Dynamic Markup Language Conversion

for Improved WAP Network Architecture and Performance

by

Kareem A. H. Akhtar

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of Bachelor of Science in

Computer Science and Master of Engineering in Computer Science

at the Massachusetts Institute of Technology

September 22, 2000

Author ______________________________
Department of Electrical Engineering and Computer Science
September 22, 2000

Certified by ______________________________
Kai-Yeung Siu
Thesis Supervisor

Accepted by ______________________________
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Dynamic Markup Language Conversion for Improved WAP Network Architecture and Performance

by

Kareem A. H. Akhtar

Submitted to the
Department of Electrical Engineering and Computer Science on September 22, 2000, in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science

ABSTRACT:

The wireless web introduces new flexibility for World Wide Web users and service providers. Users can browse the web with a WAP (Wireless Application Protocol) enabled device such as a cellular phone or a PDA (Personal Digital Assistant)., while service providers can make their services available anytime and anywhere by exploiting the wireless features of the protocol. Moreover, service providers can also make their services available at all times with the proliferation of the internet enabled handheld devices. However, although WAP promises to provide anytime, anywhere flexibility, there are fundamental problems with the new technology facing service providers and users. One significant issue is the transition to the wireless web. For the user, WAP is too slow and browsing is not efficient. Moreover, for the service provider and internet businesses, transitioning their websites to the wireless domain is not very efficient either, for WAP introduces a new markup language, and has various limitations. In this thesis, develop various applications for improving the wireless experience for service providers and users. I introduce a solution for dynamic, on-the-fly conversion of a website for improved network performance, a tool for converting and managing a complete website to the wireless domain, and finally, a method to exploit caching of developed wireless sites also for improved network performance.

Thesis Supervisor: Kai-Yeung Siu
Title: Associate Professor

ACKNOWLEDGEMENTS

I would like to thank my supervisor Kai-Yeung Sunny Siu for his wholehearted support and guidance throughout. Professor Siu allowed me generous freedom in defining and developing my thesis, for which I am especially grateful.

Moreover, this thesis was developed and written under the auspices of the MIT Auto-ID Center and its members; for whose support I am sincerely appreciative.

Finally, I would like to thank my family for their unconditional love and support. Thanks to Poppy, Mommy, Omar, Senwan, and Ali.

TABLE OF CONTENTS

CHAPTER 1[1]

Overview

1.1 Motivation

The wireless web introduces new flexibility for World Wide Web users and service providers. Users can browse the web with a WAP (Wireless Application Protocol) enabled device such as a cellular phone or a PDA (Personal Digital Assistant)., while service providers can make their services available anytime and anywhere by exploiting the wireless features of the protocol.

In the past few years, industry has witnessed the convergence of two rapidly evolving technologies, wireless data and the Internet. Not only have these two technologies grown considerably, but also the exponential growth of the Internet has influenced the creation of new and exciting information services that leverage these two technologies. WAP is the latest in these technologies and represents the convergence of wireless data and the Internet. Moreover, it is designed specifically to adapt the current network technologies specifically for wireless data services for mass marketing via the handheld device. The Internet finally has a way of reaching everyone, anytime and anywhere.

WAP provides many value-added improvements over the current mobile network to make the customer experience more useable and attractive. WAP technology can provide a customized user interface to enhance feature such as call control and call forwarding. This

[1] The images in this thesis are obtained from references [1] and [2]. I have modified some images slightly. The original images **are all © Copyright Wireless Application Protocol Forum, Ltd, 1998.**

user interface will prompt the user to decide whether to accept calls or forward them to other locations such as voicemail or to another person.

With the new improvements offered by WAP, increased usage of the mobile network has been predicted to increase. However, although the WAP enabled phone has been in industry for a while, cellular phone users still prefer to use their devices strictly for voice calls.

However, although WAP promises to provide anytime, anywhere flexibility, there are fundamental problems with the new technology facing service providers and users. One significant issue is the transition to the wireless web. For the user, WAP is too slow and browsing is not efficient. Moreover, for the service provider and Internet businesses, transitioning their websites to the wireless domain is not very efficient either, for WAP introduces a new markup language that has various limitations. Since most of the technology developed for the Internet has been designed for larger, powerful desktops, and for reliable network technologies that have the advantage of high bandwidth, handheld wireless devices, and the network technologies in which they are connected, present a more constrained computing environment.

In the wireless domain, there are limitations in power and form-factor. Wireless devices have small displays, limited power consumption, inconvenient input devices, low memory, and less powerful processors. Moreover, wireless data networks have high latency, less bandwidth, are unpredictable, and have less connection stability.

There are many ways to improve upon the limitations in the network architecture of WAP. Among them, are providing a means for speedy conversion from HTML to WML, methods for caching wireless sites at aggregate points of the wireless network, and also managerial tools to help service providers transition their websites.

1.2 Research Contribution

In this thesis, I explore methods to improve wireless network performance, by providing conversion and caching solutions among the involved markup languages.

I have developed intelligent software agents to facilitate the transition to the wireless web. I also adapted the software technology to implement on-the-fly server-based conversion from HTML to WML, and to implement caching of converted wireless sites to be stored and handled within a WAP gateway. I also adapted the software technology for use as a simple software tool for easy conversion and management of a website.

In Chapter 2, I will provide a deeper look at WAP and many of its features. Furthermore, I will discuss several advantages and disadvantages to the protocol in several capacties. Finally, I will show where improvements can be made to the protocol and to the wireless devices themselves.

In Chapter 3, I will present my solutions for dynamic conversion of websites, a method for caching for relieving bandwidth constraint problems, and a conversion and management tool.

Finally, in Chapter 4, I will present conclusions and a discussion for future research.

CHAPTER 2[2]

WAP Architecture

2.1 Architecture Overview

2.1.1 Scope and Purpose

The Wireless Application Protocol was designed as a cross-platform, cross-device standard for developing services and applications over a wireless network. It specifies an environment for developing services for PDAs, pagers, mobile phones, and other wireless handheld devices. To achieve this, WAP has leveraged the existing network technologies in both the mobile and the Internet space. For example, in the mobile space, WAP has extended digital networking standards in that WAP technologies can run over different bearers. Furthermore, WAP has extended current Internet technologies by leveraging various content formats, scripting, the Extensible Markup Language (XML), and Uniform Resource Locators (URLs). For example, WML (Wireless Markup Language), the markup language used in WAP, is a derivative of XML. Essentially, by leveraging existing network technologies, WAP enables content developers, service providers, and manufacturers to build innovative services and new implementations in an intuitive manner for the wireless space.

In bringing Internet content and improved information services to digital cellular phones and other wireless terminals, WAP has defined several objectives. A central goal to WAP is to become a cross platform protocol specification in that it works across varying network

[2] Chapter 2 utilizes information from both references [1] and [2]. The original information in the references **are all © Copyright Wireless Application Protocol Forum, Ltd, 1998.**

technologies. In this capacity, it encourages the creation of content and applications that scale across a very wide range of bearer networks and device types.

WAP also has several provisions. From a network architecture perspective, WAP seeks to provide secure access to applications and communication, optimize for high latency narrow-band bearers, provide support for a plethora of wireless networks, assist network operator and third party provisioning, and define a scaleable and extensible architecture which mimics the layered architecture of the existing web. From the point of view of the handheld device and interoperability, WAP provides for the efficient use of device resources, support multi-vendor interoperability, and provide access to local handset functionality, such as logical indication for incoming call. Finally, WAP seeks to provide a software platform for integration of telephony and data services.

2.1.2 The World Wide Web Model

As it exists today, the World Wide Web (WWW) provides a very extensible and scaleable model. Applications and content are presented in standard data formats such as HTML and XML, among others. To access information in the WWW, web browsers browse these applications and interpret the content from the standard data format to a rendering familiar to a user. A browser is able to perform its functions by sending requests to a server containing information in the standard data formats, and getting a response from the server with the requested information.

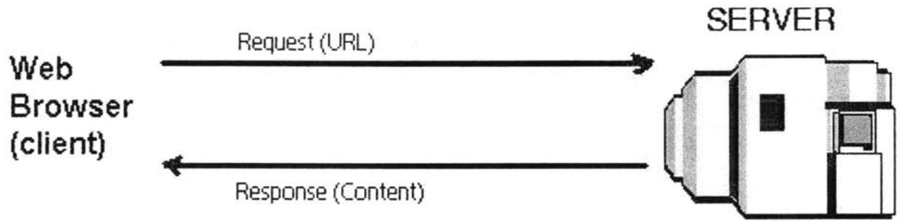

Figure1: World Wide Web Progmaming Model

The following definitions are obtained from the WAP Forum's specifications in [1] and [2].

The WWW specifies a powerful infrastructure that allows users to create services and applications to be made available to large numbers of other users. The following components of the WWW standards are defined to construct the environment for the Web as it exists today:

URLs – Essentially, this specifies a standard naming model for all servers and content. Please refer to RFC 1738 and 1808 for more information.

Content typing – This specifies a standard for WWW content. Therefore, based on its type, content can be correctly processed by the browser. Please refer to RFC 2045 and 2048 for more information.

Standard content formats – These include HTML (Hypertext Markup Language), the JavaScript scripting language, XML, and many others. It is the web browsers that render these content standards.

Standard Networking Protocols - These allow web browsers to communicate with web servers. HTTP (HyperText Transport Protocol) is the most commonly used protocol on the WWW. Please refer to RFC 2068 for more information.

The WWW protocols define three classes of servers:

Origin server – Created content resides on this server.

Proxy – This server exists between a client application and a real server. As an intermediary program acting both as a client and a server, it makes requests on behalf of other clients. The proxy program intercepts the requests to handle them. If it cannot handle the request, it forwards it along.

Gateway – This is a server that essentially links two different types of networks. Moreover, it can act like an origin server when handling requests.

2.1.3 The WAP Model

Understanding the WWW model helps one to understand the simple extension to the WAP programming model in that the two are quite similar (see Figure 2). As one can see, the

WAP programming model leverages many aspects of the WWW model in that it relies on the fact that the WWW model is a proven environment, it contains familiar components including gateways, origin servers, XML standards, and uses the same request/response mechanism for communication. This familiarity makes it easy for the programmer, application developer, and network designer to understand and extend services over this new wireless network. The key differences lie in the fact that certain optimizations were made in order to account for the wireless environment. For example, WML is designed to be lightweight when compared to XML.

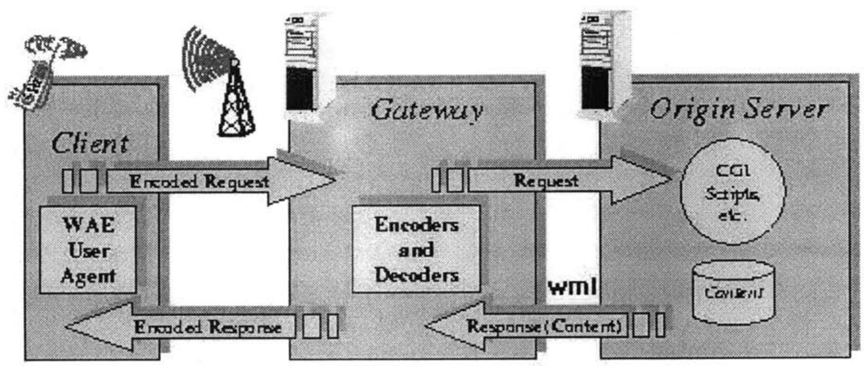

Figure2: WAP Logical Model

The WWW model heavily influences the WAP programming model with regards to content formatting and content transportation. WAP model content and application programming is actually based on the WWW content formats. Moreover, in a similar fashion to WWW content, WAP content is transported between servers, except that when WAP content travels in the air, a special WAP gateway first translates the content into a binary form to be sent over the air.

In the WWW model, a web browser is able to render the standard content formats. In the WAP model, on the other hand, a microbrowser interprets the standard content to be rendered onto the wireless device's user interface.

Much like the WWW model specifies standard components, so does the WAP model in order to coordinate communication among network servers, mobile terminals, and other components:

The following definitions are obtained from the WAP Forum's specifications in [1] and [2].

Standard naming model - WWW-standard URLs are used to identify WAP content on origin servers.

WWW-standard URIs are used to identify local resources in a device, eg call control functions.

Content typing - All WAP content is given a specific type consistent with WWW typing. This allows WAP user agents to correctly process the content based on its type.

Standard content formats - WAP content formats are based on WWW technology and include display markup, calendar information, electronic business card objects, images and scripting language.

Standard communication protocols - WAP communication protocols enable the communication of browser requests from the mobile terminal to the network web server. The WAP content types and protocols have been optimized for mass market, hand-held wireless devices. WAP utilizes proxy technology to connect between the wireless domain and the WWW. The WAP proxy typically is comprised of the following functionality:

Protocol Gateway - The protocol gateway translates requests from the WAP protocol stack (WSP, WTP, WTLS, and WDP) to the WWW protocol stack (HTTP and TCP/IP).

Content Encoders and Decoders - The content encoders translate WAP content into compact encoded formats to reduce the size of data over the network.

The WAP proxy - Allows content and applications to be hosted on standard WWW servers and to be developed using proven WWW technologies such as CGI scripting. While the nominal use of WAP will include a web server, WAP proxy and WAP client, the WAP architecture can quite easily support other configurations. It is possible to create an origin server that includes the WAP proxy functionality. Such a server might be used to facilitate end-to-end security solutions, or applications that require better access control or a guarantee of responsiveness, eg, WTA (Wireless Telephony Application).

2.1.4 Example WAP Network

The following is an example WAP network.

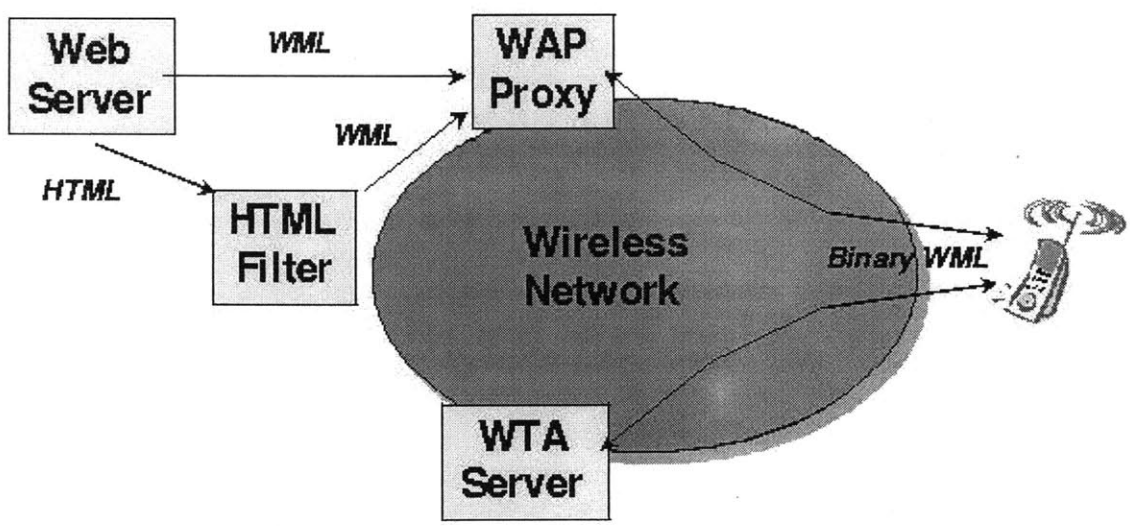

Figure3: Example WAP Network

In Figure 3, a WAP client maintains a wireless connection to a WAP proxy and a WTA server. The WTA server provides the WAP client with an interface to the existing telecommunications infrastructure. This will be explained in brief in a later section.

The function of the WAP proxy is to translate the binary, wireless WAP content into a readable WML content to be forwarded to the requested web server, if any. Moreover, it encodes readable WML from a web server into a binary form understood by the WAP client.

The function of the HTML filter is to translate HTML to WML; however, this task is not a very reliable one. Moreover, this aspect to the diagram is rather misleading in that there is no current perfect translator. I introduce a limited converter later in the thesis. However, I place it as a management tool to be used manually, or directly within the WAP proxy for limited on-the-fly conversion for potentially improved network performance.

2.1.5 Security Model

WAP's security model can provide for end-to-end security if the WAP proxy is trusted. The reason is that all communication in WAP is dependent on this intermediary.

2.1.6 WAP Telephony

The WTA environment is the WAP device's extension for telephony services. The WTA user agent extends the WAE user agent through the use of WML. Therefore, while a typical cellular phone can set up and receive calls, WML adds decision making on the device for call control and forwarding. This is an example of how WAP can enhance the existing telephony interface.

In this thesis, the focus is on the WAE side, and not on the WTA side, therefore, I do not provide significant detail about WAP Telephony. Below are two examples of WTA; the first shows how WTA fits into the scheme of WAP, and the second shows an example of a call flow.

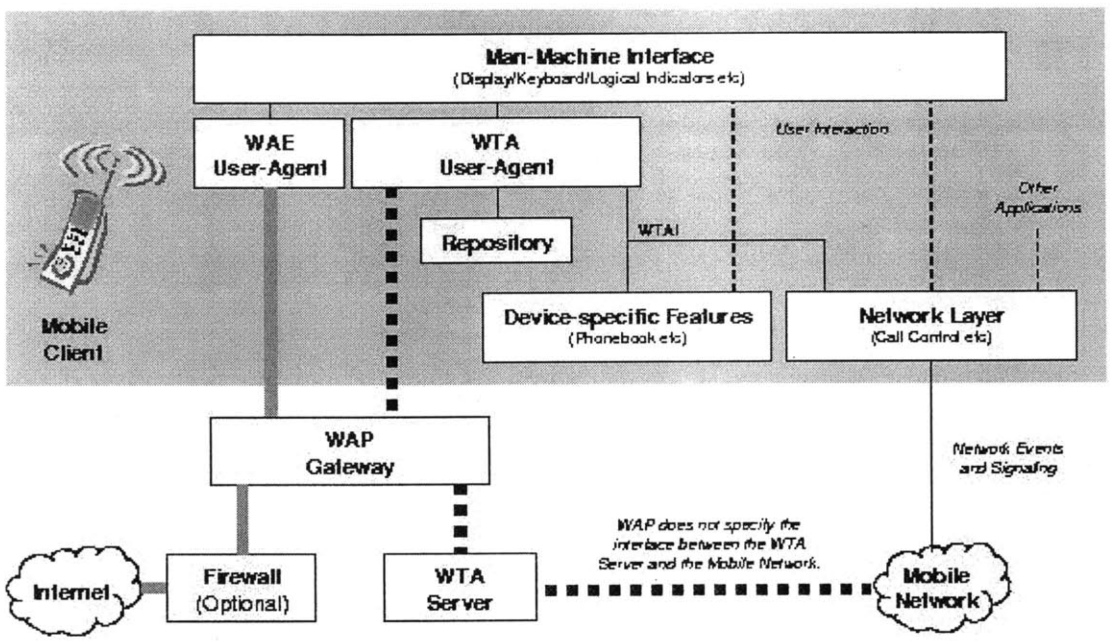

Figure4: WTA architecture

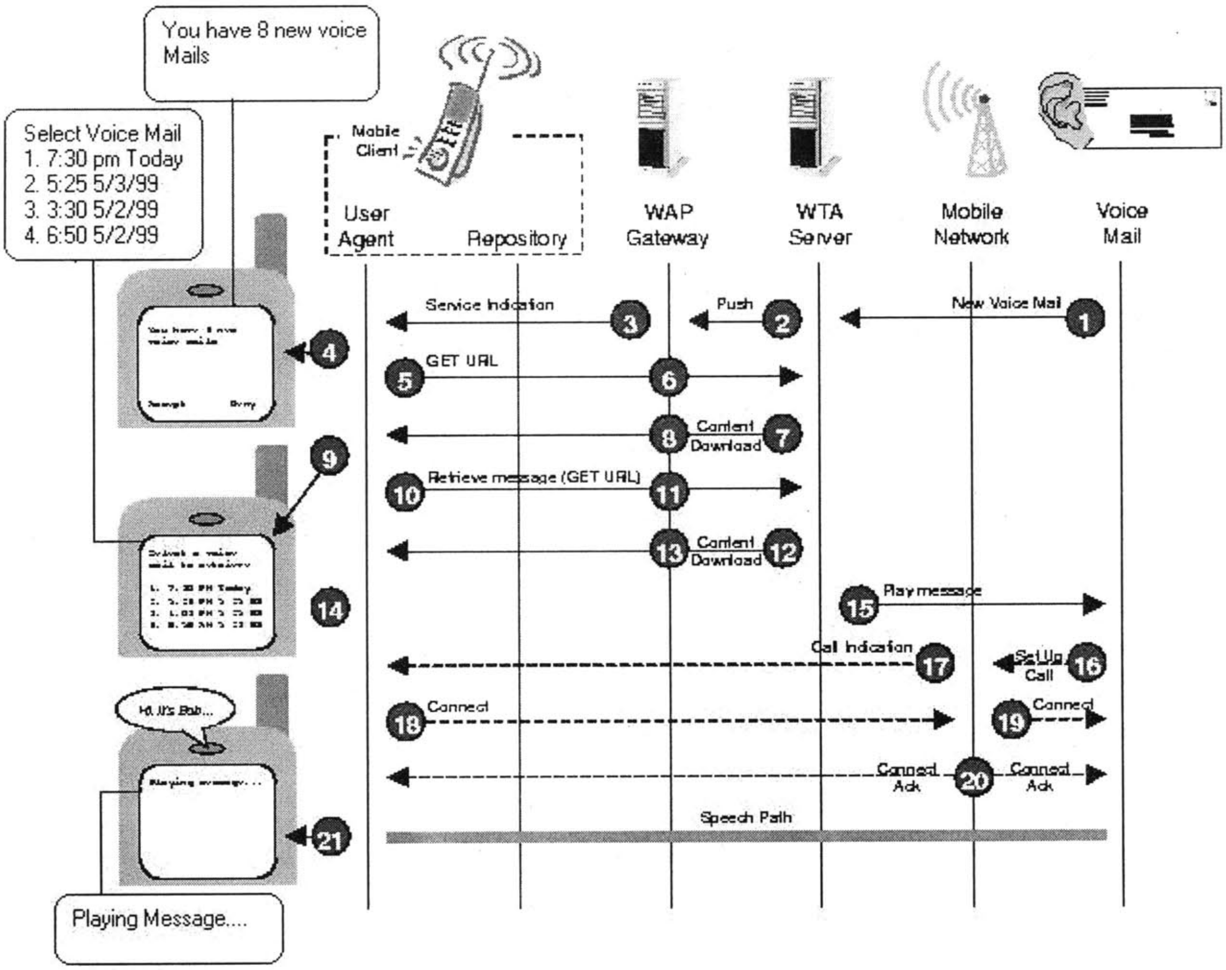

Figure 5: WTA example call flow

2.2 Components of the WAP Architecture

2.2.1 WAP Protocol Stack

The current WWW architecture is based upon the 7 layer protocol stack, known as the OSI model. These layers are: Application, Presentation, Session, Transport, Network, Data-Link, and Physical.

Not surprisingly, the WAP architecture is based upon a similar protocol stack, but optimized for the wireless environment. Like the WWW architecture, an optimized protocol stack allows the WAP architecture the ability to scale better and to be extensible. (see Figure 6).

Moreover, through a set of well-defined interfaces, WAP services and applications can take advantage of the features of the WAP protocol stack. In fact, applications can access the layers directly. These layers and interactions are described in detail in the next section.

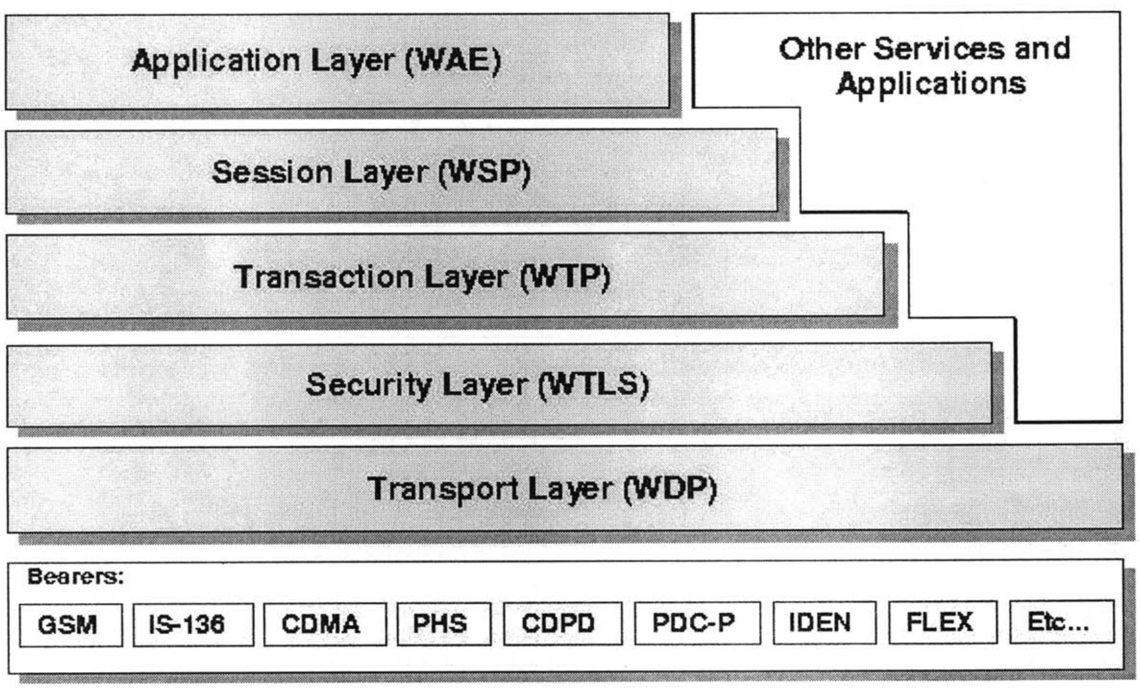

Figure6: WAP Protocol Stack

2.2.1.1 Wireless Application Environment (WAE)

This layer describes the overall WAP application environment for browsing the wireless web. It fuses both mobile telephony technology and the WWW. In this interoperable, environment, service providers, content managers, and operators can create services and applications for various wireless platforms. This layer is analogous to the application layer in the WWW OSI model. In the WWW model, the browser describes the application environment, whereas in this case, WAE supports a microbrowser to browse the wireless web and interpret phone functionality. Specifically, the microbrowser environment has support for WML, WMLScript, WTA, and Content Format support.

The following definitions are obtained from the WAP Forum's specifications in [1] and [2].

Wireless Markup Language (WML) - a lightweight markup language, similar to HTML, but optimized for use in hand-held mobile terminals.

WMLScript - a lightweight scripting language.

Wireless Telephony Application (WTA, WTAI) - telephony services and programming interfaces.

Content Formats - a set of well-defined data formats, including images, phone book records and calendar information.

2.2.1.1.1 WML (Wireless Markup Language)

Like XML, WML is a document language that is tag-based. In fact, WML is an XML document type. However, it differs in several ways, mainly in that it is particularly optimized for user interaction on limited capability devices, such as cellular phones. Furthermore, WML implements a card and deck metaphor with regards to its structure and flow in the WAE. Essentially, a user navigates through different screens on his WAP enabled device. These screens are considered cards. Several cards form a single deck, and a single deck is considered a single XML document. Several of these decks are stored on the origin server, and downloaded by the handheld device's microbrowser.

2.2.1.2 Wireless Session Protocol (WSP)

The Wireless Session Protocol can be characterized in either connectionless or connection modes.

Connectionless Mode:

The WSP/B (WSP/binary) is a stateless, binary protocol. Patterned after the HTTP World Wide Web protocol, it consists of a simple request-response pairing. No state information is maintained between requests. Moreover, the WSP uses the WAP Datagram protocol to communicate with WAP clients.

Connection Mode:

The WAP Session Protocol, WSP, is a session oriented, stateful binary protocol used in conjunction with the transaction layer below it. A superset of WSP/B, WSP defines additional protocol formats to support session initiation, suspension, and resumption, and to maintain session state information. Until explicitly disconnected, a session is started by a WAP client and is maintained throughout a connection. The transaction layer below the WSP layer exchanges WSP information.

2.2.1.3 Wireless Transaction Protocol (WTP)

WTP is the transaction protocol for exchanging WSP information. It is lightweight and can operate over secure or non-secure wireless networks. It supports three types of class differentiated services as described below.

The following definitions are obtained from the WAP Forum's specifications in [1] and [2]:

Three classes of transaction service:

-Unreliable one-way requests

-Reliable one-way requests

-Reliable two-way request-reply transactions

Optional user-to-user reliability - WTP user triggers the confirmation of each received message; Optional out-of-band data on acknowledgements; PDU concatenation and delayed acknowledgement to reduce the number of messages sent; and Asynchronous transactions.

2.2.1.4 Wireless Transport Layer Security (WTLS)

WTLS is a session oriented security protocol based upon the web's Transport Layer Security (TLS) protocol. WTLS is intended to work over narrow-band communication channels. Moreover, it is optional and can operate independently of the other layers.

WTLS provides the following features:

The following definitions are obtained from the WAP Forum's specifications in [1] and [2].

Data integrity - WTLS contains facilities to ensure that data sent between the terminal and an application server is unchanged and uncorrupted.

Privacy - WTLS contains facilities to ensure that data transmitted between the terminal and an application server is private and cannot be understood by any intermediate parties that may have intercepted the data stream.

Authentication - WTLS contains facilities to establish the authenticity of the terminal and application server.

Denial-of-service protection - WTLS contains facilities for detecting and rejecting data that is replayed or not successfully verified. WTLS makes many typical denial-of-service attacks harder to accomplish and protects the upper protocol layers.

2.2.1.5 Wireless Datagram Protocol (WDP)

The WDP is the transport protocol for the WAP architecture. It operates transparently over various bearers, and offers consistent service to the upper layers. Moreover, applications can run independently of the transport and bearers since communication among the WDP provides a common interface for the upper layers.

2.2.1.6 Bearers

As previously mentioned, WAP protocols can operate over different bearer services. Moreover, because each of these protocols can vary with regards to quality of service, the WDP layer has specific configurations to run in a specific manner over each bearer.

2.2.1.7 Sample Configurations of WAP Technology

Below are several possible protocol stack arrangements:

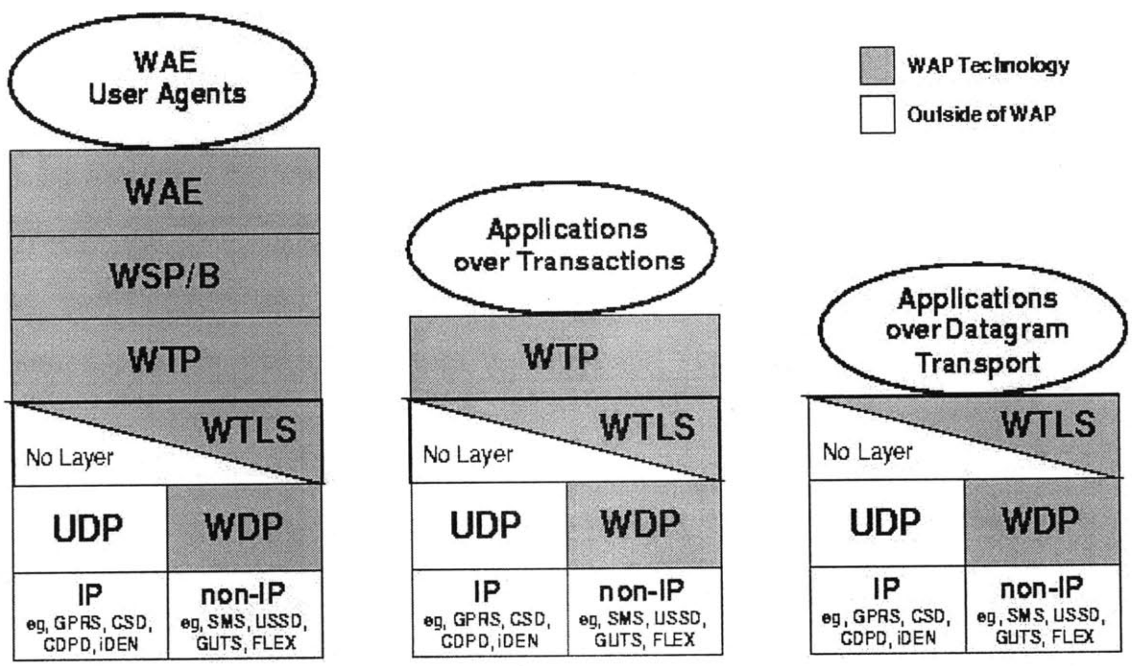

Figure7: Possible WAP configurations

- The left stack shows a WAE User Agent running over the complete WAP stack.
- The middle stack shows the technology underlying applications that need non-secure transactional services.
- The right stack is also for applications that are not secure, but do provide for datagram support.

CHAPTER 3

Software Architecture and Modular Design

3.1 Overview

As previously stated, users, service providers, and operators are affected by fundamental problems of the wireless web. These problems include quick transition to the wireless world and poor wireless network performance. For example, the wireless world presents many new issues to the Application Service Provider. They include a new language, WML that is not compatible with their existing HTML-based sites. Furthermore, given that the screen size of a WAP enabled device, typically a cell phone, is quite small, ASPs need to fit less information on the screen, but still retain their efficacy as a service provider. Finally, ASPs will incur a financial burden, as they will have to train their engineers to learn a new protocol and a new language, possibly hire more engineers to develop their wireless applications, and consider the option of outsourcing their wireless development. Furthermore, WAP enabled device users must are also affected by certain problems with the wireless network. A typical request and response between the user and the server from which they are requesting content, often takes too long, and can be very inconvenient for the user.

In this chapter, I explore several approaches to improve network efficiency, and have developed intelligent software agents to facilitate the transition to the wireless web. Moreover, I adapted the software technology to implement on-the-fly server-based conversion from HTML to WML, implement caching of converted wireless sites to be stored and handled within a WAP gateway, and in fact, to be used as a simple software tool for easy conversion of a website.

The applications of on-the-fly dynamic conversion, website caching, and a managerial conversion tool, developed in this thesis, are all constructed from three building blocks. These are an HTML parser, an efficient data structure for information storage, and a WML sheet constructor. These components are explained in subsequent sections.

The general design flow in each of the developed applications, regardless of exactly the intelligence of conversion occurs, the HTML engine first parses the page, then stores relevant information into the data structure, and finally reconstructs the page into WML. The user is abstracted away from the details of the engine, in that all he understands is that he can view his relevant sites on a wireless device.

3.2 Software Design Criteria

To implement software that performs on-the-fly conversion between languages, and caching, there are several requirements from a software engineering point of view. While server processing speeds must be maximized and network delays must be minimized, the software itself must be designed to perform its functions with speed and efficiency. However, because this software implementation is only a demonstration of future work, I implemented the conversion software in Perl, rather than faster and more efficient languages, considering the many easily accessible CGI models available on the web. One disadvantage to Perl is that it hinders server performance in that every time a CGI script is called, it forks an extra server process, preventing the server from performing other tasks. Another point is that while there are several open source parsers available, I decided to write my own in order to further understand parsing theory, and to explore certain string tokenizing innovations. This design will be explained in a later section.

Additionally, the conversion software package should be modular for several reasons. Modularly designed code is easier to modify and simpler to add new procedures or functions without significantly affecting the rest of the code. Moreover, it is critical that the software be fail safe. In other words, if one portion of the code fails, that failure should not affect the rest of the execution of the program. Also, the software must safely direct the user to a menu or a usable part of the interface, and report an informative error.

A significant aspect to the efficiency of the software technology lies within the data structure that holds the representation of the stored pages of the World Wide Web. An incorrect data structure can hinder the efficiency of the program's functionality. The data model becomes an accurate representation or mapping of the goals of your application. In a later, section, we will show the data model of this software application, and the reasoning behind its final state.

The building blocks of the software are the HTML and WML parsers, and the data model in which the information in the applications are stored. We will expand these components in detail, and finally discuss how they form the integral parts of the applications of the conversion tool, on-the-fly conversion within a WAP gateway, and website caching.

The software was written in Perl, and interfaced to a mySQL database. The software and databases ran on an Apache server running Linux.

Although validation is critical to software engineering, I performed minimally sufficient software testing as my efforts were focused on building intelligent applications for the wireless medium.

3.3 Software Design Building Blocks

3.3.1 HTML Parser

Input: A single html page.

Output: There are two outputs: The first is a dynamically reconstructed WML page, and the second is stored information about the parsed HTML page including links, URLs, and content.

The HTML parser and the WML builder are integral components of the converter. As previously explained, while there are several open source parsers available, I developed a simple tokenizing parser to explore a certain parsing algorithm.

While the input to the parser is an HTML page, there can be two distinct outputs. One output is a wirelessly enabled page in WML, built similar in structure and content to the HTML page. The second output is information about the HTML page stored in a database, such as links, link names, and content, which occur within the HTML page. The data structure for this database is explained in detail in the next section.

The lightweight parser I developed is briefly described conceptually as follows. The parser essentially treats an HTML input page as a string, and tokenizes it. To clarify, this tokenizer splits the page on the "<" tag, and stores the remaining data around this tag into an array. For example, given a sample input of: ...<a href>This is a bold link..., the tokenizer would split this string into separate elements of a newly formed array containing the following:

...n. [a href>] n+1. [b>This is a bold link] n+2. [/b>] n+3.[/a>] ..., where the variable 'n' signifies consecutive numbers in an array sequence.

Once the input is tokenized, the parser traverses through every element of the array searching for an 'href' tag. Once an 'href' tag is found, more intelligence is used to discover what kind of link exists, if any, and if there exists a link name associated with this tag. Finally, other attributes are inferred, such as whether there is an 'img' tagged within the 'href' tags. This parser's intelligence is based upon how well it incorporates all the information regarding HTML from the existing HTML, HTTP, and URL standards.

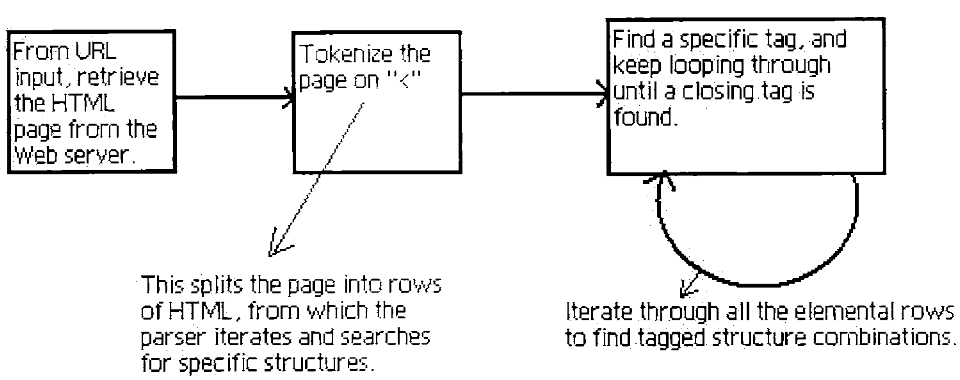

Figure 8: HTML Parser Logical Model

The following is a portion of the parsing program:

.
.
.

```
if ($find_name==1)
{
   $page_row=~ />(.*)/is;

   if($1=~ /(\S.*\S)/i) {
      $name_exists=1;
   }
   $link_name=$1;

   $n=0;
   $img_id=0;

   while (!$name_exists)
   {
      if ($page_row=~ /(.*)img(.*)/i)   {
         if ($page_row=~ /alt=(.*)/i)
         {
            if (!($1=~ /"([^"]*)"/i)) {

               $1=~ /(.*)>/i;

  $1=~ /(.*)border/i;

}
$link_name=$1;
$link_name="Image Name: ".$link_name;
$name_exists=1;
} else {
$link_name="Nameless Image";
```

.
.
.

This is the portion of the program where it traverses through each element of the array searching for certain tags and their completions, such as <img>, <alt>, and so on.

Analyzing this algorithm, we see that it is a linear time algorithm in that it iterates through every character. It is not as efficient as other parsers; however, as mentioned, this was purely an exercise in developing a string tokenizer.

3.3.2 Data Model

The design of the data structure in which information about the inputted HTML page is stored, is critical to the efficiency of the use of the database in this system. If the data model is not efficient, then the code may have to submit multiple queries to obtain information; on the other hand, an efficiently constructed data model could simply require only one query for that same piece of information.

The data model in the case of HTML conversion is essentially a mapping of an entered URL stored in a table of columns and rows of information. In this application, it stores all the link and content information within a given URL. It also keeps track of which URLs are the parents of others URLs.

In the data model, I store the relevant information in two tables: Page-links and Pages. The Pages table essentially records every page encountered and parsed, and the page's relevant information. Page-links records unique URLs, and relevant information about the URL.

The following is the data model, showing the table names and the columns of information each contains:

Page-links Table
Link_id
Link_stub
Link_content
Url_p

Pages
Page_id
Page_name
Name_change
Link_id
Parent_id
Content_stub_p
Display_p

The following describes the reasoning behind the evolved data model, and the way in which websites are represented.

The Page-links table:

The Page-links table holds only unique information about all the URLs encountered. This table records the name of every URL it ever encounters, and assigns it a unique ID. URLs are stored only once.

The attribute link_stub is simply the URL without the prefix: "http://"

The attribute link_content_id is the ID of content, which is referenced in another table not describe here.

The attribute url_p can be true or false depending on whether a URL is being stored or not, for there are certain cases in which an independent wireless screen page is created without the association of an URL.

The Pages table:

To understand the Pages table, we assume that every screen displayed in the WWW, or on a browser, is considered as a single page in the Pages table. Each page is assigned its own id, some succinct name, a link id it is associated with, and a parent id representing the id of the parent page.

The page_id is a unique ID assigned in sequential order.

The page_name is some given name by the input. For example, if "http://www.yahoo.com" is inputted to the parser, then either the parser would infer that "Yahoo!" is the URL's name, or the name would be manually entered by a user.

The link_id within the Pages table references a link_id in the Page-links table. The reason for this design choice is that many different pages encountered on the Internet can be assigned the same URL, but can have different names themselves.

The content_stub_p and display_p attributes give options within the conversion tool. When the tool builds a wireless page that contains content, it has the option of showing all the content, or the first line, or stub, of the content. This allows for more room on the small wireless screen. Finally, the display_p attribute gives the tool the option of whether or not to physically display a page on the screen.

Regarding the content, limited content is stored within the Page-links table, by the attribute: link_content_id. In this design, content is associated with a certain URL. The reason is that the content is defined as the information (excluding that which is tagged), which physically appears between the current link name and the subsequent link name. This is quite limiting in nature; however, for the purposes of a lightweight parser, it is sufficient. Also, its datatype is a number as it refers to another table that stores the actual content.

By way of example, suppose the inputted HTML page outputs:

...

First Link now here is content Second Link

....

The parser would store the URL of "First Link" and store the ID of "now here is content," stored in another content table, under the link_content attribute associated with the "First Link" attribute.

The reason for this design decision is that it makes it easier for the WML builder, explained in detail in the next section, to reconstruct an HTML page for the handheld device.

3.3.3 WML builder

Input: A specific page_id of previously parsed HTML page in the database, or a URL for an unparsed HTML page.

Output: WML code that a cell phone, or a wireless browser, can read. The wireless page displayed contains relevant information including links, URLs, and content found within the associated HTML page.

Once the program parses the HTML page and stores the data into a database, this data must be used to construct a wireless representation of the page. Therefore, the parser reads the information about the parsed HTML page, from the database, and dynamically builds a WML page to be sent to the wireless user.

The following is how the WML builder code operates:

We assume that the HTML page is parsed, and its relevant descriptive information is stored in the database. The WML builder operates given either of two inputs, depending on whether the WML builder tries to reconstruct an already parsed page, in which all the required HTML page information is stored, or if it tries to reconstruct a previously unparsed page.

When the WML builder needs to construct an unparsed HTML page, it calls the HTML parser with a given URL, so that the HTML parser finds and stores the relevant information including links, URLs, content within the page. In the second case, when the WML builder constructs an already parsed page, it constructs it from the already stored information about the HTML page.

In both cases, upon successful completion of parsing and database storage, the WML builder then simply retrieves the stored information, such as the page_id, page_name, links on the page and their associated ID's, and finally the relevant content stored on the page, to reconstruct a WML page which is very similar in content and structure to the HTML page.

The following is a standard, simple WML sheet:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card id="card1" title="Title">
<p>
This is displayed text!
</p>
</card>

</wml>
```

The following is an example where the code prepares and executes a query to the database for specific information to be used in constructing a WML page:

```
...

my $sth = $dbh->prepare("SELECT page_name,link_content,content_type
                               FROM  pages,page
                             WHERE page_links.link_id=pages.link_id
                                 AND pages.page_id='$current_parent_id'");

$dcounter=0;
$sth->execute();
while (my $ref = $sth->fetchrow_hashref()) {
   $dcounter++;
   $current_link_content=$ref->{'link_content'};
   $current_page_name=$ref->{'page_name'};
   $content_type=$ref->{'content_type'};

}

$sth->finish();
if ($dcounter==0) {
   print "Nothing found\n";
}
```

The following is a portion of code which uses the information obtained from the table to create properly formatted WML code, to be shown in screen shots wirelessly to the user:

```
...
$page_name=~ s/&/_and_/g;

$more_page_name=$dcounter;

$stub_stuff="";

if ($xshow_stub_p) {

if (($child_link_content=~ /([^\.]*)\./) || ($child_link_content=~ /([^\
?]*)\?/) || ($child_link_content=~ /([^\!]*)\!/))
{
$stub_stuff="<br/>$1";
```

```
} else {
$stub_stuff="<br/>$child_link_content";
}

$stub_stuff=$stub_stuff."    <small><anchor><go href=\"#more_$more_page_
name\"></go>[more...]</anchor></small>";

$the_new_card=&make_card($more_page_name,$child_link_content);

$card_body=$card_body.$the_new_card;


}


if ($display_bool eq "f") {
$xdisplay_bool=0;
} else {
$xdisplay_bool=1;
}


..........

$code_header= "<?xml version=\"1.0\"?>
<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\" \"http://www.wapforum.org/
DTD/wml_1.1.xml\">
<!-- Source Generated by WML Deck Decoder -->
";


#@main_name=split(/\ /,$current_page_name);
#$name_header=join(/\_/,$main_name);

$more_page_name= $current_page_name;
$more_page_name=~ s/\s/_/g;

$name_header=$more_page_name;

$first_card ="
   <wml>

<card id=\"firstcard\" title=\"$name_header\">
     <onevent type=\"onenterforward\">
       <refresh>
       </refresh>
     </onevent>
     <p>
   ";
```

```
$card_body= $card_body."

 <card id=\"info\" title=\"$name_header Info\">
  <onevent type=\"onenterforward\">
   <refresh>
   </refresh>
  </onevent>
   <p>
   $to_show_on_this_page;

 <do type=\"prev\" label=\"Previous\">
  <prev/>
 </do>
  </p>
</card>
";



$code_options= "
   </p>
   <do type=\"prev\" name=\"prev\" label=\"Back\">
     <prev/>
   </do>

   <do type=\"accept\" label=\"$name_header Info\" name=\"do4\">
     <go href=\"#info\"/>
   </do>
 </card>
";

$code_footer= "
</wml>
";


print "Content-type: text/vnd.wap.wml\n\n";
$code_result=$code_header.$first_card.$code_body.$code_options.$card_body.$code_footer;
.................
```

3.4 Applications

Now that we have an HTML parser, an extensible data model in which to store relevant data about an HTML page, and a WML builder, there are two applications which I developed around these three modular blocks of software: a conversion tool to be used by an individual interested in transitioning their existing HTML website to the wireless, and an on-the-fly converter with website caching capabilities to be used within a WAP gateway for improved network performance.

3.4.1 Conversion Tool

In this conversion tool, the overall input-output relationship is characterized such that the input is a URL for a certain website, a value for the depth and breadth of site traversal, and the output is the same site, but completely wirelessly reengineered. Essentially, leveraging the functionality of the aforementioned parsers and builders, the tool serves as a spider, which crawls through all the links within a certain site and reengineers those sites for WML capability. The reason that the tool needs values for the depth and breadth of the site is that a website could have millions of links which could be infinitely deep, and the tool would never stop processing. Therefore, the tool allows the user to limit their query.

Modularity is a key factor in this software architecture. As previously explained, the code is designed to be fail-safe and can be easily modified. We can divide the labor of the tool into several blackboxes, each with their own functions, and input-output relationships.

The following flow chart explains the overall architecture:

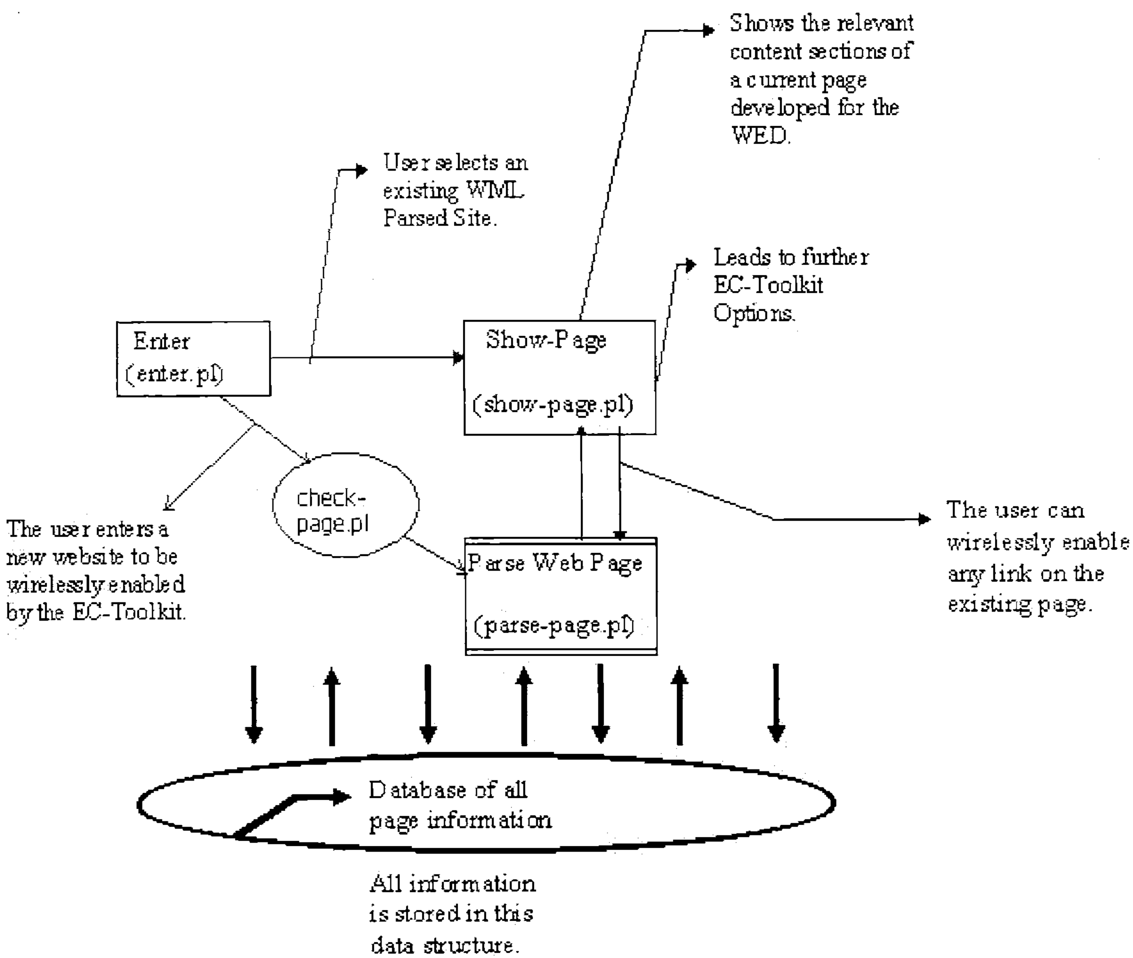

Figure 9: Conversion Tool Logical Flow

State Descriptions:

As shown from the finite state machine, a user begins at the **enter** state (Figures 10 and 11). Here, the user has the option of entering a new, previously unparsed URL to be converted, or he can select from a list of already parsed websites upon which he can perform certain operations in the next state.

If the user chooses from the list, the tool moves to the **show-page** state (Figure 12 and 13). There, the tool provides a list of all the links, URLs, and content portions found within the HTML site entered as an input. Here the user has several options as to how he would like the information displayed wirelessly. The user can choose to display or not to display certain information or link names by using the tool to **toggle** the display of a certain link or content section. Moreover, with the **create-section** state, the tool gives the user the option of creating a new content section to be displayed the wireless device. One other option, in the **edit-link-info** state (Figure 14), provides a form for editing the information displayed for a certain link or content section.

If the user decides to enter a new URL, from within the enter state, then he must also enter values for how many links deep, and how many links per HTML page traversed, should the tool parse. Given this information, the tool then moves to the **check-page** state. Here, the tool checks to see if the entered website has already been parsed. If the site has already been parsed, then the user is directed to the show-page state.

If not, the tool then proceeds to the **parse-page** state. Here, the tool uses the aforementioned HTML parser to parse through a website and gather information to be automatically displayed on the wireless device. Information is then stored into a database about the HTML page, including link names, URLs, and content. It recursively traverses a site given the breadth and depth from the user.

Finally, leveraging the WML builder in the **build-code** state, the tool dynamically ports the existing HTML information within the tool's databases to WML, for use with a WAP enabled device.

The following figures are screenshots of both the conversion tool and the phone browser:

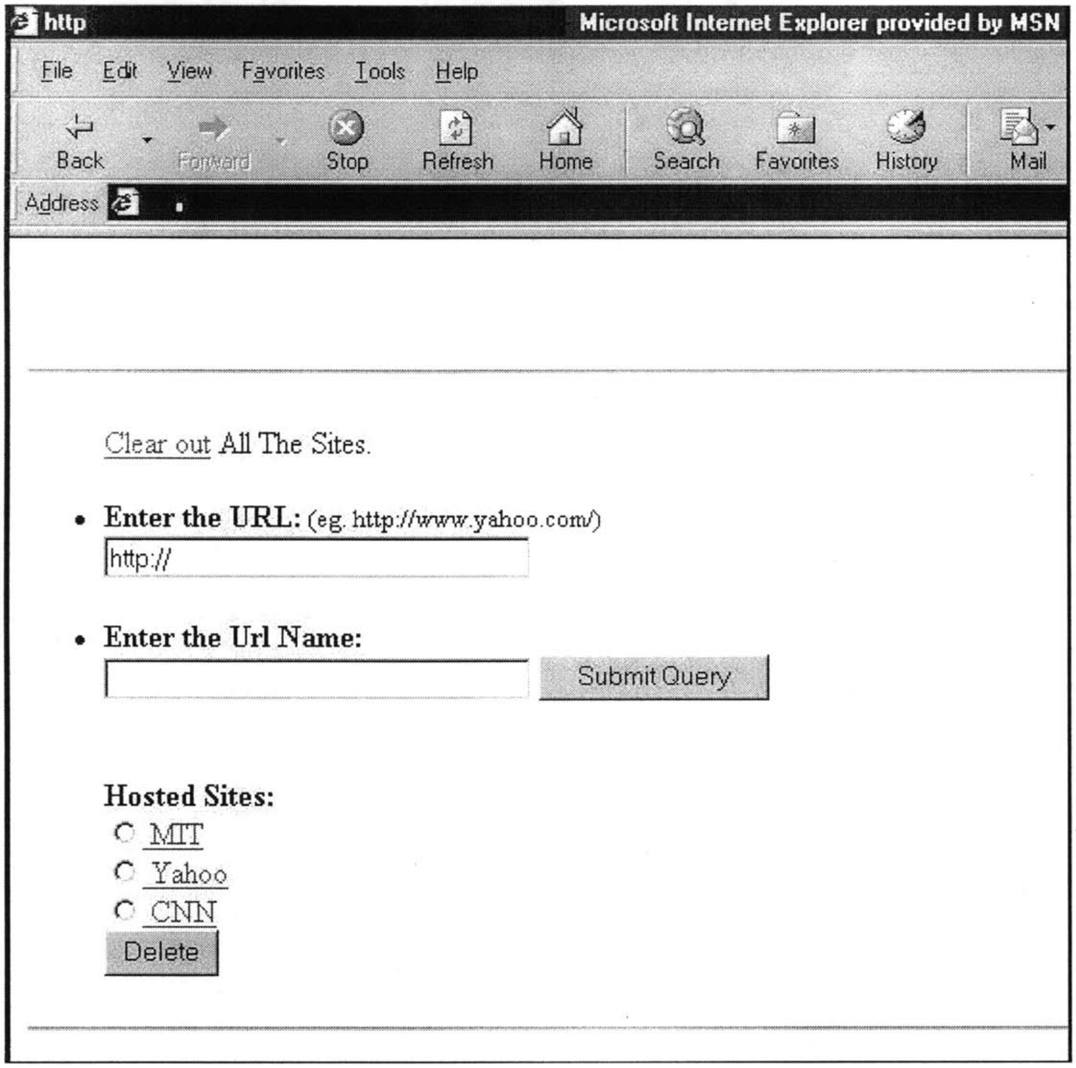

Figure10: Conversion Tool Sreenshot of the Enter Screen

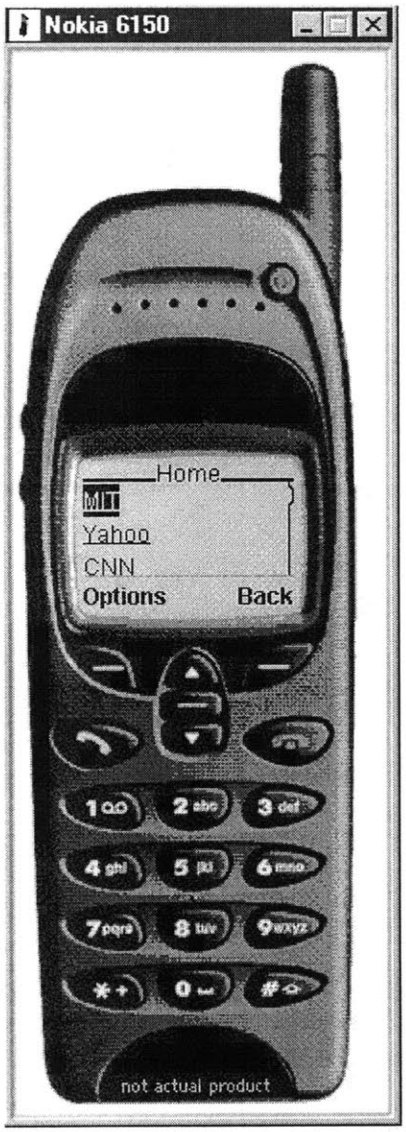

Figure 11: WAP Device Screenshot of the Enter/Home Screen[3]

[3] The images of the Nokia Wap Emulator **are all © Copyright Nokia Corporation 1999.**

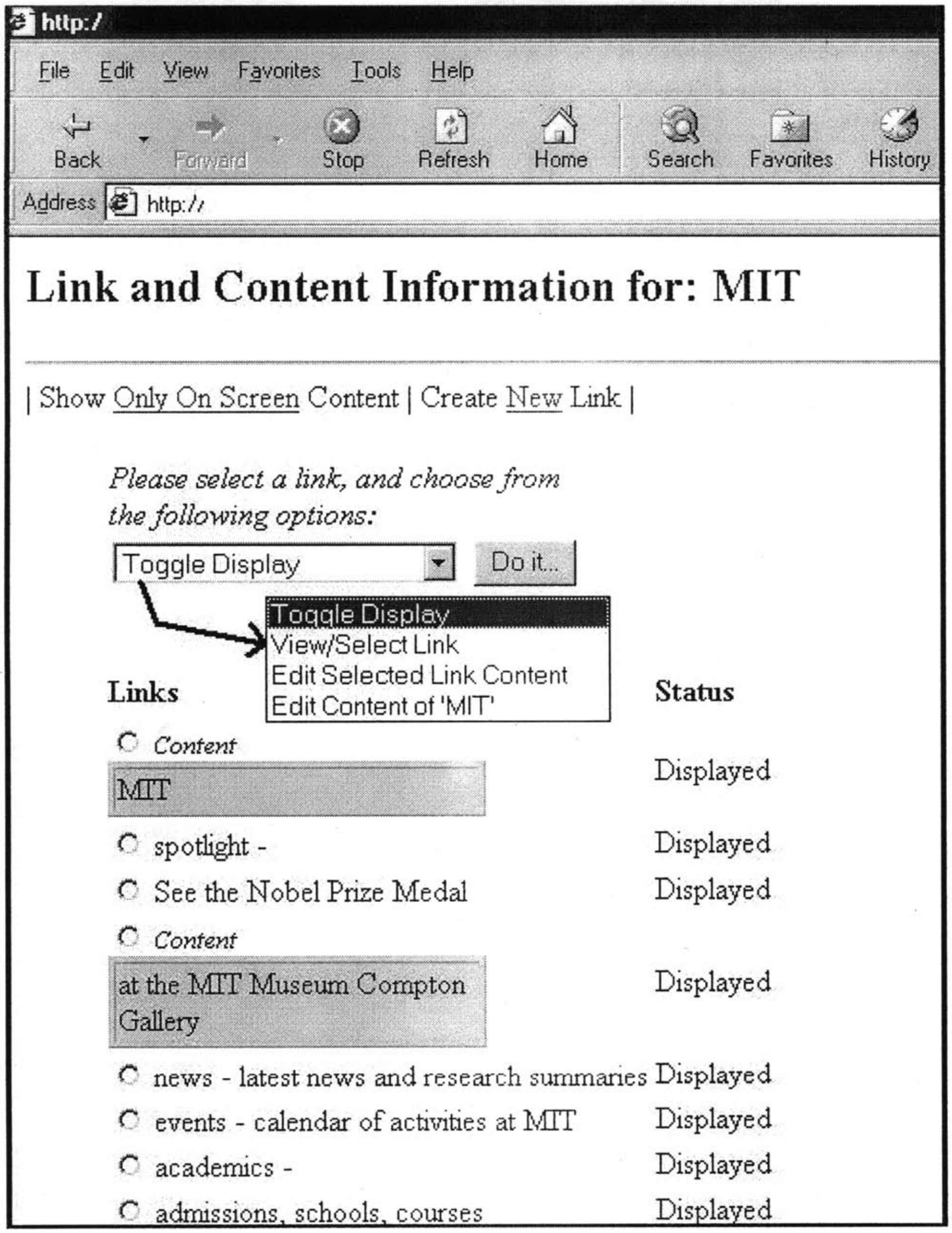

Figure12: Conversion Tool Screenshot of the Show-Page Screen

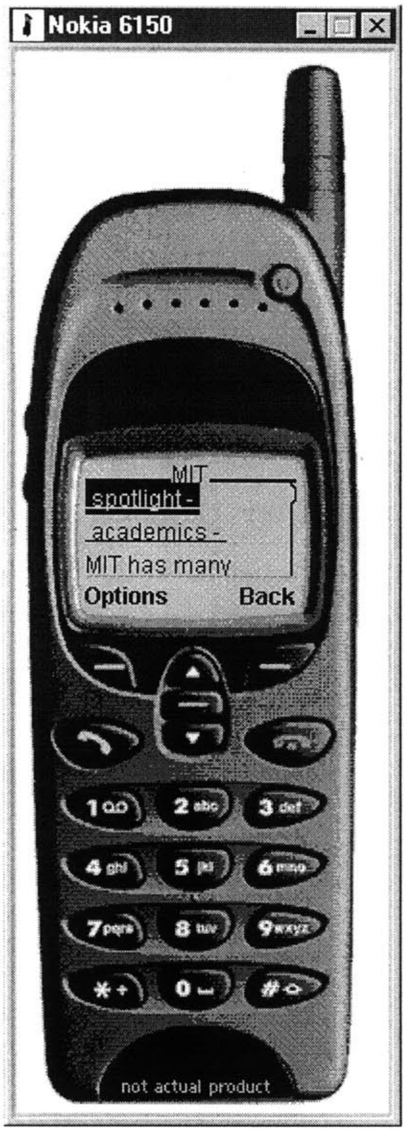

Figure13: WAP Device Screenshot of the Show Page Screen

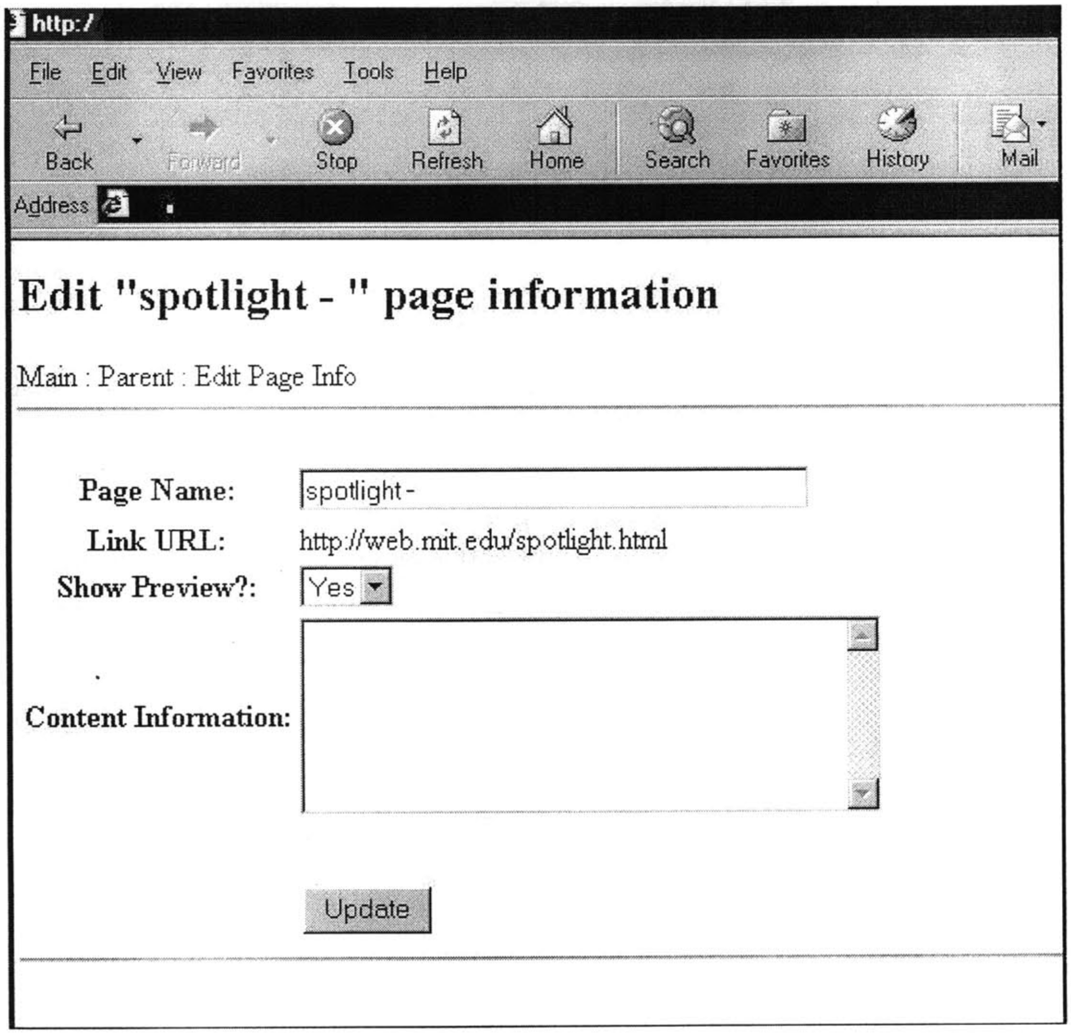

Figure14: Conversion Tool Screenshot of the Edit-Link-Info Screen

3.4.2 On-the-fly converter

We again leverage the functionality of the HTML parser and WML builder to develop an on-the-fly converter (OFC). Its input is a URL, and the output is a WML translated page. The key to the OFC, however, is that rather than storing any information in a database about the HTML page, the OFC simply rebuilds a WML page from an obtained HTML page, which was obtained from the given URL.

The OFC is intended to run in either of two places. It can be stored in a WAP gateway, or it can be stored in a server of an Application Service Provider.

If the OFC is run in an ASP server, then the process is as follows: A WAP device makes a request for a certain URL. The URL, encoded in binary WML, is pushed to the WAP gateway, which then decides to which main server the URL belongs. When the main server gets a URL request from the WAP gateway, it uses the OFC to immediately, on-the-fly, deliver a speedy WML page back to the WAP gateway. The WAP gateway then pushes WML to the requested device.

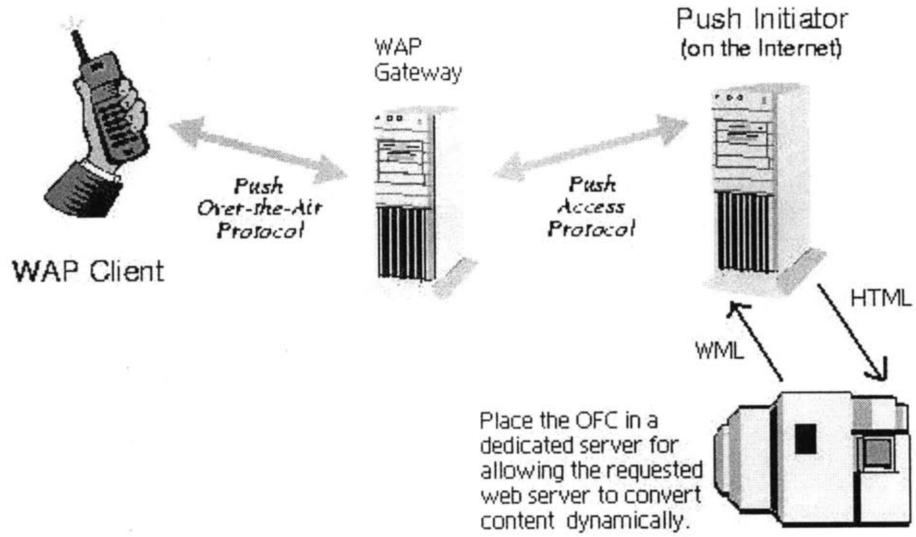

Figure15: WAP Modelwith OFC integration on the Web Server Side

On the other hand, we can run the OFC directly on the WAP gateway. In this case, when the WAP gateway receives a URL request from a WAP device, the gateway uses the OFC to query the main server of the referring URL, obtain the HTML page, and reconstruct a WML page. It is evident that this would require significant processing power within the WAP gateway to handle a myriad of requests to and from the gateway. However, by running the OFC in the gateway, this removes the burden of the wireless transition from the ASP. The problem for the ASP, however, becomes how to control what type of wireless information they can give to the end user. One solution is to place some intelligence in the ASP to interface with the OFC in the WAP gateway. Therefore, when the WAP gateway's OFC requests certain information from the ASP's server, the ASP will handle the request with its interface to allow for more personalized information wirelessly.

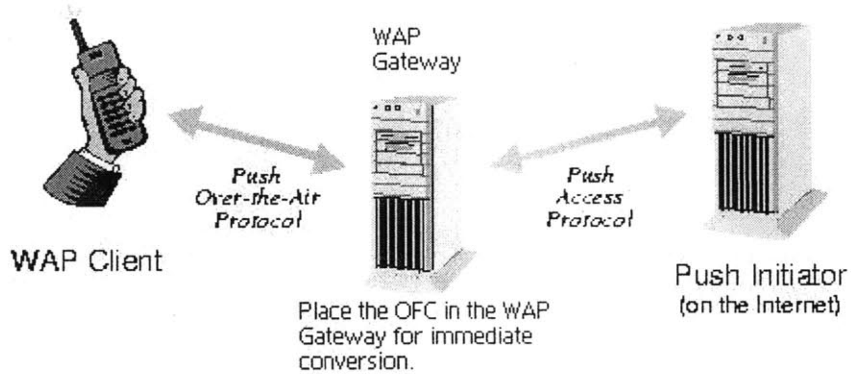

Figure16: WAP Model with OFC integration directly with the WAP Gateway

3.4.3 WML site Caching

A supplement to both methods, whether the OFC is run in the WAP gateway or in an ASP server, is caching. Caching constructed WML sites relieves network traffic and widens network bandwidth by decreasing the amount of requests, and the amount of data traversing through the Internet.

If the OFC caches every unique WML site it constructs, then when the WAP device makes a request for a certain URL, the gateway or the ASP's server (depending on where the OFC is running), can simply return the WML page that was cached. This saves processing time of the OFC and allows the servers to focus their power on handling other tasks.

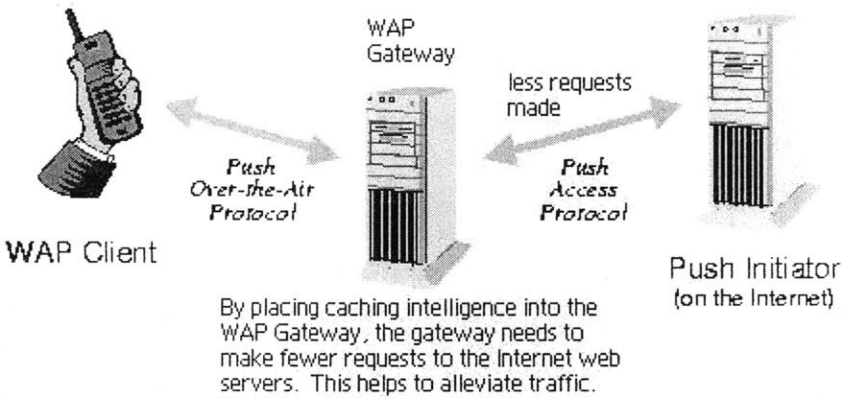

Figure17: WAP Model with caching integrated into the WAP Gateway

CHAPTER 4

Conclusions

4.1 Analysis

WAP introduces many problems of optimization, from a software perspective to that of network performance. I have addressed some of these issues in this thesis. Moreover, each of the applications proposed in this thesis has been developed to a sufficient extent to show that there is potential improvement in WAP.

The HTML to WML converter should be better optimized to offer the quickest conversion rate. Moreover, depending upon where a network designer places the converter, network performance can vary. The conversion tool offers a simple, but effective means for Application Service Providers to transition their services to accommodate for the coming wireless demand. Finally, a brief but important section of the thesis, shows how caching potentially improves WAP's performance.

4.2 Future Work

There are many areas of research for future work. Certainly, the software can be further developed for to further optimize conversion. More functionality can be added to the software toolkit for managing the conversion. Additionally, there are various caching algorithms, which could be incorporated into this work.

One significant area of development is in actually simulating the research. There are several network simulators for capacity planning and network performance that can be used for this purpose. In designing a simulation in the context of this research, one can develop the WAP protocol in the simulators and simulate the protocol stack. Then, one can run what-if scenarios to test different research hypotheses. For example, one can determine the number of cached sites in the WAP gateway that would optimize performance. It is these types of results that can serve to significantly improve WAP's performance and popularity. The Wireless Application Protocol is indeed a very developed standard with a great number of researchers around the world working to improve upon it; and there are a great many opportunities for further research in this area.

References:

1. **WAP Architecture Version 30-Apr-1998. Wireless Application Protocol Architecture Specification. Wireless Application Forum, Ltd. Copyright 1998.**
2. **WAP-195-WAE Overview Version 29-March-2000. Wireless Application Protocol, Wireless Application Environment OverviewWireless Application Forum, Ltd. Copyright 1998.**