Google Cloud Messaging (GCM): An Evaluation

Yavuz Selim Yilmaz Bahadir Ismail Aydin Murat Demirbas
Department of Computer Science and Engineering
SUNY University at Buffalo, Buffalo, New York, 14260, USA
Email: {yavuzsel | bahadiri | demirbas}@buffalo.edu

***Abstract*—This paper presents a survey on the timing performance of Google Cloud Messaging (GCM). We evaluate GCM in real world experiments, and at a reasonable scale involving thousands of real users. Our findings reveal that the GCM message delivery is unpredictable, namely having a reliable connection to Google's GCM servers on the client device does not guarantee a timely message arrival. Therefore, GCM is not suitable for time sensitive and/or *"must-deliver-to-all"* app scenarios. On the other hand, GCM delivers the push messages to a big portion of the subscribers (more than 40% in any experiment scenario) in a reasonable timeframe (in 10 seconds). Therefore, GCM may be a good fit for the application scenarios where random multicasting is sufficient, such as crowdsourcing systems. Our results provide a through evaluation of the GCM performance, and will guide developers and researchers to decide whether GCM is suitable for their intended use cases.**

I. Introduction

With the advent of mobile and pervasive computing era, smartphones became ubiquitous, and wearable devices are getting traction. A significant portion of the applications for these devices relies on remote servers on the cloud, and Google Cloud Messaging (GCM) [1] is a popular service as a client/server communication solution for Android. Today more than half of the smartphones run Android OS. With the recent release of Android Wear [2], Android extended its domain to wearable devices. Both of these platforms use GCM for central notifications. For example, Google Glass forwards most tasks to a cloud-based solution called Mirror API [3], and Mirror uses GCM for client/server communications.

GCM is a service which allows developers to send push messages to Android devices from the server. GCM handles the queuing of the messages as well as delivering those messages to the target applications on the devices. GCM is a free service by Google, and it has no quotas. It is the default push messaging solution for the Android platform.

In this paper, we present a survey on GCM message arrival times in order to investigate the realtime properties of GCM. Our aim is to provide a better understanding of this widely used service. Understanding the performance of GCM is essential for developers especially while developing applications for public avail, such as disaster alert apps, blood donation notifications etc. It is also essential to understand the realtime properties of GCM for researchers working on mobile systems such as crowdsourcing, question answering and other realtime systems.

GCM does not mention any time guarantee in its documentation [1]. However there is a need to unveil the GCM message delivery performance to caution developers and researchers from over-relying on GCM (by assuming it is fast and reliable for all cases). To this end, we evaluate GCM in real world experiments, and at a reasonable scale involving thousands of real users. Our contributions in this work are as follows:

- To the best of our knowledge, we provide the first comprehensive GCM evaluation for a variety of real world experiment scenarios, namely *offline* and *online* experiments which are elaborated based on the connection type (i.e. WiFi or cellular data) and data service providers.
- Our experiments involve thousands of real users, and the results let developers and researchers make a supervised decision on whether GCM is suitable for their intended use cases.
- We show that GCM delivers the push messages to a big portion of the subscribers (more than 40% in any experiment scenario) in a reasonable timeframe (in 10 seconds).
- We reveal that the GCM message delivery is unpredictable, namely having a reliable connection to Google's GCM servers on the client device does not by itself guarantee a timely message arrival.
- We conclude that GCM is not suitable for time sensitive and/or *"must-deliver-to-all"* app scenarios.

Overview of Our Results: In order to analyze GCM, we employ and piggyback on our CrowdReply app [4] which we built for crowdsourcing research. CrowdReply is an Android application which lets the crowd play the "Who wants to be a millionaire?" game in realtime while TV show is live. For our analysis, we collected anonymized timing data from mobile participants who have our Android app installed on their smartphones or tablets.

We define two different experiment scenarios. In the *offline* experiment, we calculate the GCM message arrival latency at a random time. Therefore, in this scenario, the server has no knowledge of whether the devices are powered on, are in use, or have network connection. In the *online* experiment, we calculate the GCM message arrival time using the online participants while the TV show is live on the TV and when the participants are playing the game. Therefore, in the *online* experiment, we know that the devices are powered on, actively in use and have network connection. We also measure how connection type affects the message arrival time, namely we detail our evaluation based on the connection types: WiFi and cellular data. Furthermore, we elaborate our experiments on the cellular connection based on different cellular data service providers.

Our experiment results show that GCM falls short for delivering the push messages to all subscribers in a timely

978-1-4799-3512-3/14/$31.00 ©2014 IEEE

manner. Moreover, the GCM message arrival latency is unpredictable even when the device is connected to Google's GCM servers. This indicates that GCM is not suitable for time sensitive and/or *"must-deliver-to-all"* application scenarios, such as fire alert, instant messaging, disaster alert etc. On the other hand, GCM delivers the messages to a big portion of the subscribers (nearly 80% in the *online* and 40% in the *offline* experiments) in a reasonable timeframe (in 10 seconds for both experiments). Therefore, GCM may be a good fit for the application scenarios where random multicasting is sufficient, such as crowdsourced question answering systems, blood donation notifications etc.

Outline of the paper: We present related work in Section II. In Section III, we detail our Android application and GCM data collection. We show comparative results in Section IV, and we conclude the paper by discussing our findings and future work in Section V.

II. Related Work

Today, all popular mobile operating systems support push notification services [1], [5], [6], [7]. There is still ongoing work to optimize publish/subscribe model for ever-developing mobile technology. For example, Mobius [8] from *Yahoo! Research* and *MIT* uses caching and makes more efficient use of ad hoc hardware for that purpose. MobilePush [9] proposes a design for an internet-scale system that supports mobile devices connected to both wireless LAN and GPRS network, and in [10], authors present an evaluation of their mobile publish/subscribe setting. To the best of our knowledge, only one retrospective study evaluates the performance of the push notification services for popular mobile operating systems [11]. In this work, the performance of Android Cloud to Device Messaging (C2DM), i.e. the deprecated predecessor of GCM, is compared to some other push notification services which are configurable to work with Android clients. Their findings reveal that, C2DM gets around 4 to 6 seconds response time given that the device which receives the push notification initiates the message, i.e. the push notification bounces back to the same device. Therefore, their results are limited to periodic notifications to a *single device*, and lacks comparative performance analysis of mass push notifications.

III. CrowdReply: Our Crowdsourced Question Answering App

In this section, we explain the architecture of CrowdReply, our crowdsourced "Who wants to be a millionaire?" (WWTBAM) application. CrowdReply enables the audience watching WWTBAM on the TV to play along on their Android smartphones in realtime.

We targeted the Turkish audience due to the high popularity of the TV show there. Our app has been installed more than 311,000 times [12]. The game enables us to build a large-scale crowdsourcing dataset. Using this data, we test several crowdsourcing algorithms in real life, and we also evaluate the GCM performance in real world experiments.

The overall architecture of the CrowdReply is shown in Figure 1. CrowdReply consists of two main parts, a mobile side for presenting the questions to the users and letting them answer the questions, and a server side for typing and dispatching the questions, collecting the answers, and providing useful statistics. We described the design, implementation, and deployment of the CrowdReply in a previous work [4]. In this paper, we leverage this app and the data for evaluating the GCM timing performance.

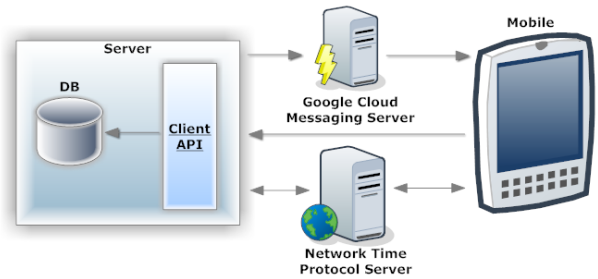

Fig. 1. CrowdReply system architecture

A. GCM Messages in CrowdReply

In WWTBAM game show, the questions appear on the TV in arbitrary moments. Because of this, CrowdReply uses push messages to send the questions, instead of the client device polling the questions from the server. This design lets CrowdReply use the connection only when there is a new question, and it delegates the data pushing job to GCM in order to resolve any scalability issues CrowdReply might encounter.

TABLE I. CrowdReply GCM Message Types

Message Type	Action	Target
B	Begin the game	all users
E	End the game	all users
Q	Send question	online users
S	Send statistics / answers	online users
T	Time Synchronization	all users

Table I shows the types of the GCM messages which CrowdReply defines and uses. At the start of the TV show, CrowdReply sends a GCM message of type B which lets all the registered devices know that the game is started. When a game is started and a participant decided to play (i.e. opened the corresponding screen), the app notifies the server to receive the questions. If a participant is not online (i.e. for that particular device, no "activity running" notification is arrived on the server side), then CrowdReply does not send the questions to her device in order to reduce the workload on the server side as well as to save energy on the offline devices.

During the game, CrowdReply sends message of type Q, which includes the question, whenever the question appears on the TV and our project members type it. The client app parses the message upon arrival and updates the user interface accordingly. Type S message is sent when a question is answered on the TV, and it consists of the correct answer and the statistics of the previous question. Type E message notifies the participant devices that the TV show and thus the game is ended. Finally, type T message is designed to synchronize the local time of the client devices with the server side in order to have accurate timestamps in CrowdReply system.

B. GCM Data Collection

Here we explain how we collected the time data in our experiments. We leverage our time synchronization mechanism (i.e. GCM message type T) to collect our data on the GCM message arrival times. To calculate precise times, we use network time protocol (NTP)[1] [13].

The GCM message arrival time is defined as the difference between the time the GCM message is initiated from our servers, and the time the client device receives that message. Therefore, in our type T GCM message flow, the difference between the server NTP time of the message initiation, and the client NTP time of the message arrival gives us precise GCM message arrival time for each device.

The client devices bounce-back to our type T message with message arrival NTP timestamp, and we calculate per device GCM message arrival times in our server. We evaluate these arrival times to elaborate how GCM performs.

In addition to this time data, we also record the network connection type (WiFi or cellular data) of the client devices as well as the cellular service provider in order to test the effects of the connection type and the data service provider.

IV. Experiments and Results

In this section, we present our data and then we evaluate the timing performance of GCM. We measured the GCM message arrival times in two different experiment scenarios: *offline* and *online*. In our *offline* experiment, we send the GCM message at a random time. Therefore, in this scenario, the server has no knowledge of whether the client devices are powered on, are in use, or have network connection. On the other hand, in our *online* experiment, we send the message to the online participants while the show is live on the TV and the participants are playing the game. Therefore, in our *online* experiment, the client devices are powered on, actively in use and have network connection.

A. The Dataset

TABLE II. Number of Devices in Each Experiment

		Experiments	
		Offline	Online
Connection Type	WiFi	1656	199
	Cellular	1299	165
	Unidentified	163	18
Device Type	Smartphone	2586	334
	Tablet	532	48
Total		3118	382

Here we provide some statistics about our dataset. Table II shows the number of devices by the connection type and the device type for each experiment.

[1]NTP is a networking protocol for time synchronization which uses packet switching delays to variable latency networks in order to synchronize the time.

TABLE III. Poisson Mean Number of Devices and Chi-Squared Goodness-of-Fit Test P-Values

	Experiments	
	Offline	Online
λ	1.89	5.92
p	0	4.7334E-203

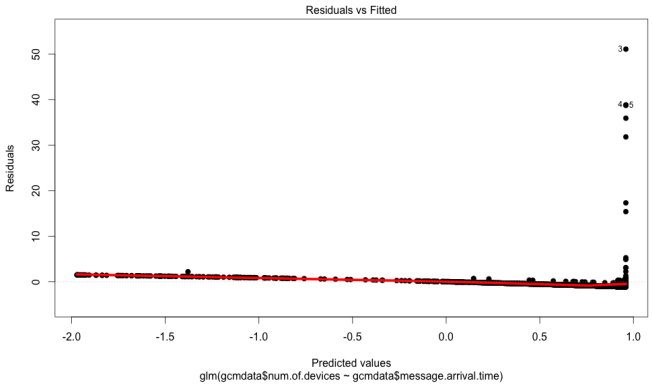

Fig. 2. Offline Experiment: Poisson Model using GLM

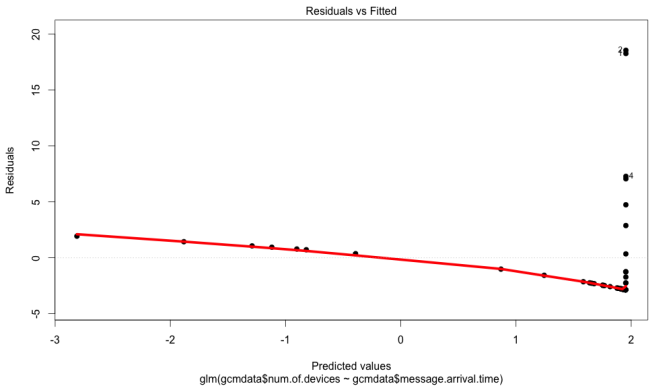

Fig. 3. Online Experiment: Poisson Model using GLM

We observed that the number of devices, which receive the GCM message, per time follows Poisson distribution. To test this hypothesis, we fitted Poisson distribution to the number of devices per time, and conducted chi-squared goodness-of-fit test on our models. Table III shows the mean number of devices (λ) and chi-squared goodness-of-fit test results ($p - value$), and Figures 2 and 3 show the residual vs. fitted values for each experiment. As seen on the Table III, the $p-value$'s are low which suggests that Poisson distribution is a plausible fit to the number of devices per time. This indicates that, message arrivals occur with an average rate in a large period. It also shows that the number of devices receiving the GCM messages is independent from the previous GCM message arrivals.

In order to calculate the average message arrival time in our experiments, we used inter-quartile range (IQR) [14] based

outlier removal method[2]. In our calculation, any arrival time beyond $(1.5 * IQR) + (3rd\,Quartile)$ is considered as outlier. Table IV shows the percentages of the number of outliers based on the experiment and the connection type in our dataset. The percentage of the outliers can be regarded as dead message delivery rate in most cases.

TABLE IV. PERCENTAGE OF IQR OUTLIERS IN EACH EXPERIMENT

		Experiments	
		Offline	Online
Overall		10%	14%
Connection Type	WiFi	7%	17%
	Cellular	20%	13%
Cellular Data Provider	AVEA	21%	16%
	Turkcell	18%	17%
	Vodafone	19%	7%

B. *Offline and Online Experiments*

Here we evaluate the GCM message arrival times in our *offline* and *online* experiments. We first give an overall performance of GCM in those experiments, and then we detail our evaluation by comparing the message arrival times based on the network type, namely WiFi or cellular data. Finally, we elaborate the GCM performance among different cellular data providers in order to show how the network infrastructure and the data service coverage affect the message arrival times.

Table V shows the statistical measures of the message arrival times in our *offline* and *online* experiments. As it is seen in the Table V, $1st\ Quartile$'s are within reasonable range in both experiments. However, the median and the average message arrival times indicate that the latency in the *offline* experiment is significantly high compared to the *online* experiment.

TABLE V. OFFLINE AND ONLINE EXPERIMENTS MESSAGE ARRIVAL TIMES (IN SECONDS)

	Experiments	
	Offline	Online
1st Quartile	5	1
3rd Quartile	8032	5
Median	334	2
Average	2996	2.8

Figure 4 shows the cumulative message arrival times in our experiments. The graph reads as follows:

The point A on the *offline* experiment plot means that the message did not reach to 58% of the devices in 10 seconds in our *offline* experiment. Similarly, the point B on the *online* experiment plot means that the message did not reach to 16% of the devices in 10 seconds in our *online* experiment. Hence, the message arrives to a significantly bigger portion of the devices in the *online* experiment than in the *offline* experiment within the same time interval.

Following the Table V, Figure 4 also reveals that the GCM message arrival latency significantly differs by the experiment type, namely having the devices powered on, in use, and connected to network have a significant effect on the message arrival time.

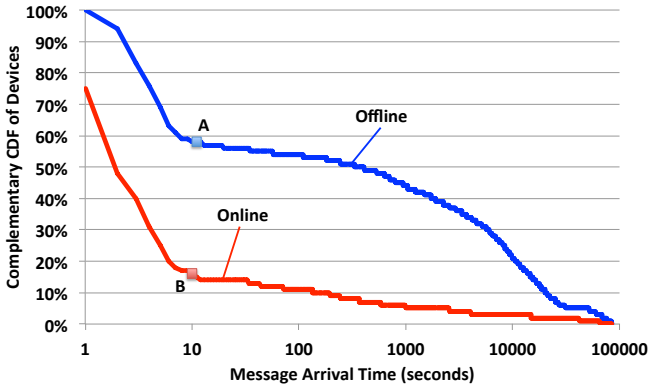

Fig. 4. Complementary CDF of Message Arrival Times for Offline and Online Experiments

In our second analysis, we analyzed how the connection type of the client device affects the message arrival time. Table VI shows the statistical measures of the arrival times based on the connection type for each experiment. As the median message arrival times indicate, when there is a connection available, the message arrival time is within reasonable range for the app scenarios which are not time sensitive, such as crowdsourced question answering. However, as $3rdQuartile$'s in the *offline* experiment indicate, the message is not able to reach to a big portion of the users in a timely manner. Hence, GCM is not suitable for time and mission critical app scenarios where message arrival to all users in a short time is required, such as emergency alerts.

TABLE VI. WIFI AND CELLULAR DATA CONNECTION MESSAGE ARRIVAL TIMES (IN SECONDS)

	Experiments			
	Offline		Online	
	WiFi	Cell	WiFi	Cell
1st Quartile	4	5	1	2
3rd Quartile	11724	1759	4	6
Median	2449	8	2	4
Average	5594	328	2	3

Figure 5 shows how the device connection type affects the message arrival time in our *offline* experiment. As figure suggests, the message arrival times are less when cellular data connection is available. Namely, more than half of the messages arrive in less than 10 seconds when cellular data connection is available. However, the remaining half of the messages still arrives too late to consider GCM as an alternative communication channel for time sensitive applications.

Figure 6 shows how the message arrival times change by connection type in our *online* experiment. As the figure

[2]IQR is defined as the difference between $1st Quartile$ and $3rdQuartile$, namely $IQR = |1st\ Quartile - 3rd\ Quartile|$

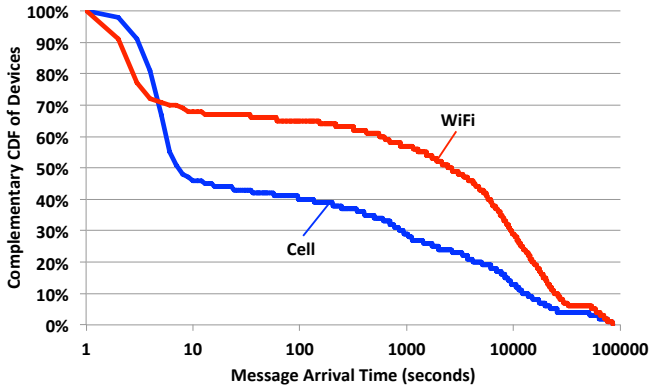

Fig. 5. Complementary CDF of Message Arrival Times for Cellular Data and WiFi Connections in Offline Experiment

suggests, when the devices are powered on, actively in use, and have network connection, then the network connection type does not have any significant effect on the message arrival time.

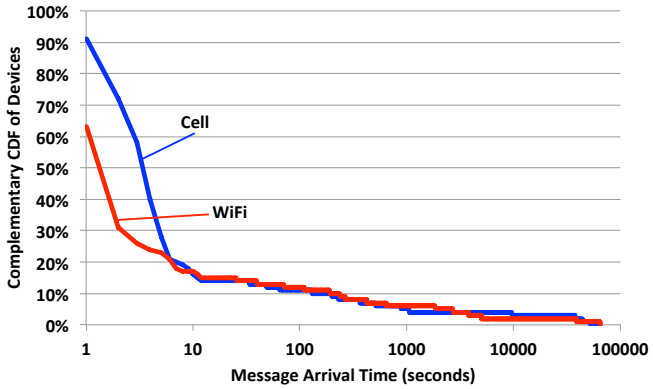

Fig. 6. Complementary CDF of Message Arrival Times for Cellular Data and WiFi Connections in Online Experiment

In Turkey (CrowdReply's target country), there are three carriers providing cellular service, which are AVEA [15], Turkcell [16] and Vodafone [17]. All three providers have 3G networks in most of the crowded cities and towns. Here we evaluate how these providers affect the message arrival latency. Table VII shows the message arrival times in our experiments based on the cellular data providers. Recall from the Table VI that, the median message arrival time is reasonably small for the cellular data in both the *offline* and the *online* experiment scenarios. However, Table VII indicates that the median message arrival time in the *offline* experiment significantly differs based on the network infrastructure (149 seconds difference between the best and the worst performing cellular network providers).

Figure 7 visualizes the changes on the message arrival times in the *offline* experiment by cellular data providers. These results suggest that the cellular data connection infrastructure has an effect on the message arrival times. Although Turkcell is the market leader and has the largest cellular data coverage area in Turkey, the GCM message arrival latency is the biggest for the client devices using the Turkcell cellular data service.

Figure 8 shows the cumulative message arrival times by

TABLE VII. CELLULAR DATA SERVICE PROVIDERS MESSAGE ARRIVAL TIMES (IN SECONDS)

	Experiments					
	Offline			Online		
	AVEA	Turkcell	Vodafone	AVEA	Turkcell	Vodafone
1st Quartile	5	6	5	2	2	2
3rd Quartile	836	2870	1994	6.5	6	5
Median	6	155	7	4	4	4
Average	110	677	427	4	3.8	3.5

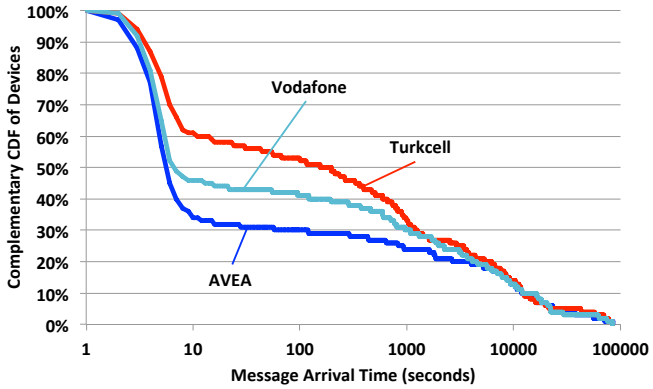

Fig. 7. Complementary CDF of Message Arrival Times for Cellular Data Service Providers in Offline Experiment

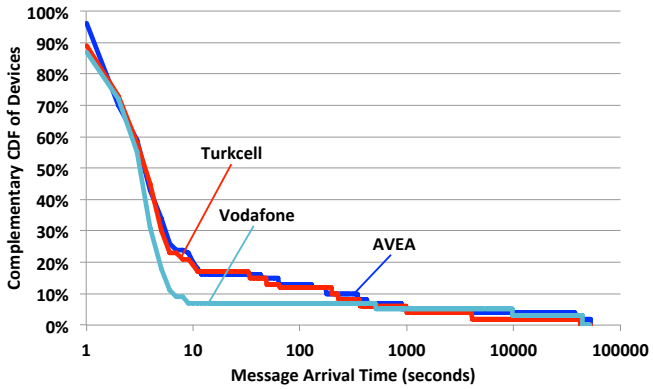

Fig. 8. Complementary CDF of Message Arrival Times for Cellular Data Service Providers in Online Experiment

cellular data service providers in the *online* experiment. It is clear from the figure that, in the *online* experiment, all cellular data providers perform similar. However, even in the *online* experiment, where we know that all devices are powered on, actively in use, and have network connection, some devices (10% of the devices in the AVEA network, and 20% of the devices in the other two networks) do not receive the GCM messages in a reasonable time, namely it takes more than 10 seconds for a message to arrive (and some devices receive the message in hours).

C. Double Message Offline Experiment

Our experimental results show that the GCM message arrival time is significantly less in the *online* experiment than in

the *offline* experiment. To further investigate the late message arrival in the *offline* scenario, we conducted another experiment which we name as *double message*. In this run, we send two GCM messages initiated at the same time on our server. Then, this time, instead of measuring the message arrival time, we measure the difference between the arrival time of the first message and the arrival time of the second message. Therefore, we know that, at the time the first message is arrived to a device, the device is reachable by Google's GCM servers.

Table VIII shows the statistical measures of the arrival time difference between the two messages in our *double message* experiment. As it is seen, although the messages are initiated at the same time, the median arrival time difference is more than 260 seconds overall, and it is 78 seconds for the cellular data connection. Moreover, Figure 9 shows that some of the devices receive the second message hours after receiving the first one. These results reveal that the GCM message arrival latency is unpredictable, namely even the connection is established between Google's GCM servers and the client devices, two GCM messages in a message queue are being sent on different times.

TABLE VIII. DOUBLE MESSAGE EXPERIMENT: PER DEVICE MESSAGE ARRIVAL TIME DIFFERENCE (IN SECONDS)

	Connection Type		
	Overall	Cell	WiFi
1st Quartile	28.5	30.5	24
3rd Quartile	11611	1483	20172
Median	264	78	631
Average	3228	344	7947

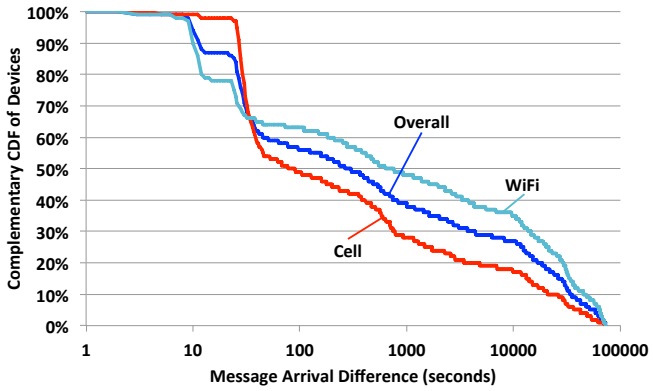

Fig. 9. Complementary CDF of Message Arrival Differential in Double Message Experiment

V. CONCLUDING REMARKS

In this paper, we evaluated the timing performance of Google Cloud Messaging (GCM) using our CrowdReply app. We conducted three different experiments in two different experiment scenarios (i.e. *offline* and *online*). Our results indicate that GCM is not suitable for time sensitive and/or *"must-deliver-to-all"* application scenarios. Furthermore, the GCM message arrival latency is unpredictable, namely having a reliable connection to Google's GCM servers on the client device does not by itself guarantee a timely message arrival. While GCM performs fairly well in our *online* experiment scenario, not all the devices receive the GCM messages in a timely manner. Hence, GCM may be a good fit for the application scenarios where random multicasting is enough, such as crowdsourced question answering systems [4] including location based services [18], [19], and blood donation notifications etc. However, GCM is not a good fit for the applications where the broadcasting is mission critical, i.e. the message arrival to all client devices is vital, such as emergency alert services, fire alert systems, instant messaging apps, disaster alert services etc.

As a future work, we will investigate other push notification options available for Android including Extensible Messaging and Presence Protocol (XMPP) [20], and Message Queue Telemetry Transport (MQTT) [21] in terms of performance and energy efficiency, and compare them with GCM for time-sensitive app scenarios. Although such external protocols need more implementation time and maintenance, we will evaluate whether they can be a good fit for the apps where GCM does not satisfy the performance requirements.

REFERENCES

[1] "Google Cloud Messaging for Android — Android Developers." [Online]. Available: http://developer.android.com/google/gcm/index.html
[2] "Android Wear." [Online]. Available: http://www.android.com/wear
[3] "Google Glass-Mirror API." [Online]. Available: https://developers.google.com/glass/develop/mirror/index
[4] B. I. Aydin, Y. S. Yilmaz, M. F. Bulut, and M. Demirbas, "Crowdreply: A crowdsourced multiple choice question answering system," *ACM/IEEE IPSN '13 3rd International Workshop on Mobile Sensing: The Future, brought to you by Big Sensor Data*, 2013.
[5] "Local and Push Notification Programming Guide: Apple Push Notification Service." [Online]. Available: http://developer.apple.com/library/mac/\#documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/ApplePushService/ApplePushService.html
[6] "Push Notifications for Windows Phone." [Online]. Available: http://msdn.microsoft.com/en-us/library/ff402537(v=VS.92).aspx
[7] "Push Service - BlackBerry Developer." [Online]. Available: https://developer.blackberry.com/services/push/?CPID=PUSHAPI00
[8] B.-g. Chun, A. Shraer, C. Curino, S. Madden, and R. Sears, "Mobius : Unified Messaging and Data Serving for Mobile Apps Categories and Subject Descriptors," *MobiSys*, pp. 141–153, 2012. [Online]. Available: http://doi.acm.org/10.1145/2307636.2307650
[9] I. Podnar, M. Hauswirth, and M. Jazayeri, "Mobile push: delivering content to mobile users," *Proceedings 22nd International Conference on Distributed Computing Systems Workshops*, p. 563568, 2002. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1030826
[10] I. Podnar and I. Lovrek, "Supporting mobility with persistent notifications in publish/subscribe systems," *Int. Workshop on Distributed Event-Based Systems*, 2004. [Online]. Available: http://www.inf.unisi.ch/carzaniga/debs04/debs04proceedings.pdf\#page=83
[11] J. Hansen, T.-M. Gronli, and G. Ghinea, "Towards cloud to device push messaging on android: Technologies, possibilities and challenges," *Intl J. of Communications, Network and System Sciences*, vol. 05, no. 12, p. 839849, 2012. [Online]. Available: http://www.scirp.org/journal/PaperDownload.aspx?DOI=10.4236/ijcns.2012.512089
[12] "Crowdreply at google play store," https://play.google.com/store/apps/details?id=edu.buffalo.cse.ubicomp.crowdonline, [Online].
[13] "Ntp: The network time protocol," http://www.ntp.org, [Online].
[14] G. Upton and I. Cook, *Understanding Statistics*. OUP Oxford, 1996, pp. 55. [Online]. Available: http://books.google.com/books?id=vXzWG09_SzAC
[15] "Avea," http://www.avea.com.tr, [Online].
[16] "Turkcell," http://www.turkcell.com.tr, [Online].
[17] "Vodafone," http://www.vodafone.com.tr, [Online].
[18] M. F. Bulut, Y. S. Yilmaz, and M. Demirbas, "Crowdsourcing location-based queries," in *PerCom Workshops*, 2011, pp. 513–518.
[19] M. F. Bulut, Y. S. Yilmaz, M. Demirbas, N. Ferhatosmanoglu, and H. Ferhatosmanoglu, "Lineking: Crowdsourced line wait-time estimation using smartphones," in *MobiCASE*, 2012, pp. 205–224.
[20] "Xmpp standards foundation," http://xmpp.org, [Online].
[21] "Mqtt: Message queue telemetry transport," http://www.mqtt.org.