WWW Unplugged: An HTML to WML Transcoding Proxy

Robert Dugas

April 29, 2001

Abstract

With the recent proliferation of wireless web clients, standard document styles and serving methods are often no longer adequate. Mobile clients present unique challenges with their limited bandwidth and display size capabilities. The Wireless Access Protocol (WAP) was developed to specifically serve the needs of such clients but proves problematic in its inability to present standard web fare, as well.

The solution proposed is a proxy server which acts as an intermediary transcoder that transforms standard HTML web pages into WAP format and transmits the data to wireless clients. Automatic transcoding issues such as semantic unit extraction and page restructuring are discussed. The working model succeeds in processing most HTML pages into readable WAP documents but further work must be done to enhance the process for optimal user interaction.

1 Introduction

Even just two years ago, designing web content to match intended clients was relatively easy. A designer could safely expect the end-user's bandwidth to meet or exceed 28.8 kbps and could target a display with at least 640x480 resolution. With the recent advent and proliferation of mobile clients, all of that has changed. Wireless Internet access is already customary overseas, and experts predict an imminent surge in US wireless usage. Nokia estimates that the number of mobile Internet users will overtake that of traditional PC internet connections in 2002 [19].

Currently, these mobile clients are vastly constrained by both display size and bandwidth. The Nokia 7100 phone, equipped with a standard micro-browser, displays 65 horizontal lines, each containing 96 pixels. This translates into roughly 6 lines of 12 point text 15 characters long [18]. By contrast, an average desktop can display roughly 40 lines of 12 point text 100 characters long. This amounts to roughly a 97 percent reduction in display size. While PDA devices boast more respectable resolutions, they still are subject to the same bandwidth constraints as the rest of their wireless peers.

Using today's technology, connection speeds for wireless access range dismally between 9.6 and 14.4 kbps [20]. Depending on local coverage and the underlying transport mechanism, these numbers can get even uglier. In the future, third generation (3G) wireless promises broadband connections for mobile users. However, 3G remains at least 2 years away in the US and most estimates put it even farther [23]. Furthermore, while display technology is likely to advance, mobile screen space will always remain a constraint.

All of this evidence points to a realization that the wireless internet is not, in fact, the web we have grown accustomed to, now available on the nearest mobile device.
Fundamental differences in the needs and capabilities of mobile clients dictate a new approach in serving the mobile Web. Furthermore, even within the class of mobile clients, the capabilities of PDA devices differ significantly from those of cellular phones and other ultra-thin clients. This project focuses specifically on cellular phones and WAP, a technology developed specifically for the mobile Internet.

1.1 WAP

In 1997, wireless providers including Ericsson, Motorola, and Nokia combined to created the Wireless Application Protocol Forum. This organization introduced an open specification that defined a protocol stack as well as markup and scripting languages for wireless data exchange. The first WAP-enabled phones became available in 1999 and currently the WAP Forum reports sales of over 40 million devices worldwide [24].

From its inception, WAP has met with criticism. The most oft-cited statistic is a usability study conducted by the Nielson group in which 70% of users reported that, after trying WAP, they did not plan to use it again within the next year [17]. The central problems critics cite are issues like sporadic coverage, long access times, and complicated navigation. At this stage in the development of wireless technology, however, any widely used service would suffer from such drawbacks. If a device cannot receive a signal, no specification can improve usability. Similarly, physics dictate that smaller screens cannot display all information of a full-size HTML page at once. Furthermore, with 40 million current users and that number expected to grow exponentially, the reports of WAP’s death are, indeed, greatly exaggerated [24].

The WAP specification both prescribes low-level transport mechanisms as well as defines a high-level markup language and scripting language [24]. The Wireless Markup Language (WML) conforms to eXtensible Markup Language and provides a platform upon which developers can create wireless applications. It retains some of the HTML tags, but also incorporates new constructs specific to mobile devices. Among these new constructs, the “<card>” tag and the “deck” concept are of primary importance, as a card tag defines one screen-full of data and a deck defines how many cards are transmitted at once. This language, combined with an associated scripting language, WMLScript, provides a customized way to deliver applications to mobile users. However, as a result of this specificity, WAP suffers from an inherent lack of compatibility with standard HTML. Users expecting to browse the web they are accustomed to on their WAP devices are sorely disappointed. This, in turn, contributes to poor usability ratings.

WAP designers, of course, would prefer to view incompatibility with HTML as a feature rather than a problem. The specificity of WAP to wireless clients allows applications to be tailored and optimized for such devices. While this facet is desirable, such rigidity deprives users of the vast content that is the Web’s central appeal. This project seeks to remedy this issue by introducing an HTML to WML automatic conversion proxy. Such a service would augment current WAP site offerings with scaled versions of all existing HTML content.

2 Previous Work

Most current efforts to bring Web content to WAP devices fall into three categories. The most obvious solution is to develop multiple copies of a site, each tailored to various client device types. This approach provides the maximum level of control for publishers as well as providing clients with an optimized user experience. This approach, however, is terribly expensive for web sites as they must rewrite code every time a new format arrives on the market. As a result, although big-name sites such as “Yahoo” and “ESPN.com” have already taken this approach with WAP, many providers never will. Without other alternatives, this deprives sites of traffic and potential clients of content.
2.1 Write Many Devices, Serve for Many Devices

Another solution content providers are using is origin site-side transcoding. IBM’s WebSphere application server, for example, contains a Transcoding Publisher module [13]. Content providers who use the product can specify XSL Transformations which will map their existing HTML offerings to a variety of mobile formats, including WML. Everypath Corporation is also a leader in this field, offering businesses mobile application servers that similarly adapt content to a variety of clients [9]. This technique provides the site developer less control than the first approach but also allows for less expense, re-training, and man-hours. In addition, users benefit from the ability to access a mobile-tailored version of the site. However, while this approach may be wise for larger, established sites, the billions of personal homepages and smaller sites will not utilize such an approach anytime soon. In the meantime, this again means that users and providers are deprived of each other’s business.

2.2 Write Once, Serve for Many Devices

An alternative method that this project focuses on is intermediary-based transcoding. This approach means that sites do not need to rewrite or transform their existing content at all. Instead, an intermediary such as a proxy server intercepts client requests. It then obtains the requested content in standard HTML format, transforms it to match the requesting device, and sends the modified content back to the client. This approach has the disadvantage of taking control almost completely away from site developers. However, it also means that mobile clients can access any and all websites they choose. Furthermore, this approach can easily be combined with the origin server-side options by mandating that the proxy first check for device-specific offerings at the location, only retrieving and modifying standard HTML if other device-specific formats are unavailable.

While still an emerging technology, mobile transcoding proxies have been developed both in the research and commercial sectors. The Digestor research project was developed as a proxy which would discover relevant semantic and syntactic ‘blocks’ and serve the results to mobile users [3]. This work attempted to preserve navigability by remaining as loyal as possible to the architecture of the original site. Google.com employs a similar strategy in its on-the-fly translation services—pages are flattened such that a single HTML page becomes a list of between 15 to 30 cards on average, only navigable sequentially [11]. In addition, Google provides no support for alt-tag identification or image map representations. To augment such work, additional schemes have been proposed to summarize and minimize text presentation [5,6]. Techniques investigated, such as keyword extraction or sentence summarization, support automated conversion processes.

In April, a middle-ware company called WiredPocket signed a deal with NEC to provide converted mobile web access in Japan [25]. Their technology, however, is currently a trade secret and no demonstrations are yet available. The techniques employed, though, claim to be based on “4 years of DARPA-sponsored document research” [25]. Thus, the market already seems to be recognizing intermediary-based solutions as a viable technology, albeit without published results.

2.3 Write Once, Serve Once

Detractors of WAP and automatic translation, however, maintain that no usable or navigable results can be obtained from automatically translating standard HTML to WML [2, 8]. Problems such as the complexity and non-uniformity of the HTML on the web are cited, as well as information extraction problems. The fact that the data on many HTML pages far exceeds the capacity of most WAP devices also poses a problem. In addition, the ability of an automatic converter to generate pages navigable by a human has been called
into question [22]. This project investigates the feasibility of an intermediary transcoding mechanism built to overcome such problems and bring the web to WAP.

3 The Proxy

A proxy-level intermediary forms the framework of this project. The intermediary, however, could also have been implemented as a client-side application. Under that scenario, the client would request and receive standard HTML but then use local, on-board software to perform translation. The resources available on the ultra-thin clients using mobile systems, however, would severely limit the sophistication possible. In addition, if performed by high-end machines server side, the latency through the required transformation process can be masked by the typical bandwidth delay of mobile devices. In this manner, allotting 2 to 3 seconds for processing per page is not overly liberal.

 Ideally, the proxy in the proposed scheme should possess facilities similar to that of a typical proxy server, but also must meet some specifications unique to its role as an intermediary. As a typical proxy, the server should listen for HTTP requests on a given port and spawn threads to handle each. However, the differences begin at this point. First, caching is far more important in a scheme in which heavy processing is performed for each request. A large and efficient web page cache will contribute to hiding the processing latency from the client. An additional consideration should be the deck size (recall that a deck defines how much WML is transmitted to a device at once). Due to limited resources, many devices support deck sizes only up to 1400 bytes. Web pages that translate into decks larger than this should be broken, cached, and served on demand. Similarly, frame set pages should all be retrieved, translated, merged and processed accordingly. To further facilitate wireless browsing, the proxy should also attempt to interpret pared-down requests, for example “yahoo” instead of “http://www.yahoo.com.”

The proxy should also filter out some file types returned by the origin-server. In the most basic scheme, only HTML files should be accepted, processed and transmitted. However, WAP does provide a graphics format called Wireless BitMap (WBMP) designed to handle small, low-resolution graphics efficiently. Thus, the server might also accept graphics files for WBMP conversion. Certainly, though, files such as Flash media, sound, or video should not be transmitted to the client at this stage in wireless infrastructure development.

For this project’s implementation, a Java-based proxy was used. Java is ideal for such a role as it can run on any server platform and the Java network package facilitates fast application development. The server is multi-threaded and extracts threads from a pool when a request comes in, thus saving the overhead of continual thread allocation and deallocation. The proxy listens for HTTP requests, retrieves HTML documents, performs the transcoding, and serves the WML documents to the requesting client. No support for image files or frame sets is included yet. In addition, due to project scope constraints, deck size is permitted to be infinite. The average time to process an incoming file and transmit it ranges from 1 to 1.5 seconds running on JDK 1.3 on a dual processor Pentium machine although little effort is devoted to process-speed optimization at this stage of development. This surprisingly good performance de-emphasized the need for caching and thus no such mechanism is yet implemented.

4 Transcoding

The goal of the transcoding process is to take HTML input and convert it to WML, the language of WAP devices. The original HTML possesses various structures for organizing and presenting information. Unfortunately, in many cases due to the swiss-army nature of HTML, the best way to markup information for presentation often is not the most logical manner to organize it. As a result, facilities such as tables and lists often serve purely as presentational tools rather than as logical grouping devices. Thus, ascertaining the
underlying structure of arbitrary HTML documents and extracting information is in no way a deterministic process. This project, however, relies on the fact that most HTML documents are not wholly arbitrary. Ideally, the following underlying properties of HTML documents should be preserved through the transformation process.

• Navigability
  HTML documents often contain table of contents entities for a site to which they belong. These take on a variety of forms and typically appear as a list of links offset from other document content. In addition, longer documents possess intra-page links to points of interest within the current document. The integrity of these structures should not be compromised in translation.

• Structure
  Often web pages contain visual break points and delimiters to aid in content presentation and readability. Although space is far more limited in a micro-browser, these elements are essential to maintain to some extent in order to make the final results comprehensible.

• Content
  Although it seems obvious that content should be preserved, some schemes proposed for translation actually do attempt to abridge unnecessary or superfluous content [5]. The appeal of the proposed project and its challenge, is to maintain as much as possible of the original HTML content and make it available to mobile users.

• Interactivity
  While static web content is indeed useful reference material, it is the interactivity of the web that appeals to many users. Although many cell phones rely on the phone keypad for input, it is important to preserve this aspect of the user experience to some extent.

In order to preserve these elements, it becomes necessary to examine not only the text contained within the HTML document, but also the tree-structure and tags themselves. The major obstacle to such an analysis is the irregularity of HTML usage encountered in most web pages. Modern browsers attempt to render any and all HTML, no matter how malformed or irregular. Moreover, even W3C HTML standards do not mandate, for example, closing tags on many inline elements. As a result, most HTML on the web today is extensively sloppy. This makes the already difficult task of interpreting HTML for structure and content nearly impossible for a transcoder to perform.

By contrast, XML and its WML subset must conform to a set of strict guidelines. This adherence produces regularity and aids in interpolation between various formats and applications. In addition, it produces predictability among XML documents and allows programmers to make useful assertions in processing. In fact, eXtensible Stysheet Language: Transformations (XSLT), which is itself a subset of XML, exists specifically to transform XML into various output formats including HTML. In order to ease in ascertaining and preserving the properties listed earlier, clearly we would like to examine structured, validated XML rather than the grab bag of irregular HTML that makes up the Web.

Fortunately, a tool exists for cleaning up arbitrary HTML and converting it into XML. This makes documents far more processing-friendly and highlights the underlying tree-structure. Both of these advantages are important for purposes of this project. Dave Raggett’s HTML-Tidy project cleans up errors and non-conformities in HTML and is capable of producing well-formed XML output. At this point in processing, XSLT can then be used to convert the document into WML for display on WAP devices. Unfortunately, as discovered in course of this project, XSLT does not provide adequate facilities to permit the fine-grained control necessary for ideal WML conversion. Thus the final conversion process used in this
project consists of three stages: HTML Tidying, XML to intermediate XML transformation via XSLT, and fine-grained post-processing for complete XML to WML conversion.

4.1 HTML Tidy

HTML Tidy tool is an official World Wide Web Consortium tool [21]. It has become an industry standard in validating and correcting HTML code. In keeping with the Java-oriented nature of this project, an open source Java port of the tool called JTidy was used [16]. This tool proved invaluable in recognizing and correcting errors present in typical HTML documents found on the web.

Typical tasks performed by JTidy for purposes of this project included the following.

- Correcting missing or mismatched end tags
- Relocating misplaced inline tags
- Adding “/” to self closing tags (ie <hr> becomes <hr /> as mandated by XML)
- Inserting missing quotes around attribute values

In addition, several modifications and additions to the JTidy source were made to optimize results for this project. First, unknown or proprietary tags encountered were transformed into <font> tags. The reasoning for this is that font information will be discarded in final processing anyway, and typically proprietary tags serve the purpose of distinguishing text in a manner similar to typical application of font tags. Additionally, JTidy was modified to examine tag attributes and ensures uniqueness by deleting duplicate attributes. This is important in that duplicate attributes are not valid XML and thus confuse XSLT processors. Finally, JTidy also strips out embedded script and comments for this project. Comments have no bearing on the final outcome, and serve only to occasionally confuse the XSLT processor later. Handling script is beyond the scope of this project, and also potentially confuses the processor in later stages of processing.

Following the HTML Tidy stage of processing, the retrieved HTML has been transformed into a valid XML document, or an error has been reported in the case of utterly and completely confused HTML code. The later case was encountered only rarely, and never in the case of commercial sites. This XML output allows the next stage to parse out via transformation only useful page elements. This, in turn, feeds the final stage of WML construction.

4.2 XSLT Processing

XSLT is a language designed primarily for transforming the structure of XML documents. [15, 26]. Originally in this project, XSLT was to take the XML output from JTidy, apply transformation rules, and output the final WML document for transmission to a WAP device. However, this proved too detailed a task for XSLT. As a rule-based, imperative language, the enormous variety in size and structure of HTML that exists on the web would require far too large and varied a set of rules for practical purposes. For example, in determining how much data from a page should be written to a WML card (recall a card is a tag used to define one screen-full) no programmatic control exists to say “keep processing until page is X bytes large.” In general, the less that is known about the incoming source, the poorer XSLT processing performs. XSLT functions optimally in mining specific data from
standard sources. Thus, for this project XSLT was used only to extract certain types of information and produce an intermediate document, which would then undergo further processing before becoming the final WML output.

The intermediate XML produced in this stage of transformation possess qualities which facilitate WML conversion, but which still retain some HTML properties. This means that all plain text is retrieved and grouped according to HTML tag analysis. Whenever ambiguity in grouping arises, the option leading to smaller unit size is chosen. As a result, high-level identification of components like navigational sidebars as in the Digestor system is forgone. However, while the full benefits of correctly identifying such elements are not reaped, the consequences of incorrectly identifying them and producing potentially confusing output are also avoided. Also avoided is the heavy overhead of implementing complex identification heuristics. Furthermore, this strategy is combined with a final processing phase that merges consecutive inadequately small sub-units together into a single larger unit. This process ultimately often mimics the semantic unit extraction high-level identification would have performed.

In extracting such units, an architectural decision was made to flatten the document tree down to a two-level structure of only units and subunits in order to simplify the later stages of processing. This also had the effect of limiting the levels of hierarchy a client will have to click through in order to find data. To compensate for poor performance cases, tree-balancing was employed later in the process.

Various heuristics were used to ascertain the proper break down of structural HTML components. For instance, most large commercial sites and portals organize content around tables. Thus, tables are parsed out as one unit which might contain many smaller subunits and nested tables are flattened into consecutive units. Other elements pulled out to construct top-level units are lists, forms, and map structures. Obviously, many components of a page will not fall into one of these elements and thus become parent-less subunits. These are grouped and dealt with in the final processing phase. Anchor tags with link information are preserved in this phase and alt tags are extracted to represent linked images.

An XSLT transformation on its own, however, is just a file. In order to apply the transformations, the open-source XSLT Java tool Xalan produced by the Apache group was used [1]. Among its many features, this application takes an XML document, an XSLT stylesheet, applies the prescribed transformations, and outputs the result file.

4.3 Final WML Conversion

At this stage, the HTML file has been tidied and transformed into a two-tiered intermediate representation that indicates grouping and content information. In order to present the two-tiered structure to the viewer, however, three views will be necessary. The first is a summarization of all top-level units. The second is a summarization of all subunits within a selected unit and the final is the full text of the subunit itself. This final phase must verify and optimize the grouping, construct the summary representations, and add markup necessary to produce a valid WML document.

In order to accomplish this task without incurring too much storage, important as some HTML documents can be excessively long, a two-tiered data structure of file pointers is constructed to represent the data. At this point, any remaining parent-less subunits are grouped together to form units. As alluded to, the first job of this data structure is to merge insignificantly small units together to produce a more coherent final output. A useful merger threshold determined empirically was 200 bytes. Increasing this number leads to large screens that require too much scrolling while reducing it leads to unintelligibly fractured representations. The 200 byte threshold seemed to encapsulate most HTML constructs well in the sense that logical semantic units such as paragraphs or lists are individualized.
The final step necessary in the grouping process is to balance the unit-subunit tree. This involves examining the number and size of units parsed then merging or splitting in such a manner as to uniformly distribute content. A full balancing process, however, was cut short due to project constraints. Instead, the balancing scheme consists of merging consecutive units with few subunit children into larger single units wherein the first unit inherits all children.

Once successfully grouped, this final processing phase must generate summarization views necessary to present the tree to the user. The method used here, again, is something of a shortcut and is referred to as first sentence elision [3]. This means optimistically assuming that enough relevant information to represent a given section or subsection is contained in its first sentence. Another, more sophisticated method that could be used instead is key-word summarization which ascertains the importance of individual words and presents a list of those deemed most significant as a summary [5,6].

Having generated the proper groupings and produced summarization views, all that remains is to output the final WML. This means merely outing the summarization and traversing the data structure, attaching deck, card, and paragraph tags wherever necessary. In addition, whitespace is added in between text groups to retain readability.

Figure 1: HTML view partitioned into corresponding top-level WAP menu view.

Figure 2: Further partitioning within a unit produces respective subunits.
5 Results

By the end of the transcoding process, the result is a WML deck which contains 3 levels of cards logically representing the hierarchy of the HTML page. This result has the effect of greatly reducing developer control over the final page presentation. In addition, it greatly diminishes the uniqueness of individual pages. Although this uniformity could be alleviated somewhat by the inclusion of limited graphics, it is not entirely a negative thing. Users really only have one fairly standard navigation scheme to master, and this combined with effective summarization techniques helps make pages more easily navigable for users.

To test the effectiveness of the proxy transcoding scheme, the Nokia WAP Developer’s Kit v2.1 was used [18]. The primary tool in this kit is a software simulation of a Nokia 7100 WAP-capable cell phone as well as a generic “blueprint” phone for testing. To reproduce the most generic results, the blueprint phone was run and pointed at a running instance of the transcoding proxy.

5.1 What Works Well

Most transcoded WAP documents turn out to be surprisingly readable and navigable. The first sentence elision-based menus are not always comprehensible, but work especially well if one is familiar with the layout of the standard HTML page already. Most structural page elements end up being encompassed correctly within a unit or across subunits which means that the page grouping mechanisms must be functioning fairly well. Stories and paragraphs of text are broken enough to facilitate reading by scroll reduction, but are not overly fragmented.

The system as whole is remarkably fault-tolerant. Even the most abstruse HTML pages around such as weather.com and nytimes.com translate into a fairly readable format. In fact, larger, portal-like sites such as yahoo.com or yalestation.org translate remarkably well, due primarily to their reliance upon tables for data separation and presentation.

Form input constructs also translated better than anticipated. Although restricted in number of input fields, both get and post method forms are supported. This allows users to enter data in for the search pages like Google or Yahoo or to check stocks on ETrade. The only drawback to this scenario is that most WAP devices rely on telephone keypads for input which can make for a frustrating user experience.

5.2 What Does Not Work Well

Not surprisingly, most the things that do not work well on the original HTML pages do not translate well. Examples of this include excessively long pages or illogically laid out pages. In addition, many pages that make heavy use of images or multimedia files without the use of alt tags did not fare well through the transcoding process.

The first-sentence elision menu system, while functional, also is not optimal. Many times, the first sentence of a section might be a picture caption, ad or disclaimer that has no bearing on the content therein. Thus, in the worst case, such menus are actually misleading and frustrating.

The uniformity of the web pages also became monotonous. Without pictures to spruce up the page, the web seemed to consist of menu after endless similar-looking menu. Although most mobile users are not surfing the web for pleasure, this could become a factor in rating usability.

Finally, the instances in which content slipped through the cracks of the grouping mechanism were few but glaring. Cases such as clicking three times only to get to one or two words of an advertisement have the potential to cause great frustration. In addition, cases in
which 95% of the content of a page becomes lumped into one enormous screen that requires excessive scrolling could be equally frustrating. Fortunately, such cases were rarely encountered.

6 Conclusions and Future Work

This project succeeds in demonstrating the feasibility of translating a large subset of the web to WAP. With more work, that subset could be expanded to include all but a very few poorly constructed pages. While the transcoder is not a perfect mechanism, certainly presenting users with the option of viewing additional web content is better than depriving them entirely. Additionally, the tiered interface often provides a more navigable route to data over purely sequential access.

Additionally, many aspects of HTML that are currently ignored in this scheme or handled sloppily could be improved. Frames, for instance, would require only some additional work at the proxy level. The addition of transcoding support for graphics files would also vastly improve user experience by relieving the monotony of viewing similar pages.

From a broader perspective, the grouping mechanism employed could also be tweaked to recognize more core HTML features heuristically. A broad-based study of the usage characteristics and patterns of appearance for various tags would aid in this process. Also in terms of navigation, adding a keyword summarization method to present menus would almost undoubtedly prove more effective than first sentence elision.

While more work is yet to be done, this relatively simple example is still capable of providing millions of users with content currently unavailable over the wireless web. As wireless technology continues to adapt over the coming years, the necessity for reducing standard HTML and perhaps even the usefulness of WAP itself may disappear. In the meantime, however, an intermediary transcoding proxy can help satisfy the needs of millions of users and website operators.

References


