Device Fragmentation: A Case Study using "NeSen"

Rakesh Kumar Mishra, Rashmikiran Pandey, Sankhayan Choudhury and Nabendu Chaki

Abstract Remote and eHealthcare Systems are designed to provide healthcare solutions catering to wide variety of requirements ranging from highly personalized to domain-specific systems. Often, a smartphone is used as an aid to port data from embedded or external sensors to remote repository. A majority of smartphones are equipped with multiple network interfaces including provisions for dual subscriber identity modules (SIMs) and a variant of Android as the operating system. Android being an open source system allows customization by the vendor or chipset manufacturer. This raises a serious concern in terms of fragmentation—a form of portability issue with application deployment. For example, App developed on API 16 from MediaTek behaves or crashes over a phone of API 16 from QualComm. We have developed a mobile App called "NeSen" to assess the parameters of all prevalent networks in an area. NeSen uses only the standardized telephony framework and is tried over various smartphones from vendors including Samsung, HTC, LG, iBall, Lava, Micromax, Karbonn, Xiaomi, and Gionee having chipset from MediaTek, QualComm, SpreadTrum, and BroadComm. In this paper, using NeSen, we have conducted first ever evaluation of fragmentation in Android's basic framework. During experimental trails, several issues concerning device fragmentation are noted.

Keywords Fragmentation · NeSen · Android · Network parameters · RHM systems

R.K. Mishra (✉) · R. Pandey
Feroze Gandhi Institute of Engineering & Technology, Raebareli, Uttar Pradesh, India
e-mail: rakesh.mishra.rbl@gmail.com

R. Pandey
e-mail: rashmikiran@hotmail.com

S. Choudhury · N. Chaki
Deparment of Computer Science & Engineering, University of Calcutta, Kolkata, West Bengal, India
e-mail: sankhyan@gmail.com

N. Chaki
e-mail: nabendu@ieee.org

P.K. Sa et al. (eds.), *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*, Advances in Intelligent Systems and Computing 518,
DOI 10.1007/978-981-10-3373-5_41

1 Introduction

Android is an open source and customizable operating system allowing the manufactures to modify the core libraries of Android. These modifications in Android OS raise a serious concern in the form of fragmentation, i.e., inability of a code to exhibit the homogeneous behavior in different Android platforms. Android compatibility definition document (CDD) [1] provides certain standards and policies to avoid the fragmentation issues. CDD is able to control fragmentation to some extent but failed to evade completely. Manufactures are also twisting basic essence of CDD in several forms. This has been exposed while deploying NeSen among various smartphones from different vendors. Fragmentation is continually reported as a serious concern for the App development community [2].

1.1 TelephonyManager Framework

Android provides an informative manager class that supplies information about telephony-related operations and details on the device. An application interacts with the TelephonyManager framework of the Android. The TelephonyManager framework is direct reflection of native telephony manager in radio interface layer (RIL). There is a mapping between the application framework and native TelephonyManager. Native TelephonyManager opens connection with RIL daemon and extending the connection down to kernel drivers.

TelephonyManager Application Framework is supposed to make the platform-specific variations transparent to the overlying application. The native TelephonyManager is a platform-dependent component and parts of it will have to be adjusted to work with the potentially proprietary vendor radio interface layer (RIL). Figure 1 contains a graphical representation of the various blocks that compose the telephony component. The RIL interactions start right above the baseband, which is the firmware-specific platform to perform various cellular communication-related actions such as dialing numbers, hanging up calls, and accepting calls, and perform the callbacks thereupon. On the other side, the Android package com.android.internal.telephony contains various classes dealing with controlling the phone.

The android.telephony package contains the TelephonyManager class which provides details about the phone status and telephony-related data. TelephonyManager can be used to access phone properties and obtain phone network state information. PhoneStateListener—an event listener—may be attached to the manager enabling an application to get aware of when the phone gain and lose service, and when calls start, continue, or end [13].

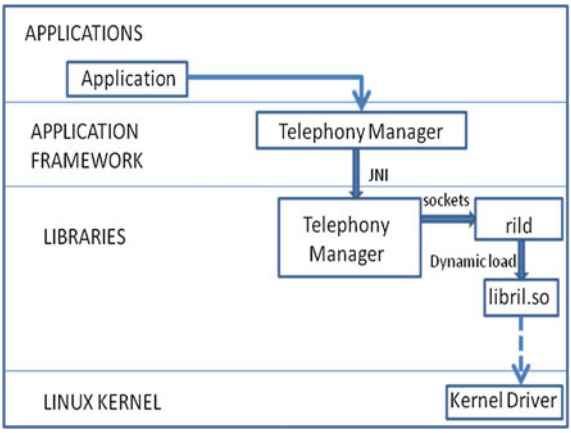


Fig. 1 Radio interface layer and TelephonyManager [3]

1.2 NeSen-The App

"NeSen" App [4] is developed to assess and record network parameters such as bit error rate (BER), signal strength, cell stability, and service connectivity. All above attributes are usually referred as performance quality parameters. NeSen is a service augmenting RHM (Remote Healthcare Monitoring) system with the capabilities of assured quality-based connection. The targeted service is supposed to harness the availability of best cellular network with dual SIM smartphones.

Vendors refer to the product marketing companies of the devices such as Samsung, Micromax, Sony, and Lava, while the manufactures refer to the companies which manufacture mobile chipsets such as MediaTek and QualComm. To further investigate the issues, another App is designed and deployed for introspecting telephony framework of each phone. The App is designed to reveal the information such as manufactures, vendor, Build version, model no, device id, and radio version. This information is extracted through Build class within the Android basic framework. It has been seen that different manufactures and vendor's ported their arbitrary information-defined formats. It is directly written in Android compatibility document [1] that such information has to be presented in homogeneous manner and is presently violated by the vendors/manufactures of mobile phone.

In one of the recent works [5], possible heterogeneities with the TelephonyManager framework of Android are discussed. Here, a first of a kind case study is presented to expose the fragmentation within the basic framework. Fragmentation is predominantly identified as API fragmentation under device fragmentation category. Lack of Google specification for dual SIM telephony framework is often considered as the major cause behind fragmentation. This has been handled by various techniques including introspection as illustrated in Table 1.

The paper is organized as follows: Related work is described in Sect. 2 and API Fragmentation manifested with NeSen is detailed in Sect. 3, while discussion with conclusions is presented in Sect. 4.

Table 1 Configuration of equipment

Manufacturer	API level	Approach for adaptation	Vendor of product	Remarks
MediaTek	16–20	SDK based	iBall, Lava, Gionee	BER is captured as −1 till API 17 and thereafter as 99
QualComm	15–16	Service call for instantiation and introspectation for invocation	Samsung, Karbonn, LG, Xiaomi	–
SpreadTrum	19	Introspection for both instantiation and invocation	Karbonn, Intex	–
BroadComm, QualComm	19	Introspection of fields for instantiation and invocation	Micromax, Xiaomi	Unusual approach but worked
QualComm	18	Service call for instantiation and introspection for invocation with different names	HTC	–

2 Related Work

A learning-based energy-efficient approach is implemented using Android App in [6] for network selection. The algorithm primarily focuses over lower power consumption and high quality of services using parameters such as network availability, signal strength of available networks, data size, residual battery life, velocity, location of users, and type of application. Battery life, location, and application type are used to determine optimal performing network based on certain predefined rules. App-based monitoring of the network is implemented in [7]. This reports the network parameters perceived by the user equipment to the network-side entity for QoE assessment.

Authors in [8] have detailed the causes of fragmentation. Two vendors, i.e., HTC and Motorola, have been chosen here to analyze the bug reports. Bug reports submitted by Android users help to identify fragmentation. Topic analysis methods have been used to extract the set of topics from the bug reports. Two topic analysis methods opted for fragmentation analysis are Labeled Latent Drichlet Allocation (Labeled LDA) and LDA. Topics extracted from the Labeled LDA and LDA are compared to find out the unique bug report topics, and these topics manifest the fragmentation.

Park et al. [9] proposed two methods to detect the fragmentation problem in Android. The methods used are code-level check and device-level test. Code-level checking method analyzes the code and finds out the part of code where the fragmentation occurs by converting code into itemized values. The itemized values are then mapped on predefined set of rules to correct the code accordingly. Code-level check

methods are generally dealt with the hardware fragmentation. Other method is used to analyze the fragmentation at API level. The method collects the test results of APIs and store in the database along with the functions of APIs. Then, these two methods compare the API's functions used by developer with the corresponding functions stored in the database and find the fragmentation.

In [10], a behavior-based portability analysis methodology is proposed to overcome the problems of portability issues in Android. The methodology lets the developer to extract the ideal behavior of application to compare it with similarity in application flows. The entire analysis includes behavior extraction, test execution, and log analysis to identify ideal pattern of App operation flow.

In another work [11], focus is on the change and fault proneness of underlying API. Two case studies are planned: First case study orients toward the change of user ratings in Google Play Store, while the second study emphasizes on experience and problems faced by Android developers and their feedbacks affecting the user ratings. Among the techniques discussed so far, work in [9] is very close and seems to be appropriate for identifying and locating the fragmentation issues of NeSen, but the technique relies on the list-based approach. In case of NeSen, this is not possible because the TelephonyManager is exhibiting valid results as expected from the single SIM phone telephony framework. There is no specification at Google repository for the dual SIM telephony framework. Thus, a list for mapping cannot be prepared. Further, customizations from manufacturers resulted in non-standardized nomenclature of the public interfaces. Hence, a generic list cannot also be prepared even after introspection. A case study-based approach using NeSen for fragmentation with telephony framework is conducted below.

3 Device Fragmentations: API Fragmentation with NeSen

Android is an open source readily available Linux-based operating system. On the one hand, openness of Android allows customizing the framework, keeping the basic library intact; on the other hand, a vendor opts Android to cut down the cost of the products.

NeSen [4] is an Android-based tool incorporated at the Mobile Terminal (MT) which is capable of assessing both static and dynamic network parameters of the smartphone. The real-time values of these parameters are logged at the file system of the MT. The customization from the manufactures of the different vendors posed NeSen a serious challenge of portability among the smartphones from different vendors as well as to different APIs of the same vendors. Fragmentation is the one of the major reason why Android Apps misbehaves and shows inconsistency in the functioning [8].

This flexibility to customize the APIs results into differences within existent framework from different manufacturer as well as platforms from same manufacturer with different APIs. The fragmentation is categorized in two types, viz. operating system (OS) fragmentation and device fragmentation [9]. Device-level

fragmentation happens because of the difference in design of underlying hardware of phones as well as customization of the APIs by the manufacturers. Device-level fragmentation is further classified into hardware-level fragmentation and API-level fragmentation. NeSen exhibited device-level fragmentation, particularly API-level fragmentation.

Initially, NeSen was tried on Samsung and Karbonn smartphones with QualComm chipset and Android API 15/16. The objects were instantiated through service instance invocation method with service name as parameter. The telephony objects thus created were given independent state listeners, and the data is logged in different files storing values of signal strength received, bit error rate, cell identity, location area code, etc. The list of phones tested with the first version is in Table 2.

NeSen is tested over different vendors and manufacturer of smartphone and has the manifestation of fragmentation. Fragmentation is observed during testing of App over several manufactures as well as API levels. Figure 2 illustrates the success and failure experience with NeSen's basic version over different phones. NeSen's initial version is tested with API 15–21, phones of 11 vendors and 4 manufacturers. NeSen is successfully installed over some phones, while on others, it is either completely failed or only GUI appeared. NeSen's reasons for the unsuccessful run or failed installation are identified as object instantiation failure, method invocation

Table 2 List of vendors and manufactures used for NeSen

Chipset manufacturer	Example
MediaTek	iBall, Lava, Gionee
QualComm	Samsung, Karbonn, LG, Xiaomi
SpreadTrum	Karbonn, Intex
BroadComm	Micromax, Xiaomi

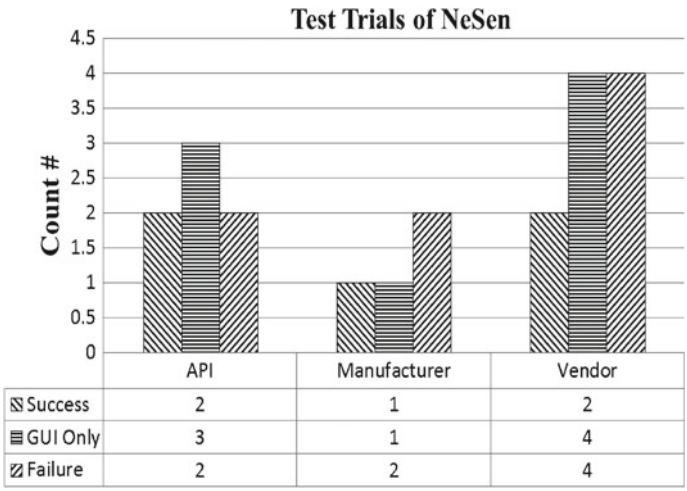
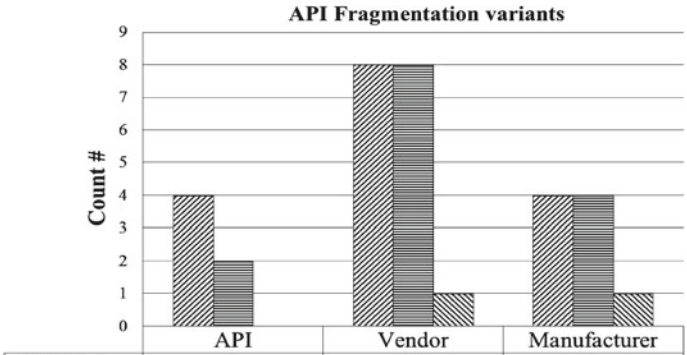


Fig. 2 Fragmentation posed to NeSen



	API	Vendor	Manufacturer
Obj Inst	4	8	4
Method Inv	2	8	4
Event Error	0	1	1

Fig. 3 Different types of fragmentation seen with NeSen

failure and event error. Certain worth mentioning failures of NeSen with different manufacturers are on account of change in the API level which elucidates in Fig. 3.

Methodology opted for the purpose was to perform introspection for each smartphone's library framework. Introspection of phone telephony framework reveals several challenges for NeSen with respect to object instantiation, method invocation, and listeners assignment.

3.1 Library Manifestation

Resources claimed that there is a gap in the Android library specifications and underlined hardware of the smartphones, hence raising the serious compatibility issues [12]. Hardware enhancements usually do not have direct support from standard library, thereby exposing the scope for the customization of Android platform by the manufacturer.

Similar instance has been observed with NeSen over several smartphones platform. Generally, as per the standard specification of Android, TelephonyManager class can be instantiated by system service invocation method with service kind literal as a parameter. This does not work for NeSen trails beyond basic platform. Meanwhile, the crucial points observed are like even a class of Android. That is, TelephonyManager has its four variants. These four variants of the class implied to their corresponding smartphone manufactures. These are being used by the different manufactures as listed in Table 3. Library has been incorporated into the framework in four different variants by the manufactures.

Table 3 TelephonyManager class variants

Chipset manufacturer	TelephonyManager class
MediaTek	TelephonyManagerEx (through SDK)
QualComm	TelephonyManager, MSimTelephonyManager, MultiSimTelephonyManager
SpreadTrum	TelephonyManager
BroadComm	TelephonyManager

3.2 Object Instantiation

Instantiation of object for NeSen require several specific ways as per their library and underlined hardware. Each manufactures force to dig out its way of object instantiation. This process makes the App development more chaotic as well as time taken. In case of NeSen, there are five different ways of instantiation were identified where each is unique to the specific API and manufacture. These are listed as in Table 4. A few of the manufacturers such as MediaTek provided their own SDK (Software Development Kit) framework which is freely available to be integrated with IDE, while the rest of the manufactures as well as vendors does not provide any such SDK. Standard Android framework library documentation is not sufficient to explore the dedicated framework libraries from different manufactures. The identification of a mechanism for instantiation of objects, for different manufactures except MediaTek, was really a thought-provoking, time-taking, and investigative task. Each smartphones' library needs to be introspected for the purpose, and each alternative is to be explored for method invocations and listener binding.

Table 4 Object instantiation mechanism

Chipset manufacturer	Instantiation mechanism
MediaTek	getSystemServices()
QualComm	TelephonyManager, static method from(), static fields instance instantiation through introspection
SpreadTrum	Instantiation through introspection
BroadComm	TelephonyManager

3.3 Method Invocation

All the smartphones enlisted here are explored deeply with the help of introspection to investigate the methods of particular class. Generally, all the required methods are present in the library, but they are also overridden or renamed in some forms. NeSen study reveals that though all the methods are present still, we need to dig out them as per the requirement. Methods are renamed by extending the standard method name of with word such as "Gemini." For example, getNetworkOperator() is renamed as getNetworkOperatorGemini(), etc. It is also observed that in some cases, few of the methods generate ambiguous or erroneous results which cannot be directly applied for NeSen. Another way to invoke such methods is through introspection of the library for the corresponding method variants and identifying the suitable candidates for the purpose.

3.4 Listener Assignment

Listener object is bound to the class object to listen the events and reciprocate with action. The required listener binding method for the NeSen was found in the library in three different variants with no particular trends of APIs and manufactures. An introspection of the library is required to choose the corresponding method for the listener accordingly. The listener binding method is found to have changed name and/or parameters; e.g., the listen method in standard framework has PhoneStateListener and an integer as parameter. The original version is sometime complimented with listenGemini method with one additional integer parameter in some libraries. In other cases, the name remains the same, but signature becomes like that of listenGemini.

3.5 Data Logs and Event Triggers

During data recording by NeSen, it was observed that BER reported by event listener in smartphone manufactured by MediaTek was incorrect as that was not similar to values in other smartphones. Parameters taken into account for comparison are operator name, time and LAC were same. MediaTek SDK is providing suspicious value for BER, while all others are as per expectation during comparison. Valid values for the BER are [0–7, 99] as in [13]. BER of -1 is reported when phone is in "out-of-service" state. One of the Sony smartphones deployed with NeSen is failed to provide CID/LAC for second interface.

Its exclusive problem observed with MediaTek smartphones, wherein the null object is being thrown by the event monitor for the listener. The problem is removed when the empty SIM slot of the smartphone is provided with the SIM. This problem

is not manifested with the smartphones of the other manufactures such as QualComm, SpreadTrum, and BroadComm. In these smartphones, events for individual interfaces are generated mutually independently. This affects cost in terms of efforts to understand and debug the problem.

4 Conclusions

Generally, most of the RHM systems require some network access and exploit all possible communication interfaces for extending communication capability with assured reliability of the data communication. NeSen is developed for this very purpose. Majority, i.e., 86%, of Android developer are considering fragmentation for rise in the cost of development [14]. Google initiatives such as Android Source Code, Compatibility Definition Code, and Compatibility Test Suite are existing to tackle the issue; however, such initiatives are far away from resolving the fragmentation problems of the domain [9]. Android compatibility document classifies the customization restrictions into MUST, REQUIRED, OPTIONAL, SHALL, SHOULD, and RECOMMENDED for ensuring the compatibility [1]. During the course of tackling the deployment issues with NeSen, we have encountered issues such as erratic exhibition of the BER value by different phone of same vendors, cell_id undetected in Sony phone C2004, and different hardware and device information by two phones with SpreadTrum chips.

In our endeavor to resolve the reported fragmentation, another tool called "Intros" reveals the platform information using the Build class. The App itself faces the challenges of fragmentation. This is an opinion that fragmentation is appearing on the account of the lack of specification for API by Google itself for multi-SIM phones and strict binding over the semantic meaning of information from vendors and manufactures of the phones.

References

1. Android Compatibility Definition Document: http://static.googleusercontent.com/media/source.android.com/en//compatibility/android-cdd.pdf. (Accessed on Oct, 2015).
2. Malavolta, I., Ruberto, S., Soru T., Teragani, V.: Hybrid Mobile Apps in Google play Store: An Exploratory Investigation. In: 2nd ACM International conference on Mobile Software Engineering (MOBILESoft). pp. 56–59 (2015).
3. http://www.nextinnovation.org/doku.php?id=android_ril. (Accessed on Jan, 2016).
4. Mishra, R. K., Pandey, R., Chaki, N., Choudhury, S.: "NeSen" -a tool for measuring link quality and stability of heterogenous cellular network. In: IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS). pp. 1–6. IEEE (2015).
5. http://www.ltfe.org/objave/mobile-network-measurements-using-android/. (Accessed on Jan, 2016).

6. Abbas, N., Taleb, S., Hajj, H., Dawy, Z.: A learning-based approach for network selection in WLAN/3G heterogeneous network. In: Third International Conference on Communications and Information Technology (ICCIT). pp. 309–313. IEEE (2013).
7. Poncela, J., Gomez, G., Hierrezuelo, A., Lopez-Martinez, F. J., Aamir, M.: Quality assessment in 3G/4G wireless networks. In: Wireless Personal Communications, 76(3), pp. 363–377 (2014).
8. Han, D., Zhang, C., Fan, X., Hindle, A., Wong, K., Stroulia, E.: Understanding android fragmentation with topic analysis of vendor-specific bugs. In: 19th Working Conference on Reverse Engineering (WCRE). pp. 83–92. IEEE (2012).
9. Park, J. H., Park, Y. B., Ham, H. K.: Fragmentation Problem in Android. In: International Conference on Information Science and Applications (ICISA). pp. 1–2 (2013).
10. Shin, W., Park, D. H., Kim, T. W., Chang, C. H.: Behavior-based portability analysis methodology for Android applications. In: 5th IEEE International Conference on Software Engineering and Service Science (ICSESS). pp. 714–717. IEEE (2014).
11. Bavota G, Linares Vasquez M: The Impact of API Change- and Fault-Proneness on the User Ratings of Android Apps. In: IEEE Transaction on Software Engineering. vol. 41(4). pp. 384–407 (2015).
12. SushrutPadhye: https://dzone.com/articles/major-drawbacks-android. (Accessed on Sep, 2015).
13. Signal Strength in Android Developer: http://developer.android.com/reference/android/telephony/SignalStrength.html. (Accessed on Sep, 2015).
14. W. Powers: Q1’11 - Do you view Android Fragmentation as a Problem? Baird Research (2011).
15. http://marek.piasecki.staff.iiar.pwr.wroc.pl/dydaktyka/mc_2014/readings/Chapter_7_Telephony_API.pdf. (Accessed on Jan, 2016).