Providing Diagnostic Network Feedback to End Users on Smartphones

Xu Ye

Department of Computer Science
University of Massachusetts of Lowell
Lowell, Massachusetts 01854
Email: xu_ye@student.uml.edu

Bo Yan

Department of Computer Science
University of Massachusetts of Lowell
Lowell, Massachusetts 01854
Email: marshimaro9913@gmail.com

Guanling Chen

Department of Computer Science
University of Massachusetts of Lowell
Lowell, Massachusetts 01854
Email: glchen@cs.uml.edu

Abstract—Despite the continued technology advances for smartphones, there are still few tools available to end users for understanding the causes of the network problems when encountered. The users often have to rely on the signal bars, which unfortunately is a poor indicator of real network issues that may arise even when the signal is good.

In this paper, we present a network troubleshooting system for Android platform, which can automatically collect network measurements and perform on-device diagnosis of likely causes of the network problems. The diagnosis model is based on a decision tree that was constructed with domain knowledge and the model's thresholds were learned statistically from empirical performance analysis. Experimental results show that the proposed approach is effective in troubleshooting network problems on smartphones. By continuously using this app, end users can make informed decisions to upgrade service plans, change mobile devices, or switch service providers.

I. INTRODUCTION

In recent years, the mobile communication industry has experienced extraordinary changes brought by new wireless technologies and huge market demands. The ubiquitously available wireless networks boost work productivity and increase user mobility. The rapid increase in capacity and real-world deployment, however, challenge mobile users with performance problems. These problems are difficult for network operators to troubleshoot the root causes manually because the manifesting anomalies from network measurements are highly interrelated. Thus fault diagnosis for such large-scale wireless networks can be an extremely challenging task.

The state-of-the-art operational approaches address this problem by capturing packet traffic or conducting active probing tests, which are then interpreted through statistical analysis and domain knowledge. This method is known as *system-level diagnosis* [16], which unfortunately does not necessarily reflect the end-users' network experience on the edge. Therefore, service providers also provide technical support service, as *customer-level diagnosis*, to detect and troubleshoot problems from the customer's perspective. When a customer calls the technical help line, the customer agent usually asks her to go through a series of routine diagnosis steps such as checking signal coverage, enabling data connection, or rebooting her mobile device. Such an interview serves as a preliminary to identify the root causes of the network problems [11].

Instead of having end users make a help call, we propose to build an app that installs on the mobile devices that can make client-side network measurements and provide diagnostic feedback on the causes of network problems. There are several benefits of this approach: 1) it does not require additional infrastructure investments; 2) it can provide network measurements on the edge that are hard to obtain for management systems in the network core; 3) it has potential to scale cheaply as more end users install the app; and 4) end users can get direct feedback on the potential cause of the network issues, even when the network may not be available.

Some operators have started to offer mobile apps that allow their customers to report problems by simply pushing a button, while the app gathers network measurement automatically and submit to the operator for future analysis, such as identifying emerging network issues [10]. While there are also several other apps that can conduct network measurements [2], [18], [19], none of them provide on-device troubleshooting and diagnostic feedback to the users.

In this paper, we introduce *MobiDiag*,¹ a light-weight network troubleshooting app for Android platform. Unlike other apps' end-to-end path measurements, *MobiDiag* focuses on performance measurements of the operator's network by conducting Ping-based tests from the mobile device to the first responsive router. By computing several key performance indicators through these tests, *MobiDiag* employs a decision-tree diagnosis model to determine the likely causes of the network problems, such as low network capacity, insufficient signal coverage, cellular tower handover, and user mobility. The diagnosis model was constructed through empirical analysis and its thresholds were learned statistically through the measurements.

Note that *MobiDiag* uses on-device diagnostic engine and thus can work offline while the network is not available, so the end user can continue receiving diagnostic feedback. The measurements will be opportunistically uploaded to a server when the network becomes available. The server will periodically re-evaluate the decision tree model using all received measurements, and send the new model to the connected devices when necessary.

¹<https://play.google.com/store/apps/details?id=swin.diagnostic/>

The rest of this paper is organized as follows. In Section 2, we present background information and system design. We conduct empirical performance analysis in Section 3 to inform the development of the decision tree diagnosis model, which is described and evaluated in Section 4. In Section 5, we introduce the implementation of *MobiDiag*. We finish by discussing related work in Section 6 and concluding in Section 7.

II. BACKGROUND AND SYSTEM DESIGN

In the context of cellular networks, it is often cost prohibitive to install and maintain a monitoring infrastructure on a large scale and geographically distributed wireless network. NETAWIN [17] attempts to measure a production 3G network by passively sniffing packets on backbone networks. However, according to 3GPP technical specifications,² hundreds of performance measurements can be collected as performance indicators to track and analyze operational status of a cellular network, which is impossible for the operators to interpret such a large number of heterogeneous observables. In practice, the operators only focus on a few *key performance indicators* (KPI) for fault detection. Unfortunately, this is still an expensive approach and the network insights are often not available to the client devices.

On the other hand, the consumers typically do not require to know the exact details of network problems in technical terms. Rather, they often just want to get a rough idea of whether the perceived network performance degradation is caused by coverage, mobility, interference, the provider's network, or Internet problems. Modern mobile devices, such as smartphones, already have the capability to perform client-side network measurements that can provide diagnostic insights directly to the end users. When appropriate, the operators can also take advantage of these client-side measurements as inputs to their network management systems, treating the clients as *crowd-sourced* distributed monitoring points.

On mobile devices, typical end-user applications are grouped into four QoS traffic classes in 3GPP TS 23.107,³ based on four different communication schemes, as shown in Table I. The main distinguishing factor among these classes is traffic delay sensitivity. The conversational class is meant for traffic with high delay sensitivity, whereas the background class is not. The expected QoS requirements of typical end-user applications are specified in 3GPP TS 22.105.⁴

From an end-user's perspective, performance should be expressed by parameters which focus on user-perceivable effects, rather than their causes within networks [1]. At user level, *latency* evaluates the time cost from establishing a particular service from initial user request to the time of receiving specific information once the service is established. When latency varies, it causes issues with real-time services which carry encapsulated sequential data bytes. The effect of latency variation on user experience is generally measured by *jitter* quantitatively. When transmission error occurs, *packet*

TABLE I
QoS TRAFFIC CLASSES OF MOBILE APPLICATIONS DEFINED IN 3GPP TS 23.107

Traffic Class	Fundamental characteristics	Application example
Background	<ul style="list-style-type: none"> • Destination is not expecting the data within a certain time • Preserves data integrity 	background download of emails
Interactive	<ul style="list-style-type: none"> • Request/response pattern • Preserves data integrity 	web browsing, games
Streaming	<ul style="list-style-type: none"> • Preserves time relation (variation) between information entities of the stream 	streaming multimedia
Conversational	<ul style="list-style-type: none"> • Preserves time relation (variation) between information entities of the stream • Conversational pattern (stringent and low delay) 	voice, videotelephony, video games

loss evaluates its direct effect on QoS presented to users. Thus, we use latency, jitter, and packet loss as our metrics to characterize network performance and understand user experience on cellular networks.

We may simulate customers' experience by analyzing the services they are using. Different from infrastructure-based monitoring, we directly measure the performance on devices that consumers really use with Internet services provided by wireless ISPs or cellular carriers. This helps us understand the customer side factors and their impact on application-level performance which customers do matter.

Radio Access Networks (RANs) are responsible for managing service access from cellular subscribers while Core Networks (CNs) provide both packet routing and traffic management, communicating with the Internet. We only focus on performance issues occurring inside cellular network. We perform Ping-based active probing towards the gateway to expect approximately replicating the cellular network condition.

As many users are only interested in comparing one service provider with another at particular locations, such as where they live and work, we focus on measuring the performance of the provider's network at those locations. This can be approximated by conducting Ping-based active probing towards the mobile device's gateway. This is better than other end-to-end approaches, such as Tulip [15], which measure per-path characteristics to destination and may cost non-trivial data service charge, especially when the customers do not have an unlimited data plan.

Note some wireless operators may disable Ping replies for security reasons. We thus have to take a traceroute-like approach, by iteratively incrementing the TTL in the Ping packets, to discover the first responsive router, which is often the first router outside of the operator's network if it is not the mobile device's gateway. We then conduct Ping-based tests to that responsive router to measure the service provider's network performance.

Figure 1 shows implementation of the traceroute-like ap-

²<http://www.3gpp.org/ftp/Specs/html-info/32814.htm/>

³<http://www.3gpp.org/ftp/Specs/html-info/23107.htm>

⁴<http://www.3gpp.org/ftp/Specs/html-info/22105.htm>

```

$ ping -s 8 -n -W 4 -c 1 -t 2 www.google.com
PING www.l.google.com (173.194.73.99) 8(36) bytes of data.
From 10.170.210.133 icmp_seq=1 Time to live exceeded

--- www.l.google.com ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet
loss, time 0ms

$ ping -s 8 -n -q -w 5 10.170.210.133
PING 10.170.210.133 (10.170.210.133) 8(36) bytes of data.

--- 10.170.210.133 ping statistics ---
5 packets transmitted, 4 received, 20% packet loss, time
4007ms
rtt min/avg/max/mdev = 35.602/46.050/64.509/11.748 ms

```

Fig. 1. Finding first responsive router to the Internet through traceroute-like approach using Ping command

```

$ ping -s 1472 -f -q -w 10 10.170.210.133
PING 10.170.210.133 (10.170.210.133) 1472(1500) bytes of
data.

--- 10.170.210.133 ping statistics ---
699 packets transmitted, 683 received, 2% packet loss,
time 9983ms
rtt min/avg/max/mdev = 37.539/509.999/2082.661/660.900 ms,
pipe 170, ipg/ewma 14.302/58.745 ms

```

Fig. 2. An example of estimating the performance of UMTS/HSDPA networks by flooding Ping echoes

proach to find first responsive router using Ping. We start to ping a destination such as `www.google.com` by setting time-to-live (TTL) value to 1 since gateway is one hop away from client and is expected to be the first responsive router. The IP address of the one hop away host is returned in a warning when TTL is exceeded. We then check the reachability of this host using Ping with default TTL value. If it is reachable, the host is identified as the first responsive router for estimating network performance. Otherwise, we repeat these two steps by incrementing TTL value until the first responsive router is found.

Round-trip measurements can be estimated by flooding Ping echoes without interval. Figure 2 gives the result of a Ping test with a specific packet size 1500 bytes in UMTS/HSDPA network. Within 10 seconds, 699 packets were sent and 683 packets were received, which means a 820 Kbps round-trip throughput with 2% packet loss. It may take only a few seconds to download a typical mobile application, but a long time to copy a large file from server. The average round-trip latency 510 milliseconds indicates a slight delay when surfing on the web, while an average deviation 660 milliseconds gives customers a worse experience on both VoIP calling and online-gaming. As we discussed before, these conversational applications often carry encapsulated sequential data bytes, packet disordering caused by latency deviation of 660 milliseconds is hard to be removed or even reduced by buffering.

Figure 3 suggests when we repeat the same test in Wireless LANs, a high throughput of 12.2 Mbps without packet loss

```

$ ping -s 1472 -f -q -w 10 10.10.10.1
PING 10.10.10.1 (10.10.10.1) 1472(1500) bytes of data.

--- 10.10.10.1 ping statistics ---
5113 packets transmitted, 5112 received, 0% packet loss,
time 9999ms
rtt min/avg/max/mdev = 1.334/1.919/14.907/1.982 ms, pipe
2, ipg/ewma 1.956/1.924 ms

```

Fig. 3. An example of estimating the performance of Wireless LANs by flooding Ping echoes

```

$ ping -n -s 1472 -w 5 -i 0.5 10.170.210.133
PING 10.170.210.133 (10.170.210.133) 1472(1500) bytes
of data.
1480 bytes from 10.170.210.133: icmp_seq=1 ttl=254
time=88.8 ms
1480 bytes from 10.170.210.133: icmp_seq=2 ttl=254
time=97.4 ms
1480 bytes from 10.170.210.133: icmp_seq=3 ttl=254
time=96.1 ms
1480 bytes from 10.170.210.133: icmp_seq=4 ttl=254
time=84.8 ms
1480 bytes from 10.170.210.133: icmp_seq=5 ttl=254
time=93.7 ms
1480 bytes from 10.170.210.133: icmp_seq=6 ttl=254
time=102 ms
1480 bytes from 10.170.210.133: icmp_seq=7 ttl=254
time=91.6 ms
1480 bytes from 10.170.210.133: icmp_seq=8 ttl=254
time=90.4 ms
1480 bytes from 10.170.210.133: icmp_seq=9 ttl=254
time=88.9 ms
1480 bytes from 10.170.210.133: icmp_seq=10 ttl=254
time=78.1 ms

--- 10.170.210.133 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss,
time 4511ms
rtt min/avg/max/mdev = 78.126/91.237/102.132/6.424 ms

```

Fig. 4. Performance estimation using Ping with a fixed interval

allows customers to download a video quickly. The relative low latency and variation makes it easier to start a group video meeting, compared to that of cellular networks.

We compare our approach with Speedtest⁵ and FCC Mobile Broadband Test⁶ which are available in Android Market. Table II suggests the result differs widely from the result of Speedtest and FCC Mobile Broadband Test. This is because Ping-based bandwidth estimation approach has a fundamental disadvantage that it confuses properties of the forward and reverse paths. The asymmetry of downlink and uplink paths in cellular networks makes it even harder to separately estimate bandwidth of both downlink and uplink using Ping-based approach.

However, round-trip time latency has no such disadvantage since a single Ping packet is relative small compared to a network flow. Alternatively, we replace flooding Ping echoes by performing Ping echoes with a large and fixed rate as shown in Figure 4. Result in Table II shows that T-Mobile USA UMTS/HSDPA network has a round-trip time latency of

⁵<http://www.speedtest.net/>

⁶<http://www.broadband.gov/qualitytest/>

TABLE II
THE COMPARISON OF BANDWIDTH ESTIMATION APPROACHES

Method	Latency	Bandwidth (Down/Up)
Ping Flooding	510 ms	820 Kbps
Ping with Fixed Rate	91ms	132 Kbps
Speedtest	102 ms	4.1/1.8 Mbps
FCC Mobile Test	97 ms	1.89/0.51 Mbps

91 ms, which is almost the same as that of Speedtest and FCC Mobile Broadband Test. Since most mobile applications do not require downloading a large file, bandwidth estimation is not important for mobile applications which are delay sensitive.

While technically we could implement Ping flooding to the first responsive router to estimate throughput, the estimation is often not accurate and the throughput is not a key performance metric for mobile applications as suggested by Table I. We thus simply conduct fixed-interval Ping tests to calculate round-trip time (RTT), jitter, and packet loss as the indicators of the network condition.

We have implemented a prototype app, *MobiDiag*, on Android platform. When the user has experienced network problems, she opens *MobiDiag* User Interface and asks for diagnostic feedback. The Performance Tester then conducts Ping tests to the first responsive router. The results are used to calculate above mentioned metrics, which are combined together with other contexts (such as device movement), to provide an explanation to the user. The network measurements and the context data during this test, called a ticket, is then passed to Ticket Uploader which opportunistically uploads tickets to our server when the network is available. The tickets will be analyzed offline to adjust the diagnosis models when necessary, which is described next.

III. EMPIRICAL PERFORMANCE ANALYSIS

We studied user experience tickets collected from *MobiDiag* and analyzed how the received signal strength (RSS), connection type, handover, and user movement affect the application-level performance. Based on this analysis, we integrated domain knowledge with statistical learning techniques to build a diagnosis model, to troubleshoot why the network performance is not satisfactory. Here we discuss empirical performance analysis that will inform the design of the diagnosis models.

A. Effects of Signal Strength

First, we studied the influence of the RSS on the round-trip time (RTT), jitter and packet loss of Ping-based tests from a Samsung Nexus S smartphone with T-Mobile USA, which supports both EDGE and UMTS/HSDPA networks. Over the four month test period, we obtained 4,000 experience tickets from 49 unique cellular towers.

The RTT was less than 1 second in 90% of all tickets, while the RSS varied from -76 dBm to -113 dBm. The jitter and packet loss, however, are still considerably large for streaming and conversational applications compared to those in wireless LANs. Jitter varied greatly from 3 ms to 984 ms and was less

than 10 ms in only 47% of all tickets, which is not sufficient for VoIP apps such as Skype. When the RSS was less than -103 dBm, jitter was larger than 10 ms in more than 95% of all tickets. Even if we used a buffer of 50 ms, a large jitter of 10 ms still existed in 70% of all tickets. Similarly, when the RSS degrades, the increasing transmission failure and consequent packet loss make network performance worse. There was at least 1 transmission failure in 30% of all tickets in which the RSS dropped below -103 dBm, but only in 3% of all tickets when the RSS is good.

B. Effects of Network Types

T-Mobile USA supports data access through both EDGE and UMTS/HSDPA connections. It is known that, UMTS provides an Internet speed from 384 kbps to 14.4 Mbps with its HSDPA extension, which is faster and more reliable than EDGE with typical speed of approximately 75 kbps to 135 kbps. The network latency also has been significantly improved. We thus split all experience tickets into two groups, *EDGE tickets* (all tickets in EDGE networks) and *UMTS/HSDPA tickets* (all tickets in UMTS/HSDPA networks).

The RTT was less than 400 ms in only 3% of all EDGE tickets, which is not good enough for most networked apps, but in 80% of all UMTS/HSDPA tickets. It is observable that the RTT has a distribution mostly around 100 ms when the RSS was greater than -103 dBm, which reflects the high-speed performance of UMTS/HSDPA connection. However, when the RSS was worse than -103 dBm, the RTT was larger than 200 ms in almost all UMTS/HSDPA tickets.

Jitter can be reduced to be less than 10 ms in more than 95% of all UMTS/HSDPA tickets with the help of jitter buffering. The QoS of both conversational and streaming apps can be guaranteed easily. When the RSS dropped below -103 dBm or the connection switched to EDGE, jitter was mostly greater than 50 ms, which was hard to smooth even with buffering. The result indicates that EDGE was much worse than UMTS/HSDPA, and it is hard to provide mobile users a good experience on conversational and streaming apps.

Packet loss existed in 70% of all EDGE tickets, or UMTS/HSDPA tickets in which the RSS was less than -103 dBm. This means that it may take a long period or even fail to load a page for web apps. However, packet loss seldom occurred with a fairly good signal coverage. Note that packet loss existed in only 3% of UMTS/HSDPA tickets in which the RSS was better than -103 dBm. Retransmission can easily recover these missed packets with a small delay.

C. Effects of Handover

The switching between EDGE and UMTS/HSDPA connections often involves two different cellular towers (handover). To identify the influence of handover on the network performance, we split all UMTS/HSDPA tickets into two groups, *multi-cell tickets* (observing two or more cellular towers during testing period which means the occurrence of handover) and *single-cell tickets* (observing only one cellular tower during

testing period implying that the mobile device stayed with current cellular tower).

When there is no handover, the network performance is mainly determined by the coverage of that cellular tower. We checked the network performance when the RSS was better than -103 dBm. Tickets show RTT was less than 100 ms in more than 94% of all single-cell tickets. Jitter can be reduced to 10 ms in 95% of all single-cell tickets with a jitter buffer. There was no packet loss in 99% of all single-cell tickets. However, when the RSS was lower than -103 dBm, the network performance became sharply worse.

Tickets show that packet loss reached 85% in 15% of multi-cell tickets, which means that the radio channel in the current cellular tower might be released before the target cellular tower was engaged during handover process. Such a handover is also known as *hard handover* or break-before-make. It often takes several seconds for a mobile device to complete the handover between different cellular towers. When the mobile device failed to attach to the target cellular tower after releasing its current cellular tower, the link will be broken.

If the radio channels of both the current cellular tower and the target cellular tower are available for mobile devices, the channel with the current cellular tower can be retained and used for a while in parallel with the channel in the target cellular tower, which is known as *soft handover* or make-before-break. When the received signal strength is good enough, the current cellular tower will be detached. The signals from two cellular towers also can be combined to produce a clearer copy of signal, which helps to maintain the QoS. 95% of multi-cell tickets had no packet loss when RSS was higher than -103 dBm.

D. Effects of User Mobility

We next study the effect of user mobility on the network performance by splitting the results of all single-cell tickets into two groups, *mobility tickets* (the user is moving) and *stationary tickets* (the user is stationary or stays in a small area). There are many approaches to detect movements using the accelerometer sensor readings. The root mean squared method for taking an acceleration vector in three dimensions (g_X, g_Y, g_Z), calculated by

$$g = \sqrt{g_X^2 + g_Y^2 + g_Z^2} \quad (1)$$

that provides a global view of the instantaneous velocity variation [6]. The abnormal movement can be detected by checking the change in g , represented as

$$\Delta g_A = \frac{1}{N-1} \sum_{i=1}^{N-1} |g_i - g_{i-1}| \quad (2)$$

where g_{i-1} is the previous acceleration sample of g_i . Since completing the measurements takes between 40 to 120 seconds in general, the average of acceleration

$$g_A = \frac{1}{N} \sum_{i=0}^{N-1} g_i \quad (3)$$

is also used to decide whether the movement lasts a long period. After testing Δg_A and g_A when sitting in the office, walking through the playground, and driving on the highway, we chose

$$\Delta g_A > 0.2 \text{ or } g_A > 0.2 \quad (4)$$

as the threshold to detect the user movement since it is simple and has a high accuracy of 87%.

We also checked the network performance when the RSS was better than -103 dBm. Tickets show that the average round-trip latency was less than 100 ms in 85% of mobility tickets but in 92% of stationary tickets. The average jitter was less than 60 ms in 94% of mobility tickets but in 97% of stationary tickets. There was no packet loss in 98% of both mobility tickets and stationary tickets. All these results indicate that mobility had no significant impact on the network performance. The network performance was still acceptable for video streaming and for conversational applications if the buffering technology is applied.

However, when the RSS became worse, the network performance varied greatly when the users were moving. The round-trip latency was greater than 1 second in 50% of mobility tickets but in only 18% of stationary tickets. 80% of mobility tickets had a jitter more than 60ms which can not be removed with buffering, but this situation existed in only 60% of stationary tickets. Packet loss existed in 48% of mobility tickets but in only 15% of stationary tickets.

IV. DIAGNOSIS AND EVALUATION

The next question is how to determine the cause of the network problems given a ticket that includes the device's network and mobility measurements. We have tried various approaches, such as Feedback Control system and Bayesian Networks [20]. The results were not satisfactory as the cellular networks provide an uncontrolled environment as we could not obtain enough training samples for all fault scenarios. Instead, we propose to use a decision tree that combines rules specified by domain experts and thresholds learned statistically from performance analysis, as described in Section III.

Table III shows four primary network and mobility states that we use for network troubleshooting, and how they are measured. Decision trees classify instances by sorting them down the tree from the root, the set of all instances, to the leaf node, a small set of instances satisfying all the rules in the path to the root. Each internal node in the decision tree conducts a test of an **attribute** of the instance, and then produces new branches corresponding to the possible values for this attribute. In the context of cellular networks, the attributes are the network symptoms while the branch descending from the attribute and exceeding the predefined threshold is the possible fault cause. Figure 5 gives the structure of the decision tree we built based on these diagnostic rules. Note that in traditional decision tree models, the order of all attributes is decided by *information gain* which measures how well a given attribute separates the training samples according to their target classification. In our case, due to the lack of a

comprehensive set of training data, the decision tree was built qualitatively based on domain knowledge and performance analysis. For example, the received signal strength may cause the online streaming performance getting worse when the users are traveling across a low coverage area. However, the performance will be sharply dropped below a basic level required by video streaming when UMTS/HSDPA connection is not available. Therefore, the connection type can be selected before received signal strength to grow the decision tree.

TABLE III
THE DIAGNOSABLE RULES FOR CONVERSATIONAL APPLICATIONS

States	Measurement Methods
Network Type	checking network type of current connection such as EDGE, UMTS/HSDPA, CDMA 1xRTT, EVDO Rev 0 and Rev A
Signal Strength	checking whether the received signal strength is lower than a threshold, which can be obtained through statistical learning or infrastructure system settings
Handover	checking the number of cellular towers attached during performance test
Mobility	checking the movement state of mobile users according to Equation 4

Fig. 5. An decision tree based model for diagnosing cellular networks

Figure 6 shows the basic architecture of the proposed diagnosis system. A rule base holds the qualitative causalities between symptoms and performance measurements obtained from protocol standards and specifications. The quantitative thresholds learned from the user experience tickets collected through *MobiDiag* make these qualitative causalities available for diagnosis. A decision tree model based on diagnostic rules is generated with the help of domain experts. When we detect bad experience from the user, the root cause can be quickly figured out through the decision tree model. As new experience tickets are collected, the decision tree model can be updated to adapt to the change of network conditions or configurations.

To evaluate the diagnosis effectiveness of the proposed

Fig. 6. System architecture of decision tree based diagnosis system

TABLE IV
RESULTS OF FAULT DIAGNOSIS IN UMTS/EDGE NETWORKS

Faults	Occurred	Detected	Accuracy
Low Capacity	34	34	100%
Weak Coverage	63	57	90.4%
Handover	17	16	94.1%
Mobility	42	40	95.2%
Total	156	147	94.2%

decision tree model, we collected experience tickets for one week and labeled 156 tickets with four possible faults using semi-controlled tests. The option of *use only 2G networks* in Android system settings allowed us to measure user experience in EDGE networks. We collected weak coverage cases by performing Ping-based tests in some areas of our department building with a bad cellular coverage. When driving on the road after some time, the handover and mobility cases were also easily obtained. To make the collected cases more diverse, we also conducted Ping-based tests when walking on the ground or going upstairs and downstairs. The results of the diagnosis accuracy are shown in Table IV.

While we correctly identified the root causes for more than 94% of all tickets, the diagnosis model did fail to identify 6 weak coverage cases, 1 handover case and 2 mobility cases. While the thresholds we chose for localizing coverage issues may not be the best ones, we used different number of training tickets to decide it. For a given number of experience tickets, we split all tickets into two groups S_L and S_H by using one of 20 possible RSS values in range from -114 dBm to -95 dBm. We then compute the difference between the average RSS of all tickets in these two groups, represented as

$$D(r_i) = |average(S_L(r_i)) - average(S_H(r_i))| \quad (5)$$

Fig. 7. Influence of the size of training tickets on diagnosis accuracy

where r_i is the RSS used to split all tickets. The average RSS is calculated by

$$average(S) = \log\left(\frac{1}{|S|} \sum_{s_i \in S} 10^{s_i}\right) \quad (6)$$

where s_i is the RSS of the ticket i in dBm. The signal strength r_k corresponds to the maximum difference $D(r_k)$ between the average RSS in $S_L(r_k)$ and $S_H(r_k)$. We repeated this procedure to determine the threshold for localizing weak coverage faults and used it to identify possible 63 weak coverage issues. The diagnosis results shown in Figure 7 suggest the diagnosis accuracy can reach a high level 95%, compared to 90.4% using -103 dBm that is taken from protocol specification, when the number of training tickets is large enough (e.g. more than 1000 tickets).

V. MOBI DIAG IMPLEMENTATION

We implemented *MobiDiag* in client-server structure as shown in Figure 8. On the *MobiDiag* client, the Diagnostic runs when the user open the User Interface on the client to diagnose the network performance, whose result is then uploaded to the *MobiDiag* Server. On the *MobiDiag* server, the Web Service provides a RESTful API for the client uploads and requests. The uploaded customer experience tickets are stored in a Backend Database. *MobiDiag* provides systematic measurements for a high-level diagnosis at customer side.

We have implemented the *MobiDiag* client on Google Android platform. The current *MobiDiag* Android client has approximately 10,000 lines of Java code using Android SDK Version 2.3, also named as Gingerbread. According to Google's statistics for active devices running different versions of the Android platform, more than 97% of Android devices are using Android 2.3 and up as of September 2013. Since Android applications are generally forward-compatible with new versions of the Android platforms, we chose this version for our development so *MobiDiag* can support most Android devices on the market.⁷ In the future work, we plan to investigate the feasibility of porting *MobiDiag* client to other smartphone platforms, such as iPhone and Windows Mobile.

⁷<http://developer.android.com/about/dashboards/index.html/>

Fig. 8. High-level view of MobiDiag's architecture

Fig. 9. The components of MobiDiag client

Figure 9 shows the components of the *MobiDiag* client. When the user opens the *MobiDiag* client interface and starts diagnosing potential problems, the User Interface invokes the Diagnostic to conduct network testing through the Performance Tester. The State Monitor also collects the network state changes. Once the testing is finished, the user interface will be updated with the results and the possible cause to the network problems analyzed by the diagnosis system as shown in Figure 10a. Then the Ticket Uploader combines network testing results and state changes into tickets and then asks the Request Manager to upload tickets to the *MobiDiag* server. The tickets are put in the Request Queue and the Task Executor processes these requests by issuing HTTP requests to the *MobiDiag* server.

Since different applications have different service features, each application has its explicit requirements to a network

Fig. 10. The MobiDiag interfaces

in order to achieve the desired QoS [8]. We categorized internet applications by their network QoS requirements into four levels: web surfing, video streaming, VOIP and video conferencing. We consider a network failing to meet any one level of internet application's requirements as a problem network and trigger the diagnosis on it. Glanced at the test result in Figure 10a, it disclosed the relative high latency and latency jitter of the network(fail to support video conferencing) and our diagnosis system computed out the possible cause that was the poor data mobile data network connection type. We can see that the tested network type is EHRPD. EHRPD is a software solution to bridge existing CDMA 3G network and LTE network with most of 3G network features and hard to support video conferencing. Moreover, we score the network according to QoS requirements of internet applications. One of our future study is to score the network more accurately to reflect the real feeling of smartphone users about the network.

The MobiDiag server is deployed by using the Play Framework that is an open source and non-blocking web server. The components in the MobiDiag server are shown in Figure 11. It accepts network measurement tickets uploaded by the MobiDiag clients through HTTP POST requests, and the data is forwarded to and stored in a MySQL database. The MobiDiag server is responsible to process systematic measurements such as number of users, devices types, regional statistics, cross-carrier statistics and so on. The primary statistic data is currently charted on the MobiDiag client interface as shown in Figure 10b. The systematic measurement results will be further studied in our future work to provide straightforward and comparable knowledge of mobile network to smartphone users.

Fig. 11. The components of MobiDiag server

VI. RELATED WORK

Client Conduit [3] is one of the first proposed architectures in IEEE 802.11 Wireless LANs, which allows disconnected wireless clients to exchange network measurements with the backend diagnosis servers via a connected client. WiFiProfiler [7] allows wireless clients to exchange network measurements they collected in a peer-to-peer fashion, so they can cooperative diagnose network problems without a server. Both systems focus on a setting where exists a number of cooperative peers sharing information within the wireless communication range, which may not be practical in reality. We also address the problem where the mobile devices experience network problems, but focus on the pre-installed diagnostic engine that leverage both rules derived from domain knowledge and thresholds empirically learned from statistical performance analysis.

Jin et al. analyzed customer tickets recorded when the users called help line to report problems [11] and also the app-based reports that were submitted by the users [10]. In the first study, the results show that the call rate correlated with user tenure and customer device and can be used to detect chronic network problems. In the second study, app-based self-reports can be used to detect emerging network problems. However, both approaches do not try to automatically diagnose network problems and thus provide little insights to the end users.

There are quite a few works in detecting and troubleshooting network problems, via expert rules [13], machine learning [4], [9], or dependency graphs [5], [12], [14]. We focus on the client-side network troubleshooting, which does not have a complete view of the networked system. Thus these traditional approaches are not directly applicable, and most end users often do not need such fine-grained diagnosis results.

VII. CONCLUSION AND FUTURE WORK

In this paper we propose a client-side system that empowers end users to understand the likely causes of wireless network problems, which can help them make informed decisions to upgrade service plans, change mobile devices, or switch service providers.

The proposed diagnosis mobile app, *MobiDiag*, runs on Android platform and measure the operator's network performance by conducting Ping-based tests to the gateway or the first responsive router. The metrics computed from these tests include round-trip time, jitter, and packet loss. These performance indicators are used to determine the network problems for common mobile apps. Through empirical analysis, we chose network type, signal coverage, cell handover, and user mobility as four likely causes that may significantly impact the performance metrics.

Through the measurements against the four factors, *MobiDiag* employs a decision tree model to decide the most likely cause of network problems as the feedback to the end user. Experimental results show that the diagnosis model was effective in troubleshooting 94% of all faulty cases. The on-device diagnosis model does not need network communication so it can work offline. The measurements are opportunistically uploaded to our server for further analysis, which can be used to adjust the model attributes and thresholds if necessary.

As a future work, we plan to extend *MobiDiag* to run in the background so it can periodically take measurements and upload to our server. With more performance reports spread over the user's daily life, she may get a better picture of her service provider's network performance. As the number of measurements received by our server grows, we will report back to the user of the network performance comparisons for the users in the same regions across different devices and operators.

ACKNOWLEDGMENT

This work is supported partly by the National Science Foundation under Grant No. 0917112 and No. 1040725. Any opinions, findings, and conclusions or recommendations expressed in this work are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

- [1] QoS performance requirements for UMTS. Nortel Networks. The 3rd Generation Partnership Project (3GPP).
- [2] Speedtest.net mobile applications. <http://www.speedtest.net/mobile.php>.
- [3] Atul Adya, Paramvir Bahl, Ranveer Chandra, and Lili Qiu. Architecture and Techniques for Diagnosing Faults in IEEE 802.11 Infrastructure Networks. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 30–44, September 2004.
- [4] B. Aggarwal, R. Bhagwan, T. Das, S. Eswaran, V. N. Padmanabhan, and G. M. Voelker. Netprints: diagnosing home network misconfigurations using shared knowledge. In *Proceedings of NSDI*, 2009.
- [5] Paramvir Victor Bahl, Ranveer Chandra, Albert Greenberg, Srikanth Kandula, David Maltz, and Ming Zhang. Towards Highly Reliable Enterprise Network Services via Inference of Multi-Level Dependencies. In *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 13–24, August 2007.
- [6] T. Ryan Burchfield and S. Venkatesan. Accelerometer-Based Human Abnormal Movement Detection in Wireless Sensor Networks. In *Proceedings of the 1st ACM SIGMOBILE international workshop on Systems and networking support for healthcare and assisted living environments*, pages 67–69, June 2007.
- [7] Ranveer Chandra, Venkata N. Padmanabhan, and Ming Zhang. WiFi-Profiler: Cooperative Diagnosis in Wireless LANs. In *Proceedings of The 4th International Conference on Mobile Systems, Applications and Services (MobiSys)*, pages 205–219, June 2006.
- [8] Yan Chen, Toni Farley, and Nong Ye. QoS Requirements of Network Applications on the Internet. *Information-Knowledge-Systems Management*, 4(1):55–76, 2004.
- [9] I. Cohen, M. Goldszmidt, T. Kelly, J. Symons, and J. S. Chase. Correlating instrumentation data to system states: a building block for automated diagnosis and control. In *Proceedings of OSDI*, 2004.
- [10] Yu Jin, Nick Duffield, Alexandre Gerber, Patrick Haffner, Wen-Ling Hsu, Guy Jacobson, Subhabrata Sen, Shobha Venkataraman, and Zhi-Li Zhang. Large-scale app-based reporting of customer problems in cellular networks: Potential and limitations. In *Proceedings of the ACM SIGCOMM Workshop on Measurements Up the Stack (W-MUST)*, August 2011.
- [11] Yu Jin, Nick Duffield, Alexandre Gerber, Patrick Haffner, Wen-Ling Hsu, Guy Jacobson, Subhabrata Sen, Shobha Venkataraman, and Zhi-Li Zhang. Making sense of customer tickets in cellular networks. In *Proceedings of INFOCOM 2011 Mini-Conference*, April 2011.
- [12] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl. Detailed diagnosis in enterprise networks. In *Proceedings of SIGCOMM*, 2009.
- [13] G. Khanna, M. Yu Cheng, P. Varadharajan, S. Bagchi, M. P. Correia, and P. J. Verissimo. Automated rule-based diagnosis through a distributed monitor system. *IEEE Transactions on Dependable and Secure Computing*, 4(4), 2007.
- [14] Ramana Rao Kompella, Jennifer Yates, Albert Greenberg, and Alex C. Snoeren. IP Fault Localization via Risk Modeling. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design and Implementation (NSDI)*, pages 57–70, May 2005.
- [15] Ratul Mahajan, Neil Spring, David Wetherall, and Thomas Anderson. User-level Internet Path Diagnosis. In *Proceedings of The 19th ACM Symposium on Operating Systems Principles (SOSP)*, pages 106–119, October 2003.
- [16] Franco P. Preparata, Gernot Metze, and Robert T. Chen. On the Connection Assignment Problem of Diagnosable Systems. *IEEE Transactions On Electronic Computers*, EC-16(6):848–854, December 1967.
- [17] Fabio Ricciato, Francesco Vacirca, Wolfgang Fleischer, Johannes Motz, and Markus Rupp. Passive Tomography of a 3G Network: Challenges and Opportunities. In *The 25th Annual IEEE International Conference on Computer Communications (InfoCom)*, pages 116–122, April 2006.
- [18] Sayandeep Sen, Jongwon Yoon, Joshua Hare, Justin Ormont, and Suman Banerjee. Can they hear me now?: A case for a client-assisted approach to monitoring wide-area wireless networks. In *Proceedings of IMC*, November 2011.
- [19] Qiang Xu, Junxian Huang, Zhaoguang Wang, Feng Qian, Alexandre Gerber, and Z. Morley Mao. Cellular data network infrastructure characterization and implication on mobile content placement. In *Proceedings of SIGMETRICS*, June 2011.
- [20] Bo Yan. *Collaborative Fault Diagnosis in Large-Scale Mobile Communication Networks*. PhD thesis, April 2012.