Push notification as a business enhancement technique for e-commerce

Conference Paper · December 2015

DOI: 10.1109/ICIIP.2015.7414815

2 authors:

Arvind Kumar
Central Scientific Instruments Organization
7 PUBLICATIONS **26** CITATIONS
SEE PROFILE

Suchi Johari
DIT University
17 PUBLICATIONS **63** CITATIONS
SEE PROFILE

Push Notification as a Business Enhancement Technique for E-commerce

Arvind Kumar[#1], Suchi Johari[2]
[#]Department of Computer Applications, Indus International University Una, Himachal Pradesh, India
[1]er.arvindkumar1989@gmail.com
[2]shuchi.johari@gmail.com

Abstract— **Push notification can have a great impact on e-commerce. It makes business to attract more customers by tracking the interest area of the customers. The notification system can be implemented using different approaches. To improve delivery speed and provide full tracking of notifications, we have used the MQTT based mosquitto broker. In this paper, we have implemented push notifications using publish and subscribe facility provided by mosquitto i.e. MQTT based broker. Comparison based on experimental results for MQTT and HTTP has proved that MQTT is better than http for push notification.**

***Keyword* - MQTT, HTTP (Hypertext transfer protocol), GCM (Google Cloud Messaging), Push Notification, Publish/ Subscribe.**

I. Introduction

Notification is the instant way to notify the user about the activities and updates done on the particular e-commerce business. Almost all the sites and mobile applications provide the facility of notification to their customers to attract more users and increase the traffic on their site. Commercial sites have the major concern about how they can increase the number of customers on their site in order to make more profit. For attracting a large amount of traffic and more user engagement on the site, notification proves to be most efficient technique. Notification make the customer aware of the products, features and facilities which are newly introduced, and hence also proves to be the fastest and most effective way for advertisement. Most commonly used notification systems used by various commercial site is GCM i.e. google cloud messaging. GCM has some issues: *(a)* uses *http protocol* which has some limitations which are discussed in further section, *(b)* tracking of notification delivery is not possible, *(c)* actions performed by users cannot be tracked. For tracking, we have to customize the code and track the notification manually. To get rid of all these issues, we have implemented MQTT based push notification system. Tracking of the user activities for commercial sites is very important, i.e. not only to bring more customers on their site but also for satisfying the requirements of existing customer. Push notification can be very useful to provide the full featured tracking of the activities of a customer. This will provide the site to attract their customers more by providing them those details for which they are more interested. For the research purpose, we have used MQTT based mosquitto broker technique and has implemented a push notification system. For testing purpose, we have simulated 64K users and had tried sending notification to these users.

Mosquitto is a MQTT protocol based broker, which uses publish/subscribe architecture for sending notifications messages. *mosquitto_pub* and *mosquitto_sub* are the two mosquitto services, which are used for sending notification to the subscribed user. *mosquitto_pub* publishes the message on to the user for which the notification is meant and *mosquitto_sub* is the service through which a user is subscribed to a particular type of a service. Whenever there is some data which is of users concern, a notification is sent to the user that there is some data related to his/her requirements. Whenever, there is some changes in the information, server generates the request for the notification, which is passed to the mosquitto broker, which is responsible for the sending of the notification to the user machine.

In the web application, the server cannot initiate the sending of the notification until the client request for it. To resolve this problem, *http* are based on the concept of *http server push* which is also sometimes called *http streaming*. Consider the web application as the client-server architecture. Client will first send a request to the server to do specific task or operation. Server will hold request of the client until there is no data to be send to the client machine. As there is some data available, server will post it on the client machine and client will again send the request to server for the new data and hence the communication continues between client and the server. If the connection is not established, then server if generates any message, the server has to wait till the next request from the client is not received and stores the message in the queue. To avoid the continuous request requirement from the client, server establishes the connection and post the message on it directly.

MQTT makes use of the long polling. This is the process similar to the *http streaming*. In polling, the server waits for the client machine to request for the connection establishment.

In long polling, server will accept the connection, even if there is no data for server to post and hold the connection for a period of time till keep alive and sends message over it whenever, there is data to post on the client.

In MQTT, we are basically concerned with the *publish/subscribe* architecture as shown in Fig. 1. Different sources of notification generator are called *publishers* and receiver, who receives the messages are called *subscriber. Publishers* do not program the different messages to be sent to *subscribers* directly or indirectly. *Publishers* mainly publishes the messages without the knowledge of *subscribers.* Similarly on other hand, *subscribers* receives only those messages which is of their own interest, without any knowledge of any *publishers*. *Publish/subscribe* architecture uses mainly two type of messaging filter: -

- ***Topic Based System***: - *Subscribers* in a topic- based system will receives those messages that are published to the topics to which they subscribed, and different *subscribers* to that topics will automatically receive the same messages.
- ***Content Based System***: - Messages must be delivered to a *subscriber,* if the content of those different messages match specified constraints defined by all those *subscriber*. The *subscriber* is also responsible for classifying the different messages.

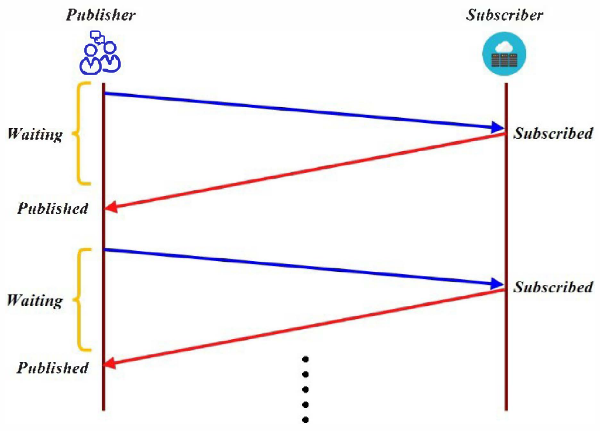

Fig. 1 *Publish/Subscribe Architecture*

II. RELATED WORK

There are number of researchers currently working on improving push notification technology. Some of techniques related to push notification system are mentioned in literature review. MQTT is generally developed by IBM. MQTT is lightweight broker that supports all kinds of platforms and used with different programming languages. MQTT is used by different researchers in order to implement push notification messaging protocol. Konglong Tang et al. [1] used MQTT broker for designing and implementing push notification system. Author has used publishing/subscribing facility of MQTT broker and solves the bottleneck of instant push of varieties of messages from server to client. MQTT protocol helps sending particular messages to specific smartphones with special support from proxy server.

Acer et al. [2] proposed network centric energy efficient scheduling mechanism that are used to provides delays to delivered messages by appropriate sensing. Due to delay of messages, author predicts user's mobile network related activities. Author uses Apple push notification server and Google cloud messaging server (GCM) in order to implement proposed approach. As according to proposed approach, author performs the experimental results on a network of 60 users for 30 days and they get significant reduction of energy consumption of 10% for delaying an average of 2 minute and 30 seconds. Yilmaz et al. [3] surveyed about timely performance of Google cloud messaging server (GCM). In most of notification system, GCM is commonly used for pushing instant messages. But nowadays, companies tries to create their own push notification cloud server. Author evaluate the performance of GCM by performing experiments on real time data as well as on thousands of real time user's. Author reveals that delivery of messages are unpredictable using GCM server and does not guarantees a timely arrival of messages to particular users. But GCM is still better because GCM uses random multicasting means a huge portion of messages are subscribed within limited time span (40% in 10 seconds).

Chetan D. Wadate et al. [4] proposed cloud-based push-styled mobile notifications that provides push services for sending messages notifications to the user and helps users to display all messages notifications on mobile smartphones. Awad et al. [5] uses push notification system for future shopping intentions purposes. For experimental analysis, author collects the data of customer having an aged in between 18 to 30 years and works on two of parameters *perceived usefulness (PU), and perceived ease of use (PEOU)* for finding the attitude and predict future intentions of young customers. Hansen et al. [6] uses push technologies on android platforms and uses different benchmarks test cases in order to find the performance four relevant push technologies named as Urban Airship, XMPP, Xtify and C2DM. Author performed analysis for three of parameters like response time, energy consumption and stability. Martin Pielot et al. [7] investigate in-situ analysis study using 15 mobile users and collect real time notification data from these users. Author found that each individual users deal with 63.5 notification average per day. Zhi Xu et al. [8] customize notification system that allows to install trojan application in order to start phishing attacks or start spam notifications. Author has performed customized push notification on androids, blackberry and IOS and found that all platforms are vulnerable. In order to prevent from these attacks, author proposed *semi-OS-controlled notification* principle and *notification logging services* and implement these approach on android platform.

III. IMPLEMENTATION DETAILS

A. Database Details

During implementation, database structure is consists of three tables *(1) device table (2) customer table, and (3) notification table*. In the *device table*, all the device related data are stored such as *device_id*, *device_type*, *device_version*, *app_installation_date*, and *device_status*. In *customer table*, details such as *cust_id* and other personal details of the customer like *first_name, last_name, address,* and *ph_no*. are stored. The third table is the *notification table,* in which the detailed tracks of the notification are stored. Notification further consists of two types of events – *(a) READ,* and *(b) SWIPED_AWAY*. These two status are of great importance for implementation purpose, as they enable system to analyse interest of customers. If a customer open, a particular notification that means he/she is interested in that kind of events, whereas if a customer ignore some message or swipe away some message, that means he/she is not interested in those kind of events. So with the help of these conditions, we can increase the number of the events related to his/her interest and decrease the number of events he/she is not interested in. So, the quality of suggestions to the customer will improve attracting more traffic towards the site.

B. Proposed Architecture

Fig. 2 shows the proposed architecture for push notification system. There is a server whose main purpose is to generate notification. Server hits the services. Services has three activities to perform, *1)* hits the mosquitto broker to send the notification and *2)* enter the notification details in the *notification table*. Mosquitto broker maintains a queue, where the undelivered notification are kept. Once, the notifications are delivered, they will be removed from the mosquitto broker queue. *3)* Then the notification is received on the customer device. There are very less or we can say negligible chances of notification being lost in the network. For surety of delivery of data over the network concept of *acknowledgement* is used.

Values used to represent the status of the notification are:

- 0 - undelivered
- 1 - delivered but not read
- 2 - delivered and read
- 3 - delivered and swiped away.

Status 0 implies that the notification could not be sent to the user machine because either the app is uninstalled or the internet connectivity is not there. Status of the notification remains 1 till no action is performed on the notification. If the notification is clicked and read, then the status changes to 2 and it implies that the user is interested in particular type of items as the notification is read by him. If the notification is swiped off that means the person is not interested in that type of notifications. By analysing these types of status, we can see the interest of the customer and can provide him opportunity to get more products of his/her interest and hence engaging him to use the app more, by which the company will make profit. Since the user will get only the appropriate suggestions and notification. So, it will attract user and user will feel that the app is personalized according to his/her requirement. Due to this, more of the people will install the app and less uninstallation would be there.

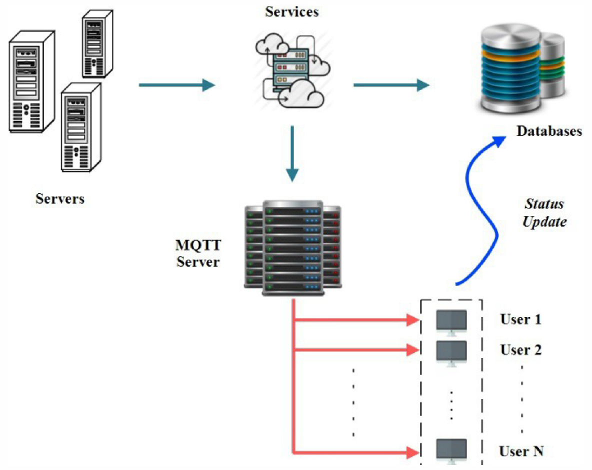

Fig. 2 *Proposed Architecture*

MQTT provides an efficient way to send the notification over http. Keep alive feature of MQTT provides better opportunity to send notification over http. In http, the connection lost is detected after waiting for a TCP timeout. But MQTT exchange the keep alive message very often to detect connection lost. For first time connection, MQTT involves more cost over the http connection. But in http since there is no keep alive feature. So, for every reconnect, http involves more cost as compared to MQTT. There are 3 cron scripts being used:

- ***1st cron script:*** - It runs once in 48 hrs. If a person has not used the app for more than 24 hrs then his status is marked as inactive. After every 24 hrs a notification is sent to all the inactive users. If they read the notification their status is active. But if they swipe away the notification, then they are active but are not interested in particular type of activity which was associated with this particular notification.

- ***2nd cron script:*** - It runs every 4 days. If a user is in Inactive status for more than 4 days and all the notifications sent in these 4 days are undelivered than the person is marked as super inactive. From this state the person can be active or doesn't exist. After every 4 days the notifications are sent to the SIA users. If notification are delivered and read than status changes to active and last active date also updates.

- ***3rd cron script:*** - It help us analyse that the person has actually uninstalled the app. It runs after every 7 days.

For mobile application the basic concern areas are:

1) response time
2) Throughput
3) Low battery usage
4) Low bandwidth
5) Less CPU usage

MQTT has rich features as compared to the http in terms of faster response time, throughput, lower bandwidth and battery usage. MQTT provides an assurance for the delivery of the message on to the receiver end.

IV. EXPERIMENTAL RESULTS

For simulation of notification system, we have used a database consisting of 64K users. For simulating 64K user, we have used 8 systems with each system having 8K devices being simulated on each using the shell script, in other words we can say that each system is acting as if 8K users are using the app at same time. Table I shows configuration of each system used for implementation purpose.

Table I Configuration of System

RAM	2GB
Processor	Intel Core 2 Duo CPU E7500 @ 2.93 GHz X 2
OS	Ubuntu Linux 32 bit
Direct X	DX11
HDD	240 GB

In experimental analysis, single user can login from different devices. Corresponding to a single user, there can be multiple *device_id*, but since if it is second time installation, then the details of the device will be updated, so single device will never can have more than one customer registered at a single instance of time. Table II shows amount of power or battery used during initial connection establishment to server and battery taken during maintenance of connection, which proves even if MQTT consumes more battery during initial connection, still it is better than http around 30% in terms of overall battery consumption. Since *http* has some drawbacks which are overcome by MQTT, and MQTT proves more efficient for the mobile applications and hence it can be concluded that MQTT based mosquitto server provide us a better opportunity over *http* based GCM server. MQTT is more reliable over *http*. Table III shows the comparison of results obtained during simulation and proves, there are more chances of notifications being lost, if sent using GCM as compared to mosquitto. Fig. 3 shows the comparison of battery consumption for MQTT and *http* for initial connection establishment and Fig. 4 shows battery consumption during maintenance of connection.

Table II. Amount of power used during connection establishment and connection maintenance

Keep Alive(Sec)	% Battery/Hour	
	3G	
	http	MQTT
Initial connection	0.02	0.04
60	1.1	0.7
120	0.46	0.34
240	0.31	0.19
480	0.14	0.05

Table III. Comparison of number of notifications received by GCM and MQTT

Number of notification generated	Number of notification delivered by GCM	Number of notification delivered by MQTT
1000	800	998
2000	1700	1907
3000	2500	2910
4000	3400	3879
5000	4500	4890

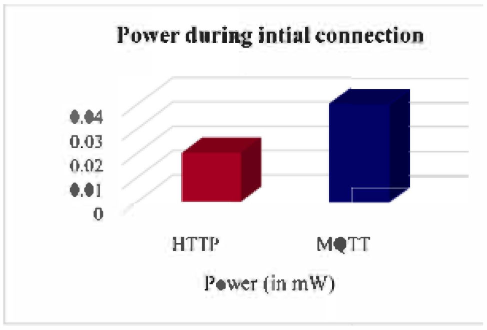

Fig. 3 Amount of power or battery used (in connection establishment (in seconds)

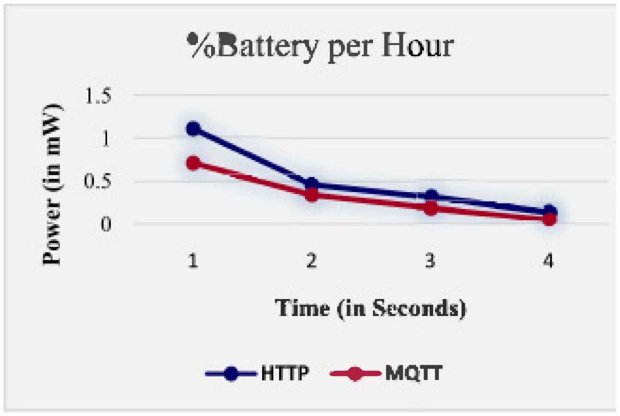

Fig. 4 Amount of power or battery used (in maintianence (in seconds)

V. FUTURE WORK

As e-commerce sites are gaining more and more popularity, so is the notification system. To enhance the quality of the existing notification system, we are planning to implement the customized notifications and will analyse the effect of these notifications for user engagement on the application.

REFERENCES

[1.] Konglong Tang, Yong Wang, Hao Liu, Yanxiu Sheng, Xi Wang and Zhiqiang Wei, *Design and Implementation of Push Notification System Based on the MQTT Protocol*, International Conference on Information Science and Computer Applications (ISCA 2013), pp. 116-119, 2013.

[2.] Utku Günay Acer, Afra Mashhadi, Claudio Forelivesi and Fahim Kawsar, *Energy Efficient Scheduling for Mobile Push Notifications*, 12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, 2015.

[3.] Yavuz Selim, Yilmaz Bahadir and Ismail Aydin Murat Demirbas, *Google Cloud Messaging (GCM): An Evaluation*, Globecom 2014 - Symposium on Selected Areas in Communications: GC14 SAC Internet of Things, pp. 2847–2852, 2014.

[4.] Chetan D. Wadate, Prashant T. Suvare and Aniket S. More, *A Survey of* Automatic *Wi-Fi based Push Notification in College Campus using Cloud*, International Conference on Advances in Science and Technology (ICAST-2014), pp. 1-4, 2014

[5.] Dr. Tamer A. Awad and Dina Ashraf El-Shihy*, Assessing the Effect of Consumers' Profiles and Attitude towards Push Notifications and Future Shopping Intentions*, Global Journal of Emerging Trends in e- Business, Marketing and Consumer Psychology, Vol. 1, No. 2, pp. 58- 93, 2014.

[6.] Jarle Hansen, Tor-Morten Grønli and Gheorghita Ghinea*, Towards Cloud to Device Push Messaging on Android: Technologies, Possibilities and Challenges*, Int. J. Communications, Network and System Sciences, Vol. 5, pp. 839-849, 2012.

[7.] Martin Pielot, Karen Church and Rodrigo de Oliveira, *An In-Situ Study of Mobile Phone Notifications*, MobileHCI 2014, pp. 1-10, 2014

[8.] Zhi Xu and Sencun Zhu, Abusing Notification Services on Smartphones for Phishing and Spamming, WOOT 2012, pp. 1-11, 2012.