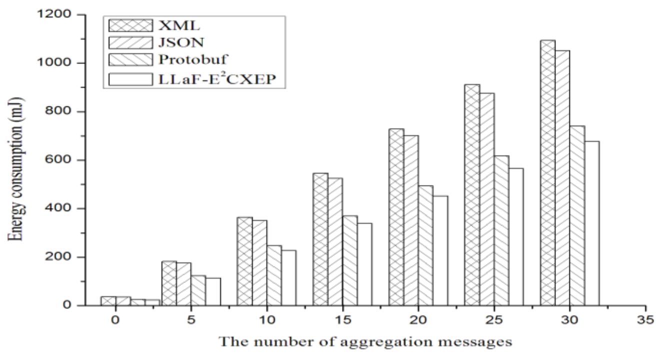


Figure 6. Energy Consumption of Different Methods

4. Conclusion

XMPP as a protocol was designed before the wide spread adoption of mobile devices, and is often cited as not being very mobile friendly as a result. Aim at the problem with high energy consumption and high-bandwidth of XMPP on mobile devices, through studying XMPP core standard, combined with the latest XMPP Extension Protocol, this paper points out the causes of the problems and give the corresponding solutions. There are three main parts included by LLaF-E^{2}CXEP. The first part is build the mechanism of establish and discover aggregation connection. The second part is multi-tasks port routing. The last part is the message serialization method. The experimental results show that LLaF-E^{2}CXEP is more suitable for running in cloud offloading environment and achieves better energy efficiency.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant No 61363067, 60963022).

References

[1] R. C. Wang, Y. C. Chang and R. S. Chang, "Design issues of semantic service discovery for ubiquitous computing", Proceedings of the IEEE International Conference on Multimedia and Ubiquitous Engineering, Seoul, Korea**, (2007)** April 26-28.

[2] H. Flores and S. Srirama, "Mobile cloud messaging supported by XMPP primitives", Proceedings of the 4th ACM workshop on Mobile cloud computing and services, Taipei, Taiwan, **(2013)** June, pp. 17-24.

[3] P. Saint-Andre, "Extensible messaging and presence protocol (XMPP): Core", RFC 3920, **(2011)**.

[4] A. Hornsby and R. Walsh, "From instant messaging to cloud computing, an XMPP review", Proceedings of the IEEE International Symposium on Consumer Electronics (ISCE), Braunschweig, Germany, **(2010)** Jun 7-10, pp. 1-6.

[5] L. Stout, M. A. Murphy and S. Goasguen, "Kestrel: an XMPP-based framework for many task computing applications", Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers, ACM, **(2009)**.

[6] D. Kovachev, Y. Cao and R. Klamma, "Building mobile multimedia services: a hybrid cloud computing approach", Multimedia tools and applications, vol. 70, no. 2, **(2014)**, pp. 977-1005.

[7] D. Schuster, T. Springer and A. Schill, "Service-based development of mobile real-time collaboration applications for social networks", Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), **(2010)**.

[8] W. R. Stevens, "TCP/IP illustrated", addison-Wesley, vol. 1, **(2011)**.

[9] S. V. Chekanov, E. May, K. Strand and P. V. Gemmeren, "ProMC: Input–output data format for HEP applications using varint encoding", Computer Physics Communications, vol. 185, no. 10, **(2014)**, pp. 2629-2635.

[10] T. R. Henderson, M. Lacage and G. F. Riley, "Network simulations with the ns-3 simulator", SIGCOMM demonstration 14, **(2008)**.

[11] C. P. Wibowo, "Evaluation of Protocol Buffers as Data Serialization Format for Microblogging Communication", Proceedings of the International Conference on Informatics for Development, vol. 602, **(2011)**.

[12] M. Bourguiba, K. Haddadou and G. Pujolle, "Packet aggregation based network I/O virtualization for cloud computing", Computer Communications, vol. 35, no. 3, **(2012)**, pp. 309-319.

[13] M. Fazio, A. Celesti, M. Villari and A. Puliafito, "Resource Management in Cloud Federation Using XMPP", Proceedings of the IEEE International Symposium on Network Computing and Applications (NCA), **(2014)**, pp. 67-70.

[14] R. Klauck and K. Michael, "Chatty things-making the internet of things readily usable for the masses with XMPP", IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), **(2012)**.

[15] S. Kosta, C. Vasile, Perta, J. Stefa and P. Hui "Clone2Clone (C2C): Peer-to-Peer Networking of Smartphones on the Cloud *", Usenix Workshop on Hot Topics in Cloud Computing San Jose Ca, **(2013)**.

[16] S. Limam and G. Belalem, "A Migration Approach for Fault Tolerance in Cloud Computing", International Journal of Grid & High Performance Computing, vol. 6, no. 2, **(2014)**, pp. 24-37.

[17] C. H. Hsu and E. Udoh, "Mobile and Wireless Computing towards Sustainable and Ubiquitous Clouds Services", International Journal of Grid & High Performance Computing, vol. 5, no. 3, **(2013)**, pp. 1-5.

[18] R. Klauck, J. Gaebler, M. Kirsche and S. Schoepke, "Mobile XMPP and cloud service collaboration: An alliance for flexible disaster management", Proceedings of the IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), **(2011)**, pp. 201-210.

[19] X. Zhang, A. Kunjithapatham, S. Jeong and S. Gibbs, "Towards an Elastic Application Model for Augmenting the Computing Capabilities of Mobile Devices with Cloud Computing", Mobile Networks & Applications, vol. 16, no. 3, **(2011)**, pp. 270-284.

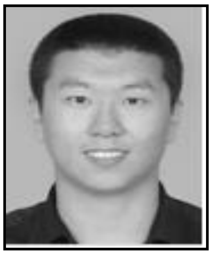
[20] S. Pearson, "Taking account of privacy when designing cloud computing services", Workshop on Software Engineering Challenges of Cloud Computing IEEE Computer Society, **(2009)**, pp.44-52.

[21] S. Belov, L. Dudko, D. Kekelidze and A. Sherstnev, "HepML, an XML-based format for describing simulated data in high energy physics", Computer Physics Communications, vol. 181, no. 10, **(2010)**, pp. 1758-1768.

[22] D. Kovachev, Y. Cao and R. Klamma, "Augmenting Pervasive Environments with an XMPP-Based Mobile Cloud Middleware", Mobile Computing, Applications, and Services, Springer Berlin Heidelberg, **(2010)**, pp. 361-372.

[23] R. Klauck and M. Kirsche, "Combining Mobile XMPP Entities and Cloud Services for Collaborative Post-Disaster Management in Hybrid Network Environments", Mobile Networks & Applications, vol. 18, no. 2, **(2013)**, pp. 253-270.

Authors



GE Zhihui was born in Hebei, China, in 1978. He received a Ph.D. degree in Computer Science from the Central South University, Hunan, China, in 2007. He is a professor in the Guangxi University. Nanning, Guangxi, China. His research interests are in wireless networks and mobile computing.

Cen Xiao was born in Guangxi, China, in 1991. He received a B.S. degree in Computer Science from the Guangxi University, Guangxi, China, in 2014, and is a M.S. student in Computer Science from the Guangxi University, Guangxi, China. His research interests are in wireless networks.

Li Taoshen received the B.S., M.S. and PhD degrees in computer science and technology from the Central South University, Changsha, People's Republic of China, in 1982, 1989 and 2008, respectively. He is now a Professor and the Dean of School of Computer, Electronics and Information in Guangxi University, China. His research interests are in the area of distributed database system, wireless mesh netwxorks, cloud computing and intelligent processing.

