Measurement on a Peer-to-Peer Package Management System for Linux Distributions

Jie Yu, Weihua Zhang, Xiaodong Tang, Shasha Li, Qiang Li, and Qingtao Shen

***Abstract*—Peer-to-Peer(P2P) technique has been successfully used in many applications, especially P2P file sharing and P2P live stream. However, there are very few work that try to employ P2P to improve package management for Linux distributions. In this paper, we first introduce the principle of a P2P package management system and a popular implementation named apt-p2p. Then, we develop a crawler to record the information of a normal peer and a bootstrapping peer in apt-p2p network. We measure and analyze the routing table, database, network traffic and network delay for this package management network. Some interesting and valuable results were found in our measurement and the results shows that it is feasible and reliable to use P2P technique to support distributions package measurement.**

***Index Terms*—DHT, measurement, package, Linux.**

I. Introduction

Linux-based open source distributions have been more and more popular in the past decade, e.g., Redhat [1] is used in more than 60% of enterprise servers and Ubuntu [2] owns ten millions of users during the last 5 years. These distributions mostly distribute free software based on the client/sever model, e.g., the Advance Package Tool (apt) for Ubuntu. Normally, there are tens to hundreds of mirror servers for a Linux distribution. When a new version (or even a security patch) is released, there are millions of upgrade requests from all over the world rush to those mirror servers. On one hand, these mirror servers would prepare a large amount of bandwidth to handle these frequently requirements; on the other hand, users may still suffer from slow response (or even denial of service) at these rush hours.

Peer-to-Peer(P2P) technique has been a hot research area in the last 20 years along with rapid improvement of Internet. Nowadays, P2P applications have been more and more popular and have lots of users, especially P2P file sharing and P2P live stream. It is reported that P2P file sharing contributes more than 70% of the traffic in some areas [3]. Given the free nature of Linux-based software, there are normally a number of users motivated by altruism to help out with the distribution, in order to promote the healthy development of this voluntary society. P2P technique is a nature choice to make use of these available resources and scale the package management system to a distributed manner.

Although it seems straightforward to use an existing P2P file sharing tool like BitTorrent for this free software package distribution, there are indeed a series of new challenges in this unique scenario [4], such as 1) 80% of the packages are less than 512 KB, 2) about 1.5% of the software archive is updated with new versions of packages, and 3) 80% of the packages are installed by less than 1% users. Hence, C.Dale and J.Liu propose a novel peer-to-peer assisted distribution system to address the above challenges. It is implemented as apt-p2p, a practical implementation based on the Debian Package distribution system. This software was first released at about 6 years ago and is still in use during these days. In this paper, we will empirically measure apt-p2p to evaluate the feasibility and reliability of a distributed package management system, which is helpful to guide the future design and optimization of a better P2P package management system.

We developed a crawler to record the information of a normal peer and bootstrapping peer in apt-p2p network. This crawler was deployed for 20 days. Our measurement shows that:

- The total number of users in apt-p2p network is estimated at 128, which is 150% more than the number estimated in 2008.
- The routing table of the boot peer changes frequently and the number of its contacts is 118% more than the normal peer.
- The normal peer stores 12 keys and 12 values, and these numbers do not change all day; the boot peer stores 427.8 keys and 501.1 values on average, and these numbers change in the range of 15%.
- The minimal bandwidth for current apt-p2p network is 211.5 Bps and the maximal bandwidth price is 1371.5 Bps, which is acceptable in today's Internet situation.
- The normal peer needs 6.0 iterations and the boot peer needs 3.2 iterations to locate the target.

The rest of this paper is organized as follows. The architecture and implementation of a P2P package management system for Linux distributions are presented in Section II. We introduce our measurement methodology and analyze the results in Section III. We then recall some relate work in Section IV and the Section V concludes the paper and offers some future directions.

II. Overview

In this section, we first present the design idea of the P2P assisted distribution system for free software package management, then introduce the principle of a DHT network,

Manuscript received October 5, 2014; revised December 15, 2014. This work was supported in part by the National Natural Science Foundation of China (61103015, 61303190).

Jie Yu, Weihua Zhang, Xiaodong Tang, Shasha Li, and Qingtao Shen are with the National University of Defense Technology, Hunan, China (e-mail: yj@nudt.edu.cn).

Shasha Li and Qiang Li are with School of Computer Science and Engineering, Beihang University, Beijing, China.

DOI: 10.7763/IJFCC.2015.V4.358

and finally give the detail of a Debian implementation of the P2P package management system.

A. Architecture

A key principle in this design is that the new functionalities implemented in the distributor should be transparent to users, thus offering the same experience as using conventional software management systems, but with enhanced efficiency.

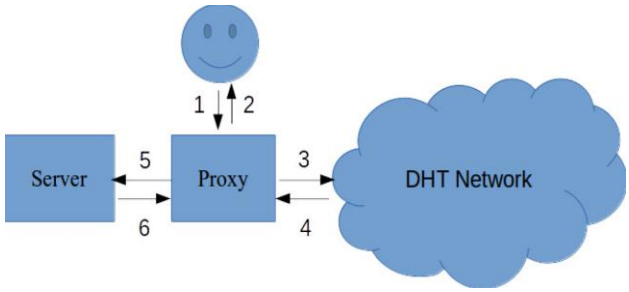


Fig. 1. The architecture of a P2P based distributed package management system.

The architecture is shown in Fig. 1. The voluntary users form a DHT (Distributed Hash Table) network with the ability to locate peers and resources in the network. When a user wants to install or upgrade a software, the request will be sent to the proxy (step 1). The proxy firstly send the request to DHT network (step 3) to find if a peer has the resource and where the peer is (step 4); if not, the proxy will sent the request to mirror server (step 5) and the mirror server then send back the data (step 6). The proxy will finally send back the data to the user (step 2).

B. DHT Network

The most important part for this P2P package management system is the building and maintenance of DHT network. In this section, we give a brief overview of a typical DHT protocol, i.e., Kademlia [5]. Each Kademlia peer has a 128-bit ID and it computes the distance between two peers by XOR metric of their IDs.

In Kademlia, when a peer *P* joins the network, it sends BOOTSTRAP request messages to n known peers. An alive peer which receives the message will respond it with a BOOTSTRAP response message. *P* then builds its own routing table and sends its location message – HELLO request message to all peers in its routing table. When a peer receives HELLO request message, it will add the location information to its routing table.

Then, *P* moves on to publish files information it is sharing. File publishing process consists of two steps as to convenient for searching and economize memory resources:

- Location information publishing: First, P hashes each file in its sharing list and obtains a 128-bit file identifier. Then it sends each file identifier and file location (IP and TCP port) to peers close to the file identifier. When peers receive this PUB_SOURCE request message, they update their local file indexes.
- Metadata information publishing: First, P extracts keyword from each name of sharing files and hashes each keyword into a 128-bit key. Then it sends each key, file identifier and metadata information of the file to peers close to the key. When peers receive this PUB_KEYWORDS request message, they update their local keyword indexes.

When searching resources, *P* hashes each keyword that user enter to search, and then send the key into Kademlia for iteratively searching. When a peer that records for this keyword receives the message, it responds corresponding records to *P*. Each record contains files identifier and meta -data information of the file. *P* then displays all matching file identifiers to user. After user selects certain file identifier *I*, *P* sends location search messages of *I* into Kademlia for iteratively searching. When a peer that has records for this file identifier receives the message, it responds corresponding to *P*. Each record contains file location (IP and TCP port). *P* then tries to establish TCP connections with these IPs and downloads that file simultaneously.

C. Implementation

C. Dale and J. Liu have created a sample implementation that functions as described above, and is freely available for other distributors to download and modify [6]. This software, called apt-p2p, interacts with the popular apt tool.

Since all request from apt are in the form of HTTP downloads from a server, this implementation takes the form of a caching HTTP proxy. Making a standard apt implementation use the proxy is then as simple as perpending the proxy location and port to the front of the mirror name in apts configuration file (i.e. http://localhost:9977/ mirror name. debian.org/...).

A customized DHT based on Khashmir is created, which is an implementation of Kademlia. Khashmir is also the same DHT implementation used by most of the existing BitTorrent clients to implement trackerless operation. The communication is all handle by UDP messages, and RPC (remote procedure call) requests and responses between nodes are all be encoded in the same way as BitTorrent .torrent files.

Downloading is accomplished by sending simple HTTP requests to the peers identified by lookups in the DHT to have the desired file. Requests for a package are made using the packages hash (properly encoded) as the URL to request from the peer. The HTTP server used for the proxy also doubles as the sever listening for requests for downloads from other peers. All peers support HTTP/1.1, both in the server and the client, which allows for pipelining of multiple requests to a peer, and the requesting of the smaller pieces of a large file using the HTTP Range request header. Like in apt, SHA1 hashes are then used to verify downloaded files, including the large index files that contain the hashes of the individual packages.

III. Measurement

A. Youker-APT Project

The apt-p2p package was initial released on 25 Apr, 2008 and five updates were released during 2008. Then the authors released the 0.1.6 version on 21 Mar, 2010. Although there is no updates during the next five years, we found that this package is still in the archives of both Debian and Ubuntu. The latest version is 0.1.6+nmul uploaded by Michael Gilbert, who fixed a Python dependence bug.

We tried to install the apt-p2p 0.1.6+nmul on Ubuntu Kylin 14.04 LTS, a Ubuntu based Chinese Linux distribution [7], and it fails due to the upgrade of python-apt library. Hence we create a Youker-APT project [8] forked from apt-p2p and released the first 0.1.7 version which updates to python-apt 0.9.3.5 API and sqlite3 API. We plan to maintain and improve this P2P-based distributed package management system in future.

B. Measurement Methodology

We developed a crawler writing in Qt to record following information of a peer: 1) the number of contacts in its routing table; 2) the number of keys in its database; 3)the number of values in its database; 4) the download traffic for DHT maintenance; 5) the upload traffic for DHT maintenance; 6) the delay for peer ping and 7) the delay for peer location.

We deployed this crawler for more than 20 days during August and September of 2014. It monitors two peers in apt-p2p network: one is a normal peer that joined from a usual PC running Ubuntu Kylin 14.04 LTS with 10 Mpbs Internet access and 2.4 GHz CPU (called as normal peer for short), the other is a bootstrapping peer that has been deployed on a public web server for a long time (called as boot peer for short).

C. Results Analysis

We found that the peer information shows a daily cycle, so we just take and analyze the information of one day as a typical example.

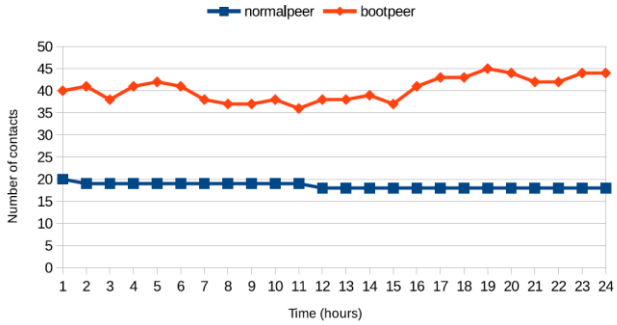


Fig. 2. Number of contacts in each routing table.

Each peer has a list of known peers, called contacts, which are structured by the routing table. Routing table is a key component in DHT. It plays the role for DHT maintenance. Fig. 2 shows the number of contacts in the routing table of each peer. The average number of contacts for normal peer and boot peer are 18.5 and 40.4 respectively. Since boot peer receives ping messages frequently, its routing table changes frequently too and its number of contacts is 118% more than the normal peer. We further monitor the estimated number of users in the DHT according to the formula, where table.buckets is a list of bucket which stores up to K contacts and $K = 8$ in this apt-p2p network. The result shows that the total number of users in apt-p2p network is estimated as 128, which is 150% more than the number estimated in 2008 [4].

Each peer has a database to store the key-value pairs for package information. Fig. 3 and Fig. 4 show the number of keys and values in the database of each peer. The normal peer stores 12 keys and 12 values, and these numbers do not change all day; the boot peer stores 427.8 keys and 501.1 values on average, and these numbers change in the range of 15%. Since boot peer would be in the routing tables of most peers, the variation of key value pair in its routing table could reflect the variation of the hole network. We found that there are two peeks everyday at about 13 PM and 23 PM in GMT+8. We also found that the rate between key and value is 1.00 for the normal peer and 1.17 for the boot peer.

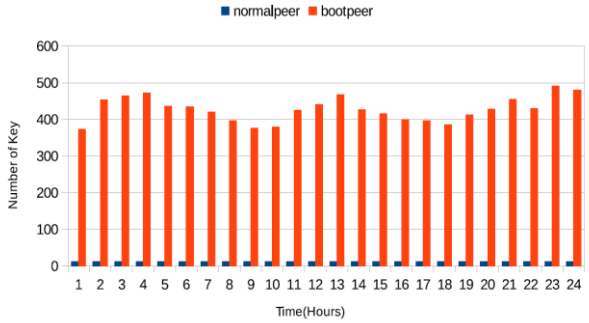


Fig. 3. Number of keys in each database.

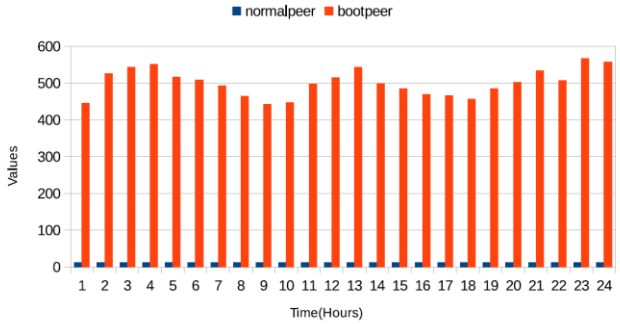


Fig. 4. Number of values in each database.

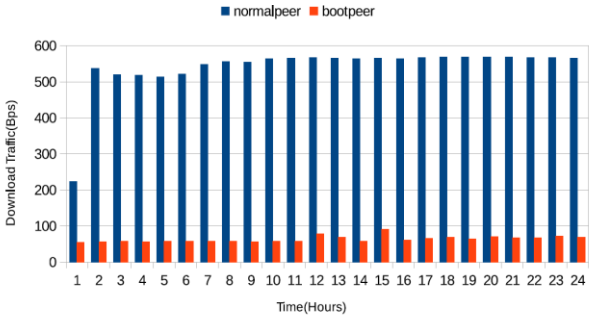


Fig. 5. Download traffic for each peer.

As introduced in Section II-B, a peer should send BOOT STRAP request messages to get contacts into its routing table, send HELLO (ping) message to keep contacting with other peers, and so on. All these actions require the traffic between the peer and the other peers, which represents the bandwidth price of network maintenance. Fig. 5 and Fig. 6 show the download traffic and upload traffic for each peer. The average download traffic are 541.3 Bps and 64.0 Bps respectively; the average upload traffic are 830.2 Bps and 147.5 Bps respectively. In our measurement, the bandwidth price for normal peer is about 6-7 times more than that for boot peer. It's due to two reasons: 1) the routing table of normal peer is very small, so it sends find node messages more frequently than the boot peer, and 2) the normal peer has several packages to share to the apt-p2p network, which results in frequently store value messages. Hence, we can conclude that the minimal bandwidth price for current

apt-p2p network is 211.5 Bps and the maximal bandwidth price is 1371.5 Bps, which is acceptable in today's Internet situation.

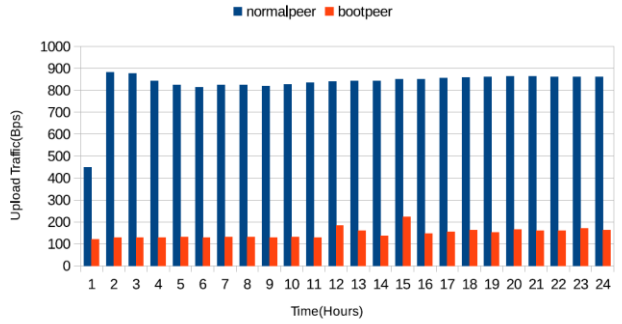


Fig. 6. Upload traffic for each peer.

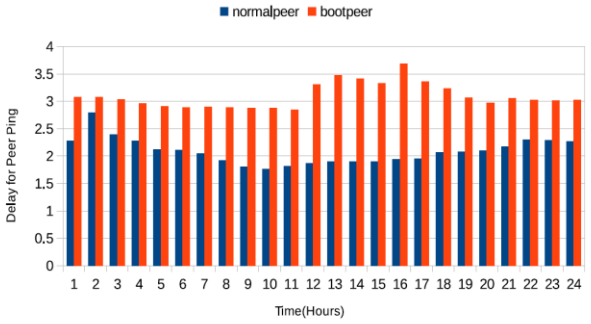


Fig. 7. Delay for peer ping.

Fig. 7 shows the delay for ping message for each peer. The average delay is 2.1 seconds for the normal and 3.1 seconds for the boot peer. We also found that the peek of delay for each peer occurs at different time. It's possibly because the normal peer is located in China and the boot peer is located Canada.

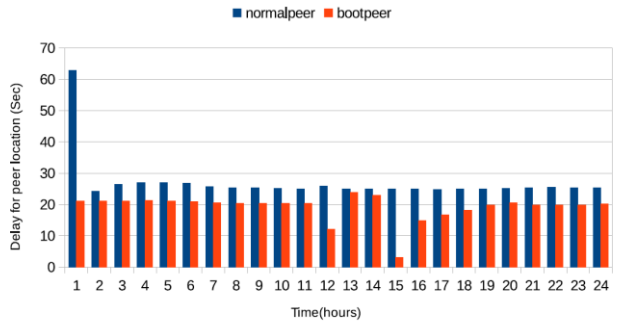


Fig. 8. Delay for peer location.

A peer iteratively sends find node messages to nearest contact until it finds the target. The delay to locate a peer depends on the network condition (i.e., the ping delay for each peer) and number of iterations. Fig. 8 shows the delay to find a target for each peer. The average delay is 25.4 seconds and 19.6 seconds respectively. We could calculate the average number of iterations as the average location delay dividing two times of a single ping delay. It means that the normal peer needs 6.0 iterations and the boot peer needs 3.2 iterations to locate the target. This is mainly because that the routing table of the boot peer is about one time bigger than the normal peer and the boot peer could start with a closer contact.

IV. Related Work

There are extensive works on measurement and modeling of P2P networks, such as file sharing workload of Kazaa [10], query behavior of Gnutella [11], swarm evolution of BitTorrent networks [12], tracker availability in BitTorrent systems [13]. Gummadi *et al.* [10] demonstrated that the fetch-at-most-once behavior causes the Kazaa opularity distribution to deviate sub-stantially from Zipf-like curves for the Web, and this deviation has significant implications for the performance of multimedia file-sharing systems. Klemma *et al*. [11]] characterized peer behavior in a form that can be used for constructing representative synthetic workloads for evaluating new P2P system designs, and The characterization is based on trace data gathered in the Gnutella P2P system over a period of 40 days. Guo *et al*. [12] found that client performance in the BitTorrent-like systems is unstable, and fluctuates widely with the peer population. Dhungel *et al*. [13] analyzed the resilience of BitTorrent leechers to two different kinds of attacks: the connection attack and the piece attack, and discovered that BitTorrent architecture is fundamentally resilient to Internet leecher attacks.

For DHT networks, a few studies are performed on Mainline and Azureus, while lots of measurements and analysis focus on Kad [14]-[16]. Stutzbach *et al.* [17] investigated how the efficiency and consistency of lookup in Kad can be improved by performing parallel lookup and maintaining multiple replicas, and they empirically showed the best operating point for the degree of lookup parallelism and the degree of replication. Kang *et al*. [18] studied the poor lookup hit rate in Kad, and found that this poor performance is due to the high level of routing table similarity, despite the relatively high churn rate in the network. We have observed that a significant portion of peers in Kad do not have unique IDs. We further analyzed the effects of ID repetitions under simplified settings and found that ID repetition degrades Kads performance on publishing and searching, but has insignificant effect on lookup process [19]. We also propose a hybrid search strategy which could both start up quickly and slow down slowly to get an accurate snapshot of a DHT network [20]. There are also several work [21]-[23] that try to investigate and analyze distributed package management, which are much similar to apt-p2p system we deeply measure in this paper.

V. Conclusion and Future Work

There are very few research work focusing on P2P assisted package distribution system. In this paper, we measure and analyze the routing table, database, network traffic and network delay for apt-p2p package management network. We found that the total number of users in apt-p2p network is estimated as 128, which is 150% more than the number estimated in 2008, and the network traffic and network delay are acceptable for normal Linux users. These measurement results are valuable to improve the design and implementation of a P2P package management system.

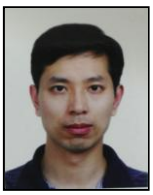
Our future work are as following: 1) develop a powerful crawler to collect the global information of apt-p2p network and investigate its DHT characteristics and user behaviors; 2) improve Youker-APT project based on apt-p2p

implementation and make it as a default service in Ubuntu Kylin distributions; and 3) design a more efficient distributed package management system based on the measurement results and user feedback.

REFERENCE

[1] Redhat Website. [Online]. Available: http://www.redhat.com/.
[2] Ubuntu Website. [Online]. Available: http://www.ubuntu.com/.
[3] J. Falkner, M. Piatek, J. P. John, A. Krishnamurthy, and T. Anderson, "Profiling a million user DHT," in *Proc. the 7th ACM SIGCOMM Conference on Internet Measurement*, vol. 7, 2007.
[4] C. Dale and J. Liu, "Apt-p2p: A peer-to-peer distribution system for software package releases and updates," in *Proc. the 28th IEEE International Conference on Computer Communications (INFOCOM)*, 2009.
[5] P. Maymounkov and D. Mazires. Kademlia, "A peerto-peer information system based on the XOR metric," *Lecture Notes in Computer Science*, pp. 1-6, 2002.
[6] The Apt-P2p Website. [Online]. Available: http://www.camrdale.org/aptp2p/
[7] The Ubuntu Kylin Website. [Online]. Available: http://www.ubuntukylin.org/
[8] Youker-APT Website. [Online]. Available: https://launchpad.net/youkerapt
[9] R. Jimenez, F. Osmani, and B. Knutsson, "Connectivity properties of mainline bittorrent DHT nodes," *Lecture Notes in Computer Science*, vol. 3, 2009.
[10] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," *Computer Science and Engineering*, pp. 1-16, 2003.
[11] A. Klemma, C. Lindemanna, M. Vernonb, and O. Waldhorsta, "Characterizing the query behavior in peerto-peer file sharing systems," in *Proc. the 4th Conference on Internet Measurement*, pp. 56-67, 2004.
[12] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurement, analysis, and modeling of BitTorrent-like systems," in *Proc. the 5th Conference on Internet Measurement*, pp. 35-48, 2005.
[13] P. Dhungel, D. Wub, and K. Ross, "Measurement and mitigation of bittorrent leecher attacks," *Computer Communications*, vol. 32, pp. 852-1861, 2009.
[14] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a DHT," present at USENIX Annual Tech. Conf., 2004.
[15] M. Steiner, T. E. Najjary, and E. W. Biersack. "A global view of KAD," in *Proc. the 7th Conference on Internet Measuremen*t, pp. 117-122, 2007.
[16] M. Steiner, D. Carra, and E. W. Biersack, "Long term study of peer behavior in the KAD DHT," *ACM Transactions on Networking*, pp. 1371-1384, 2009.
[17] D. Stutzbach and R. Rejaie, "Improving lookup performance over a widely deployed DHT," *Distributed Hash Tables*, vol. 2, 2006.
[18] H. Kang, E. Chan-Tin, N. Hopper, and Y. Kim, "Why Kad Lookup Fails," *Computer Communications*,vol. 34, no. 13, 2009.
[19] J. Yu, Z. J. Li, P. Xiao, C. F. Fang, J. Xu, and E.-C. Chang, "ID repetition in structured P2P networks," *The Computer Journal*, vol. 54, no. 6, pp. 962-975, 2011.
[20] J. Yu, P. Xiao, Z. J. Li, and Y. Zhou, "Towards an accurate snapshot of DHT networks," *IEEE Communication Letters*, vol. 15, no. 1, pp. 97-99, 2011.
[21] P. Shah, J. Pris, J. Morgan, J. Schettino, and C. Venkatraman, "A p2p-based architecture for secure software delivery using volunteer assistance," *Software Security Built For Developers*, vol. 2, 2008.
[22] Z. Xu, "Distributed package management network," Internship Report, 2010.
[23] E. Neblock, "Peer-to-peer based package management," Abstract for Master Thesis, 2013.

Jie Yu was born in 1982. He is working as an assistant professor of National University of Defense Technology (NUDT), China. He got his PHD in 2010 at NUDT. His research interests include distributed system and operation system. He has published more than 30 papers, including journals like the Computer Journal, Computer Communications, IEEE Communication Letters, and conferences like ICPP, P2P, ICPADS.



Weihua Zhang was born in 1977. He is working as an associate professor of National University of Defense Technology (NUDT), China. He got his PHD in 2006 at NUDT. His research interests include operating system design and engineering. He has published more than 10 papers.

Xiaodong Tang was born in 1973. He is working as an associate professor of National University of Defense Technology (NUDT), China. He got his bachelor's degree in 1998 at NUDT. His research interests include operation system and security. He is working on the Development of the Open Source Operating System.

Shasha Li was born in 1982. she is working as an assistant professor of National University of Defense Technology (NUDT), China. She got his PHD in 2011 at NUDT. Her research interests include big data and intelligent information processing. She has published more than 10 papers, including journals like TKDE, and conferences like ACL.

Qiang Li was born in 1982. He got his PhD candidate of Beihang University, China. His research interests include P2P and network security.

Qintao Shen was born in 1993. He is a graduate in National University of Defense Technology (NUDT), China. He got his bachelor's degree in 2010 at Henan Polytechnic University (HPU). He joined the research in 2014 which focuses on distributed system and operation system.