Date of Acceptance Grade

Instructor

Overview and Evaluation of Notifications Systems for Existing M2M Provisioning API

Linhong Sun

Helsinki November 4th, 2014

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta/Osasto–Fakultet/Sektion–Faculty/Section	Laitos–Institution–Department
Faculty of Science	**Department of Computer Science**

Tekijä–Författare–Author

Linhong Sun

Työn nimi–Arbetets titel–Title

Overview and evaluation of notifications systems for existing M2M provisioning API

Oppiaine–Läroämne–Subject

Computer Science

Työn laji–Arbetets art– Level	Aika–Datum–Month and year	Sivumäärä–Sidoantal–Number of pages
M.Sc. Thesis	**November 4th, 2014**	**87 pages + 71 appendixes**

Tiivistelmä–Referat–Abstract

Nowadays, the connected devices and services need differentiate tailored connectivity to satisfy diverse requirements. To simply provision of a large number of connected devices in a time-efficient way, Ericsson provides a SaaS solution Ericsson Device Connection Platform (DCP). Hence, a notification service is required to achieve real-time interaction, and a subscribe-notify web-services API is required to define the selected notification service based on DCP. However, it is difficult to choose a notification system due to lacking systematic approaches. In the view, I present six comparison points to compare current push notification products. My aim is to propose proposals to design a subscribe-notify web-services API. In an effort to further understand how push notification mechanisms work, I (1) explain Event Notification Service (ENS) technologies; (2) recommend to use the publish/subscribe (pub/sub) paradigm to define the API for DCP; (3) compare current push notification products by using six comparison points as the systematic approach. Six comparison points are basic information, functionalities, licenses with usage fees covering usage limits, security, supporting details and supported platforms/OS in terms of performance. In the case, Pushwoosh and AeroGear UnifiedPush Server provide appropriate notification systems to satisfy diversified demands for different scale enterprises.

ACM Computing Classification System (CCS):

Networks → Network architecture → Programming interface
Information systems and applications → Decision support systems

Avainsanat – Nyckelord – Keywords

Device Connection Platform (DCP), Push notification services, Push notification servers, publish/subscribe (pub/sub), API, Machine-to-Machine (M2M), Notification systems

Säilytyspaikka – Förvaringställe – Where deposited

Table of Contents

1 Introduction

The emergence of Machine-to-Machine (M2M) technology enables various devices to be connected across the world. It is predicted that 50 billion connected devices is a vision of the future, and more devices are using myriads of new services. Based on various requirements, these connected devices and services need to differentiate tailored connectivity packages. For example, some devices need constant connectivity and others may occasionally transfer data. In such a way, the differentiated connectivity brings benefits to all participants. Concisely, application users pay reasonable charges based on their usages, in the meanwhile, service providers utilize usage per bit distribution of network. However, provisioning a large number of connected devices is time consuming and complicated. To tackle such a task, Ericsson provides a SaaS solution called Device Connection Platform (DCP) to support the provisioning task real-timely. An API is playing a significant role in supporting diversified services and responding to competitive pressure. The core service platform of DCP is required to be exposed through APIs to both external parties and their customers. Hence, a subscribe-notify web-services API, which is able to work with third party services, should define selected notification services based on DCP to achieve real-time interactions. To design such an API, a systematic approach is required to compare current push notification products based on existing third party push notification products. Besides, proposals of such an API are required to be given.

To analyze current push notification products, four characteristics of M2M communications, including networking conditions, mobility, scalability and fault tolerance are considered. In an effort to further understand how push notification mechanisms work, this paper is conducted to explain Event Notification Service (ENS) technologies. Then, the publish/subscribe (pub/sub) paradigm is chosen to define the subscribe-notify web-services API of notification system for DCP. Based on the characteristics of M2M communications and ENS technologies, this paper presents six comparison points to analyze existing push notification products in a systematic way, and then gives proposals to meet diversified demands. Particularly, six comparison points are basic information, functionality, licenses and usage fees with usage limits, security, supporting details and supported platforms/OS. The objective of the systematic approach covers both

performance and practical cases.

The rest of the paper is organized as follows. Section I is an introduction to this paper. Section II is background which is M2M environment and user needs for M2M devices. Section III introduces DCP from the aspects of definition, user scenarios and key characteristics. Section IV overviews ENS technologies based on web service technologies and a good API definition. Then this paper gives an investigation of available notification paradigms, existing push notification services and an ENS prototype based on pub/sub paradigm. The investigation concentrates prevalent android, iOS and Windows Phone mobile platforms for different use cases. Section V categorizes existing push notification products into push notification services and push notification servers. Then, it shows comparisons of the existing push products from six comparison points. Section VI discusses and evaluates preferred options of designing the subscribe-notify API. Section VII gets a conclusion and propose a final proposal to design the subscribe-notify web-services API.

2 Background

In this section, I will introduce four key factors of M2M environment in order to see what are required from an M2M provisioning API used in a DCP notification system. Throughout this paper, service providers or application developers refer to publishers, while application users refer to subscribers.

There are billions of devices in the world, ranging from home appliances and business computers to complex utility equipment, industrial machines and transport systems. The emergence of M2M technology now allows businesses to talk to their machines over wired or wireless networks. M2M is a broad label that can be used to describe any technology which enables networked devices to exchange information without human operations. With M2M, unexpected occurred problems can be monitored, identified by issuing automatic alerts remotely. Thus, for a machine builder or a machine owner, and whatever industry, M2M communications will bring value to prosperous business. Enterprise representatives like Google, Ericsson, etc., have followed the fast development in cloud, virtual and software defined structures. In the future, M2M will bring prosperities to IT springing up, and M2M tends to be predominant in the following years.

This section will present four general characteristics and requirements of M2M environment, including networking conditions, mobility, scalability and fault tolerance. These characteristics should be considered when designing a notification system and customizing native applications for mobile devices.

2.1 Networking conditions

When machines talk, they communicate in the same language which is known as telemetry. The concept of telemetry refers to remote machines with sensors which collect and transfer data to a central point for analysis, and it can be done either by humans or computers. M2M communications are realized by wireless sensors and Internet in M2M environment, so M2M environment is affected by networking conditions.

The fundamental reasons why M2M environment is affected by networking conditions can be analyzed from lower layers of network. In order to support global software development, networks spanning across several sites are necessary. However, transitions in these networks are often slow and unreliable, because network packages are lost in the transport. The problem of lost packages results in unreliability, the low bandwidth, the low throughout and even underutilized or overload of Internet. Network Address Translation (NAT) [Smi02] devices modify the IP address information of the traffic when going through the NAT devices, which is another case of package lost problem for notification systems. NATs allow Internet connectivity between a public network with the same public IPv4 address and several devices. The original prototype of IP end-to-end connectivity is changed and causes more disruptions for traffic.

The problem of lost packages is solved by firewalls. Network packages can also be intercepted by network security systems, either software-based or hardware-based. A firewall takes responsibility of network packages analysis and receiving and sending packages control based on given rules. Hence, a firewall succeeds in creating protection for a secure trusted internal network from an unknown network.

To get better control of M2M technologies, M2M communications can be used more efficiently to monitor the condition of critical infrastructure, such as water treatment facilities or bridges, and helps businesses maintain inventory. As networking conditions rely on common technologies,

networking conditions can help a homeowner maintain a perfect lawn or create a shopping list at a button's touch as well.

2.2 Mobility

Wireless communication is ubiquitous nowadays, and service providers are making a breakthrough to support increasingly global supply chains. Real-time communication which is a form of wireless communication helps to narrow gaps in information, so that both application users and service providers like manufacturers are able to track and monitor their supply chains remotely.

Some automotive companies have increased their demands for wireless technology on alert emergency services towards accidents. This M2M evolution to alert emergency service is needed to be created based on a comprehensive architecture. IoMANET is such an architecture, which supports indirections and solves mobility issues for M2M communications [Att11]. Mobility is also required to overcome the difficulty in physical distances, so that remote counterpart components in M2M environment can be located and recognized inaccuracy in time [Bel96]. Informal communication are avoided, because informal and ad-hoc communication are dominate in detecting exceptions, correctness and recovery [Des02a]. Hence, mobility is essential factor of M2M communications especially in the large scale software development environment. Besides, mobile service providers have a challenge to provide mobility for distributed mobile applications. Because distributed mobile applications usually interact with other mobile groups which aren't in the same administrative domain in most cases. These mobility issues are argued in the edge Internet with corresponding solutions [Zha09].

Mobility is driving M2M adoption among service providers who are exploring better ways to manage supply chains. Distributed mobile service providers can support alert emergency services with mobility to connect remote counterpart components in M2M environment. Because of flexibility and ease of use, M2M is becoming one of the fastest growing manufacturing trends. Furthermore, the demand for M2M is still rising, because M2M involves mobile application users, and the demands of mobile application users drive M2M adoption especially for automotive companies.

2.3 Scalability

Ericsson predicts 50 billion users in the future [Eri11]. However, according to research from Cisco[1], the amount of Internet-enabled devices in the world will surpass 150 billion by the year of 2025. In the meanwhile, Samuel Ropert in his report [Rop13] said that there would be 80 billion connected devices by 2020, where 85 percent would include anything that connects to the Internet, even if it lacks the required electronics and relies upon intermediate devices.

The Internet-scale event notification is a trend [Des02a], and currently is implemented by the majority of notification servers. Siena [Car01] is taken as an example, Siena supports Internet-scale distributed applications. Herald [Cab01] is another example, which supports scalability and brings flexibility.

Furthermore, scalability brings benefits to the integrations, and enhances the communications of information or events exchanging between different distributed sites.

2.4 Fault tolerance

Fault tolerance reflects the ability of handling communication failures, so that reliability and stability of application users' networks and M2M communications are guaranteed. Failures can be caused by various reasons, such as power surges, static discharge, communication failures and brownouts, like lost communications, high maintenance costs, and truck roll outs, longer down times, loss of remote control and automation. Hence, it is practically useful to keep M2M communications to continuously interconnect between nodes when failures happen.

IoMANET is designed based on Chord protocol [Att11], whose robust scalable topology is implemented based on the Chord distributed hash table. Chord is able to automatically change the internal tables with existing nodes, including failure nodes. IoMANET not only solves the mobility issues of M2M communications with indirections, but allows message delivered independently from a dedicated home agent in order to support fault tolerance.

As M2M applications are deployed across critical infrastructures [Mer11], it is significantly

[1] http://newsroom.cisco.com/dlls/2010/ekits/Evolving_Internet_GBN_Cisco_2010_Aug_rev2.pdf

useful for M2M communications to support fault tolerance. M2M evolution management requires a comprehensive architecture and innovative technology solutions. Hence, it is significant that preferred solutions should eliminate single points of failures to perform in a distributed and robust way.

2.5 User needs

Application developers intend to apply an event notification system as a framework to support global software development. Both application developers and application users prefer well-known applications, such as instant messenger systems and application gauges [Des02b] and workflow systems [Cug01] to integrate notification systems. They want to use these applications for informal communication as collaborative tools. To make full use of event notification systems, application users should know the details of the events, so that the events can be captured and passed to the notification server in time.

3 Ericsson Device Connection Platform

With the emergence of new resolved opportunities, an increasing number of devices are communicating over the global networks. Ericsson proposes DCP which introduces a simple way to reducing revenues on device average and gaining profits.

DCP serves as a service. DCP is designed to allow service providers to address M2M or M2M connectivity opportunities, which accelerates expansions of revenues with minimum risks. DCP realizes functionalities of subscription management, rating with billing & policy controls, provisioning with order management, IP controls with policy enforcement, connectivity monitoring, device management and device access enablement regardless of access points and virtual private networks.

DCP uses an existing M2M provisioning API, so that DCP can be integrated with the existing operational supporting systems and business supporting systems for complete end-to-end functions. DCP enables service models to reduce delivery time to market and give a quick glance at locations on new elements. The automation of this business processes bring high operational efficiency and keeps overall costs down.

As dedicated to M2M connectivity, DCP enables service providers to focus on their enterprise customers rather than an uncomplicated technology. The portal brands DCP and it is preferred the functionality can be made available via APIs. Using the self-service portal, enterprises can manage their entire device fleet and can get easy access to reports and statistics. The devices access enablement helps smaller businesses to guarantee individual devices accessible over the Internet. Accessibility is provided in the secure application protocol acknowledgment without opinions of access points and virtual private networks.

In addition to DCP, Ericsson also provides other products in the form of services which are the key parts of M2M solutions. This includes system integration, network performance with scalability, M2M business consulting and industry specific solutions. By providing an opportunity of a low risk-development and highly-tailored M2M solutions, DCP creates values and opportunities for service providers in a long term.

3.1 Use cases in DCP notifications

Notifications are triggered by events like various provisioning requests. Even though, initial orders of subscriptions and related SIMs are left out-of-scope currently.

Notifications are user-definable, which means application users are able to choose the events. The events interest application users and become parts of notifications. Publishers publish notifications to subscribers, and subscribers receive notifications from publishers. The pub/sub process can be referred to anything from SMS text messages or e-mails to anything more synchronous. Practically, subscribers do not have to be recipients, and even have one or more recipients with one or more devices can be notified by the same notification.

In DCP, use cases are summarized in four situations. In general, application users will be notified in when operations, including changing subscription status, changing subscription package, doing batch processing and triggering alarms, are finished or have errors in DCP. In details, application users are able to choose multiple events are notified when subscription status changes in the server, when enterprises issue subscription status changed. The difference between before subscription status changed and after subscription status changed lies in how much service providers pay for bills. Because invoices depend on subscription status, and subscription status is

different among alive, deactivated, paused or terminated. Application users are also able to choose multiple events are notified when a subscription changes from a subscription package to another subscription package in server side. Again, application users can choose multiple events are notified when many subscriptions change from a subscription package to another subscription package in server side. Moreover, application users are able to choose when multiple events are notified for triggered alarms. Alarms are triggered when application users receive too many SMS text messages or too few SMS text messages. Application users receive too many SMS text messages, data will be above timeframe, and network can be overload. If application users receive too few SMS text messages, data will be under timeframe. This can be caused by some network problem or have not been used efficiently. One more user case of trigger alarms is devices connect/disconnect from networks, so that PDP context is active or deactivated.

4 Event Notification Service technologies

Event notification service uses web service technologies to transfer and manage event notifications. Subscribers initiate requests with subscription information in a specified format, and subscribers send requests to web service providers to register their interests. When interesting events happen, registered subscribers will receive notifications in a specified format. Notifications are delivered with diversified transportation mechanisms via intermediary. To get better interoperability among different service providers, service providers claim specifications or rules to illustrate notification formats and web service rules for notification delivery. The benefits of specifications and rules will bring convenience to subscription creation and event notification service management [Hua06].

The most widely used web services are RESTful web services which are also used by event notification services. RESTful web services use technologies like protocols, architecture and formats to design and implement. The following subsection gives a rough introduction to the concepts of API, Hypertext Transfer Protocol (HTTP), Hypertext Transfer Protocol Secure (HTTPS), Extensible Markup Language (XML), JavaScript Object Notation (JSON), SOAP and REST. All these web service technologies are used to realize the communications of RESTful web services which will be mentioned in Section 5.

4.1 Web service technologies

A web service is a software ecosystem to support interoperable M2M interactions over networks [Moj12]. From the whole view, a web service gives an interface in a sequential machine style like Web Service Definition Language (WSDL). Communications between systems and web service are prescribed in a manner by using SOAP messages, and typically delivered via HTTP in an XML serialization under other web standards. The most used web services are Remote Procedure Calls (RPC) which is based on a distributed function call interface, Service-oriented Architecture (SOA) and REST. SOA is the basic unit of communication, which is a message rather that operations as RPC. The architecture of REST combines HTTP with web services to constrain interfaces via a set of standard HTTP operations.

API is the abbreviation of application program interface, which goes through all the web service technologies. API is a set of routines, protocols and tools to build software applications. API specifies how software components interact and APIs are used to programme graphical user interface components. A good API is significant for both service providers and application developers. For service providers, a good API could become the most precious property for enterprises. The API would gain engagement in purchase. For application developers, modular and good code should be developed based on the API. A good API makes it easier to develop a program by providing all the building blocks. A programmer then puts the blocks together. Useful blocks even can be designed to be reused later on.

Based on Joshua Bloch's talk [Blo12], a good API has six general characteristics, including easy to learn and easy to use even without any documentation, difficult to misuse, easy to read and maintain code, appropriate enough to satisfy the requirements, easy to extend and appropriate to almost all the application developers. A good API should be learned and used easily even without any documentation, namely, a good API is simple and plain to be understandable. So that it can be memorized easily. Moreover, a good API should be more difficult or impossible to misuse. Such a good API should simply force you to do the right. A good API should be easy to read and to maintain code written to the API. Besides, a good API should be adequate to satisfy the requirements of application developers. Such a good API doesn’t have to be the most powerful API, but should be a sufficient appropriate one. A good API should evolve overtime, because

new requirements will be added later on so that application developers are able to do what they want to write. Hence, a good API should satisfy both current and future requirements of application developers. Moreover, a good API has to be appropriate to almost all the application developers, which means a good API should have a common terminology accepted by almost all the application developers.

When used in the context of web development, an API is typically defined as a set of HTTP request messages. An API is usually in an XML or JSON format along with a definition of the structure of response messages. While web API historically has been virtually synonymous for web service, the recent trend so-called Web 2.0 has been moving away from SOAP based web services and SOA towards more direct REST style web resources and resource-oriented architecture (ROA). The mentioned concepts, including HTTP, HTTPS, XML, JSON, SOAP and REST which are relevant to API, will be introduced as follows.

HTTP is an application protocol for collaborative and distributed hypermedia information systems [Fie99]. HTTP is the foundation of data communication for the World Wide Web with hypertext structured by logical links. HTTP works as a protocol to exchange or transfer hypertext. While, HTTPS is a communications protocol for secure communication over a computer network, with especially wide deployment on the Internet [Res00]. Technically, it is not a protocol in and of itself, but is the result of simply layering HTTP on top of the SSL/TLS protocol. Thus, HTTPS succeeds in adding security capabilities of SSL/TLS to standard HTTP communications. Besides, data formats of request messages over HTTP and HTTPS are XML or JSON. XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable [Bra09] produced by W3C. While, JSON is an open standard format, and JSON uses easy readable texts to transfer messages which consist of attribute-value pairs [Ecm13]. JSON is used to transmit data between servers and web applications primarily, which is an alternative to XML.

Service-oriented architecture paradigms SOAP and REST combine all the concepts of web service technologies in a coherent way. SOAP, a W3C standard which is backed by all dominant vendors, including BEA Systems, IBM, Microsoft, and SUN. SOAP allows exchanges of information between peers in a decentralized, distributed environment [Gud07]. Moreover, a huge and growing number of protocols enhance SOAP with advanced features such as reliability,

security, and transaction support, or standardize application-oriented procedures for complex interactions among businesses [Shi06].

In contrast, REST is neither a standard nor promoted by any vendor [Ric07]. REST stands for Representational State Transfer, and REST has been first described in Roy Fielding's PhD thesis [Fie00] as an architectural style for distributed hypermedia systems. Fielding, co-founder of the Apache HTTP Server project and one principal author of the Hypertext Transfer Protocol, attributes abilities of Web to sustain its exponential growth. Principles of the REST architectural style are simple interfaces, visible communication, portable components when program code changes with data, reliability for failure resistance, scalable deployment, overall performance. Hence, it is enticing to use the same principles to building web services [Shi06].

4.2 Notification paradigms

Application users of mobile devices may have applications installed that depend on remote services. Notification messages typically represent events of interests defined by applications. For the notification paradigms, the following subsection distinguishes between polling and pushes notifications, and then goes further for push notification as pub/sub technology.

4.2.1) Polling

Polling [Mar99] is a pull-based invocation pattern of notification for application users' side (client). New information is received from an external source or service and buffered at a central server. The pull-based invocation pattern is based on the request/response paradigm, where the client sends a request to the server and the server answers either synchronously or asynchronously. The client periodically establishes a connection to application developers in server side to ask for information updates as shown in Figure 1.

Polling is functionally equivalent to the client pulling the data off the server. In the polling, the data transfer is always initiated by the client like a manager. Polling is commonly used in Input/output (I/O) processes. Polling is the simplest way to ask, and it is currently used by DCP.

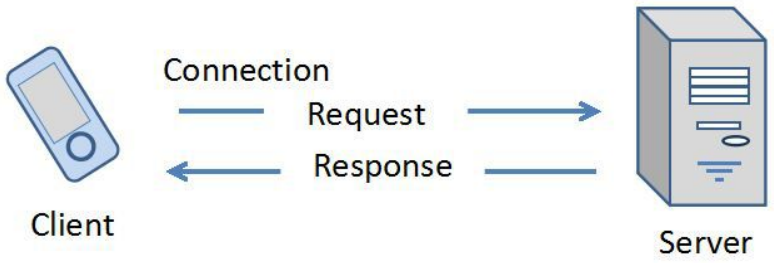

Figure 1: Overview of polling data flow.

4.2.2) Push notifications

Push notifications [Bur13] is a push-based invocation pattern based on push technology as opposed to polling. Push is a style of communication, the requests of push for a given transaction are initiated by application developers, where application users first establish a connection to its central server. A common technique for push notifications is to use a modified TCP connection and open a socket with longer timeouts. This connection is kept alive continuously, and independent from notification messages. Routers drop connection entries after a timeout if no new packets are transmitted. Application users must regularly send keep-alive messages to the server. Once the server obtains updated information for application users, it can be transmitted immediately as depicted in Figure 2.

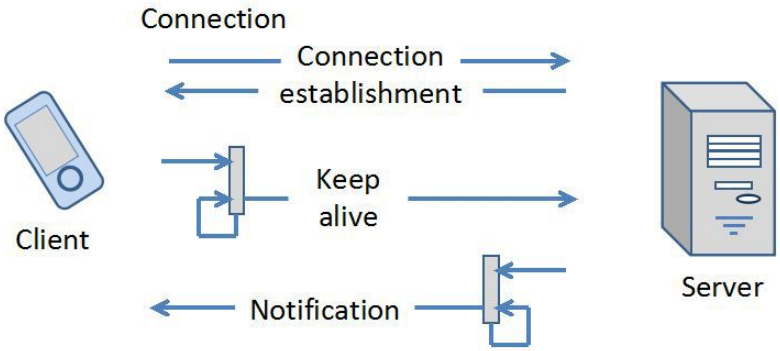

Figure 2: Overview of push notifications data flow.

In details, push notification [Guo13] is a technology based on some certain technical standard or regulation to reduce information overloaded by transmitting users' information on the Internet regularly. The key is that push notifications can transmit latest information to corresponding

application users' devices automatically according to the application users' demands, which will certainly change the way of obtaining information effectively. Push notifications are SMS-like messages [Moj12]. Unlike their GSM counterparts, push notifications are addressed to an application not to a device or a phone number. They also typically provide richer set of meta data and travel over data networks.

Depending on whether the information is customized by application users and whether it's finished by the client or server side, push technology can be realized in three different approaches [Guo13], blanket push, filtered push and pub/sub push. Blanket push sends push notifications to users without the help of filter or user selection. Most client software of blanket push is free of charge. Blanket push has multiple ways provided for information, which is another advantage. However, high frequency access to the Internet causes great consume of the broadband, advertisement appears along with Web pages, application users can't send internal data by push structure and data cannot be customized by application users. Only information of a certain channel customized by application users can be pushed in a filtered push. Filtered push allows internal application users to customize external information. However, the internal server also needs more management and auxiliary tools for the object in urgent requirements. Pub/sub push is a push technology used when a enterprise combines the internal data with the external data from information suppliers to form classifications. The form classifications are subscribed by internal application users, which are achieved by customized information in the server side and are real push notifications.

The advent of push notifications provides possibilities for the public to achieve information from Internet efficiently [Guo13], and this is why it has attracted general attention of the public. Compared with the polling in traditional Client/Server (C/S) system, Push technology possesses characteristics such as initiative, individuation, customized service of application user's contents, high intelligence and great efficiency.

4.2.3) Pub/sub paradigm

Pub/sub paradigm [Eug03] is a notification service pattern which allows publisher and subscribers to communicate with a single entity. Pub/sub paradigm is a logical intermediary propagation mechanism. Publishers are the components who sent notification messages, while

subscribers are the components who show their interests. The notification messages are the communication medium or subscriptions in the notification service pattern. All the subscriptions associated with the respective subscribers are stored in the notification service. All the notifications will be sent from publishers and will be dispatched to the correct subscribers. Then, the published notifications will be received by subscribers. In the pub/sub paradigm, publishers and subscribers deliver notification messages to each other without directly knowing each other.

Practically, a pub/sub paradigm makes publishers to publish events and subscribers to subscribe subscriptions about their interested events. The major objective of a pub/sub system is to propagate messages from publishers to interested subscribers in an anonymous decoupled fashion. Receivers are not directly oriented to publisher but indirectly targets to the content of notifications. Subscribers address their interests by publishing subscriptions, which is independently from the other components and processes.

Besides, the method of notification delivery or message filtering between publishers and subscribers is another feature of the pub/sub system. Concisely, the pub/sub system has three ways, topic-based, content-based and hybrid of them two.

For the topic-based pub/sub system, messages are published to topics or named logical channels. For the content-based pub/sub system, messages are only delivered to a subscriber if the attributes or contents of those messages match constraints defined by the subscriber. For the hybrid pub/sub system, publishers post messages to a topic while subscribers register content-based subscriptions to one or more topics.

The pub/sub system has several features of loosely coupling, scalability and security. Publishers and subscribers are loosely coupled, in other words, publishers and subscribers don’t have to know each other. When the topic of pub/sub systems is emphasized, publishers and subscribers can be allowed to ignore the topology of pub/sub systems. Both publishers and subscribers work independently from each other. In the traditional and tightly coupled C/S systems, the client can’t post messages to the server. While the server process is not running, nor can the server receive messages unless the client is running. Many pub/sub systems decouple not only the locations of the publishers and subscribers, but also decouple them temporally. A common strategy used by middleware specialists with such pub/sub systems is to take down a publisher to allow the subscriber to work through the backlog in a form of bandwidth throttling.

Pub/sub has better scalability than traditional C/S, using network-based and tree-based routing protocols, message caching, parallel operation and so on. However, in certain types of tightly coupled and high-volume enterprise environments, current vendor systems often lose the benefit that systems scale up to become data centers with thousands of servers sharing the pub/sub infrastructure. Hence, scalability of pub/sub systems under high load is a research challenge currently.

Outside of the enterprise environment, on the other hand, the pub/sub paradigm has proven its scalability to volumes far beyond those of a single data center, providing Internet-wide distributed messaging through web syndication protocols such as RSS and Atom [Ori11]. These syndication protocols accept higher latency and lack of delivery guarantees in exchange for the ability for even a low-end web server to syndicate messages to potentially millions of separate subscriber nodes.

The pub/sub system supports security and privacy. However, a subscriber might be able to receive data which is not authorized to receive. An unauthorized publisher may be able to introduce incorrect or damaged messages into the pub/sub system. This is especially true when systems broadcast or multicast their messages. Encryption, like SSL/TLS, can avoid unauthorized access but cannot prevent damaged messages from being introduced by authorized publishers. Architectures other than pub/sub, such as C/S systems are also vulnerable to authorized message senders who behave maliciously.

4.3 Push notification mechanisms for different mobile platforms

All the push notification mechanisms of different platforms are combined together in Event Notification Service (ENS) as depicted in Figure 3. The rough idea of such an ENS is that ENS works as a notification hub and uses the basic device registration with subscription information to trigger events. ENS realizes sending notifications to APNS, GCM, MPNS, web portals and even other receivers. However, devices need to be registered first, then can get channel URLs and send channel URLs to ENS, so that payloads can be sent to subscribers via ENS to realize push notification service.

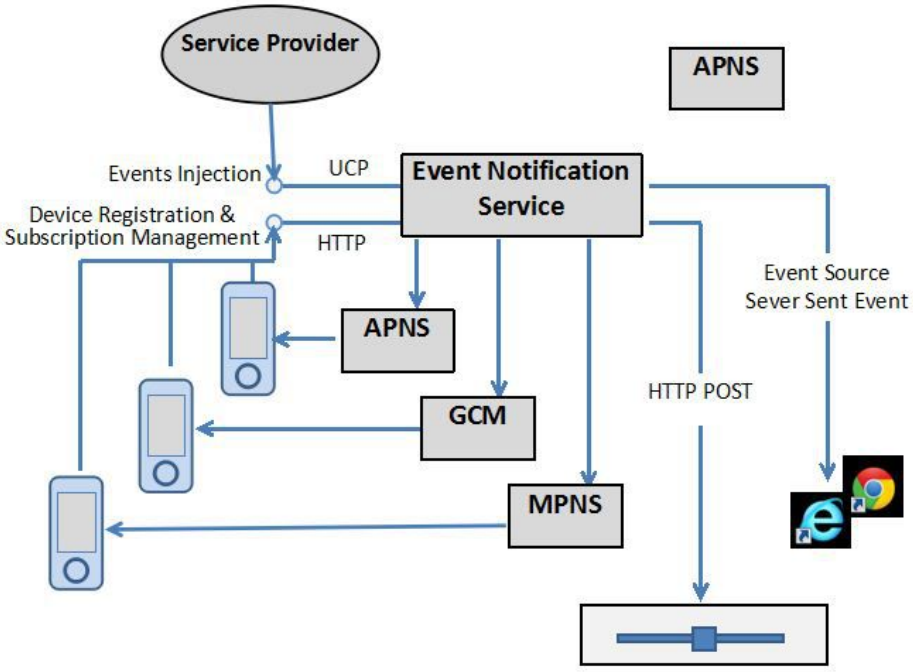

Figure 3: ENS working mechanisms.

According to the pocket guide to good push of Urban Airship [Urb13], a good push with ENS mechanism is delivering pushing messages with properiate privileges. In details, good pushing messaging should have a high privilege to deliver useful notifications, but should not interrupt application users' life anytime and anywhere.

The followings are the seven practical rules of good push from application users' point of view. The first rule is a good push should fit into application users' changing schedules. For example, an application should have an easily accessible control panel. The second rule is a good push should engage application users with relevant pushing messages. The third rule is a good push allows application users to personalize their experiences about customized contents. The fourth rule is that service providers should have hands-on visual tools like voice or tone or appearance for consistent communications to push message content and preview the messages across devices. So that a good push can stay consistent with their brands. The fifth rule is that a good push delivers entertaining and engaging experiences. The sixth rule is that a good push should continuously serve application users better and better, which means each push message works effectively. The last rule is a good push adapts to fit application users' current situations to adapt to current situation of application users, including their changing locations.

This section introduces four push notification services of different mobile platforms, including APNS, Android Cloud to Device Messaging (C2DM), GCM and the Microsoft Push Notification Service (MPNS). The information of the four prevalent platforms is from their official websites.

4.3.1) Apple Push Notification Service

Apple Push Notification Service describes the process that is design to deliver notifications to devices and computers with a push [App13]. APNS is a push design which differs from a pull design which receives the notification immediately, as the operating system passively listens for updates rather than actively polling for them. A push design makes it possible when a pull design experiences some scalability problems and timely dissemination of information.

APNS is the central part of the iOS push notification feature. It receives notifications from third party providers and routes them to target devices over a persistent long-lived connection. APNS is the gateway of pushing notifications, transmitting notifications or information to the user applications which has registered. Each application establishes an encrypted and persistent IP connection for the notification service and can receive notifications from the connection. Thus, service providers can be linked to APNS in a persistent and safe channel, and monitor the delivered information for their users' applications as well. When service providers have updated information to publish, the service providers create and deliver a notification via the channel to APNS to push notifications to the targeted application users.

APNS uses a persistent IP connection to implement push notifications. The flow of a notification is one way as depicted in Figure 4. If the user has already registered, the application user will receive a device token as an identify. When useful information generated by service providers, service providers send information to APNS server via web requests based on the registered device token. Then APNS server would judge whether certain mobile application user has registered or not. If registered, APNS server would send notifications through service providers' server to application users' devices. When service providers authenticate themselves to APNS, they sends their notifications with SSL certificates and private keys.

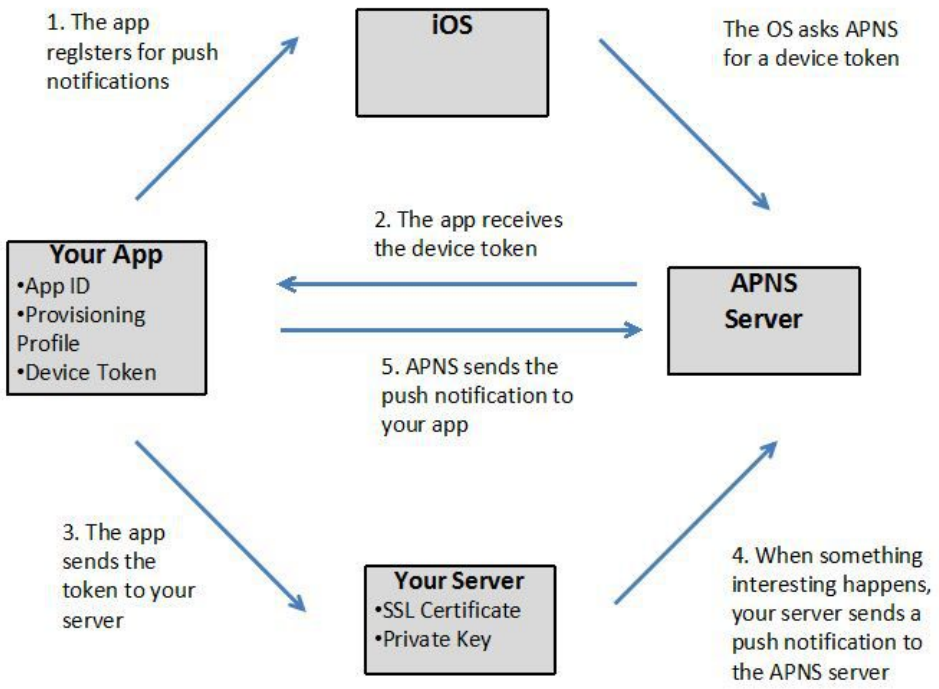

Figure 4: APNS push notification data flow.

APNS also a store-and-forward feature that saves recent notifications sent to an application. If the application is not working, APNS delivers the notification when the application is online next time. With time pass, the notification of simple format would turn into outdated messages. In other words, if a user was offline in the past three days and when he is online again, he would receive messages sent three days ago, which means little value to the user by then. The enhanced format solves this problem effectively by set an expiry time about how a device is alerted.

Another advantage of the enhanced format notification is when there are mistakes. It would receive wrong notifications and provide the reason for refusing the notification, such as wrong message, etc. There is no such function in simple format notifications. Besides, the enhanced format has common with the simple format in many aspects. Hence, enhanced format notification is recommended, when mistakes occur, we can find out the mistake in the system.

4.3.2) Android Cloud to Device Messaging

C2DM is a push notification service that enables application developers to deliver notification from servers to their Android applications and launched together with Android 2.2 by Google [Goo13]. C2DM is deprecated now, but is presented here as it is still supported in most push

notification products. The service gives a lightweight and simple mechanism for servers to make mobile applications to communicate with the server and update notifications directly. The C2DM service can manage all the queuing messages and transmission to the targeted applications.

Android applications don't have to be working to receive messages, because the applications can be triggered by Internet broadcasts when the messages arrive with correct permissions and broadcast receivers. C2DM doesn't support built-in user interface, instead, it directly delivers the received raw message to the applications. Each notification message size is limited to 1024 bytes, but Google supports messages delivered in a group without the aggregate messages limitation.

The workflow of C2DM is depicted in Figure 5. When the mobile application is first launched in a mobile device, it will perform as the following steps. First, the mobile application registers itself to one of the C2DM servers using the C2DM username, which was provided by the application developer, as well as the device ID, which uniquely identifies the Android device that hosts the application. Then the C2DM server provides a unique registration ID to the mobile application. The registration ID is a byte string that enables the C2DM server to identify the application running on a specific Android device. Thus the mobile application sends this registration ID, together with its C2DM username, to the application server, which will then record this registration ID in its database. When the application server needs to send data to a mobile device, it sends the C2DM username and password to the C2DM server, and gets an authorization token if the username and password are valid. The authorization token will be used to notify a set of mobile devices in the database. The application server then sends a C2DM request, which contains the notification message, the registration ID of a mobile application, and the authorization token to the C2DM server. The message is sent on a per device basis. Thus, if there are k devices that need the notification, then the application server will send k messages to the C2DM server. Upon receiving the message, the C2DM server looks for the specific Android device based on the registration ID. If the C2DM connection of that device is alive, then the C2DM server will send the notification message to the mobile application on that device. If the mobile device is disconnected, then the C2DM server will store the notification message, and send the message to the application on the mobile device when the device reestablishes its connection with the C2DM server. There is a persistent C2DM connection between the C2DM server and the Android device that subscribes to the C2DM service.

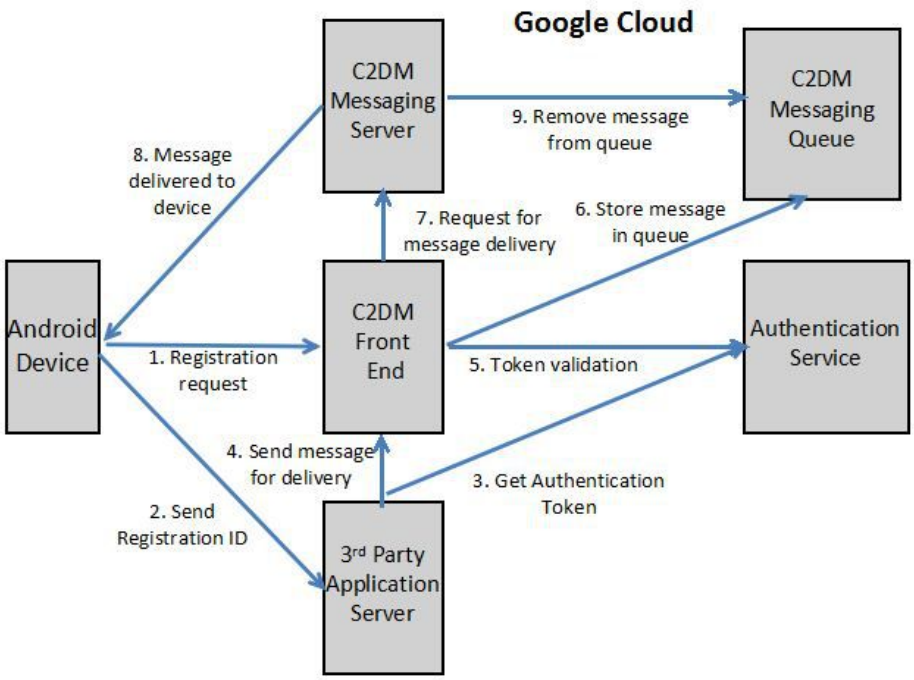

Figure 5: C2DM push notification flow.

4.3.3) Google Cloud Messaging

GCM [Goo14] is a service which enables application developers to deliver messages from servers to their Android applications or from servers to their Chrome applications and extensions. GCM replaces the depreciated Android C2DM. The free service allows application developers to deliver lightweight messages, informing the Android application users of updated messages to be fetched. Larger messages can be sent with up to 4 KB of payload data.

GCM sends notification from server side to Android applications by a sample and lightweight proxy mechanism as depicted in Figure 6. Both a server and Android applications that are registered for the GCM service required, because registration ID is required. GCM servers handle message queuing and delivering as the proxy server, guaranteeing cloud scalable communication. During the process of updating data, application server first sends notifications to the GCM servers, and then the GCM servers push the notifications to all Android applications. Then the Android applications receive the notifications, and can connect to the server and synchronize data for the future.

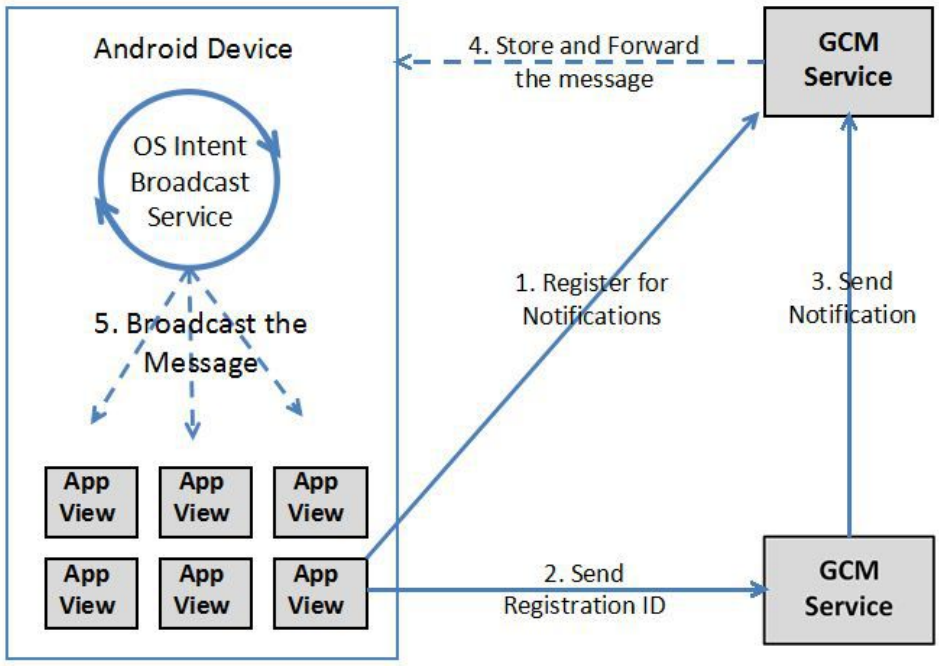

Figure 6: GCM push notification flow.

4.3.4) Microsoft Push Notification Service

MPNS [Pus13] is an asynchronous service which enables third-party application developers to deliver messages to Windows Phone applications from cloud.

MPNS has three forms of push notifications for Windows Phone, toast notifications, tile notifications and raw notifications as follows. Toast notifications which consist of two strings of texts are text-based short messages for transient data. Toast notifications are displayed on the home screen with the highest priority, clicking a toast notification will be linked to the associated applications. Then, tile notifications are to polish the appearance of application. Tile notifications can specify both local and remote resources. Tile notifications in the Quick Launch area react immediately no matter applications are running or not, and the application to be notified cannot be intercepted by tile notifications. The last is raw notifications, which are similar to tiles and toast notifications, but raw notifications do not have any special display style or predefined payload format. The contents of raw notifications are decided by the sender and handled by the applications, because raw notifications are not handled by basic operating systems. Hence, a raw notification can be received when the application is running properly.

MPNS combines unauthenticated and authenticated modes together. In unauthenticated mode, both quantity and frequency of notifications are limited by 500 per channel in a single day. For push notifications in authenticated mode, application developers should register a certificate in Push Notification Service which is authorized as a trusted Microsoft certificate. Then the certificate can be used to set up a Secure Sockets Layer (SSL) connection for web services and the Push Notification Service. MPNS doesn't limit any restriction for authenticated push notifications.

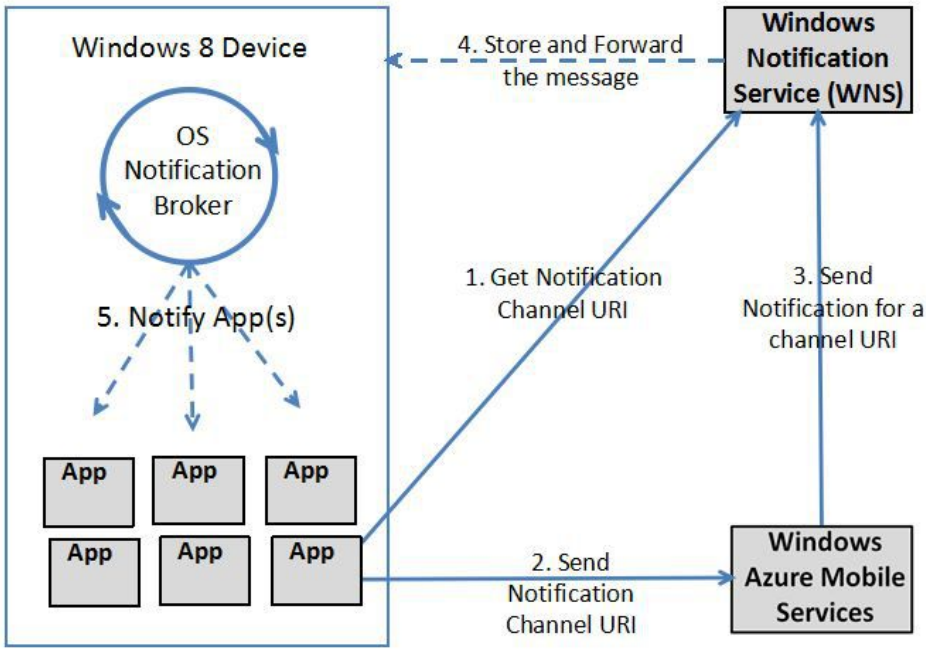

Figure 7: MPNS push notification flow.

Figure 7 illustrates how a push notification is delivered in MPNS [Pus13]. The user's application initials a request composed of a push notification URI from the push service provider and the service provider interacts with MPNS to get a notification URI. Then the application delivers the notification URI to the cloud service. The cloud service can deliver a push notification to MPNS via the notification URI when the cloud service has data to deliver to the user's application. Finally, MPNS can route the push notification to the user's application.

The types of the delivered data depend on the format of the payload attached and the push notifications. Finally, MPNS responds to the cloud service after delivery with codes, which means that notifications are received and prepare to deliver subsequently. MPNS doesn’t have

end-to-end confirmation mechanisms for reliable delivery, but MPNS can respond error codes to the cloud service instead.

4.4 An ENS prototype based on pub/sub paradigm

Applications have to develop custom notification mechanisms, because there doesn't exist a general notification mechanism for all. Based on the paradigms in the previous Subsection 4.2, a widely used approach by ENS is that each application user periodically polls the application developer's server for updates. Considering simple and easy implementation conceptually, polling creates an unfortunate tension between timeliness and resource consumption. Frequent polling allows application users to learn of updates quickly, but imposes significant loads in server side. Furthermore, most requests simply indicate that no change has occurred in practice.

An alternative is pushing notifications to application users. However, ensuring reliability in a push system is difficult, because of diversified storages, network, even server failures for the Internet scale. Further, application users may be disconnected while updating, or even keep offline for several days. Buffering messages indefinitely is infeasible. The application developers' server storage requirements must be bounded, and application users should not be overwhelmed by a flood of messages upon wakeup.

It is obvious that push paradigm is more suitable than polling paradigm for a large scale distribution. It will be better if application users show their interests first. Then, application developers or service providers provide application users' requirements as feedback. Thus, the ecosystem can work efficiently. As a result of these challenges, pub/sub paradigm to realize push paradigm is the best effort.

The following subsection will present Thialfi [Ady11], a highly scalable notification service prototype developed at Google, which means Thialfi is a client notification service for Internet-scale applications. Thialfi is presented here in details. Most of the push notification products in Section 5 are implemented based on the ENS prototype Thialfi. The notification delivery of Thialfi is depicted in Figure 8.

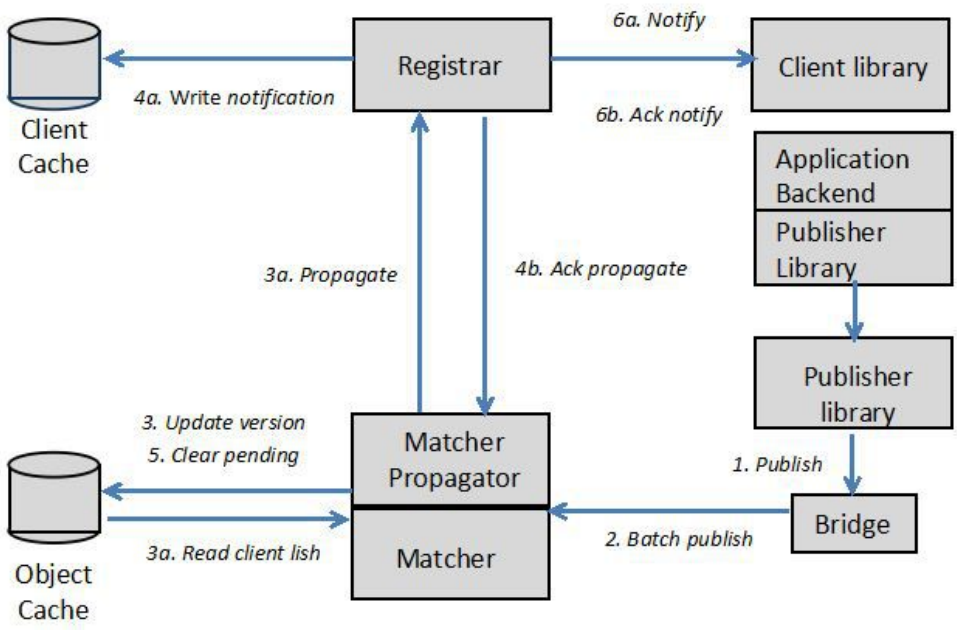

Figure 8: Notification delivery of Thialfi.

Thialfi consists of bridge servers, matchers and registrars, which is shown [Ady11] in Figure 8. Bridge servers foster or feed triggered events which are going to trigger notifications. The triggered events come from publisher library and specific update messages from Google's infrastructure pub/sub service. Bridge servers translate the triggered events into a standard notification format, and also process them into batches for transferring to matchers in a stateless and randomly load balanced way. Then, the matcher propagators of matchers compare the events with the registered applications which are subscribers, and forward the triggered events notifications to registrars as a relay agency. Matchers can maintain an overview of the state index with an event ID while disseminate triggered events. Finally, registrars track subscribers with their process registrations, and deliver reliable notifications with a user application ID.

The notification delivery is illustrated in sequential steps in Figure 8 [Ady11]. When the application server updates an authoritative copy of user data, Thialfi will be informed of the new version number. Publishers publish triggered events through the publisher's library to guarantee all the triggered events will be received by all backend servers running Thialfi. The pub/sub paradigm acknowledges the publisher's library with a reliable handoff to guarantee successful partitions to backend servers. If unsuccessful partitions, the pub/sub notifications will be buffered in a persistent log. Then, the bridge fosters the triggered events in all backend servers

and assembles them in small batches, while forwarding the notifications to the corresponding matcher. Afterwards, the matcher updates records for the new coming triggered events. The matcher propagator works to match the registered events to corresponding registrar servers by RPC operations. Subsequently, the registrar receives a notification and keeps it in a temporary record for a user application to its RPC. After all registrars have responded, the records will be removed. Moreover, the retransmission mechanism is used by registrars to deal with unacknowledged notifications of online triggered events. This mechanism realizes the reliable notification delivery of Thialfi.

Thialfi provides fundamental notification delivery for the common case and clear semantics despite failures even for entire data centers. Thialfi supports applications written in a variety of languages like C++, Java, and JavaScript, and running on diverse sets of platforms such as web browsers, mobile phones and desktops. To achieve reliability, Thialfi relies on clients to drive recovery operations to avoid the requirements for hard states in server side. Besides, the API is structured so that error handling is incorporated into the normal operations of the applications [Ady11].

5 Push notification products

In practice, push notification products are in a mass use. In order to select push notification products, subsequent comparison subsections keep the importance of general application users in mind. This means listed push notification products are not aimed for specific application users, but for general application users. Push notification products have two categories, namely push notification services and push notification servers. A push notification service is a process provided by an IT service provider to one or more application users, while a push notification server is a piece of hardware or a virtual machine which gives a specified push notification service.

This paper focuses on simple and unified push notification products across prevalent platforms, including iOS, Android, Windows Phone and Web. Besides, this section gives an outline of each push notification product based on online information which supports both APNS and GCM in the subsequent Subsection 5.1 and Subsection 5.3 with detailed information in the appendixes.

5.1 Push notification services

A push notification service is a process that provides services across diversified networks, and it runs on other service provider's server. Usually, push notification services are used remotely by small and medium scale enterprises that are limited by budgets. The following subsection introduces current push notification services with corresponding appendixes in details.

5.1.1) Mozilla SimplePush

Mozilla SimplePush[2] is a low cost service which gives a way to transmit messages from application developers to corresponding web applications. Mozilla SimplePush Server plays an important role, which is a server for SimplePush Notification alert system with Mozilla SimplePush protocol specifications in server side. Mozilla SimplePush protocol illustrates a JavaScript client API to enable backend applications to send notifications. The detailed information is in Appendix 1.

Application developers can make full use of Mozilla SimplePush API to implement notification alert systems, avoiding applications continuously polling application developers' servers for updated information. Mozilla SimplePush is used as the basis of other push notification services through an interface of Mozilla SimplePush API. In practice, Mozilla SimplePush is usually integrated in a unified push notification service or server like AeroGear UnifiedPush Server in Subsection 5.3.1, as source codes can be downloaded from websites easily. Hence, Mozilla SimplePush reduces the required number of sockets and ensures application users' privacy and ability of control, as a URL endpoint is used to wake the applications on the devices. Mozilla SimplePush reduces high costs of devices and alive-open sockets in server side, and makes it less complex to find a server for a device or to find a device for a server.

5.1.2) Pushover

Pushover[3] is a simple push notification service. Pushover can send real-time notifications to Android and iOS mobile devices easily. The basic idea of Pushover is pushing messages straight to application users' devices. Pushover provides a RESTful API based on Mozilla SimplePush API to deliver queuing messages to devices in server side, in the meanwhile, iOS and Android

[2] https://wiki.mozilla.org/WebAPI/SimplePush
[3] https://pushover.net/

application users receive those push notifications on device. Pushover is released under OpenBSD license. More detailed information is in Appendix 2.

Besides, Pushover provides users with unlimited messages for free, which means Pushover does not charge monthly fees for application developers like monthly subscription fees. Hence, most applications with Pushover notification service can send notification messages for free within the month limits. Month limits include messages like *titles* currently limited to 512 characters, and applications which are currently limited to sending 7,500 messages per month.

5.1.3) Pushwoosh

Pushwoosh[4] is a free push notification service supporting multiple platforms. Pushwoosh has a remote API similar to Mozilla SimplePush API, which supports multiple programming languages and platforms. Notification delivery mechanism of Pushwoosh is similar to Thialfi depicted in Figure 8. The detailed information is in Appendix 3.

Pushwoosh introduces geo zones, which is an important feature to trigger push notifications. Conceptually, geo zones are pins on the map. Geo zones allow automatically sending triggered push notifications exact at the moment that a user enters a specified area on the map. The range of geo zones varies from 10 to 100 meters within an operational cool down period. Geo zones bring convenience for service providers like retail owners who would like to get attention from pass-by-application users. In addition to, Pushwoosh has advanced statistics feature for service providers to keep track of their push notifications, provides developers to send a rich HTML page like rich images, formatted text and links, and enables application groups feature to deliver one push notification to several applications within a single operation as well.

5.1.4) Amazon Simple Notification Service

Amazon Simple Notification Service (SNS)[5] is a web service which gives an easy way to deliver notifications from cloud service. Amazon SNS is designed to make web-scale computing easier and cost effective to disseminate notifications for application developers' flexibly.

Amazon SNS follows the pub/sub paradigm similarly to Thialfi depicted in Figure 8.

---

4 http://www.pushwoosh.com/

5 http://aws.amazon.com/

Notifications of Amazon SNS are delivered from application developers or service providers to application users via simple APIs in the same with Mozilla SimplePush API to Apple, Google, and Kindle Fire devices. The detailed information is in Appendix 4.

5.1.5) OpenPush

OpenPush[6] is a service which enables real-time push notifications directly to its application users. OpenPush serves as an aggregator of real-time push notifications based on Thialfi depicted in Figure 8, where OpenPush uses a central dispatcher for its matcher in a contextualized view. So that application developers do not need to care about which desktop or media device or mobile platform receives the push notifications. OpenPush is open and free-of-charge with easy to share with IT community, and it realizes global development and standard push ecosystem. The detailed information is in Appendix 5.

5.1.6) Parse.com

Parse.com[7] is a push notification service which supports multiple platforms and handles device registrations easily. Parse Push makes creation, schedule and segment of push notifications as a whole based on Thialfi, where *push composer* of *matcher* and *push console* play important roles. *Push composer* helps to aggregate millions of push notifications and allows application users to preview real-time push notifications, while *push console* enables service provider freely to reach users by delivering notifications with advanced targeting controls without writing codes anytime. Parse.com also introduces multiple channel concepts to deliver notification for *Parse Push*. The detailed information is in Appendix 6.

5.1.7) Urban Airship

Urban Airship[8] is a mobile push notification service, which primarily enables application developers to add features easily like creation and delivery of enhanced push notifications with a digital wallet of feedback. Urban Airship makes application developers keep track with application user segmentation based on individual preferences, like browsing history and location with back-end data, so that personalized experiences are summarized individually.

---

6 http://openpush.im/
7 https://parse.com/
8 http://urbanairship.com/

Urban Airship offers these services[9] on android, iOS, Windows Phone 8, and BlackBerry based on Thialfi prototype as illustrated in Subsection 4.4. Urban Airship push notifications have been tested with proven results across all industries. The detailed information is in Appendix 7.

5.1.8) Xtify

Xtify[10] is called a notification platform designed for application developers, because Xtify is easily combined with mobile strategies for application developers to deliver relevantly actionable push notifications with robust notification creations and analytics. The notification delivery mechanism of Xtify is similar with Thialfi as depicted in Figure 8, but Xtify only supports Android and iOS notifications data flows. Besides, Xtify gives complete solutions for service providers to satisfy diversified demands and offers custom solutions for application developers with innovate quickly plans for mobile investment. The detailed information is in Appendix 8.

5.1.9) Push IO

Push IO[11] is a cloud based push notification service for mobile platforms. Push IO was built to satisfy demands of global leading service providers. Push IO is a unique provider who uses AutoPush to monitor application users' feeds and push personalized notifications automatically. Push IO allows service providers to push any contents as much as they want with its *web dashboard* and *composer* of *matcher* which are similar with the ones of Parse.com introduced in Subsection 5.1.6. Push IO also provides special *Live Tiles* for Windows Phone by uniquely combining images and data to maximize users' presence. The detailed information is in Appendix 9.

5.1.10) Awarly

Awarly[12] is a push notification service for mobile applications like iOS, Android or Blackberry. Awarly gives a way for application developers to send location aware notifications to their applications via RESTful APIs based on Thialfi in Subsection 4.4. Awarly uses features of advanced geo notifications&alerts, push composer and powerful analytics realize sending

---

9 http://www.oregonlive.com/silicon-forest/index.ssf/2011/11/urban_airship_raises_151_milli.html
10 http://www.xtify.com/
11 http://push.io/
12 https://awarly.com/

advanced and scalable push notifications, where features are similar to other push notification services like Parse.com in Subsection 5.1.6. The detailed information is in Appendix 10.

5.1.11) OpenMarket

OpenMarket[13] is an alert push notifications service to push short messages to applications' inboxes on Apple, Android, and BlackBerry based on Thialfi. OpenMarket supports three ways for mobile messaging, including SMS text messages, Multimedia Messaging Service (MMS) and push notifications. SMS is the fastest way to broadest consumer reach and interact directly with mobile subscribers. MMS allows longer content text, images, video, or audio to be inserted into SMS text messages as the next generation of mobile messaging. Push notifications are alerts for smart phone applications providers to engage their consumers. The detailed information is in Appendix 11.

5.1.12) PushBots

PushBots[14] is a push notifications service which can send instant notifications to Apple and Android mobile devices based on Thialfi. PushBots provides a three-minute drag-and-drop solution to fast deployment of push messages, which can be accessible from its RESTful API. PushBots helps mobile developers to minimize required time and resources to keep their application users engaged with their applications by pushing highly personalized relevant contents according to their interests. The most practical use of PushBots is to push notifications service linked with an application user's Twitter account on his application to receive Tweets pushed notifications. The detailed information is in Appendix 12.

5.1.13) Mass Notification Service

Mass Notification Service (MNS)[15] is a service to handle mass notifications like information and instructions to alert and inform the most critical asset for persons within an area site. MNS combined with emergency solutions is especially useful in an emergency, because MNS keeps everyone informed before, during and after any event. MNS has event processors of *bridge* and *notification task dispatcher* of *matcher* with the same significant roles like Thialfi. The detailed

---

13 http://www.openmarket.com/
14 https://pushbots.com/
15 http://www.everbridge.com/

information is in Appendix 13.

Cooper Notification's MNS[16] is taken as example to show MNS. Cooper Notification's MNS uses concise, timely and event-specific voice and visual messages to provide real-time information for persons in a geographic area to communicate what to do in response to threats. From a single web page, these reliable and effective MNS solutions allow its application users to send alerts and potentially life-saving instructions to unlimited communication devices, including voice sirens, indoor and outdoor speakers, digital display signage, text messaging, voice calls, desktop alerts and email notifications. It's the same with Everbridge Mass Notification[17] which can push notifications to individuals or groups based on locations, lists or visual intelligence.

5.1.14) mobDB

mobDB[18] is a free push notification service for mobile development, which provides backend service, push notifications and analytics based on Thialfi mechanism. mobDB provides both XML-based APIs and JSON-based APIs to satisfy different requirements. mobDB also gives a backend solution whose platform is independent, in the meanwhile, application developers can create feature-rich mobile applications.

mobDB is security. Because all communications between applications and mobDB are encrypted, and data is fully secured in the data store with the 256-bit encrypted SSL for application login or signing-up or sensitive data. However, servers of mobDB present an expired certificate currently, and there doesn't exist any available information to indicate whether the certificate has been compromised since its expiration. The detailed information is in Appendix 14.

5.1.15) NACapp

NACapp[19] is a mobile notification service used by iOS, Android, Blackberry and Windows Phone devices. NACapp not only pushes notifications, but sends chat messages and alerts to mobile applications. NACapp creates push notifications in three ways, HTTP to push, SOAP to push and e-mail to push. NACapp also gives application uses to set priorities for different

16 http://www.cooperindustries.com/content/public/en/safety/notification.html
17 http://www.everbridge.com/solutions/mass-notification/
18 https://www.mobdb.net/
19 http://www.nacapp.com/

notifications, so application users can give an urgent notification with a higher priority. NACapp uses either SOAP or REST, but only supports XML as its data formats. Because NACapp is in the transition from SOAP to REST, and currently both are available. The unstable reason brings disadvantages for application developers, so NACapp is not a good choice. The detailed information is in Appendix 15.

5.1.16) OpsGenie

OpsGenie[20] is a reliable IT notification service which delivers the most relevant alerts to application users. OpsGenie has several channels to deliver notifications in the forms of e-mails, SMSs, phone calls and mobile pushes of iOS, Android and BlackBerry devices. OpsGenie uses push notifications as mobile pushes. OpsGenie is reliable, because OpsGenie has a distributed and redundant architecture across diversified data centers with an end-to-end monitor system to guarantee availability, and a monitor system uses a detailed track mechanism. The detailed information is in Appendix 16.

5.1.17) SnapComms

SnapComms[21] is a push notification solution to efficient communication for staff within a company. SnapComms has four types of notifications, pop-up desktop alerts, pop-up staff quizzes, corporate screensaver messages and mobile employee communications for desktop and mobile devices. Pop-up desktop alerts are to broadcast emergent news and significant information. Pop-up desktop alerts keep staying on the desktop for targeted staff. Pop-up staff quizzes are to broaden knowledge and enhance skills for special staff. Corporate screensaver messages are to drive employees to share useful information with others in a rich style. Corporate screensaver messages help to release working burdens and enhance internal communications. Mobile employee communications are to send notifications to mobile staff. However, public documentations are not available at the moment. The detailed information is in Appendix 17.

---

20 http://www.opsgenie.com/

21 http://www.snapcomms.com/

5.1.18) Notificare

Notificare[22] is a notification platform, was originally born with Web 2.0. The Notificare team insists Web and future notification service would depend on how data is fetched. Currently, Notificare team are focusing on developing Notificare platform, and Notificare gives SDKs for smart notifications to replace Notificare applications which were used before February 15th of 2014. The detailed information is in Appendix 18.

Working mechanism of Notificare relies on a simple push method based on Thialfi in Figure 8. Notificare serves as a flexible service through API, which is system-agnostic. This means one system can function in diverse environments. Besides, Notificare supports platform development neither in native ways nor with frameworks such as Xamarin, Appcelerator Titanium and Apache Cordova to minimize development cycle and give better users' experience.

5.2 Comparisons of push notification services

In order to choose a preferred push notification service, I use six comparison points as a systematic approach to compare current push notification services. The objective of the systematic approach is to propose proposals for API design. Six comparison points are basic information, functionalities, licenses with usage fees covering usage limits, security, supporting details and supported platforms/OS in terms of performance. Basic information covers service-oriented architecture paradigms, notification transport protocols and data formats. Functionalities include batch processing, mobility, scalability and fault tolerance. Usage fees and usage limits discuss details of both free accounts and premium accounts, including the number of registered applications, the number of delivered notifications, the number of devices used by application users and prices of different push notification services per month. Security illustrates authentication model of services and SSL-supported. Supporting details show whether notifications can be delivered in the forms of e-mail, SMS and push cloud notification. Supported platforms/OS part covers GCM, C2DM, APNS, Microsoft, Firefox OS, Blackberry and ADM. This subsection uses these comparison points to compare push notification services, and these comparison points show availability in the market and popularity of push notification

[22] http://notifica.re/

services with optimality.

5.2.1) Comparisons of basic information

Table 1 summarizes basic information from three aspects, including service-oriented architecture paradigms for services, notification transport protocols for communications and data formats.

Services	Basic Information		
	Service-oriented Architecture Paradigms	**Notification Transport Protocols**	**Data Formats**
Mozilla SimplePush	REST	HTTP, HTTPS, XMPP	JSON
Pushover	REST	HTTP, HTTPS, SMTP	XML, JSON
Pushwoosh	REST	HTTP, HTTPS, SMTP	JSON
Amazon SNS	SOAP	HTTP, HTTPS, SMTP	XML
OpenPush	REST	HTTP, HTTPS, SMTP, XMPP	JSON
Parse.com	REST	HTTP, HTTPS, SMTP	JSON
Urban Airship	REST	HTTP, HTTPS, SMTP	JSON
Xtify	REST	HTTP, HTTPS, SMTP, XMPP	XML
Push IO	REST	HTTP, HTTPS, SMTP	JSON
Awarly	REST	HTTP, HTTPS, SMTP	JSON
OpenMarket	REST	HTTP, HTTPS, SMTP	XML
PushBots	REST	HTTP, HTTPS, SMTP	JSON
Everbridge Mass Notification System	REST	HTTP, HTTPS, SMTP	JSON
mobDB	REST	HTTP, HTTPS, SMTP	XML, JSON
NACapp	SOAP, REST	HTTP, HTTPS, SMTP	XML
OpsGenie	REST	HTTP, HTTPS, SMTP	JSON
SnapComms	REST	HTTP, HTTPS, SMTP	JSON
Notificare	REST	HTTP, HTTPS, SMTP	JSON

Table 1: Basic information comparisons of push notification services

From comparisons of basic information in Table 1, it can be seen that most push notification services use either SOAP or REST as their service-oriented architecture paradigms, HTTP(S) and SMPT protocol as notification transport protocols, XML and JSON as data formats. NACapp is the only push notification service which supports both SOAP and REST service-oriented architecture paradigms. This subsection compares SOAP-based implementations with REST-based implementations to show differences between SOAP and REST.

Fundamentally, SOAP service-oriented architecture paradigm implements transmissions of XML-encoded notifications over HTTP, similarly to RPC model. SOAP service sets are written in WSDL files which are essentially XML files complied with W3C-specified grammars. SOAP service-oriented architecture paradigm uses WSDL files to define a series of XML schema types

which mirrors data models in server side. Thus, XML schema types are able to generate service requests/responses and even language-specific bindings for diversified platforms by mapping corresponding structural parameters subsequently. However, XML has three disadvantages [Shi09]. XML is difficult to parse and read; XML has data models which are incompatible with most programming languages; XML has a poor output representation in serialization. As the order of output is usually irrelevant in databases, it is hard to specify sets of sub elements in disorder due to the document type definition of XML.

Compared with SOAP service-oriented architecture paradigm, REST service-oriented architecture paradigm emphasizes available resources via HTTP connections. REST service-oriented architecture paradigm identifies each available resource with a unique URL which is access by HTTP Get/Post/Put/Delete methods. REST service-oriented architecture paradigm is able to use any HTTP client library to communicate with backend REST servers without specified SOAP clients. HTTP is easy to use in practice, because new contemporary languages always come with built-in HTTP libraries [Mul09]. In addition to, service interpretation and service discovery are not required for Internet communications in most cases.

All the push notification services use data transmission of web services via HTTP(S) in either SOAP or REST service-oriented architecture paradigms. On the one hand, network-related performance is taken as the metrics to compare SOAP and REST service-oriented architecture paradigms. For example, the end-to-end response time of individual web service transactions is such a metric. Because any change of delivery latency directly will impact QoS of the whole network. In the meanwhile, the change also influences user engagement. On the other hand, average packet size of each push notification service request is another metric, which can be combined with average latency.

Currently, most push notification services adopt REST service-oriented architecture paradigm as depicted in Table 1. REST service-oriented architecture paradigm of data transmissions has been observed [Mul09]. REST service-oriented architecture paradigm is more efficient for both high network bandwidth utilization and low round-trip latency reduction. In contrast to SOAP, REST service-oriented architecture paradigm has a lower latency, a smaller average packet size and smaller message payloads [Mul09]. From discussions of performance of high-throughput and fast transmission traffic for push notification services purchases, Parse.com, PushBots and

Pushwoosh are preferred choices.

Although SOAP messages are larger with a higher average round-trip latency [Mul09], it is still easier to integrate an SOAP-based push notification service than REST equivalents for enterprises whose backend servers are implemented based on SOAP. SOAP service-oriented architecture paradigm is used by Amazon SNS and NACapp. However, NACapp uses either SOAP or REST, but only supports XML as its data formats. Both SOAP-based and REST-based APIs are available in NACapp official website. If a service provider wants to use NACapp, he needs to update and maintain both SOAP-based and REST-based APIs. This is a challenge for service providers, so NACapp is not a good choice. Hence, Amazon SNS is a primary choice for these enterprises that have SOAP-based backend servers.

Apart from HTTP and HTTPS, notification transport protocols SMTP and XMPP are used by some push notification services. If a push notification service supports e-mail, the push notification service will support SMTP. However, Mozilla SimplePush is the only one that doesn’t support e-mail, so it doesn’t support SMTP. Communications protocol XMPP works for message-oriented middleware based on XML. Distinguished from other instant messaging protocols, XMPP is defined under an open standard. XMPP also uses an open systematic approach to implement its development. Hence, any application developer is able to implement an XMPP service and interoperate with other organizations' implementations.

To exploit the differences more deeply, SOAP-based notifications have payloads containing an additional SOAP envelope, a set of XML tags, and several alternative SOAP-related headers to outbound HTTP data packets for each REST-based notification. However, REST-based notifications only use standard HTML headers to deliver the data packets across Internet without any overhead for transitions. For available information on the website, only Pushover explicitly limits its usage that notifications are currently restricted within 512 characters including titles.

Results of basic information in Table 1 indicate that the preferred push notification service should be chosen from OpenPush, Parse.com, PushBots and Pushwoosh, or Amazon SNS and NACapp. OpenPush, Parse.com, PushBots and Pushwoosh use REST service-oriented architecture paradigm, while Amazon SNS and NACapp use SOAP service-oriented architecture paradigm.

5.2.2) Comparisons of functionalities

Table 2 summarizes four functionalities, including batch processing, scalability, mobility and fault tolerance. Batch processing refers to the ability to handle a series of processes within one request without personal intervention. Batch processing is aimed to share resources, which is distinguished from interactive systems. Push technology always supports unicast, however, current push notification services also need to support batch, multicast and broadcast to satisfy increasing demands. Hence, batch processing is useful for selective notifications delivered to a group based on a category or a broadcast, as broadcasting the same update to a group of application users is always required. Batch processing can bring benefits to share resources among multiple processes and enable push notification services to make full use of computer resources.

Mobility means the ability of a service to deal with individuals or groups on the move. So that push notification services which support mobility are able to support all the services on the move seamlessly. This means push notification services keep track of application users' locations for the application users who are transferring from one access point to another, even when mobile phones are not working. Mobility brings benefits to both service providers and application developers, such that they can keep deep insight into their application users and maintain application user engagement closely with their brands. Because mobility helps service providers and application developers to keep updated with their application users, e.g. they will be informed when an application user uninstalls the application or have interested in an advertisement around him. Service providers provide such a service to gain the application user engagement and keep closely with their brands.

Scalability is the ability to handle a growing quantity of work or the ability to be enlarged to handle the expanding growth. Large scale services should be able to satisfy increasing demands, but also brings potential problems like faults.

Fault tolerance is a property that enables a service to continue operating properly in the event of some of its components have unexpected faults. Fault tolerance is significant particularly for notification on a large scale reliable delivery.

Services	Functionalities			
	Batch processing	Mobility	Scalability	Fault tolerance
Mozilla SimplePush	×	√	√	√
Pushover	√	√	√	√
Pushwoosh	√	√	√	√
Amazon SNS	√	√	√	√
OpenPush	√	√	√	√
Parse.com	√	√	√	√
Urban Airship	√	√	√	√
Xtify	√	√	√	√
Push IO	√	√	√	√
Awarly	√	√	√	√
OpenMarket	√	√	√	√
PushBots	√	√	√	√
Everbridge Mass Notification System	√	√	√	√
mobDB	√	√	√	√
NACapp	√	√	√	√
OpsGenie	√	√	√	√
SnapComms	√	√	√	√
Notificare	√	√	√	√

*) √ means the push notification service supports the specified functionality, while × means the push notification service doesn't support the specified functionality.

Table 2: Functionalities comparisons of push notification services

From comparisons of functionalities in Table 2, it can be seen that most of the current push notification services realize four functionalities including batch processing, mobility, scalability and fault tolerance.

From comparisons of functionalities in Table 2, it can be seen that most push notification services support batch operations. Push notification services use batch processing to deliver a unique payload or push scheduled notifications to a set of devices. So that push notification services can deliver a single HTTP PUT notification instead of delivering a number of individual HTTP PUT notifications to the same group. Push notification services are able to create, delete and update a batch of requests with a batch endpoint, which also reduces the time spent on network round trips.

The functionality of batch processing is packaged in the feature of group management. This means the functionality of batch processing is an important part of group management. Hence, push notification services which support batch processing have a feature of group management.

So that audience can be segmented by defining for targeted or broadcast messaging groups. For example, Pushover has a feature of group delivery to manage a list of application users, so that Pushover can deal with notifications in a single API request. Thus, most of push notification services realize batch processing in a similar way.

In addition to, some push notification services even limit the quantities of requests within one batch. For instance, PushBots allows multiple devices registrations up to 500 devices with one batch, subsequently the array of devices tokens will be added to database. However, Mozilla SimplePush doesn't support batch processing. Although it is recommended that UserAgent should try to batch all pending acknowledgements into fewer notifications. In practice, Mozilla SimplePush keeps notifications separate and individual in use scenarios, which are claimed in the specifications[23].

Push notification services use a feature of smart segments for service providers to handle mobility related to mobile push notifications as depicted in Table 2. Because the feature of smart segments gives a way for push notification services to deliver relevant and tailored push notifications based on application users' preferences or their real time locations. With smart segments, push notification services are able to collect information based on application users' real time locations. The feature of smart segments is implemented and integrated by APIs where APIs help to create audience segments. Hence, push notifications can update service provides' scheduled reminders and offer important product news for application users on the move. Besides, application developers are able to push notifications to notify their application users with mobility even when the applications are not working. For example, OpenPush enables service providers to segment their audience, and they can also provide relevant real time notifications to targeted audience.

From comparisons in Table 2, it also can be seen that push notification services support scalability. Push notification services implement scalability by APIs and cloud services. Virtual cloud service provides an option for enterprises to transfer workload to the cloud, which brings benefits to cloud deployment. Virtual cloud services are used for data replication management. So that the data can be replicated into an unlimited number of virtual copies in space, and management for replicated data are required. Thus, push notification services are able to set up

[23] https://wiki.mozilla.org/WebAPI/SimplePush/Protocol

application development, including composition and delivery of notifications through their own APIs. Push notification services also have cloud solutions, like data center, cloud communications, cloud storage and data center to support scalability. For instance, OpenPush uses XMPP servers, ejabberd with MongoDB infrastructure to support scalability. Similarly, Parse.com operates a number of high-throughput I/O intensive clusters with MongoDB infrastructure via cloud service to improve scalability and speed.

Push notification services support fault tolerance as shown in Table 2. Because push notification services provide replicated data to guarantee reliable notification delivery, and use real-time report mechanisms for management of redundant information and statistics. Service providers and application developers are able to track demographics, entitlements, preferences, frequency and duration of application usage, and even other attributes which can be used to build out user properties with smart segments. Besides, Apple also provides the unlimited Sandbox environment to detect faults for the whole APNS. Thus, push notification services can add applications into the unlimited Sandbox environment to test all characteristics, where applications are easily be used for application management.

For fault tolerance, some concrete examples of push notification services are given. Mozilla SimplePush realizes synchronization with garbage collections between server side and client side to achieve reliable notification delivery. Push IO uses the unlimited Sandbox environment to test all characteristics for the whole APNS to manage applications; OpenMarket has geo-redundant networks and reporting & message analysis to retrieve status and error information about user engagement and message deletion; Parse.com handles replicated data and data storage and disk caching for its application users; Awarly has error messages for robust analyses which use conventional HTTP response codes to indicate success or failure for an API request, and all errors will return error message describing the particular problem in JSON.

Based on comparisons of functionalities in Table 2, the recommended push notification services need to implement all the functionalities of batch processing, scalability, mobility and fault tolerance. All the push notification services except for Mozilla SimplePush implement all the four functionalities.

5.2.3) Comparisons of usage fees and usage limits

Usage fees and usage limits are summarized in Table 3 and 4, including signing-up requirements, number of registered applications, number of delivered notifications, number of devices used by application users and prices based on free accounts of each month. Followed with free accounts, this subsection also demonstrates premium accounts of the listed push notification services and gives detailed usage fees with usage limits.

Fields of usage fees and usage limits table are explained in details. Signing-up requirements means application developers need to register and sign up when they use a push notification service. Number of registered applications is the number of applications which application developers are able to register with a specified account. Number of delivered notifications is the number of push notifications sent and delivered to devices in a month. Number of devices used by application users refers to the number of devices which receive push notifications per application user within a specified account. Free accounts indicate that whether a push notification service has a free account.

Services	**Usage Fees & Usage Limits based on *Free Accounts*/month**				
	Signing-up Requirements	**No. of Registered Applications**	**No. of Delivered Notifications**	**No. of Devices Used by Application Users**	**Free Accounts**
Mozilla SimplePush	YES	UNLIMITED	UNLIMITED	UNLIMITED	√
Pushover	YES	UNLIMITED	5 for android and unlimited for iOS	UNLIMITED	√
Pushwoosh	YES	5	UNLIMITED	1 million	√
Amazon SNS	YES	UNLIMITED	1 million	UNLIMITED	√
OpenPush	YES	UNLIMITED	UNLIMITED	UNLIMITED	√
Parse.com	YES	1 million	1 million	UNLIMITED	√
Urban Airship	YES	UNLIMITED	1 million	UNLIMITED	√
Xtify	YES	N/A	N/A	N/A	×
Push IO	YES	N/A	N/A	N/A	√ (30-days-free trial)
Awarly	YES	UNLIMITED	1 million	10000	√
OpenMarket	YES	N/A	N/A	N/A	×
PushBots	YES	UNLIMITED	1.5 million	UNLIMITED	√
Mass Notification System	YES	N/A	N/A	N/A	×
mobDB	YES	1	600000	UNLIMITED	√
NACapp	YES	3	UNLIMITED	3	√ (90-days-free trial)
OpsGenie	YES	UNLIMITED	600 alert notifications	UNLIMITED	√

Services	Usage Fees & Usage Limits based on *Free Accounts*/month				
	Signing-up Requirements	**No. of Registered Applications**	**No. of Delivered Notifications**	**No. of Devices Used by Application Users**	**Free Accounts**
SnapComms	YES	N/A	N/A	N/A	×
Notificare	YES	5000	UNLIMITED	UNLIMITED	√

*) N/A means that the detailed usage fees and usage limits are not available online, and the information is only available to purchasers or is needed to contact the corresponding sales departments.
**) √ means the push notification service has a free account for trial, while × means the push notification service doesn't have a free account.

Table 3: Usage fees & usage limits comparisons of push notification services' free accounts

From comparisons of usage fees and usage limits in Table 3, it can be seen that all the push notification services need to sign up when using them. The usage fees and usage limits of push notification services like Xtify, Push IO, OpenMarket, Mass Notification System and SnapComms are not available online. Application developers have to contact with their Service Centers officially if they want to know fees. While, push notification services like Xtify, Push IO, OpenMarket, Mass Notification System and SnapComms don't have free accounts, and Push IO and NACapp limit free and trial days. To get a recommended push notification service, comparisons are conducted based on usage fees and usage limits based on their free accounts and premium accounts respectively in the following subsections.

5.2.3.1) Usage fees and usage limits of free accounts

Details of usage fees and usage limits can be seen from Table 3 based on their free accounts. Mozilla SimplePush and OpenPush push notification services don't have restrictions for usage limits. Amazon SNS, Parse.com, Urban Airship and PushBots limit their usages in the calculations of millions. However, millions of usage limits are approximately to unlimited usage in practice. Hence, Amazon SNS, Parse.com, Urban Airship and PushBots push notification services are cost efficient with large scale development based on their free accounts.

Number of delivered notifications of free accounts is the most important factor of usage limits. Different push notification services enable different quantities and types of notifications to deliver with free accounts. Quantities of delivered notifications are not limited for Pushwoosh, Awarly, mobDB, NACapp and Notificare in practical scenarios, but Pushwoosh, Awarly, mobDB, NACapp and Notificare restrict number of registered application and number of devices used. In

contrast, types of delivered notifications are limited for Pushover and OpsGenie. For instance, OpsGenie enables application developers to send alert notifications within free accounts, and pay for notifications sent as e-mails and SMS text messages with premium accounts.

In details, subsequent paragraphs will illustrate accurate information for free accounts of Parse.com and Urban Airship as examples. Parse.com enables starters to register 1 million applications and deliver 1 million push notifications to unlimited devices per month within a free account. Parse.com charges application developers $0.07 for every 1000 notifications per month over the free account limits. Within a free account, Parse.com allows maximum 20 notifications per second for burst limits, and Parse.com has APIs which needs to set priorities for professional accounts.

Urban Airship enables starters to register unlimited applications and deliver 1 million push notifications to unlimited devices per month within a free account. Urban Airship charges $0.001 for per push notification over 1 million, and $0.0025 for per rich push notification over 1 million in one month. User applications can get access to composers and core reports, including sent pushes and opened applications and utility time. But, application developers for profits only get access to reports or composers with a 45-day trial. Based on details in this paragraph, it can be concluded that Urban Airship provides such a small business edition premium account, which suits for small scale enterprises.

5.2.3.2) Usage fees and usage limits of premium accounts

Apart from free accounts, push notification services also have premium accounts which are used for production according to detailed information in Appendix 1 through Appendix 19. The detailed usage fees and usage limits are demonstrated in Table 4.

Services	**Usage Fees & Usage Limits based on *Premium Accounts*/month**				
	Packages	**No. of Registered Applications**	**No. of Delivered Notifications**	**No. of Devices Used by Application Users**	**Usage fees**
Mozilla SimplePush	×	×	×	×	×
Pushover	Basic	UNLIMITED	$150.00/30000 notifications	UNLIMITED	$4.99 for android

Services	Usage Fees & Usage Limits based on *Premium Accounts*/month				
	Packages	**No. of Registered Applications**	**No. of Delivered Notifications**	**No. of Devices Used by Application Users**	**Usage fees**
Pushwoosh	Indies	10	UNLIMITED	2 million	$39.95
	Professionals	50	UNLIMITED	4 million	$119.95
	Enterprises	100	UNLIMITED	UNLIMITED	$249.95
Amazon SNS	Basic	UNLIMITED	$1/million notifications	UNLIMITED	$30.00
OpenPush	×	×	×	×	×
Parse.com	Professionals	1.5 million	5 million notifications	1 million($0.05/1000 devices over)	$119.00
	Enterprises	N/A	N/A	N/A	N/A
Urban Airship	N/A	N/A	N/A	N/A	N/A
Xtify	N/A	N/A	N/A	N/A	N/A
Push IO	N/A	N/A	N/A	N/A	N/A
Awarly	Professionals	UNLIMITED	5 million e-mails ($0.005/notification over)	50000 ($0.015/device over)	$99.00
	Enterprises	UNLIMITED	5 million notifications	50000	$199.00
OpenMarket	N/A	N/A	N/A	N/A	N/A
PushBots	Indies	UNLIMITED	5 million notifications	UNLIMITED	$49.00
	Professionals	UNLIMITED	20 million notifications	UNLIMITED	$129.00
	Enterprises	UNLIMITED	UNLIMITED	UNLIMITED	$249.00
Mass Notification System	N/A	N/A	N/A	N/A	N/A
mobDB	Basic	UNLIMITED	4 million ($10/1 million notifications over)	UNLIMITED	$15.00
NACapp	Indies	20	UNLIMITED	20	$5.00/each
	Professionals	100N/A	UNLIMITED	100	$3.00/each
	Enterprises	N/A	N/A	N/A	N/A
OpsGenie	Indies	UNLIMITED	5 notifications/ application user	UNLIMITED	$8.00
	Professionals	UNLIMITED	5 notifications/ application user, $5 with extra customized features	UNLIMITED	$12.00
	Enterprises	UNLIMITED	UNLIMITED	UNLIMITED	$16.00
SnapComms	N/A	N/A	N/A	N/A	N/A
Notificare	Indies	10000	UNLIMITED	UNLIMITED	$199.00
	Professionals	50000	UNLIMITED	UNLIMITED	$399.00
	Enterprises	100000	UNLIMITED	UNLIMITED	$599.00

*) N/A means that the detailed usage fees and usage limits are not available online, and the information is only available to purchasers or is needed to contact the corresponding sales departments.
**) × means the push notification service doesn't have a premium account.

Table 4: Usage fees & usage limits comparisons of push notification services' premium accounts

From comparisons of usage fees and usage limits in Table 4, it can be seen that Mozilla SimplePush and OpenPush do not have premium accounts. This means application developers don't need to pay when they use Mozilla SimplePush and OpenPush. Besides, all usage fees and usage limits of push notification services like Xtify, Push IO, OpenMarket, Mass Notification System and SnapComms are not available online. Large scale service providers need to contact sales departments if they want to use Parse.com, Urban Airship and NACapp, because the prices for enterprise premium accounts are only available to purchasers.

To satisfy diversified demands of production, push notification services specify one to three packages for premium accounts. The three package premium accounts include indies who work in a small group for low profits, professionals for scale production and enterprises for large scale production. Most push notification services use three packages for their premium accounts.

Based on Table 4, number of delivered notifications is the most significant factor for certain usage fees. Different push notification services enable different quantities and types of notifications to deliver with free accounts. Quantities of delivered notifications are not limited for Pushwoosh, Amazon SNS, Parse.com, Awarly, PushBots, mobDB, NACapp and Notificare in practical scenarios. Even though Amazon SNS, Parse.com, Awarly, PushBots and mobDB limit their usages in the calculations of millions, they are approximately to unlimited usage in practice. However, other push notification services including Pushover and OpsGenie have limits for neither quantities nor types of notifications.

In details, subsequent paragraphs will illustrate accurate information for premium accounts. It is expected to conclude the cheapest and the most expensive ones, after introducing exact details of key notification services.

Parse.com gives available price for professional premium accounts. Parse.com allows professionals to register 1.5 million applications to push 5 million notifications to 1 million devices for $199 per month for scale productions. Parse.com charges $0.05 per 1000 over 1 million devices, and allows maximum 40 notifications per second for burst limits. Additionally, the following paragraphs introduce the details of Pushwoosh, PushBots and Awarly.

Pushwoosh allows Indies to register ten applications to push unlimited notifications to 2 million

devices for $39.95 per month for low profits. Pushwoosh also allows professionals to register 50 applications to push unlimited notifications to 4 million devices for $119.95 per month for scale productions. In the meanwhile, Pushwoosh allows enterprises to register 100 applications to push unlimited million notifications to 8 million devices for $249.95 per month, which means large scale service providers need to pay $249.95 for Pushwoosh. Furthermore, Pushwoosh gives a 20% discount of each premium accounts for annual purchasers.

PushBots has a similar price with Pushwoosh for enterprise premium accounts, as PushBots allows enterprises to register unlimited applications to push unlimited notifications to unlimited devices for $249 per month. Besides, PushBots allows professionals to register unlimited applications to push 20 million notifications to unlimited devices for $129 per month for scale productions and allows Indies to register unlimited applications to push 5 million notifications to unlimited devices for $49 per month.

Awarly has two packages for premium accounts, namely, professional and enterprise premium accounts. Awarly allows professionals to register unlimited applications and 5 million notifications pushed as e-mails to 50000 devices with 2.5 million API requests for $99 per month. Awarly charges $0.005 for per API request over 2.5 million, $0.005 for per notification over 5 million and $0.015 for per device over 50000. Moreover, Awarly allows enterprises to customized notifications sent as e-mails and SMS text messages for $199 per month, which means large scale service providers need to pay $199 for Awarly.

This paragraph summarizes the highest and lowest usage fees for enterprise premium accounts with detailed usage limits based on Table 4. mobDB is the cheapest push notification service for enterprise premium accounts. Basically, mobDB charges $15 per month for the premium account. mobDB enables application developers to register only five applications and charges $10 per application over five. mobDB also allows delivering 3.5 million push notifications to unlimited devices per month and charges $10 per 1 million over 3.5 million push notifications. Moreover, mobDB allows 4 million requests for API calls and charges $10 per 1 million over 4 million requests. Besides, mobDB gives 10 GB for data storage and charges $10 per 10 GB over 10 GB. On the basic level, large scale enterprises need to pay $15 for mobDB. Although mobDB is the cheapest push notification service, mobDB limits the number of registered and the number of used applications. Hence, application developers will pay more when using more applications.

In contrast to mobDB, Notificare is the most expensive push notification service for enterprise premium accounts according to Table 4. Notificare also provides three premium accounts for different scale levels, including Indies, professionals and enterprises. Notificare allows enterprises to register 100000 applications and push unlimited notifications to unlimited devices for $599 per month. This means large scale enterprises need to pay $599 for Notificare. In the meanwhile, Notificare allows professionals to register 50000 applications and push unlimited notifications to unlimited devices for $399 per month for scale productions. In addition to, Notificare allows Indies to register 10000 applications and push unlimited notifications to unlimited devices for $199 per month for low profits.

Nowadays, many developers are restricted by their limited budgets and intend to make the best of their budget. Hence, most application developers would like to try free products, while, service providers can utilize the opportunity to get their application users to buy their push notification services. Because application developers are easy to form a routine in minds after they try new technologies and services. Application developers are able to compare the free and trial service with other services. But, service providers need to make investments in time and money for the free accounts. After trying the free accounts, application developers will be more confident to buy the proved push notification service. In the meanwhile, application developers can figure out the real worth after trial and have some good ideas based on their trial experience, and even discover some potential problems so that to adjust them in time. In a long term, potential and loyal application developers are exploited; good impressions are established on products; and reliable relationship between service providers and application developers are gained. Hence, free and trial accounts bring benefits to both service providers and application developers. Push notification services with free accounts are better than those have no free accounts.

Push notification services always have limited bandwidth for free and trial accounts, so application developers need to pay to access additional bandwidth. The average usage fees are $39 for Indies premium accounts, $119 for professional premium accounts and $249 for enterprises premium accounts based on Table 4. If a free account exists, a free account is priority to premium accounts with expenses. In this view, Xtify is not the best choice, because it doesn’t support free trials as explained in this subsection. Urban Airship has a small business edition premium account, which suits for small scale enterprises. Although mobDB is the cheapest,

mobDB limits the number of registered and used applications and pay more when using more applications as detailed information as introduced in this subsection. To satisfy the demands of large scale enterprise, mobDB and Urban Airship are not good choice, either. Pushwoosh allows maximum 100 registered applications within enterprise premium accounts, but it is enough for most enterprises.

Based on usage fees and usage limits discussed in this subsection, the preferred push notification services should have reasonable prices with their provided services to satisfy the demands of large scale enterprises. Hence, the preferred push notification services are Pushwoosh, Parse.com and PushBots.

5.2.4) Comparisons of security

Table 5 summarizes security issues of different push notification services. Security issues include authentication model of services and SSL-supported which is the most widely deployed security protocol.

Services	**Security**	
	Authentication Model	**SSL-Supported (HTTPS)**
Mozilla SimplePush	API key	√
Pushover	API key	√
Pushwoosh	API key	√
Amazon SNS	AWS API key	√
OpenPush	API key	√
Parse.com	API key	√
Urban Airship	API key	√
Xtify	API key	√
Push IO	API key	√
Awarly	API key	√
OpenMarket	HTTP Basic Authentication	√
PushBots	HTTP Basic Authentication	√
Everbridge Mass Notification System	API key	√
mobDB	API key	√
NACapp	API key	√
OpsGenie	API key	√
SnapComms	API key	√
Notificare	API key	√

*) √ means the push notification service supports SSL, which means the push notification service uses HTTPS to deliver push notifications across networks.

Table 5: Security comparisons of push notification services

From comparisons of security in Table 5, it can be seen that all the push notification services need an authentication model with SSL for security. Currently, HTTP Basic authentication, OAuth and API key are three trends to secure APIs as authentication models. From Table 5, it can be seen that HTTP Basic authentication and API key are used by most push notification services.

As is known, HTTP Basic authentication is a method for an HTTP user agent to make a request of user names and passwords. HTTP Basic authentication implementation is the simplest technique to enforce access control of web resources. HTTP Basic authentication doesn't require cookies, session identifier or login pages. HTTP Basic authentication uses a static and standard HTTP header which doesn't have to use handshakes in anticipation. However, HTTP Basic authentication mechanism doesn't provide confidentiality protection for credentials in delivery. HTTP Basic authentication should be combined with SSL/TLS[24] for encryption, which is used to encrypt username and password. So that the server side can keep a long-lived access token, which is a username-password equivalent for each application user until the token is refreshed.

Especially, it is the easiest way to use HTTP Basic authentication which is used by most websites recently. HTTP Basic authentication has the advantage of better compliance with different environment than other authentication models. Nearly all application and backend servers support HTTP Basic authentication. Because of simplicity, HTTP Basic authentication doesn’t require a special processing, as long as the caller has reasonable precautions to keep the password secret. Thus, HTTP Basic authentication greatly decreases the work required for a service client, and the decreased complexity make the datacenter more scalable for service providers.

Compared to HTTP Basic authentication, API key is a code to identify the caller who intends to invoke a targeted API. API key enables both communicators to figure out who sends an API request and set the restrictions for the number of requests. In the case, identity plays a significant role in controlling service volume for API key. API key gives application developers a secure way to establish identity instead of actually authenticating application users with complicated passwords.

API key is more secure than complicated passwords in terms of interceptions. API keys are

[24] http://technet.microsoft.com/en-us/library/cc784450(v=ws.10).aspx

independent from each master credential, so refreshing passwords won't cause inconsistent problems. API key never exposes secrets in application users' interfaces. Because API key is used by each request and is delivered as a query string parameter or via an HTTP header. Basically, when application developers complete configurations of API key, application developers can easily authenticate by adding an *apikey* parameter to its query string. Because of never-exposures, API key can be tracked according to the initial download operations. Moreover, API key also brings benefits to service providers. Because API key can be used to maintain logs; service providers can establish quotas and tracks analysis. Hence, API key is more secure than HTTP Basic authentication from the viewpoints of both application developers and service providers as illustrated in this subsection.

In addition to, both HTTP Basic authentication and API key are able to use both HTTP and HTTPS. However, transitions on secure tunnels are via HTTPS and unsecure tunnels are via HTTP. Except for the case that HTTP Basic authentication delivery with HTTP connections is not secure at all, the other forms of delivery via HTTP are the same with the equivalents via HTTPS. To establish a secure connection, HTTP Basic authentication are needed to combined with HTTPS connections, not just uses HTTP connections. When accessing some critical information, API key need to be combined with other methods. So that API key will be used by application developers to invoke APIs. Hence, HTTP Basic authentication is slightly less secure than API key.

Overall, API key is slightly more secure than HTTP Basic authentication for both application developers and service providers. API key is the preferred authentication model than HTTP Basic authentication. The secure push notification service should combine API key authentication method with HTTPS-supported. OpenMarket and PushBots don't suit for the secure analysis. Hence, push notification services except for OpenMarket and PushBots can be the preferred push notification services.

5.2.5) Comparisons of supporting details

Push notifications are always sent as either e-mails or SMS text messages. Table 6 summarizes supporting details to compare the listed push notification services, including whether the push notification service supports e-mail, SMS and push cloud notification.

Services	Supporting Details		
	E-mail supported	SMS Text Message Supported	Pushing Cloud Notification Supported
Mozilla SimplePush	×	√	√
Pushover	√	×	√
Pushwoosh	√	√	√
Amazon SNS	√	√	√
OpenPush	√	√	√
Parse.com	√	√	√
Urban Airship	√	√	√
Xtify	√	√	√
Push IO	√	√	√
Awarly	√	√	√
OpenMarket	×	√	√
PushBots	√	√	√
Everbridge Mass Notification System	√	√	√
mobDB	√	√	√
NACapp	√	√	√
OpsGenie	√	√	√
SnapComms	√	√	√
Notificare	√	√	√

*) √ means the push notification service supports the specified form of push notifications, while × means the push notification service doesn't support the specified form of push notifications.

Table 6: Supporting details comparisons of push notification services

For supporting details of push notification services, each push notification service has a blog for application developers to communicate freely according to appendixes. From the comparisons of supporting details in Table 6, it can be seen that push notification services enable notifications sent in e-mail or SMS or both supported, and all the push notification services enable push cloud notifications.

Push notification services enable notifications sent in e-mail or SMS or both supported. Because push notification services are designed and realized to process incoming events which are specified as XML and JSON files, and turn the events into notifications as e-mails or SMSs on different platform mobile phones. Mobile push notifications sent by e-mail and SMS are two major forms as shown in Table 6. The push notification services are recommended to support both e-mail and SMS, as the demands of application users are increasing with the fast developing mobile platforms. For example, Amazon SNS supports e-mails in JSON formats which are delivered to registered addresses. E-mails deliver notifications as JSON objects, while e-mails

transmit text-based e-mails.

From Table 6, it can be seen that all push notification services support cloud services. The push notification services are currently realized based on cloud services. Meanwhile, the push notification services also provide full cloud solutions, so enterprises have no infrastructure costs or ongoing hardware maintenance costs. Hence, push notification services need to support pushing cloud notifications, and push notification services enable applications to send notifications from the cloud for scalability consequently. Cloud services also help push notification services with the scalability in evolvement. Besides, with cloud services,push notification services are able to test processes for APNS by simulations of triggering events in a Sandbox environment. For example, Urban Airship uses cloud services for scalability by adding servers, and Xtify supports cloud operations for SoftLayer.

From comparisons of supporting details in this subsection, the preferred push notification services should support notifications sent in the forms of both e-mails and SMS text messages, and also enable push cloud notifications with the help of cloud services. To satisfy increasing requirements of current mobile application users, push notification services like Pushwoosh, Amazon SNS, Parse.com, Urban Airship, Xtify, Push IO, Awarly, PushBots, mobDB, Notificare are preferred choices.

5.2.6) Comparisons of supported platforms/OS

Table 7 summarizes the comparisons in ubiquitous supported platforms/OS, including GCM for Android devices such as Samsung Galaxy, C2DM, APNS for iOS devices such as iPhone, iPod Touch, iPad, iOS 4+, Mac OS X, Microsoft for Windows Phone 7, Windows Phone 8, Windows 8, Firefox OS, RIM Blackberry and Amazon Device Messaging (ADM) for different products.

Services	**Supported Platforms/OS**						
	APNS	**C2DM**	**GCM**	**Microsoft**	**Firefox OS**	**RIM Blackberry**	**ADM**
Mozilla SimplePush	√	√	√	√	√	√	×
Pushover	√	×	√	×	×	×	×
Pushwoosh	√	×	√	√	×	√	√
Amazon SNS	√	×	√	×	×	×	√
OpenPush	√	×	√	√	×	√	×
Parse.com	√	×	√	√	×	×	×

Services	Supported Platforms/OS						
	APNS	C2DM	GCM	Microsoft	Firefox OS	RIM Blackberry	ADM
Urban Airship	√	×	√	√	×	√	×
Xtify	√	×	√	√	×	√	×
Push IO	√	×	√	√	×	×	×
Awarly	√	×	√	×	×	√	×
OpenMarket	√	√	√	×	×	√	×
PushBots	√	×	√	×	×	×	×
Everbridge Mass Notification System	√	√	√	√	×	√	√
mobDB	√	×	√	×	×	×	×
Services	**Supported Platforms/OS**						
	APNS	**C2DM**	**GCM**	**Microsoft**	**Firefox OS**	**RIM Blackberry**	**ADM**
NACapp	√	×	√	√	×	√	×
OpsGenie	√	×	√	×	×	√	×
SnapComms	√	×	√	√	√	×	√
Notificare	√	×	√	×	×	×	×

*) √ means the push notification service supports the specified platform, while × means the push notification service doesn't support the specified platform.

Table 7: Supported notifications/OS platforms comparisons of push notification services

From comparisons of supported notifications platforms/OS in Table 7, it can be seen that almost all push notification services support Android devices such as Samsung Galaxy, iOS devices with a little differences in details.

For APNS, Amazon SNS, mobDB, Awarly, PushSharp supports iPhone and iPad; Parse.com supports iPhone, iPad and Mac OS X; Pushwoosh supports iPhone, iPad, Mac OS X Push Notification SDK and Safari; Pushover supports iPhone, iPod Touch, iPad and iOS 4+.

For GCM, mobDB supports Android 2.2+ devices; PushSharp supports Chrome, phones and tablets; Pushover supports Android 2.0+ and optimizes for Android 4.0+.

For Windows Phone, Parse.com supports Windows 8; Pushwoosh supports Windows 8 SDK Integration and PhoneGap/Cordova SDK for Windows Phone 8; PushSharp supports Windows Phone 7/7.5/8 including FlipTile, CycleTile, IconicTile Templates and Windows 8. Besides, Pushwoosh, Amazon SNS, Everbridge Mass Notification System and SnapComms support ADM. Additionally, some push notification services also support C2DM, Firefox OS, Blackberry and

ADM.

As GCM, APNS and Windows Phone are three main platforms, preferred push notification services should support GCM, APNS and Windows Phone platforms at the same time. Hence, Mozilla SimplePush, Pushwoosh, OpenPush, Parse.com, Urban Airship, Xtify, Push IO, Everbridge Mass Notification System, NACapp and SnapComms are options to be chosen as the preferred push notification service.

5.2.7) Summary of push notification services

Based on comparisons of basic information in Table 1, it can be seen that REST service-oriented architecture paradigm is widely used and REST service-oriented architecture paradigm has the advantages of a lower latency and a smaller average packet size with a smaller message payloads than SOAP-based equivalents. For integration and purchase of push notification services, Pushwoosh, Parse.com and PushBots claim they support high-throughput and fast transmission traffic explicitly in their advertisements as Subsection 5.2.1. So Pushwoosh, Parse.com and PushBots are preferred choices. In the meanwhile, Amazon SNS is an alternative choice, because Amazon SNS are easy to be integrated for those enterprises that have SOAP-based backend servers.

As functionalities comparisons depicted in Subsection 5.2.2, it is summarized that the preferred push notification services should realize all the functionalities of batch processing, scalability, mobility and fault tolerance. All the push notification services except for Mozilla SimplePush satisfy the individual demands for functionalities, so they can be the preferred choice.

As usage fees and usage limits discussed in Subsection 5.2.3, it can be shown that the preferred push notification services are Pushwoosh, Parse.com and PushBots. Because they have reasonable prices with their provided services, which satisfy the growing requirements of large scale enterprises.

According to comparisons of security in Subsection 5.2.4, it is known that push notification services except for OpenMarket and PushBots are preferred choices. Because a secure push notification service should combine API key authentication method with HTTPS to support security, but OpenMarket and PushBots don’t.

From supporting details comparisons in Subsection 5.2.5, Pushwoosh, Amazon SNS, Parse.com,

Urban Airship, Xtify, Push IO, Awarly, PushBots, mobDB and Notificare are preferred. Because they are able to satisfy increasing requirements of current mobile users, and requirements include notifications can be sent in the forms of e-mails and SMS text messages, and push cloud notifications can work with cloud service.

From Table 7, Mozilla SimplePush, Pushwoosh, OpenPush, Parse.com, Urban Airship, Xtify, Push IO, Everbridge Mass Notification System, NACapp and SnapComms are all alternatives, considering GCM, APNS and Windows Phone platforms.

All the comparisons from Subsection 5.2.1 to Subsection 5.2.6 are summarized in Table 8. The title of the column is listed from the best choice to the last choice, and the priority is ordered from one to four. The title of the row is the six characteristics of push notification services. The contents of Table 8 are the names of push notification services. Table 8 summarizes all the comparisons from Subsection 5.2.1 to Subsection 5.2.6.

Order of Choice	**Six Chosen Characteristics of Push Notification Services**					
	Basic Information	**Functionalities**	**Usage fees & Usage limits**	**Security**	**Supporting Details**	**Supporting Platforms/OS**
1	Pushwoosh	Any except for Mozilla SimplePush	Pushwoosh	Any except for OpenMarket, PushBots	Any	Pushwoosh
2	Parse.com		Parse.com			Parse.com
3	PushBots		PushBots			
4	Amazon SNS					Push IO

*) * includes Mozilla SimplePush, OpenPush, Urban Airship, Xtify, NACapp, Everbridge Mass Notification System and SnapComms.

Table 8: Summary of six characteristics for push notification services

From Table 8, Pushwoosh and Parse.com are better than other push notification services, where Parse.com satisfies the analysis results of the six-comparison-point from Subsection 5.2.1 to Subsection 5.2.6. Pushwoosh and Parse.com are almost the same, including usage fees of the equivalent premium accounts. However, Pushwoosh has a slight benefit than Parse.com. Pushwoosh has three premium accounts for Indies, professionals and enterprises premium accounts, which are more detailed to satisfy different scale development, while, Parse.com doesn’t take indies premium accounts into account. Apart from usage fees, Pushwoosh supports

two more notification platforms compared with Parse.com. Concisely, Pushwoosh supports Blackberry and Kindle ADM, but Parse.com doesn't. Hence, Pushwoosh is more considerable than Parse.com.

To summary all the comparisons, Pushwoosh is the priority choice among all the listed push notification services. To summarize the benefits of Pushwoosh, Pushwoosh uses geo zone to trigger push notifications, statistics to track performance of campaign and application groups to multicast rich HTML pages through remote JSON API.

In addition to, other six alternatives are also given based on the comparisons. In general, alternatives are Amazon SNS, Parse.com, PushBots, Urban Airship, Xtify and Awarly. The reasons are summarized as follows.

(1) For those enterprises that have SOAP-based backend servers, Amazon SNS is a preferred choice for easy integration with their existing servers. (2) Parse.com is an alternative to Pushwoosh as discussed in this subsection, which satisfies the requirements of REST service-oriented architecture paradigm, functionality, security, notification forms, widely three widely used platforms with reasonable usage fees and usage limits. (3) PushBots is a subsequent choice, which supports all the requirements of Parse.com. But PushBots is less security than Parse.com, because PushBots uses HTTP Basic Authentication as the authentication model instead of API keys. (4) Urban Airship and (5) Xtify are alternatives as well. The fact is that the high-throughput and fast transmission traffic for REST service-oriented architecture paradigm are not explicitly illustrated by Urban Airship and Xtify. For usage fees, Urban Airship is suitable for small scale enterprises, as Urban Airship provides the small business edition premium account apart from free account. Besides, Xtify doesn't support free trials, and the prices of Xtify premium accounts are not available to non-purchasers. (6) Awarly is a minor choice, because Awarly doesn't support Windows Phone. But, Awarly satisfies security requirement with a medium price for premium accounts and supports Blackberry.

To summary the six options, the recommended push notification service should be chosen from Pushwoosh, Parse.com, PushBots, Urban Airship, Xtify and Awarly, and these options are listed from priority choice to the last choice in Table 8. One more option can be chosen from Amazon SNS and NACapp, which are used to integrate with SOAP-based backend servers. Amazon SNS is much easier to be integrated compared with NACapp.

5.3 Push notification servers

A push notification server is a piece of hardware or a virtual machine. A push notification server is composed of available services for clients in the same network. A push notification server is able to provide a specified push notification service as software in client side, and it belongs to a service provider. For example, a computer can act as a server which allows a software program to run on it. In that way, it is necessary for a service provider to establish a push notification server themselves. The following subsection introduces current push notification servers with corresponding appendixes in details.

5.3.1) AeroGear UnifiedPush Server

AeroGear UnifiedPush Server[25] is a server that delivers push notifications to multiple push networks including APNS, GCM and Mozilla Simple Push. AeroGear UnifiedPush Server makes it possible to combine mobiles and enterprises together. AeroGear UnifiedPush Server provides a Notification Service API to unify and simplify mobile development for diversified platforms as depicted in overview Figure 9. AeroGear UnifiedPush Server is released under Apache Version 2.0 license with developers' support from Red Hat, Inc. The detailed information is in Appendix 19.

[25] http://aerogear.org/

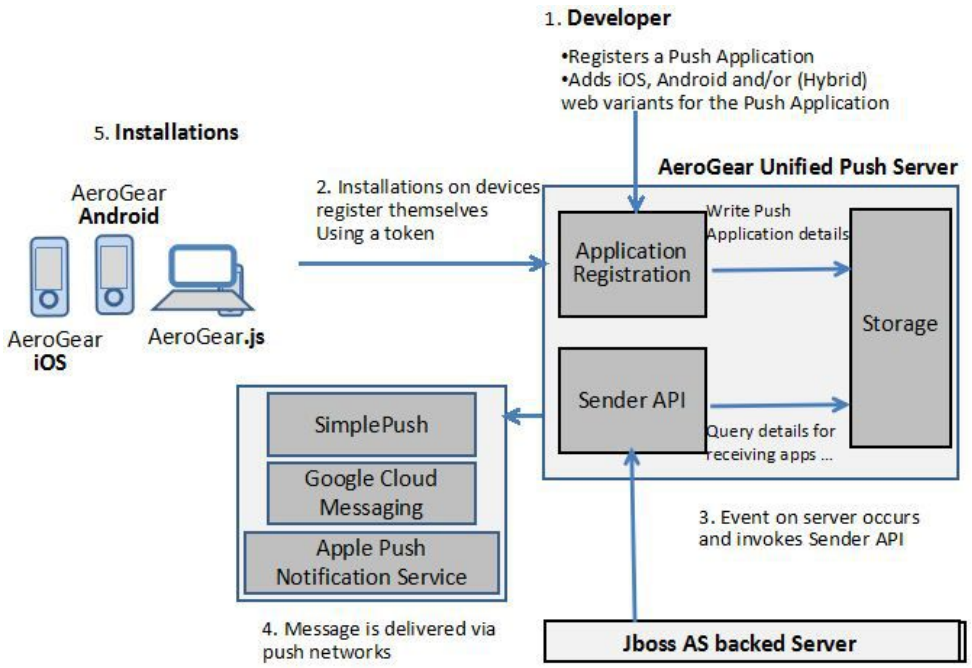

Figure 9: Overview of AeroGear UnifiedPush Server data flow.

The Notification Service API is the core of AeroGear UnifiedPush Server. The Notification Service API is a signaling mechanism to notify application users of updates in server side. The Notification Service API is used to transmit notifications to invoke corresponding applications instead of working as a data carrying system. AeroGear UnifiedPush Server also defines a logical abstraction for mobile applications to use the API, and the logic abstraction supports different Push Networks. In this way, AeroGear UnifiedPush Server is able to send notifications to different mobile clients easily.

AeroGear UnifiedPush Server introduces a concept of variant where a variant means a mobile platform. AeroGear UnifiedPush Server allows one push application to have several variants. e.g. one push application can own an iOS variant and an Android variant. Besides, AeroGear UnifiedPush Server provides a one-time-password and two-factor authentication to ensure security.

5.3.2) Pushd

Pushd[26] is a server which provides a simple unified push notification service for server side, so

[26] https://github.com/rs/pushd

that server side are able to send notifications to mobile native applications and web applications based on Thialfi as depicted in Figure 8. Pushd releases all the information in GitHub, including its RESTful API and MIT license without any fees. Especially for developer support, Pushd is mainly done by contributer Olivier Poitrey alone who contributes most than other four contributors. The detailed information is in Appendix 20.

5.3.3) PushSharp

PushSharp[27] is a push notification server which sends notifications to multiple platforms. PushSharp uses API to construct notifications for various platforms based on Thialfi notification delivery mechanism in Subsection 4.4, and supported platforms include iOS, Android and Windows Phone mobile platforms. Moreover, PushSharp is able to add notification channels in a scalable way. PushSharp is open source and released under Apache Version 2.0 license. The detailed information is in Appendix 21.

5.3.4) Uniqush

Uniqush[28] is open source software, which provides server side with a unified push notification service without fees. The overview figure of Uniqush is similar to AeroGear Overview Figure 9 in Subsection 5.3.1. Uniqush uses Apache Version 2.0 license to distribute Uniqush source codes and a Creative Commons Attribution 3.0 Unported License to deal with related documents. The detailed information is in Appendix 22.

Uniqush has several significant details. Uniqush is a standalone server program which is dedicated to push service. Hence, application developers can run Uniqush on their own servers. Uniqush has a simple and unified interface provided for application developers in server side, which is similar to Pushd in Subsection 5.3.2. Uniqush resends notifications for recoverable errors whenever possible without caring about resend implementation of failure mechanisms in server side. Uniqush uses native multicasting in APIs to extend more compatibility for GCM of Android, APNS of iOS and ADM of Kindle tablets.

5.3.5) OpenMobster

---

27 https://github.com/Redth/PushSharp
28 http://uniqush.org/

OpenMobster[29] is an open source push notification service for mobile backend applications. OpenMobster supports multiple mobile platforms, including iPhone, Android, Windows Phone, Blackberry and Symbian. OpenMobster also supports data synchronization, real-time push notifications and HTML5 hybrid applications. The features of data synchronization and real-time push notifications are realized based on Thialfi as depicted in Figure 8, while hybrid applications are implemented by HTML5, JavaScript and CSS for native application development. The detailed information is in Appendix 23.

5.4 Comparisons of push notification servers

A push notification servers is the other category of push notification products. Push notification servers are compared in a similar way as push notification services except for usage fees & usage limits. Technical characteristics of push notification servers are dominated by six comparison points from the viewpoint of QoS. The chosen characteristics are basic information, functionalities, licenses, security, supporting details and supported platforms/OS. Basic information refers to service-oriented architecture paradigms, notification transport protocols and data formats. Functionalities include client install requirements, batch processing, mobility, scalability and fault tolerance. Licenses discuss about signing-up requirements, data licenses, licenses for server/client and usage fees. Security illustrates authentication model of services and SSL-supported. A license of a push notification server enables application developers to know the restrictions of rights to use, modify, publish, merge, sublicense or sell copies within legal bounds. In this way, application developers are able to make a budget or plan when they need to buy a push notification server. Supporting details show whether notifications can be delivered in the forms of e-mails, SMS text messages and push cloud notifications. Supported platforms/OS part takes GCM, C2DM, APNS, Microsoft, Firefox OS, Blackberry and ADM into considerations.

Overall, six comparison points not only have great availability in the market, but gain popularity of the push notification platforms and optimality for the situation of large scale enterprises. This subsection compares push notification servers from these six chosen characteristics.

---

[29] http://www.openmobster.com/index.html

5.4.1) Comparisons of basic information

Table 8 summarizes basic information, including service-oriented architecture paradigms for servers, notification transport protocols for communication and data formats.

Servers	Basic Information		
	Service-oriented Architecture Paradigms	**Notification Transport Protocols**	**Data Formats**
AeroGear Push Notification Server	REST	HTTP, HTTPS, SMTP	JSON
Pushd	REST	HTTP, HTTPS, SMTP	JSON
PushSharp	REST	HTTP, HTTPS, SMTP	JSON
Uniqush	REST	HTTP, HTTPS, SMTP	JSON
OpenMobster	REST	HTTP, HTTPS, SMTP	JSON

Table 9: Basic information comparisons of push notification servers

From comparisons of basic information in Table 9, it can be seen that all the push notification servers use REST as service-oriented architecture paradigms, HTTP(S) and SMTP as notification transport protocols, JSON as data formats. Hence, all push notification servers also have their RESTful APIs.

In fact, developments should consider integrations with different data formats should be considered for practical scenarios based on the comparisons in Subsection 5.2.1. REST is a better service-oriented architecture paradigm compared with SOAP. Because REST suit for both XML and JSON data formats with HTTP(S) and SMPT as protocols for secure notification delivery.

According to Table 9, all the push notification servers implement REST service-oriented architecture paradigm in JSON data formats with HTTP(S) and SMPT protocols. Any push notification server can be the recommended push notification server to be purchased.

5.4.2) Comparisons of functionalities

Table 10 summarizes five functionalities, including client install requirements, batch processing, mobility, scalability and fault tolerancy. Clients install requirements indicate whether push notification servers can be used before they are installed. Besides, definitions of batch processing, mobility, scalability and fault tolerance are the same with the definitions in Subsection 5.2.2. Batch processing gives the ability of executing a series of processes with one request without

personal intervention. Scalability is the ability to handle the amount of burden work caused by an increasing number of application users in the server side. The server is able to deal with the increasing loads with scalability. Fault tolerance refers to the property that enables a service to continue operating properly in the event of some of its components have unexpected faults.

Servers	Functionality				
	Clients Install Required	**Batch processing**	**Mobility**	**Scalability**	**Fault tolerance**
AeroGear Push Notification Server	√	√	√	√	√
Pushd	√	√	√	√	√
PushSharp	√	√	√	√	√
Uniqush	√	√	√	√	√
OpenMobster	√	√	√	√	√

*) √ means the push notification server supports the specified functionality.

Table 10: Functionality comparisons of push notification servers

From comparisons of functionalities in Table 10, it can be seen that all the current push notification servers realize four functionalities, including batch processing, mobility, scalability and fault tolerance with client installation required. The listed push notification servers implement batch processing, mobility, scalability and fault tolerance in a similar way as push notification services in Subsection 5.2.2.

All the push notification servers support batch operations. Push notification servers use batch processing to deliver unique payloads or push scheduled notifications to a set of devices. Application developers are able to get access to databases and SDKs in server side. In the meanwhile, RESTful APIs help application developers to get access to functionalities for creation of new applications. The functionality of batch processing is usually integrated in the feature of group management. This means the functionality of batch processing is an important function of group management. Hence, push notification servers which support batch processing have a feature of group management, such that audience are segmented by targeted or broadcast messaging groups. For example, AeroGear UnifiedPush Server integrates Mozilla SimplePush in a unified push notification service. In the meanwhile, AeroGear UnifiedPush Server has a feature of delivery groups to manage lists of application users and push notifications in a single API request.

Push notification servers use a feature of smart segments to handle mobility. Because the feature of smart segments gives a way for push notification servers to deliver relevant and tailored push notifications based on application users' preferences or their real time locations. So that push notifications can be used to offer important product news, updates of service provides and scheduled reminders based on current locations to application users on the move. For example, AeroGear UnifiedPush Server enables service providers to segment their audience. So that AeroGear UnifiedPush Server can provide relevant real time notifications to target audience.

From Table 10, it also can be seen that push notification servers use scalability to satisfy growing requirements of pushing notification to various platforms. Virtual cloud services are used for data replication management. Because virtual cloud services enable enterprises to use cloud deployment approaches to transfer workloads into the cloud. Besides, push notification servers have cloud solutions, like data center, cloud communications, cloud storage and data center to support scalability.

Push notification servers support fault tolerance. Because push notification services use replicated data to guarantee reliable notification delivery, and use real-time report mechanisms to manage of redundant information and statistics. Thus, service providers and application developers are able to track demographics, entitlements, preferences, frequency, duration of application usage, and even other attributes which can be used to build out user properties with smart segments. Besides, push notification servers use Apple's unlimited Sandbox environment to detect faults for the whole APNS. For example, Pushd has a feature of data synchronization and a feature of real-time push notifications which are realized based on Thialfi.

Based on the comparisons of functionalities in Table 10, the recommended push notification server needs to realize all the functionalities of batch processing, scalability, mobility and fault tolerance with client install requirements. All the push notification servers realize batch processing, mobility, scalability and fault tolerance functionalities with client install requirements. Hence, any listed push notification server can be the recommended push notification server.

5.4.3) Comparisons of licenses

Table 11 summarizes issues related to licenses, including signing-up requirements, data licenses,

licenses for server/client and usage fees. In details, signing-up requirements means whether application developers need to register and sign up when they use push notification servers. Data licenses refer to licenses to be used when application developers develop projects based on a given push notification server. Licenses are used either by server side or by client side. Usage fees mean the price to be paid when application developers use a push notification server.

Servers	Licenses			
	Signing-up Requirements	Data Licenses	Licenses for S/C	Usage Fees
AeroGear Push Notification Server	√	Apache License Version 2.0	Server	Free and open source
Pushd	√	MIT License	Server	Free and open source
PushSharp	√	Apache License Version 2.0	Server	Free and open source
Uniqush	√	Apache License Version 2.0	Client	Free and open source
OpenMobster	√	Eclipse Public License 1.0	Server	Free and open source

*) √ means the push notification server supports the specified functionality.

Table 11: Comparisons of licenses for push notification servers

From comparisons in Table 11, it can be seen that all the push notification servers need signing-up requirements. Because account and phone number verification are required when application developers sign up in practice. All the listed push notification servers are free and open source in terms of usage fees. Thus, differences lie in different open licences used by different push notification servers.

It is seen from Table 11 that the listed push notification servers use open licenses including Apache Version 2.0 License, MIT and Eclipse Public License 1.0. Among these licenses, Apache License Version 2.0 is the friendliest one. Apache License Version 2.0 contains a patent grant. In this way, projects authors are able to satisfy any required right proposed by application developers.

It is also seen from Table 11 that Apache License Version 2.0 is used by most push notification servers, while MIT License is used by Pushd and Eclipse Public License 1.0 is used by OpenMobster. These licenses are used to limit the rights of server side or client side when application developers use the push notification server. In practice, most of the listed push

notification servers use their licenses to limit the rights of server side. While, AeroGear UnifiedPush Server and PushSharp use Apache License Version 2.0 for server limits, and Uniqush uses Apache License Version 2.0 for client limits. Thus, the preferred license for push notification servers is Apache License Version 2.0 and Apache License Version 2.0 is used for server limits. Hence, AeroGear UnifiedPush Server and PushSharp are preferred proposes for server development with Apache License Version 2.0.

5.4.4) Comparisons of security

Table 12 summarizes security issues of different push notification servers. Security issues are authentication model of servers and SSL-supported, which shows the most widely deployed security protocols.

Servers	**Security**	
	Authentication Model	**SSL Support (HTTPS)**
AeroGear Push Notification Server	HTTP Basic Authentication & HTTP Digest	√
Pushd	HTTP Basic Authentication	√
PushSharp	API key	√
Uniqush	API key	√
OpenMobster	API key	√

*) √ means the push notification server supports the specified security.

Table 12: Security comparisons of push notification servers

From the comparisons of security in Table 12, it can be seen that push notification servers use authentication models combined with HTTPS. Besides, all the push notification servers support SSL. AeroGear UnifiedPush Server uses both HTTP Basic authentication and HTTP Digest as authentication models. Pushd uses HTTP Basic authentication. The other push notification servers use API key.

Based on the comparisons of HTTP Basic authentication and API key in Subsection 5.2.4, it is known that HTTP Basic authentication is slightly less secure than API key in theory. However, HTTP Basic authentication has advantages from three viewpoints. HTTP Basic authentication is the easiest approach to securing APIs among all the authentication models, and HTTP Basic authentication is widely used by most websites. In the meanwhile, HTTP Basic authentication is supported by nearly all application users and backend servers such that HTTP Basic

authentication is better compliant with different environment than other authentication models. Besides, HTTP Basic authentication doesn’t require a special processing. HTTP Basic authentication greatly decreases the work required for a service client, and the decreased complexity makes the datacenter more scalable for service providers. Hence, HTTP Basic authentication with HTTPS-supported is not a bad authentication model.

Additionally, AeroGear UnifiedPush Server uses HTTP authentication of both basic and digest access authentication together. HTTP Digest authentication uses MD5 security to protect application users from leaking cleartext password to attackers. In the case, HTTP Digest authentication makes a supplement for HTTP Basic authentication. HTTP connections protected by Digest authentication are equally secure with HTTPS connections. Because the only difference between HTTPS and HTTP connections is the way to securing connections, where HTTPS connections give a secure way but HTTP don’t. In details, HTTPS connections encrypt everything with *Public Key Encryption*, and HTTP connections sent in the cleartext are protected by Digest authentication. Hence, HTTP Digest authentication makes up for the slight disadvantage of HTTP connections compared to API key.

HTTP Basic authentication is less secure than API key in theory as discussed in Subsection 5.2.4, however, HTTP Digest authentication makes up for the slight disadvantage. Moreover, integration and development issues should be considered in practical cases when service providers plan to purchase push notification servers or use open source codes for redesign. With predefined specifications, application developers should implement push notification servers for different application users to interact with each other easily. From the viewpoint of easy-to-use, it is better to adopt a basic authentication model which can be in compliance with any environment. Besides, authentication models should be simple enough to make the push notification server to serve as a fundamental service easily. For API key cases, API key should be able to solve a problem of downloadable applications in web 2.0 services when API key values have to be embedded in distributed applications. However, HTTP Basic authentication doesn’t embed any value in the distributed applications, so HTTP Basic authentication doesn't have such a problem. Hence, HTTP Basic authentication is simpler than API key in practice.

It is necessary to integrate available knowledge into practical open source projects. According to discussions in this subsection, HTTP Basic authentication with HTTP Digest authentication is

more suitable than API key for push notification servers in practice. From Table 12, AeroGear Push Notification Server is the only push notification server which uses HTTP Basic and Digest access authentication. Hence, AeroGear Push Notification Server is a priority choice in terms of security.

5.4.5) Comparisons of supporting details

Push notifications are always sent as either e-mails or SMS text messages. Table 13 summarizes supporting details of listed push notification servers, including whether push notification servers support e-mail, SMS and push cloud notification.

Servers	Supporting Details		
	E-mail Supported	**SMS Text Message Supported**	**Pushing Cloud Notification Supported**
AeroGear Push Notification Server	√	√	√
Pushd	√	√	√
PushSharp	√	√	√
Uniqush	√	√	√
OpenMobster	√	√	√

*) √ means the push notification server supports the specified detail.

Table 13: Supporting details comparisons of push notification servers

From comparisons of supporting details in Table 13, it can be seen that push notification servers enable notifications sent in e-mail or SMS or both supported, and all the push notification servers enable push cloud notifications. Push notification servers enable notifications sent in e-mail or SMS or both supported. Push notification servers use the same approaches to support e-mail, SMS and push cloud notifications as push notification services in Subsection 5.2.5. Practically, AeroGear UnifiedPush Server supports e-mails in JSON formats, and e-mails are delivered to registered addresses. AeroGear UnifiedPush Server also provides full cloud solutions to support push cloud notifications to expand scalability.

From comparisons of supporting details in this subsection, the recommended push notification servers should support notifications sent in the forms of both e-mails and SMS text messages, and also allow push cloud notifications with cloud service. It is seen from Table 13 that all the push notification servers realize notifications sent in the forms of both e-mails and SMS text

messages, and also support push cloud notifications with cloud service. Hence, any listed push notification server can be the recommended push notification server.

5.4.6) Comparisons of supported platforms/OS

Table 14 summarizes the comparisons in supported ubiquitous platforms/OS, including GCM and C2DM for Android devices like Samsung Galaxy, APNS for iOS devices, e.g. iPhone, iPod Touch, iPad, iOS 4+ and Mac OS X, Microsoft for Windows Phone 7/8 and Windows 8, Firefox OS, RIM Blackberry and ADM for different products.

Servers	Supported Platforms/OS						
	GCM	**C2DM**	**APNS**	**Microsoft (WP 7/8, Windows 8)**	**Firefox OS**	**RIM Blackberry**	**ADM**
AeroGear Push Notification Server	√	×	√	×	×	×	×
Pushd	√	√	√	√	√	×	×
PushSharp	√	×	√	√	√	√	×
Uniqush	√	√	√	×	×	×	√
OpenMobster	√	×	√	√	×	√	×

*) √ means the push notification server supports the specified functionality, while × means the push notification server doesn't support the specified functionality.

Table 14: Supported platforms comparisons of push notification servers

Based on comparisons of supported notifications platforms/OS in Table 14, it can be seen that almost all push notification servers support Android devices such as Samsung Galaxy, iOS devices with a little differences in details.

For APNS and GCM, AeroGear Push Notification Server, Pushd, PushSharp, Uniqush, OpenMobster all support iPhone, iPad and android devices. Besides, PushSharp also supports Chrome and tablets. For Windows Phone, Pushd, PushSharp and OpenMobster support Windows Phone; PushSharp supports Windows Phone 7/7.5/8, including FlipTile, CycleTile, IconicTile Templates and Windows 8. Additionally, some other push notification servers also support C2DM, Firefox OS, Blackberry and ADM.

As three prevalent platforms are GCM, APNS and Windows Phone, preferred push notification servers should support the three platforms at the same time. Pushd, PushSharp and OpenMobster

support all the three platforms. Another alternative is AeroGear UnifiedPush Server which supports both GCM and APNS.

5.4.7) Summary of push notification servers

Based on comparisons of basic information in Table 9, all the push notification servers implement REST service-oriented architecture paradigm in JSON data formats with HTTP(S) and SMTP. So any listed push notification servers can be taken as an option to be purchased.

As functionalities comparisons depicted in Subsection 5.4.2, it can be seen that the preferred push notification servers should have client installation requirements and realize all the functionalities of batch processing, scalability, mobility and fault tolerance. All the push notification servers satisfy such demands of functionalities. Hence, any listed push notification servers can be the recommended push notification server.

As licenses discussed in Subsection 5.4.3, it can be shown that the preferred push notification servers are AeroGear UnifiedPush Server and PushSharp which use Apache License Version 2.0.

According to comparisons of security in Subsection 5.4.4, it is known that AeroGear Push Notification Server is a priority choice. Because, the combination of HTTP Basic and Digest access authentication is more suitable than API key to act as an authentication model for push notification server, particular for integration of available knowledge into practical open source projects. AeroGear Push Notification Server is a push notification server which use HTTP Basic and Digest authentication together.

From supporting details comparisons in Subsection 5.4.5, any listed push notification server can be the preferred push notification server to satisfy increasing. They support notifications sent in the forms of e-mails and SMS text messages, and also enable push cloud notifications with the help of cloud service.

From Table 14, it should be taken GCM, APNS and Windows Phone platforms into consideration. The preferred push notification servers should be Pushd, PushSharp and OpenMobster. Besides, another alternative is AeroGear UnifiedPush Server which supports both GCM and APNS.

All the comparisons from Subsection 5.4.1 to Subsection 5.4.6 are summarized in the following table. Titles of columns are listed from the best choice for a specified characteristic, and the

priority is ordered from one to two. Titles of rows are six characteristics of push notification servers. Contents of Table 15 are names of push notification servers. Table 15 demonstrates all the comparisons from Subsection 5.4.1 to Subsection 5.4.6.

Order of Choice	**Six Chosen Characteristics of Push Notification Servers**					
	Basic Information	**Functionalities**	**Licensing**	**Security**	**Supporting Details**	**Supported Platforms/OS**
1	*	*	AeroGear UnifiedPush Server, PushSharp	AeroGear UnifiedPush Server	*	Pushd, PushSharp, OpenMobster
2						AeroGear UnifiedPush Server

*) * means the push notification server supports the specified characteristics.

Table 15: Summary of six characteristics for push notification servers

Based on Table 15, AeroGear UnifiedPush Server is a preferred option in the viewpoints of Apache License Version 2.0 and security. But, AeroGear UnifiedPush Server only supports both GCM and APNS without login requirement.

PushSharp could be an option, where PushSharp considers Apache License Version 2.0 and security with login requirements. However, PushSharp supports complicated API key for security issues, which is not suitable for practical cases compared to other authentication models. Hence, PushSharp is not a good option for the preferred push notification server.

Pushd is an alternative, because Pushd considers secure issues and supported platforms. However, Pushd uses MIT License, which is less friendly than Apache License Version 2.0. Moreover, Pushd is almost maintained by Contributer *Olivier Poitrey* individually, while AeroGear UnifiedPush Server is supported by *Red Hat, Inc*. Hence, Pushd has less developer supported than AeroGear UnifiedPush Server.

To summary the six options, the recommended push notification server should be chosen from AeroGear UnifiedPush Server and Pushd. (1) AeroGear UnifiedPush Server is the priority choice; (2) Pushd is an alternative.

6 Summary and discussion of push notification products

In the Subsection 5.2.7 and Subsection 5.4.7, the recommended push notification services and push notification servers are given. However, in terms of the two categories for push notification products, different scale enterprises have distinct requirements for push notification services and push notification servers during different development periods. This subsection distinguishes push notification services from push notification servers, and then gives several compromise proposals.

6.1 Summary of push notification services

A push notification service is a process that supports services to run across diversified networks. A push notification service can run either on the purchaser's server or on push notification service owner's server. This means purchasers can use either their own servers or the push notification service owner's servers. Because of a limited budget on IT service, small and medium-sized enterprises usually choose a push notification service to make the best of their limited IT budgets.

By using push notification services, enterprises don't have to purchase their own servers. When enterprises use push notification service owner's server, host servers of the chosen push notification service are adjusted by the owner of the chosen service. The chosen push notification service provides enterprises with a stable host server and enables enterprises to use high speed network to get access to the service. The chosen push notification service also supports fixed broadband and standard maintenance services by combining a data center to like providing free server surveillance service and 24-hour telephone technical support. Thus, the push notification service is able to cover the whole world and deliver business to every corner of the globe.

With push notification service, enterprises are able to use the chosen push notification service by installing system software or application software via an equipped high-performance server exclusively. The network service functionalities cover available dns, ftp, web and mail with high costs of leased line in network devices. Hence, the initial input by enterprises is reduced, and enterprises are able to focus more on business.

6.2 Summary of push notification servers

A push notification server, belonging to the enterprise, is a piece of hardware or on a virtual machine. A push notification server is able to support a specified push notification service as software in client side. So a push notification server is also software which consists of services available to clients on the same network. However, it is essential to establish their own host servers for enterprises to realize push notification servers. A host server enables an enterprise to centralize and protects automatically file backups from data losses. A host server also allows remote access to database in a secure and cost-efficient way, as today's servers aren't so much costly or complex when compared with a standard desktop.

Enterprises can use their existing hardware to work as host servers for their chosen push notification server. Enterprises need to select an appropriate push notification server to equip with their existing hardware. Thus, several points need to be taken into considerations as follows when considering purchasing a push notification server.

First of all, enterprises need to have essential power and Ethernet cables to wherever the servers are located. When enterprises don't have network across the working environment, a wireless network and some wireless gateway devices or router devices are also needed to be bought and installed. With networks, enterprises are able to select data and applications to be shared on their host server.

Besides, enterprises should estimate how much storage space is required for the chosen push notification server. It is necessary to evaluate the number of user devices which already enrolled in business with a prospective increasing number in the near future.

Moreover, backup options and security are important issues as well. Currently, cloud-based backup services support offsite storage to retrieve data in case of accidental data loss due to user errors or equipment failures. As most servers centralize most data in one place, the location of the server is critical and should be considered before purchasing. Hence, enterprises should consider both physical security and online risks. For physical security, enterprises should have enough room and computing power to provide adequate ventilation and airflow, including cooling and humidity controls and I/O management and even space for extensions. For online risks, enterprises should have firewall software to isolate internal office network from Internet. As a majority of servers and routers have built-in firewall features, host servers should also

support anti-virus software. For example, enterprises need to think about detection of automatically scan mechanism for host servers' hard drives and even protection from malicious software spread by USB thumb drives.

Furthermore, enterprises need to pay for bandwidth when using the chosen push notification server. In addition to, the maintenance of the server is another cost which should be handled by either a particular employee or an outsourcing group.

6.3 Discussion of push notification products

Demands vary based on different scales of enterprises. To satisfy diverse requirements of enterprises, it is necessary to categorize enterprises into two parts, namely small or medium scale enterprises and large scale enterprises. This subsection illustrates the differences between push notification services and push notification servers, then matches different categories of push notification products with different scales of enterprises.

For a small or medium scale enterprise, it is better to select a push notification service rather than a push notification server. A small or medium scale enterprise is able to use a push notification service via an equipped high-performance server, and the way of using the push notification service is as simple as installing a system software or an application software exclusively. Hence, a push notification service works efficiently within a given budget, and an enterprise is able to focus more on business research and development by reducing its initial input.

Especially, responsibilities of maintenance can be taken as a part-time job for a small scale enterprise. A cost efficient solution to maintenance for a small-scale enterprise is that an employee charges maintenance with other delegates at the same time. In addition to, physical security is a big issue for small businesses, where a careless employee might damage a host server easily. Small scale enterprises are hard to manage physical security issues, so small scale enterprises are better not choose a push notification server.

For a small or medium scale enterprise, options of the preferred push notification service are discussed in Subsection 5.2.7. Pushwoosh is the priority choice among all the push notification services. In addition to, alternatives are Amazon SNS, Parse.com, PushBots, Urban Airship, Xtify and Awarly.

For a large scale enterprise, compromise proposals vary with budgets and development requirements. When a large scale enterprise intends to set up a simple notification service, it is recommended to choose a push notification service as the same reasons for small and medium scale enterprises.

A large scale enterprise should use a push notification server when the enterprise intends to develop further push notification service or provides a large scale service to satisfy increasing demands. Because push notification service of push notification servers is running on enterprises' own servers, so that enterprises are able to add server configuration and optimize development anytime they need. This also means technical employees are able to adjust changes in server side in time without contacting other organizations. If large scale enterprises intend to use free and open source push notification servers like AeroGear UnifiedPush Server, large scale enterprises can download source codes from products' websites and set up host servers without any fees. If large scale enterprises would like to use existing hardware and enough room, large scale enterprises can use existing equipment such as electricity-supported, ventilation-supported, airflow-supported, cooling-supported, humidity-supported and I/O management in their computer rooms. In this case, large scale enterprises will pay less for an optimal push notification server option than those who don't have available resources. Hence, it is cost efficient for a large scale enterprise to purchase a push notification server in a long-term development.

Based on Subsection 5.4.7, AeroGear UnifiedPush Server is the priority choice and Pushd is the second choice among all the push notification servers for large scale enterprises.

7 Evaluation

In this subsection, it is necessary to figure out the best compromise between Pushwoosh and AeroGear UnifiedPush Server. In this section, Pushwoosh and AeroGear UnifiedPush Server are compared from three aspects, including six comparison points based on Subsection 5.2.7 and 5.4.7, characteristics of good APIs introduced in Subsection 4.1 and differences between push notification services and push notification servers given in Section 6.

7.1 Evaluations from six comparison points

First of all, Pushwoosh is the best choice of push notification service and AeroGear UnifiedPush Server is the priority choice of push notification server from six comparison points. Based on Subsection 5.2.4 and Subsection 5.3.4, it can be seen that Pushwoosh uses API key as the authentication model, while AeroGear UnifiedPush Server uses HTTP Basic Authentication. However, it is shown that API key is slightly more secure than HTTP Basic authentication based on comparisons between Pushwoosh and AeroGear UnifiedPush Server. Hence, Pushwoosh is a more secure option compared with AeroGear UnifiedPush Server.

Based on Subsection 5.2.6 and Subsection 5.3.6, it can be seen that Pushwoosh supports API key; Pushwoosh APNS, GCM, Microsoft, RIM Blackberry and ADM; AeroGear UnifiedPush Server only supports GCM and APNS. Considering three widely used platforms GCM, APNS and Microsoft, Pushwoosh supports all the three platforms but Pushwoosh doesn't. Hence, Pushwoosh supports better platforms than AeroGear UnifiedPush Server. In the view, Pushwoosh is better AeroGear UnifiedPush Server considering secure issues and widely platforms supported from six comparison points.

7.2 Evaluations from good APIs

Push notification services and push notification servers both expose services via APIs, APIs of all the push notification products are compared based on six characteristics of good APIs in Subsection 4.1. APIs of Pushwoosh and AeroGear UnifiedPush Server are taken as examples. Both Pushwoosh and AeroGear UnifiedPush Server provide well-documented and easily used RESTful APIs. They both enable service providers to establish their own servers to customize push notifications via their APIs. In general, both Pushwoosh and AeroGear UnifiedPush Server support six characteristics of a good API, but details of six characteristics vary from each other.

For good APIs, Pushwoosh supports backwards compatibility. Because Pushwoosh has an *onPushAccepted:withNotification:onStart:* method to inform delegates of the fact whether the application user has pressed *OK* of their received notifications. Besides, Pushwoosh depreciated a function of automatic push notifications registration for its SDKs. Pushwoosh allows application users to force old behaviors to add *Pushwoosh_AUTO keys* to *Info.plist* with a value

YES, where Pushwoosh adds -*ObjC flag* to *Linker Flags* in the projects. While, AeroGear UnifiedPush Server keeps compatibility for APIs only by AeroGear Security updates, and AeroGear UnifiedPush Server doesn't support backwards compatibility. Hence, Pushwoosh has a more appropriate RESTful APIs for current expected and potential functionalities compared with AeroGear UnifiedPush Server. In this way, the RESTful API of Pushwoosh is able to debug and implement future requirements.

Basically, Pushwoosh and AeroGear UnifiedPush Server define their semantics for RESTful APIs in a clear and simple way. But, AeroGear Unified Server introduces definitions of UnifiedPush and installation, where a variant consists of several push applications and an installation is an identity for an application user who has multiple devices. The two definitions are new for application developers who intend to use AeroGear UnifiedPush Server. Pushwoosh define its RESTful APIs in a more easily accepted semantics than AeroGear UnifiedPush Server.

From the viewpoints of a good API, Pushwoosh supports a better API than AeroGear UnifiedPush Server. Because Pushwoosh gives easily accepted semantics RESTful APIs in an appropriate way, while, AeroGear UnifiedPush Server has novel semantics definitions and doesn't support backwards compatibility. Hence, Pushwoosh supports a more suitable RESTful APIs than AeroGear UnifiedPush Server.

7.3 Evaluations from services/servers

The last point is based on differences between push notification services and push notification servers given in Section 6. Large scale enterprises can use Pushwoosh when they start to set up a notification system, and they will pay less when they use Pushwoosh than AeroGear UnifiedPush Server which is an optimal push notification server option. Hence, it is cost efficient for a large scale enterprise to purchase a push notification server in a long-term development.

To summary the preferred push notification product, Pushwoosh is better than AeroGear UnifiedPush Server for start-up, while, AeroGear UnifiedPush Server is preferred than Pushwoosh for long terms.

8 Conclusion

Nowadays, M2M devices are used to connect people according to their diverse requirements across all over the world. DCP is proposed by Ericsson to provide a time-efficient way of simply provisioning M2M devices. As the core service platform of DCP is exposed through APIs to external parties and their customers, a subscribe-notify web-services API is needed to define the selected notification service based on DCP to achieve real-time interactions.

This paper proposes Pushwoosh is the best option for push notification services and AeroGear UnifiedPush Server is the best choice for push notification servers. As discussed in Section 6, Pushwoosh and AeroGear UnifiedPush Server both support six comparison points in a systematic way, including basic information, functionality, licenses & usage fees with usage limits, security, supporting details and supported platforms/OS. In an effort to further analysis, options of Pushwoosh and AeroGear UnifiedPush Server are evaluated from two aspects of service/server and a good API respectively. Based on evaluations, Pushwoosh is the final proposal for a small or medium scale enterprise to buy, and even for a large scale enterprise to get started. Moreover, AeroGear UnifiedPush Server is the optimal proposal for a large scale enterprise to buy in a long term.

REFERENCES

Ady11 Adya, A., Cooper, G., Myers, D. and Piatek, M., "Thialfi: a Client Notification Service for Internet-scale Applications", ACM, New York, USA, 2011, pp. 129 - 142.

App13 Apple Inc., "Apple Push Notification Service", iOS Developer Library, Local and Push Notification Programming Guide, February 11th, 2014, retrieval from https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html at November 7th, 2013.

Att11 Attwood, A. , Merabti, M. and Abuelmaatti, O., "IoMANETs: Mobility architecture for wireless M2M networks", GLOBECOM Workshops (GC Wkshps), 2011 IEEE, December 2011, Houston, US, pp. 399 - 404.

Bel96 Bellotti, V. and Bly, S., "Walking Away from the Desktop Computer: Distributed Collaboration and Mobility in a Product Design Team", *Proceedings of the ACM Conference in Computer Supported Cooperative Work*, Cambridge, MA, ACM Press, 1996, pp. 209-218.

Blo12 Bloch, J., "How to Design a Good API and Why it Matters", Google Inc., August 22nd, 2012, retrieval from http://lcsd05.cs.tamu.edu/slides/keynote.pdf at May 11th, 2014.

Bra09 Bray, T., Paoli, J. and Sperberg-McQueen C.-M., "XML Extensible Markup Language 1.0", W3C Recommendation, http://www.w3.org/TR/1998/REC-xml-19980210.html at 25th November 2013.

Bur13 Burgstahler, D., Lampe, U., Richerzhagen, N. And Steinmetz, R., "Push vs. Pull: An Energy Perspective", The 6th IEEE International Conference on Service-Oriented Computing and Applications (SOCA), IEEE, Koloa, USA, December 2013.

Cab01 Cabrera, L.-F., Jones M.-B. and Theimer, M., "Herald: Achieving a Global Event Notification Service", *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, Elmau, Germany. IEEE Computer Society, May 2001.

Cug01 Cugola, G., Di Nitto, E. and Fuggetta, A., "The JEDI Event Based Infrastructure and Its Application to the Development of the OPSS WFMS", IEEE Transactions on Software Engineering, 27(9), September 2001, pp. 827 - 850.

Car01 Carzaniga, A., Rosenblum, D. and Wolf, A., " Design and Evaluation of a Wide Area Notification Service", ACM, Transactions on Computer Systems, 19(3), 2001, pp. 332 - 383.

Des02b De Souza, C.-R.-B., Basaveswara, S.-D. and Redmiles, D.-F., "Lessons Learned Using with Notification Servers To Support Application Awareness", Human Computer Interaction Consortium Workshop 2002, January, 2002.

Des02a De Souza ,C.-R.-B., Basaveswara, S.-D., Redmiles, D.-F., "Supporting Global Software Development with Event Notification Servers", International Workshop on Global Software Development ICSE 2002, Orlando, Florida, USA, May 21st, 2002.

Ecm13 Ecma International, "ECMA-404 The JSON Data Interchange Standard", October 2013, retrieved from http://json.org/ at November 25th, 2013.

Eri11 Ericsson, "More than 50 Billion Connected Devices - Taking Connected Devices to Mass Market and Profitability", White Papers, February 2011, 284 23-3149 Uen, retrieval from http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf at November 7th, 2013.

Eug03 Eugster, P.-Th, Felber, P.-A., Guerraoui, R. and Kermarrec, A., "The Many Faces of Publish/Subscribe", ACM, Computing Surveys, vol. 35, issue 2, New York, USA, June 2003, pp. 114 - 131.

Fie00 Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", PhD thesis, University of California, Irvine, USA, 2000.

Fie99 Fielding, R., Getty, J., Mogul, J., Frystyk, H., Masinter, L., Leach P. And Berners-Lee, T., "Hypertext Transfer Protocol - HTTP/1.1", RFC 2616, June 1999, http://www.w3.org/Protocols/rfc2616/rfc2616.html at November 25th, 2013.

Goo13 Google Inc. "Android Cloud to Device Messaging Framework", Google Developers, retrieved from https://developers.google.com/android/c2dm/?csw=1 at November 25th, 2013.

Goo14 Google Inc., "Google Cloud Messaging", April 18th, 2014, retrieved from http://developer.android.com/reference/com/google/android/gms/gcm/GoogleCloudMessaging.html at November 25th, 2013.

Guo13 Guo, W. and Liu, H., " The Analysis of Push Technology Based on iphone Operating System", Measurement, Information and Control (ICMIC), 2013 International Conference, vol. 1, Harbin, China, August 2013.

Gud07 Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., Nielsen, H.-F., Karmarkar, A. and Lafon, Y., "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)", W3C Recommendation 27th April 2007, retrieved from http://www.w3.org/TR/soap12-part1/ at November 25th, 2013.

Hua06 Huang, Y. and Gannon, D., “A Comparative Study of Web Services-based Event Notification Specifications”, Parallel Processing Workshops, International Conference on Columbus, OH, 2006, pp. 8 - 14.

Mar99 Martin-Flatin, J.P., “Push vs. Pull in Web-Based Network Management”, *Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management*, 1999, Distributed Management for the Networked Millennium, Boston, US, May 1999, pp. 3 - 18.

Mer11 Merabti, M., Kennedy, M. and Hurst, W., "Critical Infrastructure Protection: A 21st Century Challenge", International Conference on Communications and Information Technology (ICCIT), March 2011, Aqaba, Aqaba Governorate, pp. 1 - 6.

Moj12 Mojzisova, A. and Mojzis, M., “Unified Platform for the Delivery of Notifications to Smartphones Notificatio”n, 13th International Carpathian Control Conference (ICCC), Slovak Republic, May 2012.

Mul09 Mulligan, G. and Gracanin, D. “A Comparison of SOAP and REST Implementations of a Service Based Interaction Independence Middleware Framework”, *Proceedings of the 2009 Winter on Simulation Conference (WSC)*, Austin, TX, December 2009, pp. 1423 - 1432.

Ori11 O'Riordan, A., O'Mahoney, O., "Engineering an Open Web Syndication Interchange with Discovery and Recommender Capabilities", *Journal of Digital Information 12 (1)*, 2011.

Pus13 Microsoft Inc., “Push notifications for Windows Phone”, Windows Phone Dev Center, retrieved from http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff402558(v=vs.105).aspx at November 25th, 2013.

Res00 Rescorla, E., “HTTP Over TLS”, RFC 2616, Internet Engineering Task Force (IETF), May 2000, https://tools.ietf.org/html/rfc2818 at November 25th, 2013.

Ric07 Richardson, L. and Ruby, S., “RESTful Web Services”, O'Reilly Media, May 2007, ISBN 10:0-596-52926-0.

Rop13 Ropert, S., Bonneau, V., Ramahandry, T., “Internet of Things - Outlook for the Top 8 Vertical Markets”, Market Report, IDATE, M13112MR, August 26th, 2013, pp. 123, ISBN 978-2-84822-340-7, retrieved from http://www.idate.org/en/Research-store/Internet-of-Things_785.html at December 19th, 2013.

Shi06 Shi, X., “Sharing Service Semantics using SOAP-Based and REST Web Services”, IEEE Educational Activities Department Piscataway, vol. 8, issue 2, NJ, USA, March 2006, pp. 18-24.

Shi09 Shi, S. and Zhang, R., “SXML, an Enhancement of XML Documents in

Mobile Learning", International Conference on Computational Intelligence and Software Engineering (CiSE), December 2009, pp. 11 - 13.

Sim10 Simpson, K., "Defining Safer JSON-P", 2010, retrieved from http://json-p.org/ at November 25th, 2013.

Smi02 Smith, M. and Hunt, R., "Network Security using NAT and NAPT", The 10th IEEE International Conference (ICON) on Networks, 2002, pp. 355 - 360.

Urb13 UrbanAirship, "The Pocket Guide to Good Push, Push is a Permission-based Mobile Customer Communication Channel", retrieved from http://www.urbanairship.com/talesofgoodpush at 25th December 2013.

Zha09 Zhang, P., Durresi, A. and Barolli, L., "A Survey of Internet Mobility", International Conference on Network-Based Information Systems (NBIS), August 2009, Indianapolis, IN, pp. 147 - 154.

Appendix 1

Mozilla SimplePush	
Server/Service(System)	A push notification service to web applications
Protocols	REST
Notification Transport Protocol	HTTP, HTTPS, XMPP
Data Formats	JSON
API home	Web APIs https://wiki.mozilla.org/WebAPI
Functionality	
Client Install Required	YES
Service Endpoint	Android, desktop, open OS for mobile platforms
WSDL	\
Batch processing	NO, Mozilla SimplePush doesn't support batch processing, because SimplePush keeps notifications separate and individual in most use scenarios in the specifications despite of the recommendation that the UserAgent should try to batch all pending acknowledgements into fewer notifications.
Scalability	YES, Based on the implementation of the SimplePush protocol, SimplePush Server supports scalability with the help of the Client side engineers. In details, SimplePush Server doesn't store user data, nor limited utility for the hacking or grieving vector. Meanwhile, Mozilla SimplePush deals the client connection between the web socket and the proxy server by identifying a GUID4 UserAgentID which is unique to each device. The client sends a request of a new Entry point for a third party server which is combined with the GUID4 channel id. The connection between the client and the server will be established to receive notifications after the server accepts the connection and responds with any GUID4 channels. Thus, Mozilla SimplePush realizes the Internet-scale scalability across various

	networks.
Mobility	YES, Mozilla SimplePush notification service uses Mozilla cloud services to support mobility. The virtual cloud services are used in management and seamless application users' experience. Mozilla SimplePush can reach their online application users through Mozilla presence after the applications register in Mozilla Cloud Services.
Fault tolerance	YES, If network failures or database failures or anything else makes PushServer in an inconsistent situation, Mozilla SimplePush may drop all states, and disconnect all active applications to force to reconnect and begin a handshake to get synchronized. In such a way, Mozilla SimplePush realizes the synchronization with garbage collections between the server and client of fault tolerance to achieve reliable notification delivery.
Backward compatibility	YES
Sign up and Licensing	
Sign up Requirements	YES
Developer Key Required	YES
Account Required	YES
Commercial Licensing	Yes with service
Provider	https://wiki.mozilla.org/
Managed By	Mozilla
Non-Commercial Lic.	
Data Licensing	Mozilla Public License Version 2.0
License for server(service)/client	Service
Usage Fees	Free. It's the Web, based on Open standards, where everyone can participate. No need to ask permission nor pay a fee to use a proprietary technology.

Program Fees	
Certification Program	
Usage Limits	NO
Terms of Service	
Security	
Authentication Model	API key
SSL Support	HTTPS
Read-only Without Login	NO
Support	
Vendor API Kits/Support the third party API	YES
Community API Kits	
API Blog	
API Forum	http://www.mozilla.org/about/forums/
Site Blog	https://wiki.mozilla.org/Talk:WebAPI/SimplePush/Protocol
Developer Support	
Console URL	
Support E-mails	NO
Support SMS Text Messages	YES
Support Pushing Cloud Notifications	YES
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	YES
C2DM	YES
Apple Push Notification Systems (APNS	YES
Microsoft (Windows Phone 7, WP 8, Windows 8)	YES
Firefox OS	YES
RIM Blackberry	YES

Amazon Device Messaging (ADM)	NO

Appendix 2

Pushover	
Server/Service(System)	A mobile push notification service
Protocols	REST
Notification Transport Protocol	HTTP, HTTPS
Data Formats	JSON, XML
API home	https://pushover.net/api
Functionality	
Client Install Required	YES
Service Endpoint	Android, iOS devices and desktop https://api.pushover.net/1/
WSDL	/
Batch processing	Delivery Groups of Pushover makes it possible to manage lists of application users and push notifications to them in a single API request. With Pushover, batch jobs can be stored when power off and execute after recharged automatically. Even the plugin Zapier enables the integration of Batchbook to support management.
Scalability	Pushover uses cloud service to scale by adding servers and to support scalability.
Mobility	When a mobile device has an active connection, a notification will be pushed to the application based on the subscription to push services agreed between the application user and Pushover.
Fault tolerance	Pushover supports reliability in each Pushover application users releases. From Device Client Changelog, the features which support reliability and compatibility are added, such as Pushover can fix crashes after synchronizing notifications of Android 2.0.1 release, and Pushover can fix crashes if saving alerts of a specific length of iOS 2.0.2 release.
Backward compatibility	YES
Sign up and Licensing	

Sign up Requirements	YES
Developer Key Required	YES
Account Required	YES
Commercial Licensing	YES
Provider	superblock.net/ Red Rat
Managed By	Superblock, LLC
Non-Commercial Lic.	NO
Data Licensing	OpenBSD
License for server(service)/client	Service
Usage Fees	Pushover has both free and premium accounts. Pushover has no monthly fees for iOS devices within the free accounts. But Pushover allows only five notifications for android devices per day for free, and charges $4.99 for android devices to get unlimited notifications.
Program Fees	
Certification Program	
Usage Limits	Pushover limits the usage of free accounts as follows, notifications are currently restricted within 512 characters, including titles, and applications are currently restricted to push 7500 notifications per month. If intends to use Pushover on both iOS and Android devices, application users must pay $4.99 for it twice, once from the App Store and once from Google Play. Besides, when the publishers want to push more than 7,500 notifications per month or need additional capacity, they should purchase from Purchase Additional Message Capacity of Pushover with 10000 additional notifications cost for $50.00. Moreover, high traffic public applications like IFTTT allows unlimited notifications.
Terms of Service	https://pushover.net/terms

Security	
Authentication Model	API key, By setting up a Pushover account, you will be provided with a user key. You may also set up one or more applications, each of which receives its own application key. All Pushover messages require, at a minimum, a user key, an application key and a message. You may send additional information such as a custom title, sound effect and priority as well. By default, Pushover will display the application name as the title (based on the application key specified), but you can override this by passing a custom title.
SSL Support	YES, HTTP library's SSL verification
Read-only Without Login	YES
Support	
Vendor API Kits/Support the third party API	Perl, PHP, Ruby support the third party API
Community API Kits	
API Blog	
API Forum	
Site Blog	http://www.reddit.com/r/pushover/
Developer Support	
Console URL	
Support E-mails	E-mail gateway allows clients to send e-mails to *Your-Pushover-User-Key@api.pushover.net* then convert them into Pushover notifications
Support SMS Text Messages	YES, Instant Messages (IM)
Support Pushing Cloud Notifications	YES
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	Android devices
C2DM	NO
Apple Push Notification Systems (APNS	iPhone, iPod Touch, iPad

Microsoft (Windows Phone 7, WP 8, Windows 8)	NO
Firefox OS	NO
RIM Blackberry	NO
Amazon Device Messaging (ADM)	NO

Appendix 3

Pushwoosh	
Server/Service(System)	A push notification service
Protocols	REST
Notification Transport Protocol	HTTP, HTTPS, SMTP for e-mails
Data Formats	JSON
API home	Latest JSON API version 1.3 http://www.pushwoosh.com/programming-push-notification/pushwoosh-push-notification-remote-api/
Functionality	
Client Install Required	YES
Service Endpoint	iOS, Android, Blackberry, Windows Phone, Mac OS X, Windows 8, Amazon 8, Safari, others with Wordpress plugin
WSDL	\
Batch processing	Pushwoosh enables service providers to segment their audience based on their marketing strategies, then they can provide relevant facilities and contents to target push notifications. Service providers also schedule push notifications in batches to support group multicast with reports.
Scalability	Pushwoosh is highly scalable with the help of cloud service and is able to send millions of notifications very fast but the traffic is prioritized for Premium accounts.
Mobility	Pushwoosh pushes notifications based on real time location. The application developers and service providers are able to reach application users located in a particular location immediately, and application users can precisely select targeted zones by setting the distance in meters in longitude and latitude of geo zones.
Fault tolerance	Advanced Statistics feature allows publishers to keep track of their push notifications. Pushwoosh has report mechanism to track campaigns and devise corrections in case of potential faults or errors.

Backward compatibility	Push notifications plugin for WordPress helps to send push notifications when publish or update a post or a page.
Sign up and Licensing	
Sign up Requirements	YES
Developer Key Required	YES
Account Required	YES
Commercial Licensing	Yes with service
Provider	www.pushwoosh.com
Managed By	Pushwoosh
Non-Commercial Lic.	
Data Licensing	MIT
License for server(service)/client	Service
Usage Fees	Fees depend on functions. http://www.pushwoosh.com/pricing/
Program Fees	
Certification Program	
Usage Limits	Unlimited push notifications
Terms of Service	http://www.pushwoosh.com/terms-of-use/
Security	
Authentication Model	API key
SSL Support	YES, HTTP library's SSL verification
Read-only Without Login	NO
Support	
Vendor API Kits/Support the third party API	Any reference to any third party on the Website is provided to you for informational purposes only.
Community API Kits	

API Blog	
API Forum	https://community.pushwoosh.com/
Site Blog	http://www.pushwoosh.com/blog/
Developer Support	
Console URL	http://www.pushwoosh.com/another-billion-pushes-wordpress-plugin-update-and-a-new-pushwoosh-control-panel-sneak-peek/
Support E-mails	YES
Support SMS Text Messages	YES
Support Pushing Cloud Notifications	YES, Pushwoosh uses cloud service to synchronize data in a secure way.
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	Android devices such as Samsung Galaxy
C2DM	NO
Apple Push Notification Systems (APNS	iPhone, iPad, Mac OS X Push Notification SDK, Safari
Microsoft (Windows Phone 7, WP 8, Windows 8)	Windows 8 SDK Integration, PhoneGap/Cordova SDK for Windows Phone 8
Firefox OS	NO
RIM Blackberry	Native Blackberry SDK
Amazon Device Messaging (ADM)	Native Amazon SDK

Appendix 4

Amazon SNS	
Server/Service(System)	A push notification service for web/application
Protocols	SOAP
Notification Transport Protocol	Notifications over HTTP, HTTPS, Email-JSON and SQS transport protocols consists of a simple JSON object.
Data Formats	XML
API home	http://aws.amazon.com/sns/
Functionality	
Client Install Required	YES
Service Endpoint	iPhone, iPad, Android, Kindle Fire, and internet connected smart devices, as well as pushing to other distributed services.
WSDL	http://sns.us-east-1.amazonaws.com/doc/2010-03-31/SimpleNotificationService.wsdl
Batch processing	Amazon SNS uses batch process to securely access remote resources. Amazon SNS supports an unique payload sent to different platforms, namely, Amazon SNS can publish a single message for all platforms.
Scalability	The cloud delivers mobile push notifications to targeted applications via APN, GCM etc.
Mobility	Amazon SNS can send mobile push notifications when the applications are not online.
Fault tolerance	All messages published to Amazon SNS are stored redundantly avoiding lost. e.g. For notification message by HTTP, an SNS Delivery Policy can be used to control the retry pattern (linear, geometric, exponential backoff), maximum and minimum retry delays, and other parameters. Application developers should have notifications delivered to an SQS queue with other notifications. Amazon SNS also supports feedbacks and token management for reliable token delivery to each platform.
Backward compatibility	AWS compliance

Sign up and Licensing	
Sign up Requirements	AWS account, phone number verification
Developer Key Required	YES
Account Required	YES
Commercial Licensing	Yes with service
Provider	www.amazon.com
Managed By	Amazon
Non-Commercial Lic.	No
Data Licensing	GNU General Public License (GPL version 2)
License for server(service)/client	Client
Usage Fees	Amazon SNS allows 1 million notifications within free account, and $1 for per million notifications over the free account. http://aws.amazon.com/sns/pricing
Program Fees	Free
Certification Program	
Usage Limits	Amazon SQS should have enough connections or threads to support the number of concurrent message producers and consumers which will be sending requests and receiving responses. For instance, instances of the AWS SDK for Java AmazonSQSClient class maintain at most 50 connections to Amazon SQS by default. http://aws.amazon.com/agreement/
Terms of Service	aws.amazon.com/terms/
Security	
Authentication Model	AWS API key
SSL Support	Both publishers and subscribers can use SSL to help secure the channel to send and receive messages. Publishers can connect to Amazon SNS over HTTPS and publish messages over the SSL channel. Subscribers should register an SSL-enabled end-point as part of the

	subscription registration, and notifications will be delivered over a SSL channel to that end-point.
Read-only Without Login	NO
Support	
Vendor API Kits/Support the third party API	Amazon SNS provides a set of simple APIs to enable event notifications for topic owners, subscribers and publishers. http://aws.amazon.com/code/Amazon%20SNS?_encoding=UTF8&jiveRedirect=1
Community API Kits	
API Blog	
API Forum	https://forums.aws.amazon.com/forum.jspa?forumID=72#
Site Blog	aws.typepad.com/
Developer Support	aws.amazon.com/documentation/sns/
Console URL	http://aws.amazon.com/developertools/Amazon%20SNS?_encoding=UTF8&jiveRedirect=1
Support E-mails	E-mails with Email-JSON transport protocol are sent to registered addresses as email. Email-JSON sends notifications as a JSON object, while Email sends text-based emails.
Support SMS Text Messages	YES
Support Pushing Cloud Notifications	It enables applications to send notifications from the cloud.
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	Android devices such as Samsung Galaxy
C2DM	NO
Apple Push Notification Systems (APNS)	iPhone, iPad
Microsoft (Windows Phone 7, WP 8, Windows 8)	NO
Firefox OS	NO
RIM Blackberry	NO
Amazon Device Messaging (ADM)	Kindle Fire

Appendix 5

OpenPush	
Server/Service(System)	A push notification service
Protocols	REST
Notification Transport Protocol	HTTP, HTTPS, SMTP, XMPP
Data Formats	JSON
API home	
Functionality	
Client Install Required	YES
Service Endpoint	iOS, Android, Blackberry, Windows Phone, desktop
WSDL	\
Batch processing	OpenPush enables service providers to segment their audience, then they can provide relevant real time notifications to target audience. Service providers also schedule push notifications in batches to support group multicast with reports.
Scalability	OpenPush is highly scalable with the help of cloud service. OpenPush uses ejabberd and XMPP servers to support scalability. OpenPush also uses MongoDB on ejabberd for management.
Mobility	OpenPush supports mobility, because OpenPush customizes and replace ejabberd behavior with hooks which can be located on another node. OpenPush pushes notifications based on real time location.
Fault tolerance	Advanced Statistics feature allows publishers to keep track of their push notifications.
Backward compatibility	
Sign up and Licensing	
Sign up Requirements	YES
Developer Key Required	YES

Account Required	YES
Commercial Licensing	Yes with service
Provider	www.openpush.im/
Managed By	ProcessOne
Non-Commercial Lic.	
Data Licensing	MIT
License for server(service)/client	Service
Usage Fees	Free
Program Fees	
Certification Program	
Usage Limits	NO, unlimited push notifications
Terms of Service	
Security	
Authentication Model	API key
SSL Support	YES, HTTP library's SSL verification
Read-only Without Login	NO
Support	
Vendor API Kits/Support the third party API	Any reference to any third party on the Website is provided to you for informational purposes only.
Community API Kits	
API Blog	
API Forum	
Site Blog	
Developer Support	http://openpush.im/#beta
Console URL	
Support E-mails	YES

Support SMS Text Messages	YES
Support Pushing Cloud Notifications	YES, OpenPush is offered as a service via cloud.
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	YES
C2DM	NO
Apple Push Notification Systems (APNS	YES
Microsoft (Windows Phone 7, WP 8, Windows 8)	YES
Firefox OS	NO
RIM Blackberry	YES
Amazon Device Messaging (ADM)	NO

Appendix 6

Parse.com	
Server/Service(System)	A third party push notification service
Protocols	REST
Notification Transport Protocol	HTTP, HTTPS
Data Formats	JSON
API home	Latest JSON API version 1.3 http://parse.com/docs/api_libraries
Functionality	
Client Install Required	YES
Service Endpoint	iOS, Android, Blackberry, Windows Phone 8, Mac OS X
WSDL	\
Batch processing	Parse.com supports batch operations via Parse RESTful APIs to reduce the time spent on network round trips, because Parse.com can create, delete and update 50 requests with a batch endpoint.
Scalability	Parse.com supports scalability with the help of cloud cloud and Parse.com operates a number of high-throughput, I/O intensive MongoDB clusters and needed to improve scalability and speed.
Mobility	Parse.com supports mobile cloud service, because Parse.com has enterprise mobility suites and mobile backend as a service to support geo queries based on application user's real time location.
Fault tolerance	Parse.com has reliable backends and handles user account management, data storage and disk caching for its application users and usage can fluctuate on a daily basis. Parse.com SDKs work properly with custom implementation. Besides, Parse.com can work when applications are offline, and application developers is able to get error responses which can be handled later.
Backward compatibility	Push notifications plugin for WordPress helps to send push notifications when publish or update a post or a page.

Sign up and Licensing	
Sign up Requirements	YES
Developer Key Required	YES
Account Required	YES
Commercial Licensing	Yes with service
Provider	www.parse.com
Managed By	Parse
Non-Commercial Lic.	
Data Licensing	BSD
License for server(service)/client	Service
Usage Fees	Fees depend on functions. http://www.parse.com/plans/
Program Fees	
Certification Program	
Usage Limits	Parse.com has maximum notifications limit for burst limit according to https://parse.com/plans
Terms of Service	http://www.parse.com/about/terms
Security	
Authentication Model	API key
SSL Support	YES, HTTP library's SSL verification
Read-only Without Login	NO
Support	
Vendor API Kits/Support the third party API	Cloud Modules are the easiest way to integrate your Parse app with third-party services and libraries
Community API Kits	
API Blog	
API Forum	

Site Blog	http://blog.parse.com/
Developer Support	
Console URL	
Support E-mails	YES
Support SMS Text Messages	YES
Support Pushing Cloud Notifications	YES, Parse.com uses cloud code to support scalability. Parse.com provides cloud-based backend services for mobile application developers.
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	YES
C2DM	NO
Apple Push Notification Systems (APNS	iPhone, iPad, Mac OS X
Microsoft (Windows Phone 7, WP 8, Windows 8)	Windows 8
Firefox OS	NO
RIM Blackberry	NO
Amazon Device Messaging (ADM)	NO

Appendix 7

Urban Airship	
Server/Service(System)	A mobile push notification service
Protocols	HTTP POST, REST
Notification Transport Protocol	HTTP, HTTPS
Data Formats	JSON, JSONP
API home	Latest UrbanAirship Library 2.1.0 http://docs.urbanairship.com/
Functionality	
Client Install Required	NO
Service Endpoint	iOS, Android, Blackberry, Windows Phone 8
WSDL	\
Batch processing	Urban Airship has batch push which supports push multiple notifications in one call, and all options of batch push are available in the normal push API. The batch push API payload is a list of JSON objects, where each object is a valid push API request. Beside, Urban Airship also supports multicast.
Scalability	Urban Airship allows to scale by adding servers with the help of cloud service.
Mobility	Urban Airship has mobile relationship management solutions for better contextual aware of mobility.
Fault tolerance	Urban Airship has near real time report mechanisms, notifications are sent and received in diversified time slices with time-in-application, uniques and conversions analyses to guarantee accurate and reliable delivery.
Backward compatibility	Urban Airship provided the first end-to-end cloud push notification service in the market, via a service we called Helium for C2DM.
Sign up and Licensing	
Sign up Requirements	YES
Developer Key Required	YES

Account Required	YES
Commercial Licensing	
Provider	www.urbanairship.com
Managed By	Urban Airship
Non-Commercial Lic.	
Data Licensing	Apache license 2.0
License for server(service)/client	Client
Usage Fees	Fees depend on how to use it. https://support.urbanairship.com/customer/portal/articles/1061364-pricing-billing-and-invoice-questions
Program Fees	
Certification Program	
Usage Limits	User applications can get access to the composers or core reports including pushes sent, application opens and time in application, but applications for profits only have access to the reports or composers with a 45 day trial.
Terms of Service	http://urbanairship.com/images/uploads/documents/URBAN_AIRSHIP_TERM_OF_SUBSCPRITION_SERVICE_17_JAN_2014.pdf
Security	
Authentication Model	API key (Urban Airship API uses a key and secret combination to authenticate for Urban Airship app setup)
SSL Support	YES
Read-only Without Login	NO
Support	
Vendor API Kits/Support the third party API	A third party providing a web based or offline software application or platform can't be integrated by API. No third party beneficiaries under its agreement.
Community API Kits	

API Blog	
API Forum	
Site Blog	http://urbanairship.com/blog
Developer Support	http://urbanairship.com/images/uploads/documents/URBAN_AIRSHIP_SUPPORT_SERVICES_DESCRIPTION_17_JAN_2014.pdf
Console URL	
Support E-mails	YES
Support SMS Text Messages	YES
Support Pushing Cloud Notifications	YES, Urban Airship allows scaling by adding servers with the help of cloud service.
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	Android devices
C2DM	Depreciated
Apple Push Notification Systems (APNS	iPhone, iPad
Microsoft (Windows Phone 7, WP 8, Windows 8)	WP 8, PhoneGap/Cordova SDK for WP 8
Firefox OS	NO
RIM Blackberry	YES
Amazon Device Messaging (ADM)	NO

Appendix 8

Xtify	
Server/Service(System)	A push notifications service
Protocols	REST
Notification Transport Protocol	HTTP, HTTPS, XMPP, Xtify opens an XMPP connection between the server and device with an SDK to extracting locatin from the device.
Data Formats	XML, RSS
API home	http://developer.xtify.com/dashboard.action
Functionality	
Client Install Required	YES
Service Endpoint	iOS, Android, Blackberry
WSDL	\
Batch processing	Xtify can send to a group of devices in a batch by segmenting with tags.
Scalability	Xtify allows services to get data centers from SoftLayer for scalability.
Mobility	Xtify enables application developers to deliver push and location-triggered messages directly to users devices using SDK and messaging engine even when the application is closed. Xtify uses the geo-indexes property of MongoDB to realize location data store. Because when location enabled devices send updates back to Xtify, Xtify turns to MongoDB's location datastore to track application users movement from time and space. Xtify also allows service providers to schedule application users as based on time zones to create geo-fences with a CRM-defined segmentation for better precision.
Fault tolerance	Xtify uses the geo-indexes property of MongoDB to realize location data store, because MongoDB's location datastore can track application users movement from time and space. Thus, Xtify has message-level tracks and analytics, which can be integrated with fulfillment management systems and CRM via

	API.
Backward compatibility	YES
Sign up and Licensing	
Sign up Requirements	YES
Developer Key Required	YES
Account Required	YES
Commercial Licensing	YES
Provider	developer.xtify.com
Managed By	IBM
Non-Commercial Lic.	YES
Data Licensing	BSD
License for server(service)/client	Service
Usage Fees	Varies based on number of application users and notifications. http://www.xtify.com/pricing.html#tab=tab-A
Program Fees	
Certification Program	
Usage Limits	The limits depend on different packages. http://www.xtify.com/pricing.html#tab=tab-A
Terms of Service	http://developer.xtify.com/terms-of-use/
Security	
Authentication Model	API key
SSL Support	YES, HTTP library's SSL verification
Read-only Without Login	NO
Support	
Vendor API Kits/Support the third party API	YES
Community API Kits	
API Blog	http://developer.xtify.com/display/APIs/Push+A

	PI+2.0
API Forum	
Site Blog	http://blog.xtify.com/
Developer Support	*support@xtify.com*
Console URL	https://console.xtify.com/login
Support E-mails	YES
Support SMS Text Messages	YES
Support Pushing Cloud Notifications	Xtify supports cloud operations to SoftLayer.
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	Android devices
C2DM	NO
Apple Push Notification Systems (APNS	iPhone, iPad
Microsoft (Windows Phone 7, WP 8, Windows 8)	Windows Phone 8
Firefox OS	NO
RIM Blackberry	YES
Amazon Device Messaging (ADM)	NO

Appendix 9

Push IO	
Server/Service(System)	A cloud based push notification alert or service for mobile and mobile data delivery
Protocols	REST
Notification Transport Protocol	HTTP, HTTPS
Data Formats	JSON
API home	https://docs.push.io/API_&_cURL_Information/Overview
Functionality	
Client Install Required	YES
Service Endpoint	iOS, Android, Blackberry, Windows Phone, Mac OS X, Windows 8,
WSDL	/
Batch processing	Push IO supports auto push in dashboard which automates both building and delivering notifications. Auto push includes batch processes and broadcasts, so application groups can be used to send one push notification to several applications within one group with one click.
Scalability	Push IO can add applications easily to use application management, because Push IO can add applications to the unlimited Sandbox environment to test all characteristics for the whole system.
Mobility	Push IO allows application developers to use smarted tailored feature to segment audience and trigger push notifications to application users, so the notifications of Push IO can reach globally.
Fault tolerance	Push IO has Sandbox environment to test all characteristics for the whole system, the sandbox gives fault detections.
Backward compatibility	YES
Sign up and Licensing	
Sign up Requirements	YES

Developer Key Required	YES
Account Required	YES
Commercial Licensing	Yes with service
Provider	push.io/
Managed By	Oracle
Non-Commercial Lic.	YES
Data Licensing	BSD
License for server(service)/client	Service
Usage Fees	A free 30-day trial to the service preferred by professionals
Program Fees	
Certification Program	
Usage Limits	Push IO gives a free 30 days trial for push notification service in the Push IO's homepage.
Terms of Service	http://push.io/legal/tos/
Security	
Authentication Model	API key
SSL Support	YES, HTTP library's SSL verification
Read-only Without Login	YES
Support	
Vendor API Kits/Support the third party API	API Key
Community API Kits	
API Blog	
API Forum	
Site Blog	http://push.io/blog/
Developer Support	
Console URL	

Support E-mails	YES
Support SMS Text Messages	YES
Support Pushing Cloud Notifications	YES
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	YES
C2DM	NO
Apple Push Notification Systems (APNS)	iPhone, iPad, Mac OS X Push Notification SDK, Safari
Microsoft (WP 7, WP 8, Windows 8)	Windows Phone, Windows 8
Firefox OS	NO
RIM Blackberry	NO
Amazon Device Messaging (ADM)	NO

Appendix 10

Awarly	
Server/Service(System)	A push notification service for application
Protocols	REST
Notification Transport Protocol	Notifications over HTTP, HTTPS
Data Formats	JSON, all data is sent and received as JSON.
API home	https://awarly.com/docs/rest-api
Functionality	
Client Install Required	YES
Service Endpoint	iPhone, iPad, Android,Blackberry
WSDL	/
Batch processing	Awarly supports batch processing, which means Awarly can push a unique notification to a variety of devices, although it is inconvenient to send multiple requests. So application developers could group several notifications in one call, then send the POST body as a JSON array.
Scalability	Awarly offers push notification management and mobile SDKs for quickly deployment, which supports scalability.
Mobility	Awarly has advanced geo-notification features to send automatically triggered push notifications right at the moment to application users when they enter the specified area on the map.
Fault tolerance	Awarly has error messages which use conventional HTTP response codes to indicate success or failure of an API request. The all errors will return error message describing the particular problem in JSON.
Backward compatibility	
Sign up and Licensing	
Sign up Requirements	To authenticate users' applications with Awarly's API must use OAuth. Awarly supports App Login using the OAuth 2.0 Client

	Credential flow.
Developer Key Required	YES
Account Required	YES
Commercial Licensing	Yes with service
Provider	https://awarly.com/
Managed By	
Non-Commercial Lic.	No
Data Licensing	BSD
License for server(service)/client	Service
Usage Fees	https://awarly.com/pricing
Program Fees	
Certification Program	
Usage Limits	Awarly enables starters to register unlimited applications, and also allows to deliver 1 million push notifications as e-mails sent to 10000 devices per month with the free account in one month.
Terms of Service	https://awarly.com/terms
Security	
Authentication Model	API key
SSL Support	YES
Read-only Without Login	NO
Support	
Vendor API Kits/Support the third party API	The underlying REST API is simple, but there are also lots of pre-built libraries for interacting with Awarly.
Community API Kits	
API Blog	
API Forum	

Site Blog	http://blog.awarly.com/
Developer Support	
Console URL	Push Composer https://awarly.com/products/notifications/
Support E-mails	YES
Support SMS Text Messages	YES
Support Pushing Cloud Notifications	Awarly uses cloud service to send notifications.
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	Android devices such as Samsung Galaxy
C2DM	NO
Apple Push Notification Systems (APNS)	iPhone, iPad
Microsoft (Windows Phone 7, WP 8, Windows 8)	NO
Firefox OS	NO
RIM Blackberry	YES
Amazon Device Messaging (ADM)	NO

Appendix 11

OpenMarket	
Server/Service(System)	A push notification alerting service for mobile device
Protocols	REST
Notification Transport Protocol	HTTP, HTTPS
Data Formats	XML
API home	http://www.openmarket.com/openmarket-push-notifications/
Functionality	
Client Install Required	No
Service Endpoint	Apple, Android, BlackBerry
WSDL	/
Batch processing	OpenMarket can broadcast important news or emergency information to application users, and the batch processing is implemented with smart segments feature of OpenMarket.
Scalability	OpenMarket uses a unified API to satisfy the growing needs of pushing notification to various platforms, and the unified API is easy to realize global scalability. Beside, OpenMarket has its self-service tools to set up application development, create audience segments, compose and send push notifications via the unified API.
Mobility	OpenMarket has its self-service tools to set up application development, create audience segments, compose and send push notifications via the unified API.
Fault tolerance	OpenMarket uses geo-redundant network and reporting & message analysis to retrieve status and error information about user engagement and message deletion, thus OpenMarket can learn how users are responding to your messages.
Backward compatibility	NO

Sign up and Licensing	
Sign up Requirements	Yes, need registered account
Developer Key Required	No
Account Required	Yes
Commercial Licensing	
Provider	www.openmarket.com/
Managed By	OpenMarket
Non-Commercial Lic.	YES
Data Licensing	GNU GPL
License for server(service)/client	Service
Usage Fees	Application developers need to pay when they use OpenMarket to send push notifications, but the fees supported by sales department are not transparent. http://www.openmarket.com/payments/mobile-payments-overview/
Program Fees	
Certification Program	
Usage Limits	
Terms of Service	http://www.openmarket.com/terms-of-service/
Security	
Authentication Model	HTTP Basic Authentication
SSL Support	YES
Read-only Without Login	No
Support	
Vendor API Kits/Support the third party API	YES
Community API Kits	
API Blog	

API Forum	
Site Blog	
Developer Support	http://www.openmarket.com/wp-content/uploads/2011/12/PushNotifications_IntegrationGuide.pdf
Console URL	
Support E-mails	NO
Support SMS Text Messages	YES
Support Pushing Cloud Notifications	OpenMarket supports to push notifications via APIs or through a customizable cloud-based mobile engagement platform.
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	YES
C2DM	YES
Apple Push Notification Systems (APNS	YES
Microsoft (Windows Phone 7, WP 8, Windows 8)	NO
Firefox OS	NO
RIM Blackberry	YES
Amazon Device Messaging (ADM)	NO

Appendix 12

PushBots	
Server/Service(System)	A push notification service for mobile device
Protocols	REST
Notification Transport Protocol	HTTP, HTTPS
Data Formats	JSON
API home	https://pushbots.com/developer/rest
Functionality	
Client Install Required	YES
Service Endpoint	Apple, Android
WSDL	/
Batch processing	PushBots allows to register multiple devices up to 500 devices within per batch request, then the array of devices tokens can be added to database.
Scalability	PushBots support scalability via flexible APIs by cloud service which support custom integration.
Mobility	PushBots has geo targeting feature to send automatically triggered push notifications right at the moment to application users when they enter the specified area on the map.
Fault tolerance	PushBots provides reliable delivery via flexible APIs and real-time analytics for redundant statistics management, which also makes custom integration.
Backward compatibility	
Sign up and Licensing	
Sign up Requirements	YES
Developer Key Required	YES
Account Required	YES
Commercial Licensing	
Provider	https://pushbots.com/

Managed By	PushBots, Inc.
Non-Commercial Lic.	
Data Licensing	Apache 2.0
License for server(service)/client	Service
Usage Fees	https://pushbots.com/pricing
Program Fees	
Certification Program	
Usage Limits	
Terms of Service	https://pushbots.com/home/policy
Security	
Authentication Model	HTTP Basic Authentication
SSL Support	YES
Read-only Without Login	YES
Support	
Vendor API Kits/Support the third party API	YES
Community API Kits	
API Blog	
API Forum	
Site Blog	https://blog.pushbots.com/
Developer Support	
Console URL	
Support E-mails	YES
Support SMS Text Messages	YES
Support Pushing Cloud Notifications	PushBots support scalability via flexible APIs by cloud service which support custom integration.
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	YES

C2DM	NO
Apple Push Notification Systems (APNS)	YES
Microsoft (WP 7, WP 8, Windows 8)	NO
Firefox OS	NO
RIM Blackberry	NO
Amazon Device Messaging (ADM)	NO

Appendix 13

Everbridge Mass Notification System (MNS)	
Server/Service(System)	An emergency mass notification or communication service
Protocols	REST
Notification Transport Protocol	HTTP, HTTPS, SMTP for e-mails
Data Formats	JSON
API home	http://www.everbridge.com/
Functionality	
Client Install Required	YES
Service Endpoint	Smart phones, tablets
WSDL	\
Batch processing	YES
Scalability	Remote API
Mobility	Allow for localized global communications
Fault tolerance	Powerful reports and robust analytics
Backward compatibility	YES
Sign up and Licensing	
Sign up Requirements	YES
Developer Key Required	YES
Account Required	YES
Commercial Licensing	Yes with service
Provider	http://www.everbridge.com/
Managed By	Everbridge
Non-Commercial Lic.	
Data Licensing	Proprietary
License for server(service)/client	

Usage Fees	
Program Fees	
Certification Program	
Usage Limits	
Terms of Service	
Security	
Authentication Model	API key
SSL Support	YES, HTTP library's SSL verification
Read-only Without Login	NO
Support	
Vendor API Kits/Support the third party API	
Community API Kits	
API Blog	
API Forum	
Site Blog	www.everbridge.com/category/our-company/blog/
Developer Support	
Console URL	
Support E-mails	YES
Support SMS Text Messages	YES
Support Pushing Cloud Notifications	YES
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	YES
C2DM	YES
Apple Push Notification Systems (APNS	YES
Microsoft (Windows Phone 7, WP 8, Windows 8)	YES
Firefox OS	NO

RIM Blackberry	YES
Amazon Device Messaging (ADM)	YES

Appendix 14

mobDB	
Server/Service(System)	A push notification service for mobile development
Protocols	REST
Notification Transport Protocol	HTTP, HTTPS
Data Formats	XML, JSON
API home	https://www.mobdb.net
Functionality	
Client Install Required	NO
Service Endpoint	iOS, Android (2.2 +)
WSDL	\
Batch processing	mobDB is able to push scheduled notifications to a set of devices via SDKs and RESTful APIs, which means mobDB supports batch operations via SDKs and RESTful APIs. Because mobDB can create, delete and update a batch of requests with a batch endpoint to reduce the time spent on network round trips.
Scalability	mobDB runs the push notification service on Amazon web services and is able to automatically scale with the demands of applications.
Mobility	mobDB enables application developers to push notifications to notify their offline application users.
Fault tolerance	Advanced statistics feature of mobDB allows you to keep track of push notifications.
Backward compatibility	YES
Sign up and Licensing	
Sign up Requirements	YES
Developer Key Required	YES
Account Required	YES
Commercial Licensing	Yes with service

Provider	https://www.mobdb.net
Managed By	Amazon Web Services
Non-Commercial Lic.	YES
Data Licensing	GNU GPL
License for server(service)/client	Service
Usage Fees	https://www.mobdb.net/pricing
Program Fees	Free to $4.99/month
Certification Program	
Usage Limits	
Terms of Service	http://www.mobdb.net/tandch.jsp
Security	
Authentication Model	API key
SSL Support	YES, HTTP library's SSL verification
Read-only Without Login	NO
Support	
Vendor API Kits/Support the third party API	iOS, Android, Java
Community API Kits	
API Blog	
API Forum	
Site Blog	
Developer Support	support@mobdb.net
Console URL	
Support E-mails	YES
Support SMS Text Message	YES
Support Pushing Cloud Notification	YES
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	Android (2.2 +) devices

C2DM	NO
Apple Push Notification Systems (APNS)	iPhone, iPad
Microsoft (Windows Phone 7, WP 8, Windows 8)	NO
Firefox OS	NO
RIM Blackberry	NO
Amazon Device Messaging (ADM)	NO

Appendix 15

NACapp	
Server/Service(System)	A push notification service for mobile devices.
Protocols	SOAP, REST
Notification Transport Protocol	HTTP, HTTPS, SMTP
Data Formats	XML
API home	http://www.nacapp.com/Support/Resources/
Functionality	
Client Install Required	YES
Service Endpoint	iOS, Android, Blackberry, Windows Phone, iPad
WSDL	\
Batch processing	NACapp can push notifications to a group of devices. Because NACapp has CollaChat to allow group chatting among staff, and CollaChat is available on Android, iPhone and iPad devices currently.
Scalability	NACapp uses cloud service for scalability.
Mobility	NACapp is able to deliver critical alert notifications to mobile applications in a unique way based on the urgency, no matter where they are.
Fault tolerance	NACapp makes duplicated and redundant data to a server, which brings backup, reliability and speed notification delivery.
Backward compatibility	Cisco NAC appliance has been tested with stand alone CAS and NM Module CAS(Swiffer).
Sign up and Licensing	
Sign up Requirements	YES
Developer Key Required	YES
Account Required	YES
Commercial Licensing	

Provider	www.nacapp.com
Managed By	TNZ Group Limited
Non-Commercial Lic.	
Data Licensing	Apache Version 2
License for server(service)/client	Service
Usage Fees	Fees vary on different situations. http://www.nacapp.com/Pricing/
Program Fees	
Certification Program	
Usage Limits	
Terms of Service	http://www.nacapp.com/AboutUs/Terms/
Security	
Authentication Model	API key, NACapp can integrate other services such as Two Factor Authentication.
SSL Support	YES
Read-only Without Login	NO
Support	
Vendor API Kits/Support the third party API	NACapp can integrate other services such as Two Factor Authentication.
Community API Kits	
API Blog	
API Forum	http://www.nacapp.com/JoinUs/
Site Blog	
Developer Support	*developers@nacapp.com*
Console URL	
Support E-mails	YES
Support SMS Text Message	YES
Support Pushing Cloud Notification	NACapp is a full cloud solution.

Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	YES
C2DM	NO
Apple Push Notification Systems (APNS)	YES, iPhone, iPad
Microsoft (Windows Phone 7, WP 8, Windows 8)	YES
Firefox OS	NO
RIM Blackberry	YES
Amazon Device Messaging (ADM)	NO

Appendix 16

OpsGenie	
Server/Service(System)	A multichannel IT alert notification service
Protocols	REST
Notification Transport Protocol	HTTP, HTTPS
Data Formats	JSON
API home	http://support.opsgenie.com/customer/portal/topics/495390-web-api/articles
Functionality	
Client Install Required	YES
Service Endpoint	iOS, Android and Blackberry
WSDL	\
Batch processing	OpsGenie allows to send 10 alert notification in a batch via the group API.
Scalability	OpsGenie integrates MongoDB to realize scalability and high-performance.
Mobility	OpsGenie is able to push relevant notifications to the right people even when applications are offline, because OpsGenie uses incident management to implement the offline triggers.
Fault tolerance	OpsGenie is reliable, because OpsGenie has a distributed and redundant architecture across diversified data centers with an end-to-end monitor system to guarantee the availability, and the monitor system uses a detailed track mechanism.
Backward compatibility	Integration plugins
Sign up and Licensing	
Sign up Requirements	YES
Developer Key Required	YES
Account Required	YES
Commercial Licensing	Yes with service

Provider	http://www.opsgenie.com
Managed By	iFountain LLC
Non-Commercial Lic.	
Data Licensing	Apache Version 2
License for server(service)/client	Service
Usage Fees	Fees depend on functions. http://www.opsgenie.com/pricing
Program Fees	
Certification Program	
Usage Limits	OpsGenie does not limit alert messages to a few simple characters. Alerts can contain free form text, tags, additional fields, and even files.
Terms of Service	http://www.opsgenie.com/tos
Security	
Authentication Model	API key
SSL Support	YES
Read-only Without Login	NO
Support	
Vendor API Kits/Support the third party API	Supporting multiple integration plugins http://support.opsgenie.com/customer/portal/topics/255609-integration-plugins/articles
Community API Kits	
API Blog	
API Forum	
Site Blog	http://blog.opsgenie.com/
Developer Support	
Console URL	
Support E-mails	YES
Support SMS Text Messages	YES

Support Pushing Cloud Notifications	OpsGenie is cloud based notification
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	YES
C2DM	NO
Apple Push Notification Systems (APNS	YES
Microsoft (Windows Phone 7, WP 8, Windows 8)	NO
Firefox OS	NO
RIM Blackberry	YES
Amazon Device Messaging (ADM)	NO

Appendix 17

SnapComms	
Server/Service(System)	A push notification service
Protocols	REST
Notification Transport Protocol	HTTP, HTTPS
Data Formats	JSON
API home	http://www.snapcomms.com/products/desktop-alert.aspx
Functionality	
Client Install Required	YES
Service Endpoint	iOS, Android, Windows Phone, desktops
WSDL	\
Batch processing	SnapComms can deliver notifications in batches.
Scalability	SnapComms supports scalability with cloud solutions including data center, cloud communications, cloud storage and data center.
Mobility	The SnapComms messaging platform allows message administrators to target messages by role or (in some instances) location.
Fault tolerance	The SnapComms platform provides full reporting on message readership as well as collects and collates.
Backward compatibility	NO
Sign up and Licensing	
Sign up Requirements	YES
Developer Key Required	YES
Account Required	YES
Commercial Licensing	
Provider	http://www.snapcomms.com/
Managed By	SnapComms

Non-Commercial Lic.	
Data Licensing	BSD
License for server(service)/client	Service
Usage Fees	Fees depend on channels. http://www.snapcomms.com/learning-center/pricing-and-savings.aspx
Program Fees	
Certification Program	
Usage Limits	The polling frequency default is a minimum of 120 seconds, but this can be adjusted to any interval required.
Terms of Service	http://www.snapcomms.com/terms-of-use.aspx
Security	
Authentication Model	API key
SSL Support	YES
Read-only Without Login	YES
Support	
Vendor API Kits/Support the third party API	SnapComms API makes it possible to integrate with other solutions or platforms.(for example, intranet and contact center solutions)
Community API Kits	
API Blog	
API Forum	
Site Blog	
Developer Support	
Console URL	
Support E-mails	YES
Support SMS Text Messages	YES
Support Pushing Cloud Notifications	SnapComms is an employee communications software company, which gives cloud solutions including data center, cloud communications,

	cloud storage and data center.
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	YES
C2DM	NO
Apple Push Notification Systems (APNS	YES
Microsoft (Windows Phone 7, WP 8, Windows 8)	YES
Firefox OS	YES
RIM Blackberry	NO
Amazon Device Messaging (ADM)	YES

Appendix 18

Notificare	
Server/Service(System)	A mobile push notification platform
Protocols	The API uses RESTful protocol.
Notification Transport Protocol	HTTP, HTTPS
Data Formats	Responses are formatted in JSON.
API home	http://notifica.re/features/api/
Functionality	
Client Install Required	YES
Service Endpoint	iOS, Android, web
WSDL	\
Batch processing	Notificare has free SDK inside smart push notifications across all channels. Notificare also has user segmentation to categorize users based on users' geo locations, providing a powerful search query and easily targeting different groups of users with batches of notifications.
Scalability	Notificare has smarter push notifications SDK which helps to set up a scalable messaging service.
Mobility	Notificare has location services which can transform a location into a relevant push and gather enormous amounts of data in just 24 hours.
Fault tolerance	Upload certificates for APNS to validate them and manage the services and keep track of expiration.
Backward compatibility	
Sign up and Licensing	
Sign up Requirements	YES
Developer Key Required	YES
Account Required	YES
Commercial Licensing	Yes with service

Provider	www.notifica.re
Managed By	Notificare
Non-Commercial Lic.	
Data Licensing	BSD
License for server(service)/client	Client
Usage Fees	http://notifica.re/pricing/
Program Fees	
Certification Program	
Usage Limits	Free push provides unlimited notifications to unlimited devices up to 5,000 users.
Terms of Service	http://notifica.re/terms/
Security	
Authentication Model	API key
SSL Support	YES
Read-only Without Login	NO
Support	
Vendor API Kits/Support the third party API	PHP, Python, .NET, ColdFusion, Ruby. Support integration requests to existing APIs. Provide libraries for several back-end technologies.
Community API Kits	
API Blog	
API Forum	
Site Blog	http://notifica.re/blog/
Developer Support	
Console URL	
Support E-mails	YES
Support SMS Text Messages	YES

Support Pushing Cloud Notifications	YES
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	Android devices
C2DM	NO
Apple Push Notification Systems (APNS	iPhone, iPad
Microsoft (Windows Phone 7, WP 8, Windows 8)	NO
Firefox OS	NO
RIM Blackberry	NO
Amazon Device Messaging (ADM)	NO

Appendix 19

AeroGear UnifiedPush Server	
Server/Service(System)	A push notification server for web/application
Protocols	REST
Notification Transport Protocol	Notifications over HTTP, HTTPS
Data Formats	JSON
API home	http://aerogear.org/docs/specs/aerogear-rest-api/
Functionality	
Client Install Required	YES
Service Endpoint	iOS, Android,Web,Hybrid
WSDL	/
Batch processing	AeroGear UnifiedPush Server allows a push application to add multiple variants, and a variant to add multiple installations/devices.
Scalability	AeroGear UnifiedPush Server provides a cloud infrastructure to support scalability.
Mobility	AeroGear UnifiedPush Server is able to integrate with existing enterprise environment to support scalability with cloud service.
Fault tolerance	AeroGear UnifiedPush Server supports password recovery which is a bare minimum set of cryptographic primitives with Two-Factor Authentication and One-Time-Password features to improve security, where Objective-C One Time Password API is an iOS library for generating one time passwords according to RFC 6238.
Backward compatibility	API compatibility
Sign up and Licensing	
Sign up Requirements	YES
Developer Key Required	YES
Account Required	YES
Commercial Licensing	Yes with service

Provider	http://aerogear.org/
Managed By	AeroGear
Non-Commercial Lic.	No
Data Licensing	Apache License Version 2.0
License for server(service)/client	Server
Usage Fees	Free
Program Fees	Free and open source
Certification Program	
Usage Limits	NO
Terms of Service	http://www.waonlinestore.com/downloads/aerogear_gift_card_terms_12013.pdf
Security	
Authentication Model	HTTP Basic Authentication
SSL Support	YES
Read-only Without Login	NO
Support	
Vendor API Kits/Support the third party API	YES
Community API Kits	
API Blog	
API Forum	
Site Blog	http://aerogear.org/community/
Developer Support	*aerogear-dev@lists.jboss.org*
Console URL	
Support E-mails	YES
Support SMS Text Messages	YES
Support Pushing Cloud Notifications	YES, AeroGear provides a cloud infrastructure.

Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	YES
C2DM	NO
Apple Push Notification Systems (APNS)	YES
Microsoft (Windows Phone 7, WP 8, Windows 8)	NO
Firefox OS	NO
RIM Blackberry	NO
Amazon Device Messaging (ADM)	NO

Appendix 20

Pushd	
Server/Service(System)	A push notification server
Protocols	REST
Notification Transport Protocol	Notifications over HTTP, HTTPS
Data Formats	JSON
API home	https://github.com/rs/pushd/blob/master/settings-sample.coffee/
Functionality	
Client Install Required	YES
Service Endpoint	iOS, Android,Windows Phone, Web
WSDL	/
Batch processing	Pushd enables broadcast and multicast.
Scalability	Pushd provides a cloud infrastructure to support scalability. Pushd not only disseminates push notifications to support web applications, mobile platforms and even HTTP server from a single unified entry point, but has the ability to extend to other platforms in a appropriate way
Mobility	Pushd is able to integrate with existing enterprise environment to support scalability with cloud service.
Fault tolerance	YES
Backward compatibility	API compatibility
Sign up and Licensing	
Sign up Requirements	YES
Developer Key Required	YES
Account Required	YES
Commercial Licensing	Yes with service
Provider	Olivier Poitrey cooperated with Juho Vähä-Herttua, Antti Ajanki, Andrew Naylor,

	yanncoupin.
Managed By	Olivier Poitrey cooperated with Juho Vähä-Herttua, Antti Ajanki, Andrew Naylor, yanncoupin.
Non-Commercial Lic.	No
Data Licensing	The MIT license
License for server(service)/client	Server
Usage Fees	Free
Program Fees	Free and open source
Certification Program	
Usage Limits	NO
Terms of Service	
Security	
Authentication Model	HTTP Basic Authentication
SSL Support	YES
Read-only Without Login	NO
Support	
Vendor API Kits/Support the third party API	YES
Community API Kits	
API Blog	
API Forum	
Site Blog	http://aerogear.org/community/
Developer Support	*rs@dailymotion.com*
Console URL	
Support E-mails	YES
Support SMS Text Messages	YES

Support Pushing Cloud Notifications	YES, Pushd provides a cloud infrastructure.
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	YES
C2DM	YES
Apple Push Notification Systems (APNS)	YES
Microsoft (Windows Phone 7, WP 8, Windows 8)	YES
Firefox OS	YES
RIM Blackberry	NO
Amazon Device Messaging (ADM)	NO

Appendix 21

PushSharp	
Server/Service(System)	A third party push notification server
Protocols	REST
Notification Transport Protocol	HTTP, HTTPS
Data Formats	JSON
API home	
Functionality	
Client Install Required	YES
Service Endpoint	iOS (iPhone/iPad APNS), OSX (APNS 10.8+) Android (C2DM and GCM), Chrome, Windows Phone, Windows 8, Blackberry (PAP), Amazon (ADM)
WSDL	\
Batch processing	YES
Scalability	PushSharp uses cloud services via flexible APIs to support scalability.
Mobility	PushSharp provides push notification service to mobile devices.
Fault tolerance	PushSharp provides a reliable service
Backward compatibility	YES
Sign up and Licensing	
Sign up Requirements	YES
Developer Key Required	YES
Account Required	YES
Commercial Licensing	Yes with service
Provider	http://xamarin.com/evolve/2013#keynote
Managed By	Xamarin EVOLVE 2013
Non-Commercial Lic.	
Data Licensing	Apache License Version 2.0

License for server(service)/client	Server
Usage Fees	Free and open source
Program Fees	
Certification Program	
Usage Limits	Unlimited push notifications
Terms of Service	
Security	
Authentication Model	API key
SSL Support	YES, HTTP library's SSL verification
Read-only Without Login	YES
Support	
Vendor API Kits/Support the third party API	A third party push notification service
Community API Kits	
API Blog	
API Forum	
Site Blog	
Developer Support	
Console URL	
Support E-mails	YES
Support SMS Text Messages	YES
Support Pushing Cloud Notifications	YES
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	YES, Chrome, Phones,Tablets
C2DM	NO
Apple Push Notification Systems (APNS)	iPhone, iPad
Microsoft (Windows Phone 7, WP 8, Windows 8)	WP 7 / 7.5 / 8 (e.g. FlipTile, CycleTile, and IconicTile Templates), Windows 8
Firefox OS	NO, but coming soon

RIM Blackberry	YES, BIS and BES via PAP
Amazon Device Messaging (ADM)	YES

Appendix 22

Uniqush	
Server/Service(System)	A mobile push notification server
Protocols	REST
Notification Transport Protocol	HTTP, HTTPS
Data Formats	Responses are formatted in JSON.
API home	http://notifica.re/features/api/
Functionality	
Client Install Required	YES
Service Endpoint	iOS, Android, ADM
WSDL	\
Batch processing	Uniqush supports wildcard multicast and native multicast, so that you multicast notifications can be sent to the specific group.
Scalability	Uniqush-front handles scale-out growth with cloud service.
Mobility	YES
Fault tolerance	Uniqush provides a reliable unified push service, because Uniqush will send the notification again as a recoverable error whenever possible.
Backward compatibility	
Sign up and Licensing	
Sign up Requirements	YES
Developer Key Required	YES
Account Required	YES
Commercial Licensing	Yes with service
Provider	www.uniqush.org
Managed By	Uniqush
Non-Commercial Lic.	

Data Licensing	Apache License Version 2.0
License for server(service)/client	Client
Usage Fees	
Program Fees	Free and open source
Certification Program	
Usage Limits	
Terms of Service	http://uniqush.org/terms/
Security	
Authentication Model	API key
SSL Support	YES
Read-only Without Login	NO
Support	
Vendor API Kits/Support the third party API	YES
Community API Kits	
API Blog	
API Forum	
Site Blog	http://uniqush.org/documentation/index.html
Developer Support	https://groups.google.com/forum/#!forum/uniqush
Console URL	
Support E-mails	YES
Support SMS Text Messages	YES
Support Pushing Cloud Notifications	YES
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	Android devices
C2DM	YES
Apple Push Notification Systems (APNS)	iPhone, iPad

Microsoft (Windows Phone 7, WP 8, Windows 8)	NO
Firefox OS	NO
RIM Blackberry	NO
Amazon Device Messaging (ADM)	YES

Appendix 23

OpenMobster	
Server/Service(System)	A push notification server
Protocols	REST
Notification Transport Protocol	HTTP, HTTPS
Data Formats	JSON
API home	http://openmobster.googlecode.com/svn/wiki/content/app-developer-guide/html/index.html
Functionality	
Client Install Required	YES
Service Endpoint	iPhone, Android, Windows Phone, Blackberry and Symbian
WSDL	\
Batch processing	YES
Scalability	PhoneGap enables application developers to authorize native applications.
Mobility	Geozones are pins on the map that allow sending automatically triggered push notifications right at the moment a user enters the specified area on the map.
Fault tolerance	OpenMobster has synchronized data which is replicated across devices and platforms for fault tolerance. If something unexpected happens, application users can get access through the cloud server.
Backward compatibility	NO
Sign up and Licensing	
Sign up Requirements	YES
Developer Key Required	YES
Account Required	YES
Commercial Licensing	Yes with service
Provider	http://www.openmobster.com/index.html

Managed By	OpenMobster
Non-Commercial Lic.	
Data Licensing	Eclipse Public License 1.0
License for server(service)/client	Server
Usage Fees	Free
Program Fees	
Certification Program	
Usage Limits	No limits
Terms of Service	http://www.openmobster.com/service.html
Security	
Authentication Model	API keys
SSL Support	YES, HTTPS
Read-only Without Login	YES
Support	
Vendor API Kits/Support the third party API	Any reference to any third party on the Website is provided to you for informational purposes only.
Community API Kits	
API Blog	
API Forum	
Site Blog	https://groups.google.com/forum/#!forum/openmobster-users
Developer Support	
Console URL	
Support E-mails	YES
Support SMS Text Messages	YES
Support Pushing Cloud Notifications	YES
Support Notifications Platforms/OS	
Google Cloud Messaging (GCM)	YES

C2DM	NO
Apple Push Notification Systems (APNS)	YES
Microsoft (Windows Phone 7, WP 8, Windows 8)	YES
Firefox OS	NO
RIM Blackberry	YES
Amazon Device Messaging (ADM)	NO