Enabling Uniform Push Services for WAP and WWW

Yen-Cheng Chen
Department of Computer Science
Ming Chuan University
Taoyuan, Taiwan 333, Republic of China
yencheng@mcu.edu.tw

Abstract

A uniform push framework for both WAP and WWW is proposed in this paper. In WAP 1.2, a major enhancement over WAP 1.1 is the support of push services. That is, an original server providing contents can asynchronously send information to a WAP client without any explicit request from the WAP client. In the Internet WWW environment, there are also demands for web push services. Instead of developing a new suite of protocols for web push, we propose the use of the WAP Push Access Protocol (PAP) in web push services. This approach enables uniform push services for WAP and WWW. Our approach requires each web client to support PAP and push user agent functionality. A system architecture for a web client is thus proposed. The implementation of a web client on PCs is also discussed.

1. Introduction

Wireless Application Protocol (WAP) [1], developed by WAP Forum [2], is to define a suite of industry-wide specifications for enabling applications over wireless communication networks. The one of initial goals of WAP is to introduce the web browsing-like capabilities into mobile devices [3,4]. Regarding this issue, two major limitations were carefully considered. One is the limitation of mobile devices in their small display, little memory space, and less computing capability. The other is the unreliable communication and low transmission support in current wireless network infrastructure. To reduce the possible implementation complexity, a WAP protocol gateway is introduced between the mobile devices and the original servers providing WML/WMLScript pages [5]. A WAP gateway receives a request from WAP clients, retrieves the desired WML pages stored in Internet web servers via HTTP, encodes the WAP contents, and finally sends the encoded binary results to the WAP clients. Since WAP gateway divides the complicated communication procedures into wireless and wired parts, WAP content providers in wired Internet can easily provide WML pages just like HTML ones. No additional hardware or software is required but the HTTP server. Indeed, there have been a lot of web servers providing both HTML and WML pages in the Internet.

In 1999, WAP Version 1.2 was proposed. The major enhancement over WAP 1.1 is the support of "push" services [6]. As known in a client/server model, a client can request a service or information from a server. The server then responds in transmitting information to the client. This is known as "pull" technology. In contrast, we may wish that the server also could send information to the client without any explicit request from the client. This capability comes from the "push" technology. In brief, a "pull" transaction of information delivery is initiated by the client, while a "push" transaction is initiated by the server. WAP Push service is proposed to make the servers to the push content to mobile devices in a standard manner within the WAP domain. Obviously, if we would like to get information as soon as it is available, letting the information be automatically pushed to us is an efficient way. Stock quote updates, important or urgent news, traffic reports, or event notifications may be typical examples demanding push services. WAP 1.2 will be the very first standard approach enabling push services in wireless network environments. We can foresee that a lot of WAP servers will turn into WAP push servers. As a result, we can get information efficiently from a mobile device without performing any manual operation but just keeping the mobile phone powered on.

Compared with the WAP Push services in wireless networks, current WWW services in the wired Internet are typical pull applications. We believe that there is demands for web push services in Internet. In fact, there have been a number of so-called "push" web applications such as Pointcast, Infogate [7], Netscape's Netcaster, and Microsoft I.E. Channels. However, the above applications are not really push applications. Quietly in the background, their client-side programs retrieve information from severs according to a predefined schedule. Users get the up-to-date contents as if they are pushed automatically by the servers. In [8], such kind of pushes is called "*Smart Pull*". The frequent information retrievals by smart pulls may consume considerable network bandwidth. In the recent years, a few true push approaches are proposed for efficient web information delivery [9-11]. The common features among them are channel-based information delivery and the use of multicast protocols. These features may limit the types of the delivered

information and the network environments. By WAP Push specifications [6], on the other hand, the push initiators can be from anywhere in the Internet and the information are delivered via unicast communications in the wired Internet side. Furthermore, the pushed information can be per-user based. All these features can lead to a more general push communication infrastructure. Encouraged by the WAP forum's effort in promoting public push services for general purposes, we think that if there is a similar open approach for web push services in the Internet, the WWW information will be delivered more efficiently. Consequently, like WAP push service, people can use desktop PCs to get web information easily without taking long time in opening a WWW browser, typing URLs, and surfing among tons of hyperlinks.

In this paper, we will propose an open approach enabling Internet web push services. Our idea is the use and adaptation of WAP push protocols in wired Internet. Although WAP Push protocols are designed for WAP wireless environment, we found that the wired Internet environment can be regarded as a particular and simplified configuration with respect to the WAP Push architecture. Hence, WAP Push protocols could be a possible candidate enabling web push services. We will adopt the WAP Push Access Protocol (PAP) [12] as the push transfer protocol between web clients and web push servers. Accordingly, both web clients and servers should support PAP to provide push services. Since PAP is a lightweight protocol over HTTP and most messages carried in PAP is of XML format [13], we will demonstrate that only modest overhead is introduced over HTTP and the PAP messages can be efficiently parsed by XML tools. Another reason why we choose the same push protocol as WAP push services is in order to make a WAP push server capable of pushing contents to the Internet. That is, under our approach, a WAP push server itself can also be a web push server except for the different content formats. On the other hand, a web push server can push information into wireless devices. Our uniform approach also facilitates integrated and intelligent push services, in which a push server can possibly determine an appropriate push method to deliver information to the client according to the predefined schedule or client location information.

The remainder of this paper is organized as follows. Section 2 is an overview of the WAP Push framework. In Section 3, we will propose a uniform push framework for both WAP and WWW. The use of WAP Push Access Protocol in web push services will be presented. Then, a possible architecture for a web client will be proposed in Section 4. A web client implementation for desktop PCs will also be discussed. Finally, conclusions and future work are given in Section 5.

2. WAP Push Framework

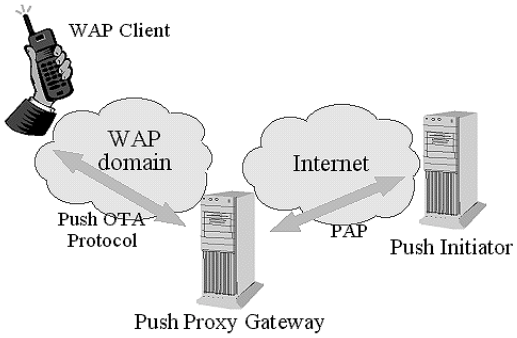

Figure 1. WAP Push Framework

Figure 1 illustrates the WAP Push framework [6]. The procedure of a push operation is usually started from the push server that wishes for delivering information to mobile clients. The push server, located in the Internet, is called Push Initiator (PI) in the WAP context. The push communication is not directly performed by PI and the mobile client. Similar to the WAP gateway in WAP 1.1, a WAP Push Proxy Gateway (PPG) [14] is introduced between the mobile clients in WAP domain and Push Initiators in the Internet. When a Push Initiator would like to send a push message to one or more WAP clients, it first transmits the push message together with control information to the Push Proxy Gateway (PPG) via Push Access Protocol (PAP), which is an application protocol on top of HTTP. When PPG receives a push request from a Push Initiator, it will retrieve the control information sent with the push content to determine the addresses of WAP clients to be pushed and the specified QoS requirements for the push communication over the air. The push content is then transmitted via the Push Over-The-Air (OTA) Protocol [15] from PPG to WAP clients in the wireless domain. Since it is possible that push message was not successfully delivered to the WAP clients for some reasons, or the push message is rejected or pended by WAP clients, the PPG may notify the Push Initiator about the result of the push operation. PPG may also provide the Push Initiator with client capability query services. The client capabilities query helps a Push Initiator to select the most appropriate format of the push content for the particular WAP client. The WAP client, the terminal side of the WAP Push framework, is also recommended to have two additional components: Session Initiation Application (SIA) and Application Dispatcher. The SIA establishes an active WSP [16] session between the WAP client and the PPG. The activated session is to enable connection-oriented push. Application Dispatcher is responsible for forwarding the received push messages to the appropriate upper applications.

3. A Uniform Push Framework

It can be seen that several concepts of WAP were derived from the Internet WWW services. In WAP

1.1, the Internet side of the WAP architecture is almost the same as WWW except for the markup languages. The advantage of the much similarity between WAP and WWW is the reuse of web technology in delivering WAP services. At the present, in WAP 1.2, WAP Push services have been well defined. However, there is still no open approach that makes possible the Internet web push services. Nevertheless, the much resemblance in the Internet side of these two services tells us that it is possible to adopt WAP Push technology in web push services. Our first idea is the use of PAP in the delivery of web push messages.

As shown in Figure 1, WAP Push services use PAP in the Internet side. PAP defines five operations between PI and PPG. PI can initiate the following operations to PPG:

a. Push Submission
b. Push Cancellation
c. Push Status Query
d. Client Capabilities Query

In the opposite direction, PPG is able to initiate the Result Notification operation to PI. A push message is sent primarily by means of the push submission operation. Push status query is to let PI understand the current status of a message submitted previously. Additionally, push cancellation allows PI to cancel a previous push submission. The client capabilities query operation helps PI to prepare the push content in an appropriate format suitable for the display of the push content in a wireless device. Via result notification, the PPG informs PI the final result of a previous push submission. We can find there is much effort taken in PAP to handle the potential communication issues caused by the latter transimisson of push messages over the air.

Now, let us start to consider the use of PAP in the web push services on Internet. To be consistent with WAP push, we also call the web push server Push Initiator. A web client is a node in the Internet which is able to receive push messages from PIs. Since both web clients and PIs reside in the Internet, it is not necessary to introduce any gateway between them. Recall that PAP is designed for communication between PIs and PPGs. Under the first glance, it seems to be problematical to adopt PAP between the web clients and PIs. In the following, let us consider an extreme case in the WAP environment to show the possibility of using PAP in web push services. For simplicity, we discard the wireless portion of the WAP Push architecture. Assume there is WAP Push message subscribed earlier by a client. From the standpoint of PI, the client is conceptually an address field in the push message since the actual wireless communication is performed by PPG. Hence, PI just fills in the necessary fields in the push message, and then sends it to the PPG via PAP. Let us assume the PPG itself is the client to receive the push message.

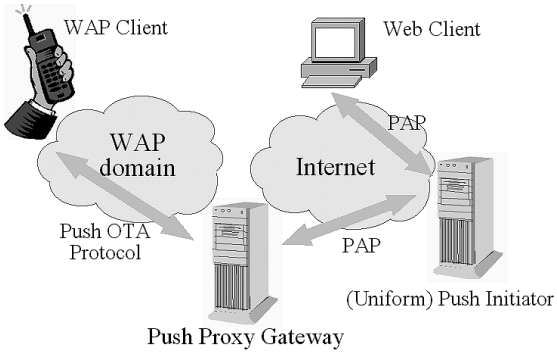

Figure 2. A Uniform Push Framework

Therefore, the client address field in the push message is the IP address of PPG. As a consequence, the client receives a push message directly from the PI, and the WAP Push transaction is done. In fact, this particular WAP Push transaction is a web push because the client is on the Internet and no wireless communication occurs. At this extreme case, only PAP is used for push message transfer. Therefore, if a web client can behave like the Internet side of a PPG when facing a PI, PAP can be used in web push sevices. The adoption of PAP in web push services can also lead to a uniform push framework. In the uniform push framework, each WAP PI itself is a Web PI without any modification in protocol level. No difference exists between WAP PIs and Web PIs. Accordingly, there will be only one kind of PIs in the Internet, and any client wherever it resides can receive push messages in a standard fashion. Figure 2 is exhibition of the uniform push framework. In the uniform push framework, the WAP clients and PPG as well as the protocols in the WAP domain are kept unchanged. In the following, the functionality of PI and Web Client is described in details.

A. Uniform Push Initiator (PI)

Within the uniform push framework, the uniform PI should fully conform to WAP 1.2. This is to achieve the seamless integration of WAP and Web Push services. In the protocol level, we recommend that the uniform PI behave in the same way wherever the client is located. That is, a uniform PI may issue any operation defined in PAP even if some of the operations are not necessary for web pushes. In addition, as in WAP Push, a uniform PI can also send Service Indication (SI) [17] and Service Loading (SL) [18] messages to a web client in the Internet. A web client may utilize these features to facilitate upper applications. The requirement of identical behavior in the protocol level allows the existing WAP PIs to push messages to web clients without any modification.

Beyond the protocol conformance, there still exist difference in PI implementation. For example, a WAP PI usually uses one or a mall number of PPGs for WAP Push services. In web push services, however, PIs should directly push information to

each web client. Therefore, a uniform PI should communicate with each web client as if there is a unique PPG for the web client. Accordingly, additional maintenance of PPG information will be required for a uniform PI. In addition, PAP allows a WAP PI to push an identical message to multiple WAP clients in one push operation. This feature cannot be supported if the recipients are web clients. Thus, the uniform PI should initiate a push operation for each web client. One possible solution is the use of multicast addresses, which are not implemented usually. Another potential issue is the availabilities of web clients. In WAP Push, push messages can be buffered in PPG if the WAP client is not available. Because the PPG is supposed to be always ready to receive push requests, a WAP PI will not be troubled when a WAP client is not active. On the other hand, if a web client is not active, PI cannot successfully send out a push message to the web client. Then, the uniform PI should take care of the successive retrials for ensuring eventually successful delivery of push messages.

B. Web Client

A Web Client receives push message from uniform PIs via PAP. Since uniform PIs treat Web Clients the same as PPG, it is required for a Web Client to support the functionality of the PPG components that interact with uniform PIs. A Web Client should at least provide the following features of PAP [12]:

a. Respond to a push submission,
b. Validate the XML in control entity of the push submission
c. Initiate a result notification message.

The above mandatory features are to satisfy the minimum static conformance requirements specified in PAP. Operations support for push cancellation, status query, and client capabilities query, are optional. Some other optional features include multiple recipient support, delivery time constraints, QoS, and capabilities entity support in push messages. Most of these optional features are only meaningful for WAP Push. Therefore, it is not necessary to implement these features in a Web Client.

In addition to the PAP support, a web client should have two mandatory components: Application Dispatcher and Push User Agent. Application Dispatcher is responsible for forwarding the push content to the proper application according to application identifier specified in push message headers. Push User Agent (PUA) is the default application receiving common push messages. The contents pushed to PUA can be HTML or WML documents. PUA can use appropriate browsers to show the contents to the user. PUA may receive WAP Service Indication (SI) or Service Loading (SL) messages, both of which are XML documents. SI provides a standard way to sending notifications to end-users. SL allows PUA to load and execute a service on the web client.

4. A Web Client Implementation

The major idea of the uniform push service is the employment of WAP Push technology. To be compliant with WAP Push specifications, the uniform push framework keeps PIs, PPG, and WAP Clients unchanged. Therefore, the key to enable a uniform push service is how to easily provide web client functionality in end-users' computers. As known, most users use desktop PCs to access the Internet. In the following, we propose the architecture for a web client implementation in desktop PCs running Microsoft Windows 9x, NT, or 2000. Figure 3 illustrates the proposed architecture of a web client.

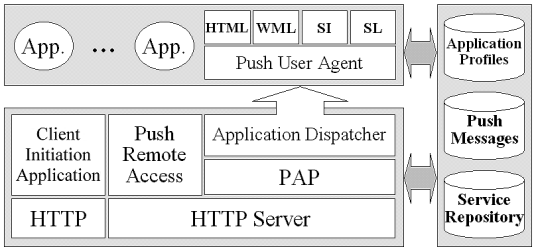

Figure 3. Web Client Architecture

In our implementation of a web client, the required run-time platform is composed of only three components: Microsoft Personal Web Server (PWS) or Internet Information Services (IIS), MS Access 2000, and Microsoft XML Parser (MSXML) 3.0 [19]. Most modules of the web client are developed using VBScript and Active Server Pages (ASP) [20] technologies. A Web-based user interface is developed. In the following, we will describe the functionality of the components respectively. How the components will be implemented is also discussed.

A. HTTP Server/HTTP

Since PAP is on top of HTTP and the web client should be often ready to receive push messages, we must use a HTTP server to provide the underlying communication service. PWS for Windows 9x or IIS for Windows NT/2000 can be used in our implementation. In addition, a web client may asynchronously notify a PI about the outcome of a previous push. The XMLHTTP object in MSXML 3.0 can support this feature.

B. Push Access Protocol (PAP)

Since both PWS and IIS support Active Server Pages (ASP), the PAP layer is entirely implemented in VBScript/ASP. The format of push messages from PIs can be multipart/related [21] or in pure XML. The PAP layer should first divide the received HTTP POST messages into entities. The control entity is transformed into an XML document object. The PAP layer will faithfully react according to the type of the PAP element carried in the control entity. If there is a

content entity, the PAP layer will directly forward to the Application Dispatcher. The application identifier specified in push message headers is also sent in company with the content entity. The capabilities entity is not implemented at present.

C. *Application Dispatcher (AD)*

Application Dispatcher (AD), implemented as an object in ASP, is responsible for forwarding push messages to proper applications. AD queries the application profiles stored in database to obtain the information about how to dispatch the push content. The information may include the execution path of the application, or the actions to be taken.

D. Push User Agent (PUA)

Push User Agent (PUA) is the default user agent in the client for receiving push messages. PIs can assume that each web client has a PUA to receive the push messages in standard formats, e.g. WML, HTML. In addition, PUA also receive Service Indication (SI) or Service Loading (SL) messages. As shown in Figure 3, PUA makes use of four modules to handle the push messages. In general, the HTML and WML modules are web and WML browsers [22], respectively. The necessary operations upon reception of a SI (SL) message are performed by the SI (SL) module. In general, a Uniform Resource Identifier (URI), indicating the address of a service, will be carried in a SI or SL message. SI and SL modules may use HTTP to further retrieve the service. The services may be kept in the service repository in database.

Although SI and SL are designed especially for WAP environments, their standard approach for service management is also applicable for desktop applications. Recently, for example, many users send and receive e-mails via web-based E-mail systems. The web-based mail server may use a SI to inform a user about the arrival of new e-mails. The user can directly retrieve the e-mails via the URI specified in the SI.

E. Push Remote Access (PRA)

Push Remote Access (PRA) allows the end-user to access push messages remotely through WWW. When a push message arrives, perhaps the end-user is not present in the front of the web client. A more common scenario is that a number of push messages are sent to the web client in the user's office but the user maybe is off duty at home. With the support of PRA, we can use a web browser in any other computer to remotely access push messages. The PRA is an ASP application in our implementation. It retrieves push messages stored in database, displays them in the web browser, and may accept requests to take actions upon the push messages. PRA is also the local user interface of the web client to browse push messages.

F. Client Initiation Application (CIA)

Client Initiation Application (CIA), adapted from the Session Initiation Application (SIA) defined in WAP, is developed to inform PIs that the web client is ready to receive push messages. CIA should be performed every time when a web client is turned on. CIA first retrieves the information about the previous subscribed push services from the service repository in database, and then sends a readiness message to each PI.

CIA is designed to overcome two possible drawbacks in practice. Firstly, it may happen that a web client is turned off or disabled. If a PI wants to push a message to a web client that is now unavailable, the PI will retry the transmission. PI may stop the retrials after a finite number of failures. These outstanding push messages will be buffered in the PI. If there is no any further pull action from the web client to the PI, the PI cannot know when the web client will be available. As a result, the only way to eventually achieve a successful push is continuous retrials until the web client is available. This may result in performance degradation in PIs and waste network traffics. The use of CIA can significantly improve PIs' performance and reduce communication overheads. Another drawback overcome by CIA is the IP mobility issue of web clients. The IP address of a web client may be changed dynamically due to the use of DHCP or dialup access. To be able to receive push messages, a web client should tell PIs its current IP address. This can be achieved by CIA. Currently, the communication from CIA to PI is not a standard approach. However, the communication is very simple. HTTP requests containing only authentication information for subscribed push services can offer the functionality required by CIA.

G. Database

Microsoft Access 2000 is used for the database management of the web client. Application profiles, push messages, and services information are the three categories of data stored in database. The modules of the web client access the database via Active Data Objects (ADO) [23]. In fact, any database supporting ODBC can be accessed by ADOs. Microsoft Access can be replaced for serious applications.

5. Conclusions and Future Work

In this paper, we have proposed a uniform push framework for WAP and WWW via the use of PAP in web push services. This approach extends the applicability of WAP PAP and utilizes the facilities provided in WAP Push services. In the new push framework, web clients are the key to enable uniform push services. We have proposed a possible architecture of a web client. A web client implementation on PCs is developed at present.

Although PAP runs on the wired Internet, there are a few of features designed particularly for handling the possible problems indirectly caused by the wireless communication in the WAP domain.

Indeed, these features are not necessary for web clients. In the future, we will propose an implementation profile with conformance statements for the use of PAP in a web client. In addition, the current push framework disallows the web clients located in intranets to receive push messages from the Internet. A possible solution is the placement of a web push proxy between web clients and PIs. The functionality of a web push proxy is to be studied.

Reference

1. "Wireless Application Protocol Architecture Specification", WAP Forum, 30-Apr-1998. URI: http://www.wapforum.org/
2. WAP Forum, http://www.wapforum.org/
3. "Wireless Application Environment Overview", WAP Forum, 29-March-2000. URL: http://www.wapforum.org/
4. "Wireless Application Environment Specification", WAP Forum, 19-February-2000. URL: http://www.wapforum.org/
5. "Wireless Markup Language Specification", WAP Forum, 04-Nov-1999. URI: http://www.wapforum.org/
6. "WAP Push Architectural Overview", WAP Forum, 08-Nov-1999 URL: http://www.wapforum.org/
7. Infogate, http://www.infogate.com/
8. Tie Liao, "WebCanal White Paper", http://webcanal.inria.fr/white/index.html
9. Trecordi, V.; Verticale, G., "An architecture for effective push/pull Web surfing," Proceedings of IEEE International Conference on Communications, Volume: 2 , 2000, pp.1159 -1163.
10. Kinoshita, S.; Shiroshita, T.; Nagata, T., "The realpush network: a new push-type content delivery system using reliable multicasting," IEEE Transactions on Consumer Electronics, Volume: 44 Issue: 4, Nov. 1998, pp. 1216 –1224.
11. Liao, T., "Global Information Broadcast: an architecture for Internet push channels," IEEE Internet Computing, Volume: 4, Issue: 4, July-Aug. 2000, pp.16-25.
12. "WAP Push Access Protocol Specification", WAP Forum, 08-Nov-1999. URI: http://www.wapforum.org/
13. "Extensible Markup Language (XML), W3C Recommendation 10-February-1998, REC-xml-19980210", T. Bray, et al, February 10, 1998. URL: http://www.w3.org/TR/1998/REC-xml-19980210
14. "WAP Push Proxy Gateway Specification", WAP Forum, 16-August-1999. URI: http://www.wapforum.org/
15. "WAP Push OTA Specification", WAP Forum, 08-Nov-1999. URI: http://www.wapforum.org/
16. "Wireless Session Protocol Specification", WAP Forum, Ltd., 05-Nov-1999. URI: http://www.wapforum.org/
17. "WAP Service Indication Specification", WAP Forum, 08-Nov-1999. URI: http://www.wapforum.org/
18. "WAP Service Loading Specification", WAP Forum, 08-Nov-1999. URI: http://www.wapforum.org/
19. Microsoft, "MSDN Online XML Developer Center," http://www.msdn.microsoft.com/xml/default.asp
20. Microsoft, "Active Server Pages Guide", http://www.msdn.microsoft.com/library/psdk/iisref/aspguide.htm
21. E. Levinson, "The MIME Multipart/related content type", August 1998. URI: http://www.ietf.org/rfc/rfc2387.txt.
22. MyWap.To, "MyWap 手機模擬器," http://mywap.to/html/ad/ad000809/brower.htm
23. Microsoft, "ADO Version 2.6," http://www.msdn.microsoft.com/library/psdk/dasdk/ados4piv.htm