A mobile campus information push system is developed based on the proposed system architecture shown in Fig. 1, which mainly consists of push resource website, push proxy server, android push client, and streaming media server. The communication between push resource website and push proxy server is based on the PubSub protocol—PubSubHubbub. The publisher, which is the push resource website, publishes various information resources, such as intramural news, notices, and video courseware. And also takes the burden of visit from the browser. The communication protocol between push proxy server and android push clients is the instant messaging protocol—XMPP. The push proxy server acts as a subscribe proxy, forwards not only the subscription requests to the Publisher, but also the updated information to the android push clients. Push proxy server is the interface of PubSub protocol and instant messaging protocol, and is separated from the push resource website, which exempts the website from keeping persistent connections with android clients, and greatly reduces its burden.

The streaming media server, which affords streaming media service, uses RTMP protocol to handle the streaming media requests from both browser clients and android clients. Android push clients are the terminals of this information push system, can subscribe and receive information.

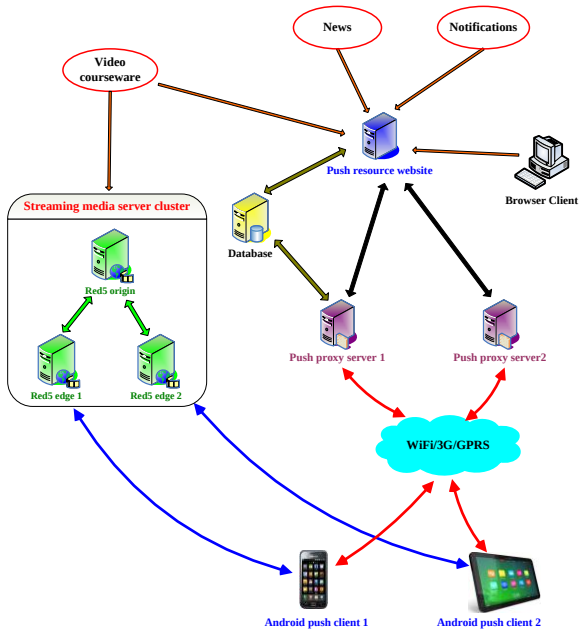

Figure 1. Fig 1. System architecture

A. PubSubHubbub Protocol

PubSubHubbub [4] is a server to server PubSub protocol between the push resource website and push proxy server in this system. PubSubHubbub adopts HTTP non polling mode which is real-time and cost-effective, enables subscribers to receive the updated information from publishers as soon as the information subscribed is updated. Hence the subscribers do not need to query the resource website continually.

The PubSubHubbub consists of three parts: Publisher, Hub and Subscriber, and two stages: the subscribe stage and the publish stage.

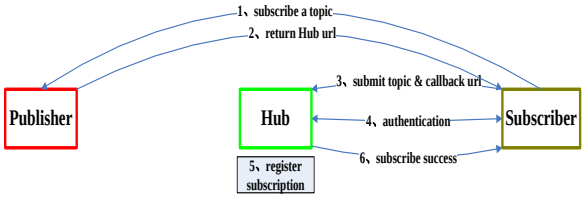

Figure 2. Fig 2. Subscribe stage in PubSubHubbub

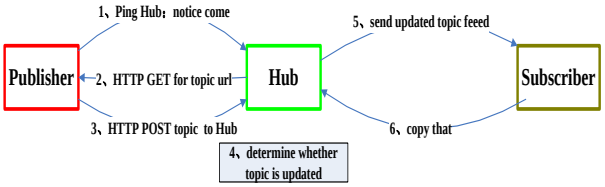

Figure 3. Fig 3. Publish stage in PubSubHubbub

In the subscribe stage shown in Fig. 2, the Subscriber submits the subscribed topics to the Publisher, and the Publisher returns the Hub's url, and then the Subscriber submits his topics to the Hub with his own callback url. After the mutual authentication process between the Hub and the Subscriber, the Hub registers this subscription with the Subscriber's callback url.

In the publish stage shown in Fig. 3, when the topic subscribed is updated, the Publisher sends a ping message to notice Hub the u

pdate. The Hub sends a HTTP GET request for the topic feed, and then the Publisher returns a HTTP POST message to the Hub with the topic feed. The Hub determines whether the feed is updated, if updated, it extracts the updated topic feed, sends a HTTP POST request to the Subscriber. At last, the Subscriber receives the updated topic feed, and parses the feed to get the title, content, url and other information.

B. XMPP Protocol

XMPP [5] is one of the most flexible and prevailing instant messaging protocols. It is a real-time C/S architecture over TCP. In XMPP, loading of the server is onerous and that of the client is simple, which is suitable to mobile applications. Also the XMPP is open and can adopt SASL and TLS to guarantee the information security.

The XMPP defines three roles: server, client and gateway. The transferred message is in XML stream format which contains three basic elements: Presence, Message, and Iq. The Presence element is used to determine the state of the user (client or server). The Message element is used to send messages between the two users, and the Iq element based on request-response mechanism allows two users to inquiry and response in XML format.

C. RTMP protocol

Traditional information push systems push text message only. Multimedia information such as video

courseware is usually required by campus information push system. RTMP is a streaming media protocol [6], and is employed to provide streaming media service in our system. RTMP builds over TCP or polling HTTP protocol and mainly transports audio, video and AMF format data between flash player and the RTMP streaming media server. RTMP uses different working mode to transfer video compared with traditional HTTP approach. In the traditional HTTP mode, the client downloads the video file to the local disk first and then plays it. The downloaded video can be found in the local cache, which is not secure. Also, if the content server is not speed-limited, the greater the client's bandwidth is, the more bandwidth the server consumes. If is speed-limited, the user experience may downgrade.

In the RTMP mode used in our system, client can play video in real-time, and choose arbitrary playback position to play without downloading the former parts of the video. Also no any video is cached locally, so it is suitable for copyright protection.

RTMP has several variant protocols: RTMPT protocol in which the RTMP messages are encapsulated over HTTP hence the RTMP can traverse firewall; RTMPS protocol which is over HTTPS and offers a secure connection; and RTMPE protocol which is an encryption version of RTMP. These versions can be adopted in our system.

III. TECHNICAL REALISATION

The push resource website is built based on the open-source blog platform — wordpress, and the Hub in PubSubHubbub protocol is built with the wordpress plug-in—pushpress. Here we combine the Publisher and Hub together to eliminate the HTTP request and response process between them. As a result, the whole system delay is lower. Fig. 4 explains this.

Figure 4. The Publisher /Hub combined

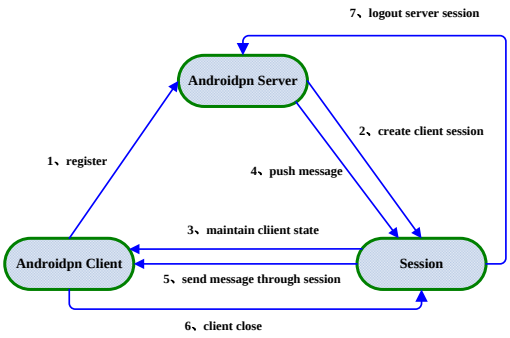

Figure 5. Androidpn workflow

The realization of the instant messaging part between the push proxy server and android push clients is based on the second development of the open-source android push platform—androidpn. Androidpn [7] is developed in java, implements the XMPP protocol both on server and client. XMPP adopts Mina framework to manage socket connections, which is really suited for high concurrent information push situation because of Mina's non-blocking feature. The workflow of androidpn is shown in Fig. 5.

1) The client opens up the client side application, sets up a persistent connection to androidpn server, and sends registration information.

2) After finishing registration, the server establishes a corresponding session for the client registered, which is used to manage client's state.

3) Server sends message to the sessions alive via socket connections. Client receives the message, parses it and shows the message.

Spring, Hibernate technology and Mina framework are used in androidpn server. In the top of the androidpn server structure shown in Fig. 6, there are four important parts: Session Manager, Auth Manager, Presence Manager and Notification Manager. Session Manager is responsible for the session management between clients and server. Auth Manager is responsible for user authentication. Presence Manager is responsible for clients' status management. And Notification Manager is in charge of pushing information from server to clients.

Androidpn client has integrated the asmack, which is the XMPP client side implementation for android. Androidpn leaves a power of work such as manage connection， send message and other laboring work to server, but creates lightweight clients. This is praisable for reducing android client power consumption and client burden.

The subscriber of the PubSubHubbub protocol is implemented in java, and integrated with the androidpn server together to form the push proxy server. The push proxy server shares the user database with the push resource website.

Subscribe, view details, video display and video broadcast functions are all implemented in the android push client.

To provide streaming media service as well as possible, red5 is adopted as the streaming media server. Red5 [8] is an open-source server supports RTMP, RTMPT, RTMPS, RTMPE protocols, and flv, f4v, mp4 format video and mp3, aac format audio. Since streaming media consumes huge swathes of bandwidth, it is critical on server performance. So cluster the red5 to provide better QoS is a must. The cluster of red5 divides into two classes: the red5-origin and the red5-edge. Red5-origin is the server where releases videos, and the red5-edges are cluster servers, do not release videos but obtain videos from the red5-origin and then forward to clients. In the red5 cluster, a red5-origin corresponds to several red5-edges. Fig. 7 shows the cluster architecture. The connections between red5-origin and red5-edges are MRTMP connections on port 9035, and the connections between red5-edge and clients are RTMP or RTMPT connections. Actually, one MRTMP connection is the RTMP or RTMPT

connections multiplexing. By this way, the workload of streaming media servers is reduced, and the clients can not access directly to the red5-origin, which protects the security of the media.

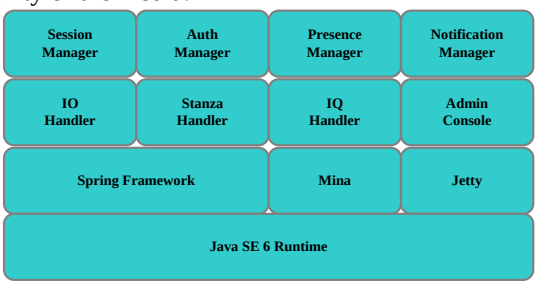

Figure 6. Androidpn server structure

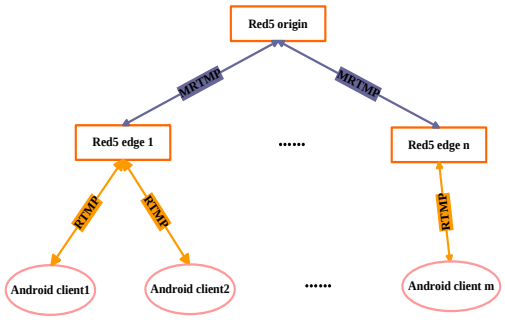

Figure 7. Red5 cluster

TABLE I. COMPARISON OF PULL, PURE PUSH AND PUSH-PUBSUB

	Pull	Pure Push	PubSub-Push
Description	Android clients polling for message from website continually	Website broadcast message to all android clients immediately	Clients subscribe topics, website publishes message, push server pushes notifications, clients pull detail message
Latency	Depend on polling interval, high commonly	Low	Low
Server consumption	Little	Website takes most work	Website is only responsible for publishing, while push server is responsible for pushing
Client consumption	Large	Little	Little
Subscribe capability	Good	None	Good

IV. SYSTEM TEST RESULTS

This novel hybrid mobile campus information push system is based on the PubSub-Push scheme proposed above. Table I shows the comparison of the traditional http pull scheme, pure push scheme and the proposed pubsub-push scheme, and we can find the advantage of the pubsub-push scheme on mobile campus information push system obviously.

The proposed system is not only designed but also implemented in a real environment. It is very convenient and prompt for the android push clients to subscribe topics and receive information over WiFi/WCDMA/GSM. Fig. 8 shows the system push delay between the time when push resource website publishes one message ("Hello World!") and when all android push clients receive it.

The result in Fig. 8 shows that the way of pushing information over WiFi has the lowest delay, about 200 milliseconds to 40 android clients, while the GSM way consumes about 900 milliseconds, the delay of the WCDMA way falls in between them, about 400 milliseconds. Either way, the system proves efficient in its practical use. And the delay increases very slowly with the android push clients growing. So, this mobile information push system proves stable.

The system effectiveness is demonstrated in Fig. 9, Fig. 10 and Fig. 11, Fig. 11 shows the received video playing on android mobile and the live video broadcasting on one android mobile and playing on another android mobile and our website.

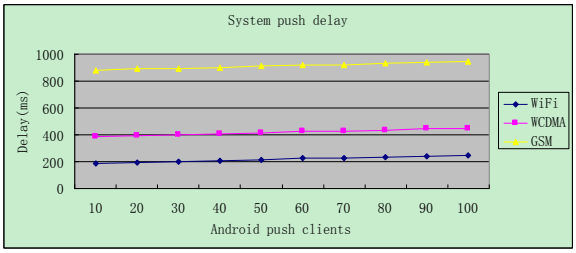

Figure 8. Fig 8. System push delay

Figure 9. Push resource website

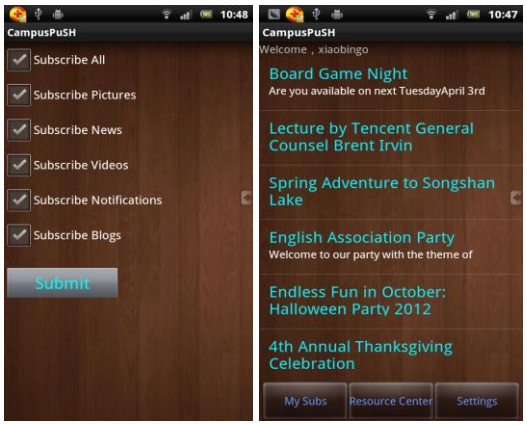

Figure 10. Subscribe and receive information on android

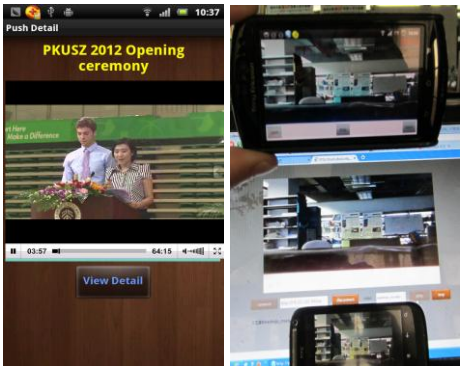

Figure 11. RTMP video play and live broadcast

V. Conclusion

Traditional mobile instant messaging systems and mobile PubSub systems cannot meet the need of mobile digital campus system. A novel mobile campus information push system, which combines the PubSubHubbub, XMPP and RTMP, is designed and implemented in this paper. This system intends to enable publishers to publish intramural information like notices and video courseware to subscribers effectively, and enable subscribers to accept notices and watch videos including live videos both on android mobile terminals and on our website. The experimental results of this system indicate that the combination proposed is an effective way to establish a real-time, subscribable, content-rich, stable, reliable and cost-effective mobile campus information push system.

Acknowledgment

The work described in this paper was supported by the 973 Program 2012CB315904, China, and the Research Program of Shenzhen, China. The authors thank Chao Wang, for his praisable art designing work on our system. Also, we wish to thank the Information & Technology Office of Peking University Shenzhen Graduate School, for their network and devices support.

References

[1] J. Brustel and T. Preuss. "A universal push service for mobile devices," *IEEE Conference Publications*. *Complex, Intelligent and Software Intensive Systems*, pp. 40-45, July 2012.

[2] V. Pimental and B. G. Nickerson, "Communicating and displaying real-time data with WebSocket," *IEEE Journals & Magazines. Internet Computing*, vol. 16, pp. 45-53, July-Aug 2012.

[3] Z. H. Wang, X. L. Xiong, and Y. G. Hou, "The Study of mobile learning based on information push technology," *IEEE Conference Publications Education Technology and Computer Science*, vol. 2, pp. 1140-1142, March 2009.

[4] Google. (19-Nov-2012). PubSubHubbub Core 0.3 – Working Draft. [Online]. Available: http://pubsubhubbub.googlecode.com/svn/trunk/pubsubhubbub-core-0.3.html

[5] Network Working Group. (19-Nov-2012) Extensible Messaging and Presence Protocol (XMPP): Core. [Online]. Available: http://www.ietf.org/rfc/rfc3920

[6] Adobe. (19-Nov-2012). Real-Time Messaging Protocol Specification. [Online]. Available: http://www.adobe.com/devnet/rtmp.html

[7] Source Forge. (19-Nov-2012). Android Push Notification. [Online]. Available: http://androidpn.sourceforge.net/

[8] Red5. (19-Nov-2012). Red5–Reference Documentation Version 1.0, [Online]. Available: http://www.red5.org/downloads/docs/red5-reference-1.0.pdf.

