JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 33, 1417-1432 (2017)
DOI: 10.6688/JISE.2017.33.6.2

Peer-to-Peer File Sharing Framework for Smartphones: Deployment and Evaluation on Android

FARRUKH ASLAM KHAN[1,3], UMAR MANZOOR[2], AZHAR KHAN[3],
AFTAB ALI[3], HAIDER ABBAS[4] AND MARUF PASHA[5]
[1]*Center of Excellence in Information Assurance*
King Saud University
Riyadh, 11653 Saudi Arabia
[2]*Faculty of Computing and Information Technology*
King Abdulaziz University
Jeddah, 21589 Saudia Arabia
[3]*Department of Computer Science*
National University of Computer and Emerging Sciences
Islamabad, 44000 Pakistan
[4]*Department of Information Security*
National University of Sciences and Technology
Islamabad, 44000 Pakistan
[5]*Department of Information Technology*
Bahauddin Zakariya University
Multan, 60000 Pakistan
E-mail: fakhan@ksu.edu.sa

Peer-to-Peer (P2P) applications especially in the domain of file sharing have become very popular during the past few years. These applications have huge traffic volume on the Internet as compared to any other application. In this paper, we propose a mobile P2P file-sharing framework that uses dedicated Session Traversal Utilities for NAT (STUN) servers for Network Address Translation (NAT) traversal. In our case, we divide the nodes into two types: Node Peers (mobile applications) and Super Peers (dedicated servers connected to a fixed P2P network). We develop the proposed framework for Android operating system and test it on a 3rd Generation (3G) network testbed. The experimental results obtained are quite positive and support the implementation of the system.

Keywords: mobile peer-to-peer (P2P), peer-to-peer file sharing, Android operating system, network address translation (NAT), smartphones

1. INTRODUCTION

File sharing [1, 2] in mobile phones is usually done using two methods *i.e.*, either by using Bluetooth, which has the distance limitation, or by using Multimedia Messaging Service (MMS), which has the size/type limitation [3]. With the evolution in mobile/ telecom networks (*i.e.*, 3G, 4G), the leading IT companies (such as Skype) have supported their existing Peer-to-Peer (P2P) applications [4-6] to work for mobile phones [7-9]. To the best of our knowledge, pure Mobile Peer-to-Peer (MP2P) application does not exist. The reason behind our opinion is that each mobile phone/smartphone is assigned a private IP address when it is connected to the mobile Internet, whereas each computer connected to

Received November 22, 2016; revised December 14, 2016; accepted April 4, 2017.
Communicated by Kashif Saleem.

the Internet is assigned a public IP address. This means that the mobile device cannot be accessed publicly even if the IP address is known, whereas the computer can be accessed from outside if its public IP address assigned is known. Therefore, address translation is required on behalf of mobile phones and it is done usually at the router, which has a public IP address. The service provider provides the Network Address Translation (NAT) service [10-13]. Mobile phones are assigned IP addresses of the private network and whenever some mobile phone needs to use the Internet, it sends out an IP packet. The service provider's router receives that packet and replaces the private IP address with its own public IP address and sends the packet to the Internet. The router maintains a table of mappings between private IP addresses and port numbers through which it keeps track of the sources. The problem with NAT is that the connection can be initiated from inside the private network to any destination that has a public IP address, but the opposite is not possible.

During the last decade, researchers have proposed many efficient techniques and protocols for NAT traversals. Some of the proposed techniques are Relaying, UDP hole punching, TCP hole Punching, and connection reversal [14] *etc.* Most commonly considered protocols are NAT-PMP, TURN, STUN, and ICE. All of these solutions require some sort of fixed structure in the form of servers; therefore, these solutions cannot be used for a pure MP2P network. Few P2P applications have been developed *e.g.*, Skype, which do the NAT translations themselves. Skype is a very famous and highly efficient P2P VoIP application. It uses propriety protocol for its application, which is not published yet. Researchers have tried and studied the architecture of Skype through reverse engineering [15-17]. What we know about Skype is that it uses an overlay P2P network, which has two types of nodes: normal nodes and super nodes. A super node is an endpoint for a normal node. Due to this structure, Skype is not considered as a pure P2P application. Skype uses a variation of STUN for NAT traversals and super nodes play a vital role in it. After initialization, the Skype client registers itself with a super node and through that super node; it can be accessed behind a NAT.

Due to the limitations of Bluetooth and MMS technologies, researchers have started to explore other solutions for file sharing in the MP2P domain, where files of any type/size can be shared from anywhere in the world. Understanding the importance of the issue, we propose and develop a MP2P file-sharing framework, which uses dedicated Session Traversal Utilities for NAT (STUN) servers for NAT traversal. In our case, we divide our nodes into two types: Node Peers (NPs) that work as a mobile application, and Super Peers (SPs), which are dedicated servers. NP is a file sharing application running on the smartphone, whereas SP is an application running on a computer system or a mobile device having a public IP address. Any mobile device with a public IP address, sufficient processing power, memory, and network bandwidth, can become a SP. When a NP application starts, it sends a request to the Registration Server (RS) for SP configuration. After receiving the request, it sends a connect request to the SP. After a successful connection, NP can either share files with other NPs or search for a specific file. We test the effectiveness of the proposed framework on a 3rd Generation (3G) network. The results obtained are very promising and support the implementation of the system.

The remainder of the paper is organized as follows: In section 2, the related work is discussed in detail. The proposed system design for MP2P file sharing is discussed in section 3. Section 4 presents the details of our experiments and results. Finally, section 5 concludes the paper with possible future directions.

2. RELATED WORK

P2P networks and their file sharing applications have become very popular during the last decade. These applications have huge traffic volume on the Internet as compared to any other application. Application developers and designers have developed many file sharing architectures *e.g.*, Kazaa, eDonkey, Gnutella, Napster, and BitTorrent *etc.* Later, it was realized that P2P is also a very attractive domain for mobile networks [7-9]. Because of the challenges involved in mobile networks such as limited processing [18] and memory capacity [19] of mobile devices and wireless media, the existing wired network solutions cannot be directly used in a mobile network environment. Therefore, various P2P architectures have been proposed for mobile networks.

Mobile eDonkey is a P2P file sharing application based on eDonkey protocol with modifications [20]. The main components of Mobile eDonkey are eDonkey client and eDonkey index server. Since TCP is used to transmit data in eDonkey, it suffers packet loss due to mobile environment, which in turn delays the network traffic [21]. The other disadvantage is the mechanism of scoring system in which a peer behind a firewall or a NAT gets a low ID, which consequently reduces the priority of the service. JXTA [22] is P2P file-sharing technology based on a layered architecture. JXME, a mobile P2P file sharing based on JXTA was proposed in [22]. Any Mobile Information Device Profile (MIDP) device is able to collaborate with other MIDP devices as well as with other peers running JXTA for desktops with some restrictions. The disadvantage is limited bandwidth and high latency. The other critical issue is the library supporting XML parsing.

Moby Architecture proposed in [23] is based on existing technologies; Jini and the Jini Technology Surrogate Architecture Specification [24]. Mobile P2P over Session Initiation Protocol (SIP) is proposed in [25]. SIP is a signaling protocol, which is based on Hypertext Transfer Protocol (HTTP). The work in [26] also presents a SIP-based architecture for MP2P content sharing services in mobile networks. The prototype is implemented in C++ on Symbian Series 60 platform in a mobile handset with GRPS/UMTS access. SymTorrent is a full-featured BitTorrent client for Symbian platform [27]. By installing SymTorrent, mobile devices can connect to the same BitTorrent trackers and enable the mobile devices to transfer files among peers. However, it has several bugs that crash the application during downloading, installation, or execution. Mobile proxy [28] is a hierarchical mobile P2P architecture, which is connected to a fixed P2P network.

The authors in [29] proposed a pricing scheme for resource allocation in P2P file-sharing networks. In [30], a file-sharing framework for content-based file searching in mobile ad hoc networks is proposed. The authors in [31] describe four P2P applications developed for smartphones using cellular communication. The applications include distributed computing as well as content sharing. The work presented in [32] uses ZigBee protocol for MP2P using Android. The authors in [33] present the design and implementation issues regarding P2P file sharing for mobile devices. However, they provide a proof-of-concept file sharing application for mobile phones that uses Bluetooth technology. In [34], authors developed an Android application for P2P file sharing named "Mobile Torrent" used for sharing files between smartphones within a campus without the use of the Internet. They implemented it for single hop only using Wi-Fi Direct, without using any P2P ad hoc network connection.

3. SYSTEM DESIGN

In the proposed MP2P framework, we have two types of nodes namely Node Peers (NPs) and Super Peers (SPs). NP is a File Sharing application running on the smartphone, which is used to share files of any type with other connected NPs, whereas SP is an application (running on PC or a mobile device having public IP address) with the following functionalities: 1) It maintains a list of connected NPs and a list of file names shared by these NPs, and 2) It can search for file(s) on the request of the NP. Any mobile device with a public IP address, sufficient processing power, memory, and network bandwidth can become a SP. In order to share files, every NP must connect to the SP; however, the NP has to get public IP address of its SP from the Registration Server (RS), whose role is to provide help in the NAT traversal. Each SP, when starts, sends a registration request to the RS whose IP address is pre-configured in the SP application.

RS stores the Public IP address and geographic location information of the registered SPs and shares this information with the NP upon request. RS is also responsible to monitor the status of SPs (*i.e.*, offline/online) and for this purpose, periodic ping messages are sent to each SP. If any of the SPs is found offline, the status of the same is updated on the RS. Whenever a NP application starts, it sends request to the RS for SP configuration. RS checks the geographic location of the NP and sends the three IP addresses of the SPs, which are closely located with it with reference to the location. After receiving the list of SP IP addresses, NP connects to the first SP in the list. Upon connection failure, it sends the request to another SP listed in the list and so on. In this way, the MP2P architecture is built where peers can communicate with each other any time through NAT-mapped IP addresses. The numbers of SPs in a city totally depend on the number of NPs. For example, if the numbers of NPs in a city increase and become more than the already specified number that the existing SPs can handle, then the number of SPs can be increased dynamically. In Fig. 1, each SP is representing a different region and the SPs can communicate with each other by the 3G network provided by the Telecom company.

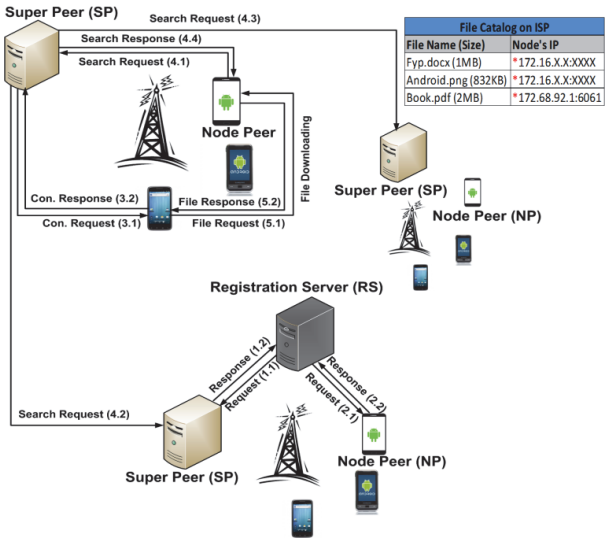


Fig. 1. High level architecture of P2P file sharing framework for smartphones.

3.1 Super Peer Registration with Registration Server

In the initialization phase, the SP application performs the following tasks: (1) it loads the configuration file, which includes the RS information (*i.e.*, IP address); (2) it sends registration request (which includes IP address of the SP) to the RS. Assume that the IP address of the system/smartphone running the SP application is 172.16.4.2 in a public network; this IP address is embedded in the registration request. RS stores the IP address and the geographic location of the requested SP in the SP registration list and upon successful registration, it sends the registration confirmation message to the SP as shown in Fig. 2. The SP registration list contains the names of the cities under the geographic location column.

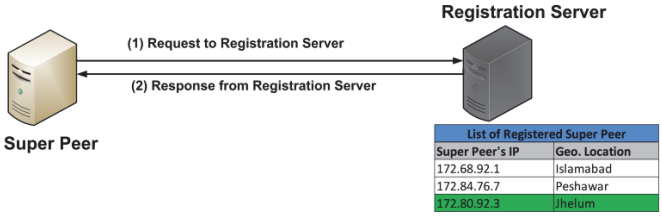


Fig. 2. Super peer registration process.

3.2 Node Peer Registration

NP sends registration request to the RS, which checks the geographic location of the requesting node and searches three closest SPs with respect to the geographic location. RS sends the IP addresses of the selected SPs to the NP as shown in Fig. 3.

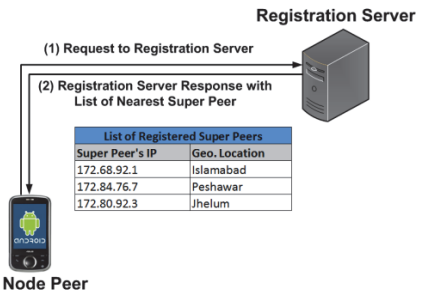


Fig. 3. Super peer registration process.

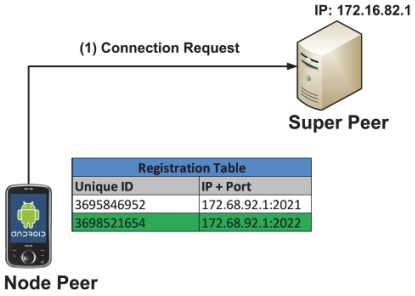


Fig. 4. Node peer connection request.

3.3 Node Peer Connection Request

After receiving the list of public IP addresses of the SPs from the RS, NP sends connection request (which contains the unique ID of the NP) to the first SP 172.16.82.1, as shown in Fig. 4. After receiving the unique ID of the NP, the SP assigns a forwarding port corresponding to this unique ID. Since each NP has a private IP address (NAT traversal required), it cannot accept any request; therefore the SP assigns a forwarding port and starts listening on behalf of the NP.

3.4 File Sharing

Assume a NP in City X wants a file (*e.g.*, Book.pdf). It sends the search file request containing the file name to the connected SP. The following scenarios can occur:

3.4.1 File found in local catalog

When SP receives the search file request from one of the connected NPs, it searches the file in its file catalog. If the SP finds the requested file (Book.pdf) from its local file catalog, it sends the response back to the requesting NP with the IP address of the NP who has the file (Book.pdf), as shown in Fig. 5.

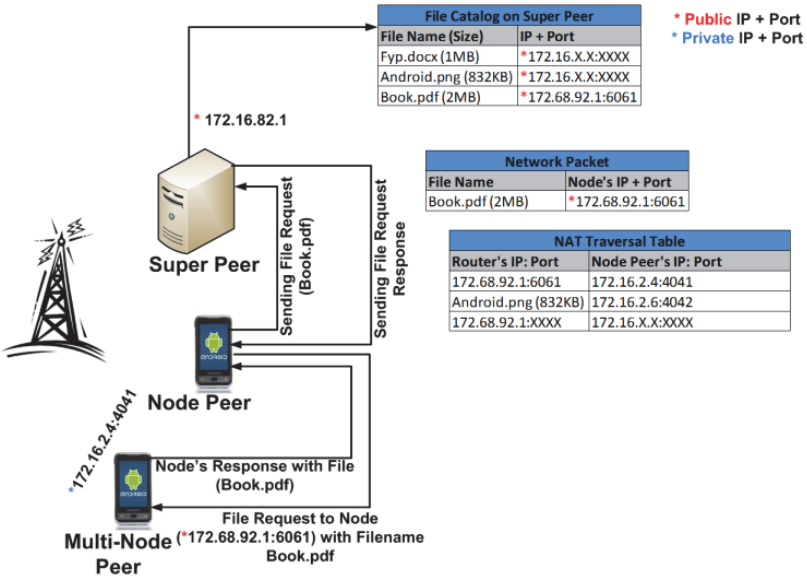


Fig. 5. Super peer finds file in its file catalog.

3.4.2 File found in remote catalog

If the SP does not find the requested file in its file catalog; then it forwards the search request containing the file name to the neighboring SP and this process goes on until the file is found, or the limit (configurable) on forwarding the search request to the neighboring SP is reached, or a specific time limit (configurable) is reached. After receiving the search file request from the neighboring SP, the receiving SP searches the requested file from its file catalog. If it finds the file in its file catalog, then it sends the response to the neighboring requesting SP with the IP address and Port no. (172.16.82.2: 4041) of the NP who has the requested file. After receiving the response from the neighboring SP, the requesting SP forwards the response to the requesting NP, as shown in Fig. 6. After receiving the IP Address of the NP who has the requested file, the requesting NP sends the file request to the NP with the address 172.16.82.2:4041, which respond back to the requesting NP with the file (Book.pdf) and the NP in City X gets the file as shown in Fig. 6.

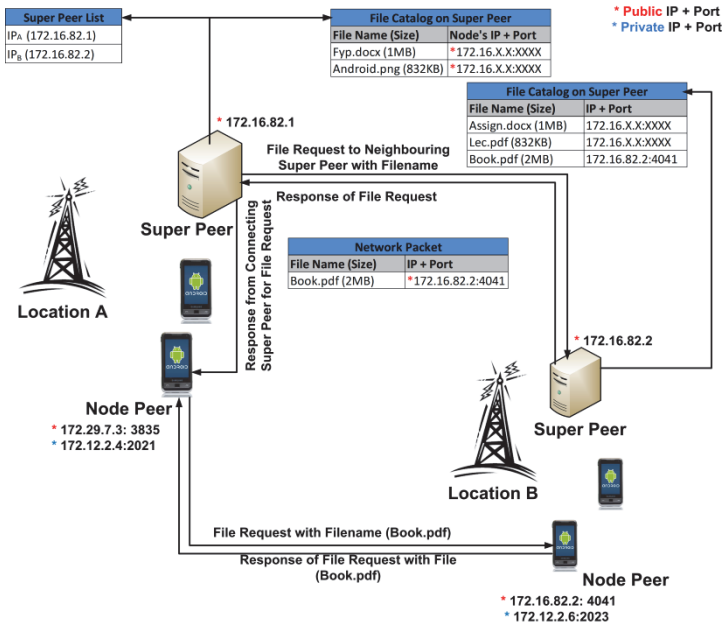


Fig. 6. SP failed to find requested file in its Catalog but found in connecting SP.

3.4.3 File not found

It is quite possible that a user requests for a file *e.g.*, abc.txt, which is not available on the network. After sending the file requests to all its neighboring SPs and receiving "file not found" response from all of them; the requesting SP sends "file not found" message to the requesting NP as shown in Fig. 7.

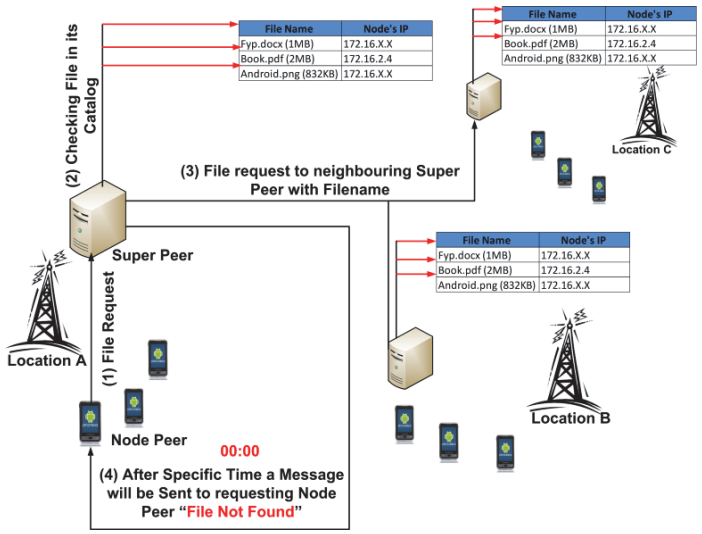


Fig. 7. File does not exist.

3.5 Super Peer Catalog Updating

In a live network, changes such as Joining/Leaving of NPs, and addition or removal of file(s) in NPs occur rapidly. If any of the above change occurs in the network, SP can respond to the requesting NP with wrong information. Therefore, some type of mechanism is required to keep the SP updated with the current status of its connected NPs to answer the requesting NP with effective and accurate information. To cope with the changes above, we have incorporated periodic update file catalog request, which is sent by a SP to all connected NPs. On receiving the request, NPs respond with the updated file list or "no update" message. After receiving response from the NPs, the SP updates its file catalog (if required), as shown in Fig. 8.

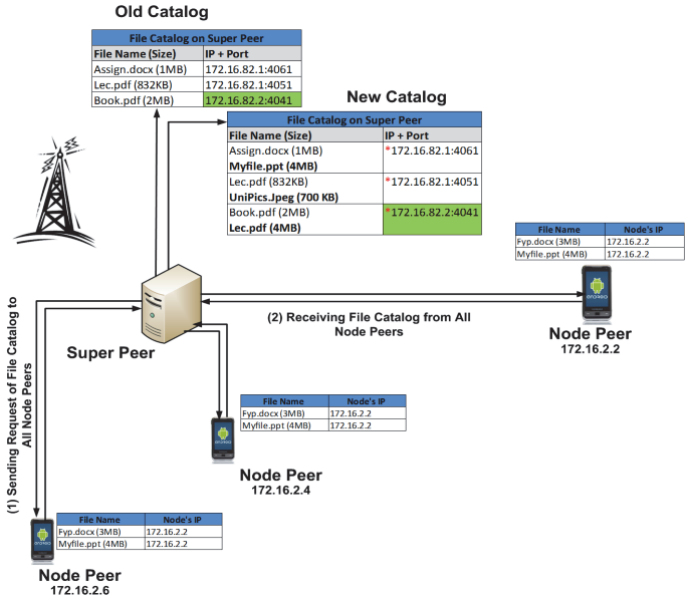


Fig. 8. SP updating its file catalog.

3.6 Fault Tolerance in File Transfer

It is quite possible that the NP that has the file disconnects from the network during the file transfer. Therefore, to incorporate fault tolerance in the file transfer, SP sends a list of IP addresses of NPs that have the requested file instead of one NP to the requesting NP. Assume a NP searches for a file "Book.pdf"; the request is forwarded to the SP, which checks its file catalog and finds the requesting file on the multiple NPs as shown in Fig. 9. SP responds with the list of IP addresses of the NPs who have the file "Book.pdf" to the requesting NP, which then sends the file request to the first IP address from the list (*i.e.*, 172.16.2.4). After receiving the file request, NP (172.16.2.4) starts sending the file "Book.pdf" to the requesting NP. Let us assume that after time X, NP (172.16.2.4) crashes or disconnects while transferring the file. In this case, the requesting NP selects the next IP address (172.16.2.6) from the list and sends file request to the NP

(172.16.2.6), which responds to the requesting NP with file (Book.pdf) and the process of file transferring resumes, as shown in Fig. 9.

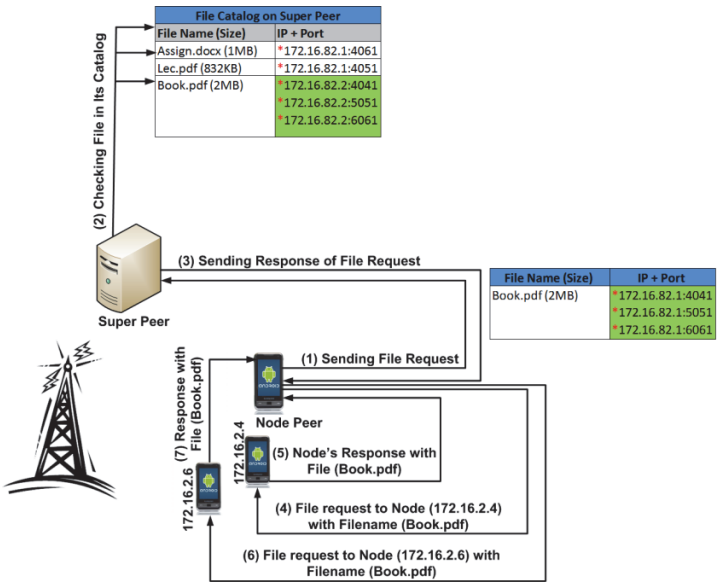


Fig. 9. Fault tolerance mechanism in file transfer.

4. EXPERIMENTS AND RESULTS

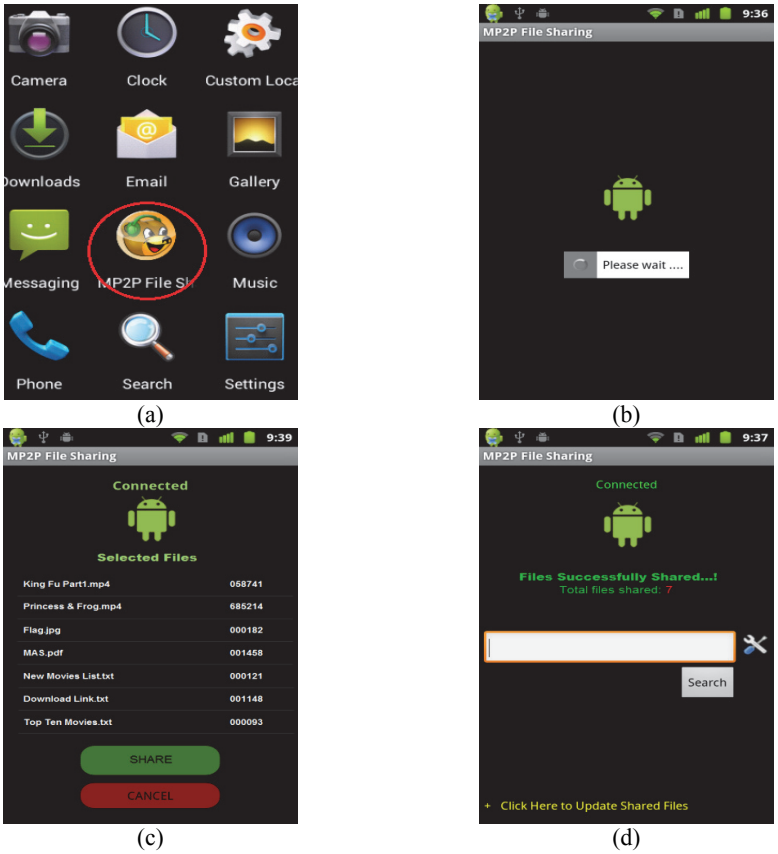
The proposed MP2P file sharing application is developed on Android version 2.2 (Froyo), which is compatible with all next versions of the Android OS. The simulator used for testing and debugging of the proposed application is Android Virtual Device (AVD) 2.2. The input parameters used in the simulations are shown in Table 1.

Table 1. Input parameters used in simulations.

Parameter	Value
Number of Node Peers	15
Number of Super Peers	5
Number of Shared files	80
Fault-Tolerance Ratio	5%

The following assumptions are made in the experimentation: The shared files are available on more than one NPs to test the fault tolerance mechanism. There is no network constraint such as bandwidth during the execution of these experiments. There are no faults in the network other than those that are induced in the tests to evaluate the results. The maximum file size for sharing is limited to 100MB.

Fig. 10 (a) shows the proposed application installed on the smartphone. If the user runs the application, it requests the registration server for the SP information and after receiving that information, it connects to that SP, as shown in Fig. 10 (b).



(a) (b) (c) (d)

Fig. 10. (a) Application screen; (b) Connecting to the SP; (c) List of files selected for sharing; (d) Shared files confirmation.

Once the connection is established, the main screen of the application is shown to the user. Here, either the user can search for a specific file or can share its files with the other users of the application by using the upload shared file(s) link available on the screen. If the user opts to share its files with the other users, a file selection screen is shown using which the user selects the file(s) he/she wants to share. Once the files are selected, a confirmation screen with all the selected files is shown, as seen in Fig. 10 (c). Once the user reviews the selection and clicks the share button, the information is sent to the SP, which updates its catalog accordingly and the confirmation message is shown to the user, as in Fig. 10 (d). If the user opts to search some file, he/she has to specify the file name in the text box, as shown in Figs. 11 (a) and (b). Once the user clicks the search button, the search request is sent to the SP, which checks its local catalog. If similar file(s) are found in the catalog, it returns the same to the requesting node; otherwise it contacts its neighboring SP and waits for the response. Once the SP receives the response from the neighboring SP, it forwards it to the requesting node. The list of similar files along with node IP address and file size is shown to the user, as seen in Fig. 11 (c). The user can download any file from the above result by simply clicking on the file name, as shown in Fig. 11 (d).

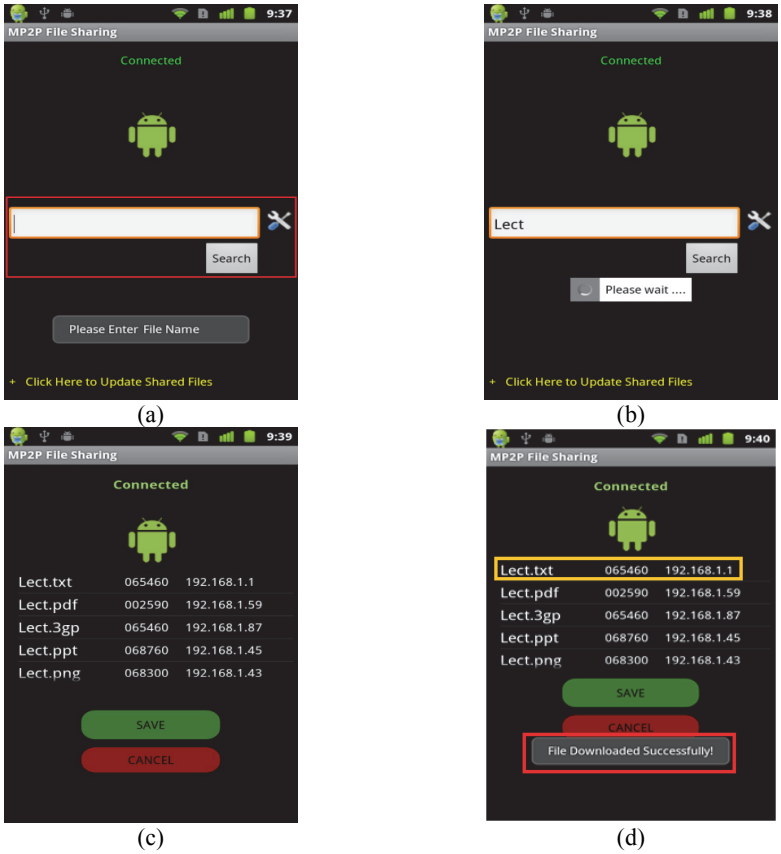


Fig. 11. (a) Search the file; (b) Searching similar files given as input; (c) List of similar files; (d) Selected file downloaded.

Fig. 12 shows the activity chart of MP2P file sharing framework with reference to three different NP (search file) requests. When the application starts on the user's smartphone, it takes around 1.21 seconds for the initialization of the system (*i.e.*, retrieving SP information from the RS and connecting with the SP). After initialization, if the user searches for some file, the request is forwarded to the concerned SP, which searches the requested file in its local catalog; and if the file is found, it returns the result. This operation takes 3.1 seconds (Fig. 12: case 1, first from bottom). If the SP does not find the requested file in its catalog, it performs the following tasks: (1) it forwards the request to three neighboring SPs, which return the result after checking their catalogs, and this takes 3.7 seconds (Fig. 12: case 2, middle NP request), (2) it forwards the request to five neighboring SPs and this activity takes 4.19 seconds to complete (Fig. 12: case 3, top NP request). Once the result is returned to the user, the file transfer step is started (we are assuming that the size of the file is same for all the three cases *i.e.*, 1 MB in this scenario). For the first case, it takes 45.6 seconds to complete the file transfer. For the second case, it takes 48.6 seconds, and for the third case, it takes 49.5 seconds to transfer the file, as shown in Fig. 12.

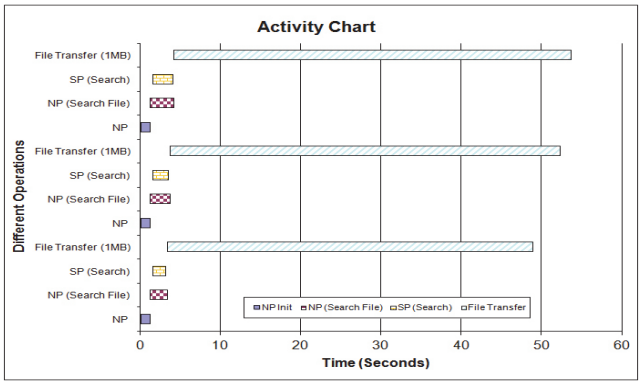


Fig. 12. Activity chart of MP2P file sharing framework.

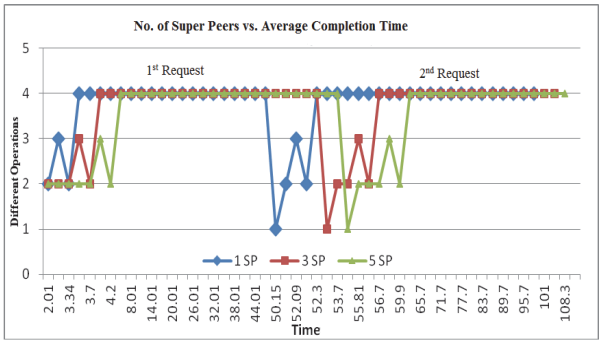


Fig. 13. Number of super peers vs. average completion time.

Fig. 13 shows the average completion time of different operations with respect to the number of SPs involved. Searching the user requested file on different number of neighboring SPs takes different times, as the user request is forwarded from one SP to the other for searching. Fig. 13 shows the comparison of different operations with reference to two user requests initiated one after the other. As we increase the number of SPs, the average completion time slightly increases. To avoid delay in responding to the user request, we have limited the search file request to a maximum of five SPs.

Fig. 14 shows the comparison of average file transfer time with reference to SP local transfer (*i.e.*, both the NPs, one that has the file and the other that wants the file, are connected to the same SP) and SP remote transfer (*i.e.*, both the NPs, one that has the file and the other that wants the file, are connected to different SPs). It is observed that the same network transfer takes less time as compared to the remote transfer. This is because in remote transfer, two different SPs are involved in the file transfer process, which causes delay, as shown in Fig. 14.

5. CONCLUSIONS

In order to overcome the limitations of existing file-sharing techniques for smart-phones, we have proposed a Mobile Peer-to-Peer (MP2P) file-sharing framework

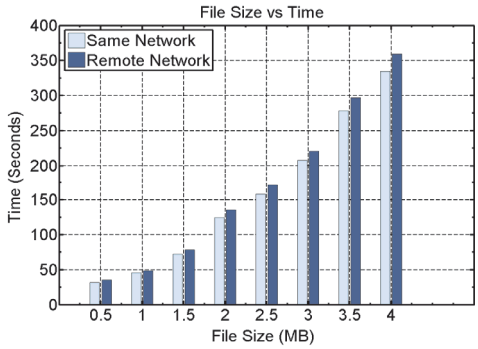


Fig. 14. File size vs. average file transfer time.

using 3rd Generation (3G) telecommunication network. In the proposed mechanism, we have two kinds of nodes *i.e.*, Node Peers (NPs) and Super Peers (SPs). The proposed solution has been successfully tested and the experimental results show efficiency of the proposed system. The proposed work can be extended in two directions: (1) Extending this framework to support pure MP2P, by implementing SP as a smartphone application (*i.e.*, mobile client), which means that NAT traversal will also be required for the SP. For this purpose, the Registration Server (RS), which also works as a STUN server in this case, can be assigned this responsibility. When a SP sends the registration request, the RS will send a series of packets to the SP to determine what type of NAT server is between the RS and the SP. RS can do this by observing the changes NAT servers make to the source address of the reply packet. Based on the nature of these changes, it can determine NAT server's type, address, and port that can be used to send packets to the SP from the NP. However, this task is very challenging and requires in-depth study of NAT servers. Furthermore, additional mechanism would be required to keep track of the SP. (2) Extending this framework to support on-the-go multimedia file streaming for non-supported streaming files (such as avi, mp4 *etc.*). This means that when the file is transferred between the NPs, the content of the file is shown to the user while it is being downloaded.

ACKNOWLEDGEMENT

The authors would like to extend their sincere appreciation to the Deanship of Scientific Research at King Saud University for its funding of this research through the Research Group Project no. RGP-214. The authors would also like to thank Mr. Asif Kamal, Mr. Muhammad Ashar, and Mr. Muhammad Faheem Amjad for their contributions in implementing the framework as part of their BS Final Year Project.

REFERENCES

1. J. Chiang, Y. Tseng, and W. Chen, "Interest-intended piece selection in bit torrent-like peer-to-peer file sharing systems," *Journal of Parallel and Distributed Computing*, Vol. 71, 2011, pp. 879-888.
2. A. Pasala and D. J. Ram, "FlexiFrag: A design pattern for flexible file sharing in distributed collaborative applications," *Journal of Systems Architecture*, Vol. 44,

1998, pp. 937-954.
3. S. Jung, U. Lee, A. Chang, D. Cho, and M. Gerla, "BlueTorrent: Cooperative content sharing for bluetooth users," *Pervasive and Mobile Computing*, Vol. 3, 2007, pp. 609-634.
4. L. Mekouar, Y. Iraqi, and R. Boutaba, "Peer-to-peer's most wanted: Malicious peers," *Computer Networks*, Vol. 50, 2006, pp. 545-562.
5. Y. Zhang and M. Schaar, "Peer-to-peer multimedia sharing based on social norms," *Image Communication*, Vol. 27, 2012, pp. 383-400.
6. M. Xu, S. Zhou, and J. Guan, "A new and effective hierarchical overlay structure for Peer-to-Peer networks," *Computer Communications*, Vol. 34, 2011, pp. 862-874.
7. F. S. Tsai, W. Han, J. Xu, and H. C. Chua, "Design and development of a mobile peer-to-peer social networking application," *Expert Systems with Applications*, Vol. 36, 2009, pp. 11077-11087.
8. W. Yang and S. Hwang, "iTravel: A recommender system in mobile peer-to-peer environment," *Journal of Systems and Software*, Vol. 86, 2013, pp. 12-20.
9. A. Chen and T. Ho, "Location aided mobile peer-to-peer system," *Pervasive and Mobile Computing*, Vol. 2, 2006, pp. 300-312.
10. C. Tseng, C. Lin, L. Yen, J. Liu, and C. Ho, "Can: A context-aware NAT traversal scheme," *Journal of Network and Computer Applications*, Vol. 36, 2013, pp. 1164-1173.
11. Y. Chen and W. Jia, "Challenge and solutions of NAT traversal for ubiquitous and pervasive applications on the Internet," *Journal of Systems and Software*, Vol. 82, 2009, pp. 1620-1626.
12. E. Knipp, B. Browne, W. Weaver, C. T. Baumrucker, L. Chaffin, J. Caesar, V. Osipov, and E. Danielyan, "Chapter 5 – Network address translation/port address translation," *Managing Cisco Network Security*, 2nd ed., 2002, pp. 233-272.
13. J. R. Vacca and S. R. Ellis, "Chapter 15: Network address translation deployment," *Firewalls: Jumpstart for Network and Systems Administrators*, 2005, pp. 249-268.
14. J. L. Eppinger, "TCP connections for P2P apps: A software approach to solving the NAT problem," Technical Report CMU-ISRI-05-104, Carnegie Mellon University, 2005.
15. N. M. Markovich and U. R. Krieger, "Statistical analysis and modeling of Skype VoIP flows," *Computer Communications*, Vol. 33, 2010, pp. S11-S21.
16. L. De Cicco, S. Mascolo, and V. Palmisano, "Skype video congestion control: An experimental investigation," *Computer Networks*, Vol. 55, 2011, pp. 558-571.
17. D. Rossi, M. Mellia, and M. Meo, "Understanding Skype signaling," *Computer Networks*, Vol. 53, 2009, pp. 130-140.
18. S. Sharafeddine and R. Maddah, "A lightweight adaptive compression scheme for energy-efficient mobile-to-mobile file sharing applications," *Journal of Network and Computer Applications*, Vol. 34, 2011, pp. 52-61.
19. Budiarto, S. Nishio, and M. Tsukamoto, "Data management issues in mobile and peer-to-peer environments," *Data and Knowledge Engineering*, Vol. 41, 2002, pp. 183-204.
20. J. O. Oberender, F. Andersen, H. de Meer, I. Dedinski, T. Hoßfeld, C. Kappler, A. Mäder, and K. Tutschku, "Enabling mobile peer-to-peer networking," *Wireless Systems and Mobility in Next Generation Internet*, LNCS, Vol. 3427, 2005, pp. 219-234.

21. K. Brown and S. Singh, "M-TCP: TCP for mobile cellular networks," *ACM Computer Communications Review*, Vol. 27, 1997, pp. 19-43.
22. J. Bistrom and V. Partanen, "Mobile P2P – Creating a mobile file-sharing environment," Technical Report Tik-111.590, Helsinki University of Technology, 2004
23. T. Horozov, A. Grama, V. Vasudevan, and S. Landis, "MOBY-A mobile peer-to-peer service and data network," in *Proceedings of International Conference on Parallel Processing*, 2002, pp. 1-8.
24. Jini Technology Surrogate Architecture Specification: https://svn.apache.org/repos/asf/river/jtsk/skunk/surrogate/refdocs/sa.pdf.
25. L. Li and X. Wang, "P2P file-sharing application on mobile phones based on SIP," in *Proceedings of the 4th International Conference on Innovations in Information Technology*, 2007, pp. 601-605.
26. M. Matuszewski, N. Beijar, J. Lehtinen, and T. Hyyryläinen, "Mobile peer-to-peer content sharing application," in *Proceedings of the 3rd IEEE Consumer Communications and Networking Conference*, 2006, pp. 1324-1325.
27. SymTorrent, Budapest University of Technology and Economics, Department of Automation and Applied Informatics: http://symtorrent.aut.bme.hu/.
28. R. Yrjö, "A peer-to-peer overlay architecture for mobile networks," Technical Report T110.7190, Helsinki University of Technology, 2005.
29. S. Li, W. Sun, and C. E, L. Shi, "A scheme of resource allocation and stability for peer-to-peer file-sharing networks," *International Journal of Applied Mathematics and Computer Science*, Vol. 26, 2016, pp. 707-719.
30. K. L. S. Sindhu and A. M. Priyadarshini, "Efficient file sharing using gossiping algorithm in mobile adhoc network," *International Journal of Control Theory and Applications*, Vol. 9, 2016, pp. 6335-6340.
31. B. Bakos, L. Farkas, and J. K. Nurminen, "P2P applications on smart phones using cellular communications," in *Proceedings of IEEE Wireless Communications and Networking Conference*, 2006, pp. 2222-2228.
32. A. N. A. Rahman, M. H. Habaebi, and M. Ismail, "Android-based P2P file sharing over ZigBee radios," in *Proceedings of the 5th International Conference on Intelligent Systems*, *Modelling and Simulation*, 2014, pp. 591-596.
33. C. E. Palazzi, A. Bujari, and E. Cervi, "P2P file sharing on mobile phones: Design and implementation of a prototype," in *Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology*, 2009, pp. 136-140.
34. R. Shah and Z. Narmawala, "Mobile torrent: Peer-to-peer file sharing in Android devices," *International Journal of Computer Science and Communication*, Vol. 7, 2016, pp. 20-34.

Farrukh Aslam Khan is an Associate Professor at the Center of Excellence in Information Assurance (CoEIA), King Saud University, Riyadh, Saudi Arabia. He did his MS in Computer System Engineering from GIK Institute of Engineering Sciences and Technology, Pakistan, and Ph.D. in Computer Engineering from Jeju National University, South Korea, in 2003 and 2007 respectively. His research interests include cyber security, mobile computing, and internet of things.

Umar Manzoor received the MS degree in Computer Science from National University of Computer and Emerging Sciences, and the Ph.D. degree in Multi-Agent Systems from the University of Salford, Manchester, UK, in 2005, and 2011, respectively. Currently, he is working at King Abdulaziz University, Jeddah, Saudi Arabia. His research interests include multi-agent systems, autonomous systems, and network management.

Azhar Khan received his BS degree in Computer Science from National University of Computer and Emerging Sciences, Islamabad, Pakistan, in 2011. His research interests include peer-to-peer networks, wireless networks, software engineering, and mobile computing.

Aftab Ali received his MS and Ph.D. degrees in Computer Science from National University of Computer and Emerging Sciences, Islamabad, Pakistan, in 2009 and 2015 respectively. He is currently working as a CBT security consultant at National Center for Assessment (NCA), Riyadh, Saudi Arabia. His research interests include wireless body area networks, key management, cyber physical Systems security, e-health, and cloud computing.

Haider Abbas is a Senior Member IEEE and Cyber Security Professional who took professional trainings and certifications from Massachusetts Institute of Technology (MIT), USA, Stockholm University, Sweden, IBM and EC-Council. He received his MS in Engineering and Management of Information Systems (2006) and Ph.D. in Information Security (2010) from KTH, Sweden. He is the principal advisor for several MS and Ph.D. students at King Saud University, National University of Sciences and Technology, and Florida Institute of Technology.

Maruf Pasha is currently serving as an Assistant Professor/ Head of Department of Information Technology at Bahauddin Zakariya University, Multan, Pakistan. He received his MS from National University of Sciences and Technology, Pakistan and Ph.D. from ENSIBS Engineering School, France. Pasha has authored several international publications and worked on various ICT related projects.