Quick Implementation of a WAP Push Gateway

Yi-Ming Wen **Hsing Mei**

Web Computing Lab
Computer Science and Information Engineering Department
Fu Jen Catholic University
http://weco.csie.fju.edu.tw

Abstract

The WAP gateway is the key component standing between the mobile handset and web servers. Accessing Internet through WAP gateway using mobile phone becomes popular. However, lacking of the operation models and evaluation tools cause difficulties on analyzing the wireless applications and wireless gateway environment. This paper introduces our experience of a quick implementation of WAP Push Proxy Gateway (PPG).

We transformed a HTTP proxy into WAP gateway by modifying the differences between the two systems. The original HTTP proxy was designed under the object-oriented technology. The converted WAP gateway inherits the structure of HTTP proxy. To translate the traditional HTTP requests/replies into the coded forms, a mapping table is required. Our original HTTP proxy server is designed with coded HTTP header to enhance the internal operation speed. Hence, by replacing the internal operation code to WSP header code, the mapping table is implemented. Following the newly announced WAP push protocols, the right information will be delivered to the handset on time. To support the push capability, we analyze the related protocols, and propose a possible push operation sequence. To investigate the performance of the gateway, a WAP testbed is under development.

This research was supported in part by the National Science Council of Taiwan, ROC, under contract NSC88-2213-E-030-002.

1. Background

The number of mobile phone holders grows significantly in the past few years, and the trend will continue in the next couple years. This implies that mobile phones will be popular client system in the network. Facing such great demands, WAP(Wireless Application Protocol) forum proposed a series of protocols which permit Internet accessing through mobile phone [1].

There are two possible mechanisms to access Internet in the WAP environment: the first way is based on WSP(Wireless Session Protocol [2])/HTTP exchange via WAP gateway. The WAP gateway is the key component standing between the mobile handset and web servers. The alternative is directly access over pure WSP. Although many WAP protocols have been finalized, some WAP gateway related protocols are still in draft.

Instead of building a WAP gateway from scratch, we transformed a HTTP proxy into WAP gateway by modifying the differences between the two. The coded header and push operation of WAP are two major improvements over HTTP [3].

WAP use coded binary data to improve the transmission efficiency. The header and content are compactly compiled. For the traditional HTTP network, the packet header is in string format. In order to adapt to the WAP network, encode and decode technique are required. To translate the traditional HTTP requests/replies to the coded forms,

a mapping table is required. Coincidentally, our HTTP proxy server is designed with coded HTTP header to enhance the internal operation speed [4]. Hence, by replacing the internal operation code to WSP header code, the mapping table is implemented.

Push operations deliver the right information to the handset on time. To support the push capability, we analyze the related protocols, and propose a possible push operation sequence. To investigate the performance of the gateway, a WAP testbed is under development.

This paper addresses our experience on turning a HTTP proxy to a WAP gateway. Section 2 illustrates the operation sequences of a HTTP proxy and a WAP gateway. The converted gateway architecture is introduced. In Section 3, the push operation of a Push Proxy Gateway (PPG) is proposed. Our testbed structure is illustrated. Conclusion and future works are given in Section 4.

2. Gateway Structure

This section illustrates the structure of a WAP gateway. Firstly, we describe the operation of HTTP Proxy and WAP Gateway. Then, the functionality of each module of our WAP gateway is introduced. Finally, the tasks to convert a HTTP Proxy to WAP Gateway are discussed.

2.1 HTTP Proxy Operations

Accompany the rapid growth of Internet population, insufficient network bandwidth becomes a major problem. HTTP proxy is the most effective and widely adopted mechanisms to shorten the response time [5, 6].

Figure 1 shows the request/response sequence chain between a client and web sever through a HTTP proxy. The sequence is consisted of following six steps as the six labels in the figure.

1. User agent (i.e. client system) sends a request to the proxy, and asks a particular resource (e.g. a page).
2. User agent (i.e. client system) sends a request to the proxy, and asks a particular resource (e.g. a page).
3. After receiving and interpreting the request, proxy checks if the resource had been cached in the disk. It's called a Hit when the resource is found and fresh enough, then proxy sends a response to user agent. This effect of a cache is that the request/response chain is shortened. Otherwise, a Miss situation occurs, and the sequences continue to next step.
4. Proxy rewrites all or part of the request, and forwards the reformatted message toward the origin server.
5. The web server receives the request, and replies a suitable response.
6. After receiving the response sent by origin server, proxy would cache the message if it is a cacheable message.
7. At last, the response is replied to user agent.

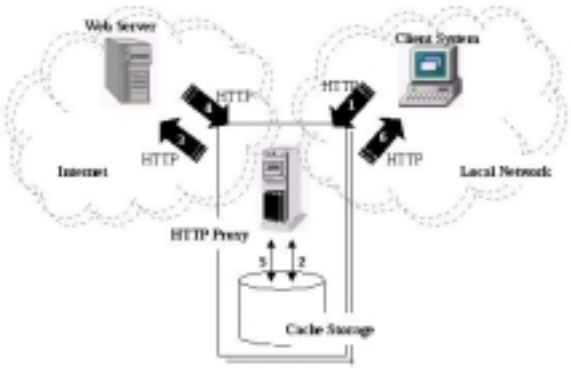

Figure 1. Traditional HTTP Proxy operation.

2.2 WAP Gateway Operations

In general, WAP Gateway is expected to complete three tasks: header translation, push operation, and content compiling. The header translation allows the client system to access Internet via different protocol. The push operation allows the server sending right information to client. The content compiling compacts the data for low

bandwidth support.

To turn a HTTP proxy to WAP gateway, header translation is the first task to be done. Although the other two tasks are not supported in HTTP Proxy, the push operation will be discussed in Section 3. As to the content compiling, it is not a concerned point in this paper.

In WAP, the WSP headers are defined in binary format; but the HTTP headers are in string type. In order to enable handset to access the IP network, translation of the coded WSP and traditional HTTP requests/replies is needed.

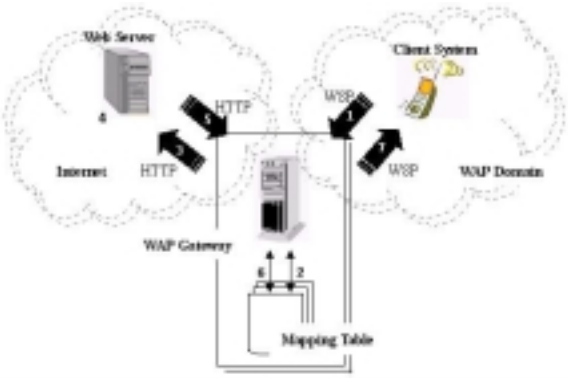

Figure 2. WAP Gateway operation sequence.

The operation sequence chain between a mobile phone and web server through a WAP gateway is shown in Figure 2, and the six steps are shown as labels in the figure.

1. User agent (i.e. client system) sends a URL request to a WAP gateway following the WSP protocol.
2. The WAP gateway decodes the request massage and translates the request line and request header (in binary-format) to HTTP format by a mapping table.
3. The WAP gateway creates a connection to the web server and sends a HTTP request to it.
4. The HTTP request is processed by the web server. If the URL refers to a static file, web server fetches the file to response with a HTTP reply header; if the URL refers to a script application, web server starts the application.
5. The web server returns a HTTP reply message, which contains data or result from script application.
6. The WAP gateway encodes the received reply message, and translates HTTP well-known-formatted reply line and reply header to WSP binary-format using the mapping table.
7. The WAP gateway creates a connection, and a WSP response containing the WML [7] is returned to the client system.

Notice that the WAP client side environment is different from that of traditional HTTP Proxy. The network domain of client system in Figure 2 had been changed to WAP domain. And the protocol used between client and gateway is WSP now. Besides, the caching is an optional operation in the WAP Gateway.

2.3 WAP Gateway Implementation

This subsection introduces the design and implementation of our WAP Gateway. We have finished the header translation part, and the push operation part is designed, and to be implemented. Thus, we focus the description on the header translation here. Figure 3 shows the our WAP Gateway system architecture.

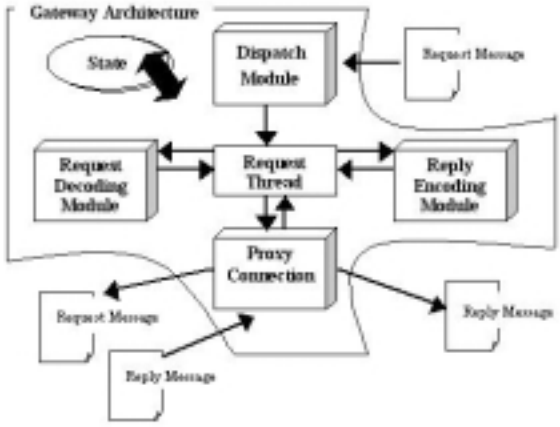

Figure 3. WAP gateway architecture.

The architecture is composed of six modules :

Dispatch , State, Request thread, Request Decoding, Proxy Connection, and Reply Encoding module. The functionality of each module is illustrated as following.

Dispatch Module:

When a client sends a request to our WAP gateway, the gateway will establish a socket connection, and put the socket into a Listen Queue. The dispatcher itself is a thread. Dispatcher keeps a Server Socket, which can get the socket connection. Then, dispatcher will forward the request by calling RunSet method to record request's state and start request thread.

We set the dispatcher with low priority. It means that CPU would spend much more time on handling requests than dispatching them. Since the action of handling requests can consume numbers of input request but not dispatching, our gateway still runs smooth under busy condition.

State Module:

State module is implemented as an object-oriented class. The state class binds the resources with the request. It puts the data-gram packet into a PushbackInputStream object from standard JDK. State module also handles the Request Thread, Request Line, Request Header, Proxy Connection, Reply Line and Reply Header for client request task.

Request Thread Module:

The major part of the Request Thread is a run method. This method accomplishes all tasks starting from receiving client request, and up to responding reply message.

The six tasks must do in thread request module:

1. Parse the request line and decode binary format to string format fitting for HTTP.
2. Parse the request header and decode binary format to well-known-header string format fitting for HTTP.
3. WAP Gateway creates a connection and sends a HTTP request translating from WSP request to the web server which client original request. Then receiving the HTTP reply message.
4. Parse the reply line and encode string format to binary format fitting for WSP.
5. Parse the reply header and encode well-known-header string format to binary format fitting the WSP.
6. WAP Gateway sends a WSP reply message to the client.

Request Decoding Module:

Request message decoding module includes two parts: request line decoding and request header decoding. Request line parser gets the TID (Transaction ID), PDU type, and URI of the request packet. Then it translates the PDU type to a string format by using a defined mapping table.

Request header parser gets the well-known-header values and translates those to the string type header's names by a defined mapping table. According to those values, parser calls the corresponding method to get string type header's values.

The mapping table mentioned above can be found in WSP Specification.

Proxy Connection Module:

After WAP Gateway has decoded a request message, it must forward this request to the original web server that holds the content. Thus, WAP Gateway connects the web server using this Proxy Connection module. This module firstly gets the resource processed and bound in the State module. Then it connects to the web server. Finally, WAP Gateway rewrites a HTTP request and sends to the Internet.

Another task of Proxy Connection module is writing response message to the client. When WAP

Gateway receives the reply message from the web server, WAP Gateway will encode the message for WSP adapting. Then, the Proxy Connection module will write this reply message to the session created by WAP Gateway and client to finish the primal request.

Reply Encoding Module:

Reply message encoding module includes two parts: response status line encoding and response header encoding. Response status line parser gets the Version, Response Status Code, and Reason of the reply packet. Then it encodes these information to a binary format by using a defined mapping table.

Response header parser gets the string type header's names and translates those to the well-known-header values by a defined mapping table. According to these values, parser calls the corresponding method to get binary format header's values.

The mapping table mentioned above can be found in WAP WSP Specification.

Example:

The example below illustrates the translation of headers.

WSP coded data:

```
01 40 25 68 74 74 70 3a 2f 2f 77 65 63 6f 2e
63 73 69 65 2e 66 6a 75 2e 65 64 75 2e 74 77
2f 69 6e 64 65 78 2e 77 6d 6c 81 ea 81 84 80
94 80 95 80 a1 80 9d a9 4e 6f 6b 69 61 2d 57
41 50 2d 54 6f 6f 6c 6b 69 74 2f 31 2e 33 62
65 74 61 00
```

Decoded HTTP data:

```
TID=1
Type=GET
URL=http://weco.csie.fju.edu.tw/index.wml
Accept-Charset:Somali ,Amharic
Accept:application/vnd.wap.wmlc,
       application/vnd.wap.wmlscriptc,
       image/vnd.wap.wbmp, image/gif
User-Agent:Nokia-WAP-Toolkit/1.3beta
```

WSP code	WSP Value (Hex.)	Filed Name	HTTP Value
01	0x01	Transaction ID	1
40	0x40	PDU Type	GET
25	0x25	URI Length	37
(Following 37 bytes) 68 74 74 70 3a 2f 2f 77 65 63 6f 2e 63 73 69 65 2e 66 6a 75 2e 65 64 75 2e 74 77 2f 69 6e 64 65 78 2e 77 6d 6c		URI	http://weco.csie.fju.edu.tw/index.html
81	0x01	Header Name	Accpet-Charset
ea	0x6A	Header Value	Somail
81	0x01	Header Name	Accpet-Charset
84	0x04	Header Value	Amharic
80	0x00	Header Name	Accept
94	0x14	Header Value	application/vnd.wap.wmlc
80	0x00	Header Name	Accept
95	0x15	Header Value	application/vnd.wap.wmlscriptc
80	0x00	Header Name	Accept
a1	0x21	Header Value	image/vnd.wap.wbmp
80	0x00	Header Name	Accept
9d	0x1D	Header Value	image/gif
a9	0x19	Header Name	User-Agent
4e 6f 6b 69 61 2d 57 41 50 2d 54 6f 6f 6c 6b 69 74 2f 31 2e 33 62 65 74 61 00		Header Value	Nokia-WAP-Toolkit/1.3beta.

For this example, a WSP coded header is sent to the WAP Gateway. Then Request Decode Module parses the data and translates them to a http value by using a mapping table. The mapping tables refer to WAP WSP Specification.

2.4 Test Environment

This subsection explains the current testing environment of our WAP Gateway. This testing environment is part of the WAP testbed.

In the client system, we use the Nokia WAP

Toolkit 1.3beta [8]. This Toolkit simulates a mobile phone, and it supports WAP content development.

The Toolkit can send a request through two different paths: HTTP over TCP, and WSP over UDP. When using HTTP, the Toolkit connects directly to the web server as traditional web connection. When using WSP via GSM platform, the Toolkit is not a real phone set, thus UDP is chosen to simulate the WAP connection.

In the server side, we use MS-IIS 4.0 web server. We put WML/WML-Script contents on the server, and set MIME type to let web server aware of the content type. WML/WML-Script contents can be built using the Nokia WAP Toolkit or a pure text editor. The MIME type about WAP content type must follow WAP WSP Specification. The complete WAP testbed structure is given in Section 3.2.

2.5 Convert HTTP Proxy to WAP Gateway

A HTTP Proxy could to transfer to a WAP Gateway by adding some mechanisms as Figure 4.

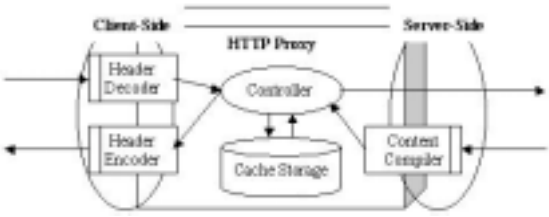

Figure 4. HTTP Proxy working with WAP

In the client side, a header decoder and a header encoder for protocol translation are attached. In the server side, we add a content complier for content compacting. When a WSP request is sent to a HTTP proxy, the Header Decoder converts the request to a normal HTTP one. Then Proxy do its regular work. After replying back a HTTP response, the Content Compiler compiles WML/WML-Script decks or cards. Before replying to client, the Header Encoder translates a normal HTTP reply to a WSP reply.

3. Push Operation and WAP Testbed

This section proposes a possible push operation sequence and illustrates our testbed structure in order to examine the performance of the gateway.

3.1 Push Operation

A push operation in WAP occurs when a Push Initiator transmits content to a mobile client[9]. However, push operation is not directly between Push Initiator and mobile client. The Push Initiator is on the Internet, and the mobile client is in the WAP domain. Thus, the Push Proxy Gateway (PPG) is needed[10].

The Internet-side PPG access protocol is called Push Access Protocol (PAP) which is on top of HTTP. The WAP-side OTA delivery protocol on top of WSP is called Push Over-The-Air Protocol (Push OTA Protocol). These two protocols define the required elements of push operation. The push operation sequences are as Figure 5:

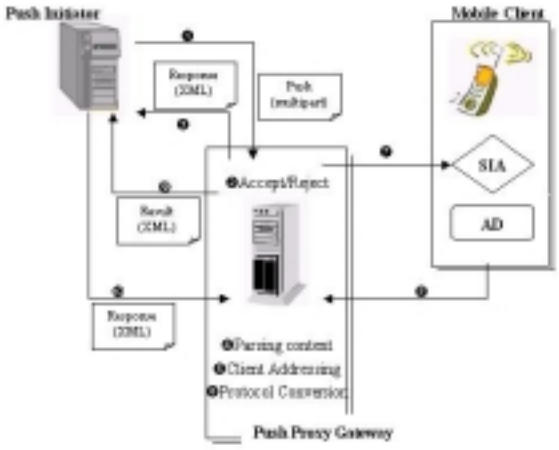

Figure 5. Push operation with PPG.

1. A Push Initiator submits a push message to a PPG. The push message is a multipart type, which contains a XML control entity and a content entity, and may contain a RDF-format

capabilities entity defined in the user agent profile.

2. The PPG determines to accept or to reject the received push message. If the PAP push message element is not valid with respect to its document type definition (DTD), the PPG must rejects it.
3. The PPG reports the acceptance/rejection result in the response which is an XML document.
4. The PPG parses control information of push message to determine where and how should deliver.
5. A PI must identify client addresses for submitting. A client address is composed of a client specifier and a PPG specifier. The client specifier is used to find out a target client, and the content is either User-defined identifiers (e.g. E-mail address) or Device address (e.g. phone number). The PPG specifier permits a push message to be routed through proxy, and the content is a site of PPG. In this step, the PPG parses the client addresses for delivering.
6. The PPG converts the protocols between the Internet and WAP domain.
7. If there is no error, the PPG must deliver either confirmed or unconfirmed push primitives. In a confirmed push, an active WSP session which only the client can create is required. This situation is also called connection-oriented push. If PPG gets a connection-oriented push message to a client, and there are no active sessions to that client, PPG cannot deliver the push message. In order to solve this problem, PPG sends a request, which contains necessary information for creating a session, to a Session Initiation Application (SIA) in the client side. Let SIA determines either establishing a session or not. Of course, Push delivery may also be performed without the use of sessions in a unconfirmed push.
8. When a client receives a pushed message, Application Dispatcher (AD) in the client parses the push message header to determine its destination application and to run it. If the Push delivery method is confirmed push, the client responses a primitive to acknowledge receiving the PPG's push content.
9. The PPG sends a Result Notification to inform the PI of the outcome of the push submission. This notification reports the push message was sent, delivered, expired, cancelled, or occurred an error.
10. The PI returns a response message to the Result Notification message. Above two message are XML documents.

Above push operation sequences are all required operations. It omits the Optional operation part from WAP Push specification.

3.2 WAP Testbed

In order to investigate the performance of our WAP gateway, a WAP testbed is under development. The testbed structure is illustrated as Figure 6.

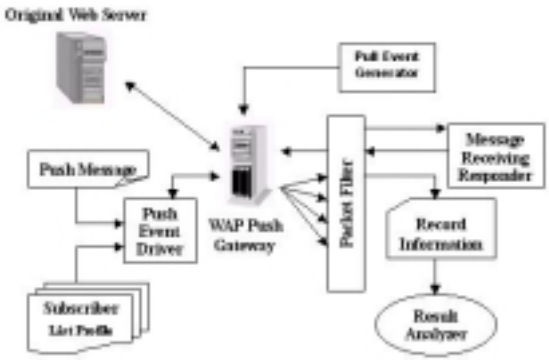

Figure 6. WAP Testbed

There are two major behavior in WAP Testbed: pull and push. In pull condition, the Pull Event Generator simulates a great deal of clients and sends requests to gateway. Then gateway does its regular work and responses packets to the Packet Filter. The Packet Filter records the information about packets such as amount. In push condition, the Push Event Driver starts scheduled push delivery with push messages according to subscriber list.

After parsing push messages by gateway, the packets are sent to the Packet Filter. The Packet Filter classifies these packets and records the related information respectively. Finally, according to the recorded information, the statistic analysis will be able to do.

4. Conclusion and Future Work

In this paper, we firstly introduce the replacing internal operation code of HTTP proxy server to WSP header code and construct a WAP Gateway. Base on the replacing, quickly turning a HTTP Proxy Server to WAP Gateway by adding extra mechanisms is feasible. Next, a possible push operation sequence is proposed. Finally, the developing WAP Testbed is illustrated.

Now our WAP Gateway has the ability of protocol translation, but lacks of the management and efficiency evaluation modules. Besides, the push operation part is under construction. For future work, we will (1) continue implementing the functionality of our WAP Push Gateway; (2) complete the WAP Testbed and investigate the gateway performance; (3) estimate the rationality of the proxy caching operation within WAP Gateway.

Reference

[1] WAP Forum, "Wireless Application Protocol Architecture Specification", April 1998. http://www.wapforum.org/

[2] WAP Forum, "WAP Wireless Session Protocol Specification", May 1999. http://www.wapforum.org/

[3] H. Mei, and Y.M. Wen, "Turning a HTTP Proxy to WAP Gateway", *Proceedings of INET 2000*, July 2000 (Accepted).

[4] W. J. Lin, and H. Mei, "A High Performance Java-based HTTP Proxy Server", *Proceedings of 1999 Workshop on Distributed System Technologies & Applications*, May 1999, pp. 654-663.

[5] R. Fielding, J. Gettys, J. Mogul, H. Frystyk and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997.

[6] Paul S. Hethmon, "Illustrated Guide to HTTP", Manning Publications Co, 1997.

[7] WAP Forum, "WAP Wireless Markup Language Specification", November 1999. http://www.wapforum.org/

[8] Nokia WAP Toolkit SDK 1.3 http://www.forum.nokia.com/

[9] WAP Forum, "WAP Push Architecture Overview", November 1999. http://www.wapforum.org/

[10] WAP Forum, "WAP Push Proxy Gateway Service Specification", August 1999. http://www.wapforum.org/