On Measuring Cloud-Based Push Services

Wei Chen, Nanjing University of Posts and Telecommunications, Nanjing, China
Shiwen Zhou, Nanjing University of Posts and Telecommunications, Nanjing, China
Yajuan Tang, Shantou University, Shantou, China
Le Yu, Nanjing University of Posts and Telecommunications, Nanjing, China & The Hong Kong Polytechnical University, Hong Kong, China

ABSTRACT

The push services provide an incredibly efficient and reliable way of sending information to mobile device users. However, very little is known about how push service providers implement their services. This understanding is valuable as a guideline for building well-performing mobile applications. In this paper, the authors propose a methodology to assess push services and conduct the first systematic study on evaluating and comparing 4 popular push services (i.e., GCM, Gexin, JPush, and ZYPush). They dissect the protocols used by these services and discuss the implications. Moreover, they measure and compare their performances, including server distribution, message delay, heartbeat frequency, data compression and traffic consumption. The experimental results show that GCM outperforms others and JPush is one of the best providers in China. The authors’ methodology and results are useful as both benchmark and guideline for mobile applications development.

KEYWORDS

Heartbeat Packet, Measurement, Mobile Internet, Push Service, Smartphone

1. INTRODUCTION

Today push services play an important role in smartphone applications (apps) and they disseminate billions of messages through their gateways on a daily basis. Push services deliver push messages notifying users to news, chatting messages, ads and promotions. Adoption of push message can remind users of an app’s existence and encourage its continued use. They can generate personalized notifications and satisfy the real needs and interests of an individual subscriber. Personalized notifications depend on contextual information, such as a user’s location, time, or activity.

Push technologies are suitable for developing mobile apps to keep their users informed of updated changes. Push services deliver content from a push initiator on an application server to a push-enabled app on mobile device. They can reliably deliver notifications to apps running on different mobile
operating systems and provide low latencies on the order of a few seconds. With the development of various mobile platforms, there are different deployments of this push service, such as Google’s Cloud Messaging (GCM) service for Android (Google cloud message, 2014), Apple’s Push Notification Service (APNS) for iOS (Apple Inc., 2014) and Microsoft Push Notification Service (MPNS) for Windows Mobile (Microsoft Inc., 2014). With push technology, the push initiator doesn’t need to wait for a request from the push-enabled application to deliver the content. The push services provide an incredibly efficient and reliable way of sending information to mobile device users. They notify users of information in a timely manner, thus offering real-time and always-on experience for users.

Besides GCM, APNS and MPNS, there are many third-party push service providers, which enable mobile application developers to send notifications to various mobile applications. For examples, Amazon provides Simple Notification Service (SNS) (Amazon inc., 2014), a web service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients. Airbop (Airbop, 2014) simplifies the complexity of developing push notification applications based on GCM. Gexin (Gexin, 2014), JPush (JPush Service inc., 2014), and ZYPush (ZYPush, 2014) are popular push services providers in China and widely used by many mobile apps.

However, very little is known about how these providers design and implement their services and the implications of different designs. This understanding is valuable as a guideline for building well-performing mobile apps. Properties of push services play a key part in developing apps, which include client handling of incoming notifications, gateway policies for storing notifications, configurability for QoS, and runtime operation. The design features and performance, including server distribution, message delay, heartbeat frequency, data compression and traffic consumption, affect the choice of third part mobile apps developers.

We investigate the protocols employed by different service providers through capturing traffic from both push initiator side and push receiver side. We also propose a methodology to evaluate the performances of push services and apply it to compare 4 popular services, revealing differences on message delay, heartbeat frequency, traffic consumption etc. We benchmark each selected service under the same conditions, highlighting differences manifested in various usage scenarios. These comparisons provide guideline for mobile app developers to select appropriate push services. To our best knowledge, we are the first to conduct a systematic evaluation on real push services and examine the implications.

Our results reveal some interesting insights:

- The heartbeat protocol design is important to the performance of push services.

  Long connections usually have less pushing delay than short connections. The frequency of sending heartbeat messages affects the traffic consumption. We observe that some long connection based push services, such as GCM, use less heartbeat traffic than the short one while still keep shorter push delay. It is because GCM selects an appropriate heartbeat interval that is suitable for their global server deployment. On the contrary, ZYPush, a short connection based push service, sends heartbeat packets rather frequently, thus leading to unnecessary traffic consumption.

- Heartbeat reusability can effectively reduce the heartbeat traffic by sharing heartbeat packets among apps in the same mobile devices. For example, Facebook, Gmail and Hangouts share the same GCM heartbeat connection. However, some push services, such as JPush, do not share the heartbeat information among different apps. Each JPush-based app establishes a separated connection, through which heartbeat packets are sent. Such design can dramatically increase heartbeat traffic when the number of apps grows.

- Some push services do not support secure communications (e.g., HTTPS) and therefore have potential security vulnerability. For example, some services only support HTTP POST method to submit push requests in plaintext. These requests usually include sensitive secret information,
An Adaptive Approach to Optimizing Tradeoff Between Service Performance and Security in Service-Based Systems
www.igi-global.com/article/adaptive-approach-optimizing-tradeoff-between/55237?camid=4v1a

Selective Service Provenance in the VRESCo Runtime
www.igi-global.com/chapter/selective-service-provenance-vresco-runtime/66968?camid=4v1a