WAPS: Web Application Server

Jacob I. Torrey

June 6, 2007

ii

Contents

1.1 Introduction

The landscape of how computers are used, and how business is conducted is changing. As the internet becomes more widely used, and as network speeds increase, there is a growing demand for smart websites. These sites provide a service, and sometimes are just as powerful as the desktop counterparts they replace. However, it takes much more time and effort to create the same interactive software over the web, than on a desktop computer. Also, there is a wealth of great software already out there for the desktop. The Web Application Server (WAPS) would let software developers use their existing software and serve it over the web to remote users, with almost no extra effort.

WAPS would replace the current libraries used for writing desktop GUI applications. By recompiling the program with these new libraries, the program would have the same logic, but the graphics calls would be converted to HTML, Javascript and Ajax, then sent to the user via an application server. User actions would be sent to the server using Ajax, and then forwarded to the application. In this manner, the user can use the application as if it's running on the local machine, and not over the internet.

Since this is such a large project, I plan to develop a proof of concept library that replaces the ncurses library. I choose ncurses since it is a small and simple library, which would be a good place to start. From there, I should have a working application server that serves ncurses programs to web users. After the programming stage is complete, I will decide if replacing other, more complex libraries would indeed ease web-based program development.

This paper will cover other ways to develop web-based applications, why WAPS is needed in today's digital age, how WAPS works, considerations in the development of WAPS, and finally, where to go from here.

1.2 Alternatives to WAPS

There are two ways to take a program written for the desktop and convert it into web based software, either by converting the code by hand, or by using GWT, which provides automatic conversion from Java to web languages.

1.2.1 Converting by Hand

This is the most common method for conversion, it is done by re-implementing all the features of the desktop application in a web-based language and rewriting the interface by hand, or with the help of a library. Using this method, it is possible to take advantage of all the pros of web-based software development, and find ways to avoid small things that are very difficult to program in a web environment. An example of this, is Meebo, which implements all the functionality of a desktop-based instant message program, but takes advantage of technologies such as Ajax, Javascript and SQL databases. Meebo is an example of a team of developers taking existing functionality from a standard desktop application and re-implementing it in web languages using more web-friendly technology.

However, this is a time consuming, and difficult process, as every feature needs to be re-coded, and some web languages are quite different from their desktop counterparts. As this is pretty much the only option for converting desktop applications to the web, this is the most popular method.

1.2.2 Google Web Toolkit

GWT is a relatively new project, and is still in beta. Aside from this, it shows great promise to ease the development of cross-browser development. GWT allows for develops to write software in Java, and then have it converted to web languages. It also provides built in functionality for detecting, and correcting differences in the rendering engines of different browsers. Another advantage of GWT is that it lets developers debug their application in Java, which has very power debugging and re-factoring support. Once the code is working, the GWT will convert it to web-based languages.

Even though GWT helps develops create GUI applications in a more powerful language, it's aim is to ease the development of web-based applications, not to convert existing programs to the web environment. In addition, it only supports Java, which is a relatively new language. Because of this, there isn't too much existing software that needs to be converted, where as there are many programs written in C/C++ (Google (2007)).

1.3 How it Works

WAPS is made up of a few different parts that all work together to allow for ncurses applications to run over the web. There is a C++ library that is included instead of the normal `ncurses.h`, this library has all the same functions that the standard ncurses library does, but writes commands to a file, so that the next part of the WAPS system can work. The second part is a web-based PHP backend and a Javascript/Ajax frontend to display the GUI that's described by the output file from the C++ library. Figure 1.1 shows how the different parts of WAPS work together (Wikipedia (2007)).

1.3.1 C++ Library

The C++ library, called `webncurses.h` is included in the application instead of the standard ncurses library, it has a version of all the normal ncurses functions. Due to time constraints and the fact that WAPS is just a proof of concept, the webncurses library only implements a subset of the standard ncurses functionality. While the number of functions supported by WAPS is quite small compared to the total, most of the others are functions which aren't used as much, or are a specialized version of another function which WAPS supports. However, if WAPS was to be taken over by either a company, or an open source

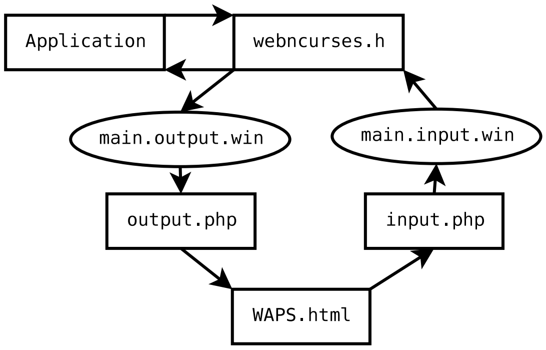

Figure 1.1: A diagram of the WAPS system

project, adding this functionality would be rather simple. The C++ library opens two files, `main.output.win` and `main.input.win` (technically a named pipe) for IO with the client. Whenever the application calls an output function, the C++ library then append the output file with the position and the content of the string.

1.3.2 Web System

There are two parts the web part of WAPS, a PHP backend consisting of `output.php` and `input.php`. The PHP output file merely displays the commands from the file to the Ajax request. It uses a session variable to save how much of the file has been displayed.

The second part is the `WAPS.html` file which has a number of Javascript functions to update the window, and handle user input. Every second, a Javascript function is called which makes an Ajax request to the `output.php` file. If there is text to be output, it adds the text at the right location to the window. If the application wants to input text from the user, is sends a `INPUT [TYPE]` command, and the Javascript makes a form that gets the input and sends it to the `input.php` file using an Ajax request.

1.4 Why WAPS

WAPS has a very large potential to speed up the transition to a web based computer world, from the tradition desktop model. Some of it's advantages that should help its success are as follows:

1.4.1 Simplicity

Simplicity is one of the most important features. Once a WAPS has been setup (see 1.6.1), it should take almost no effort to recompile a program and configure it to be run by web users. WAPS should be easy enough to use to allow for computer users with only a slight knowledge of programming and server administration to setup and run a WAPS. Also, it lets any user recompile source code to work with WAPS, so the original authors can focus on developing their software, and not on configuring it to work over the web. Since WAPS lets developers recompile their software with a different library, there is no new language to learn, and no downtime for quirks to be worked out. This is a great help, seeing as many programmers are hesitant to drop their favorite language and recode their software in an entirely foreign language.

1.4.2 Fits the Current Trend of Computing

WAPS is needed at this time as more users get connected to the internet at ever faster speeds. This allows richer and more powerful web applications to be developed and used by more users. Another technology that is helping the move towards web applications is the mobile and hand-held infrastructure. As mobile phones become smarter and more connected to the internet, companies want to have their applications web accessible so their employees can stay productive on the road. It also helps keep all important data in one place, so there are no duplicates, or different versions of documents.

1.4.3 Remote Application Invocation

Another advantage of WAPS is that since it uses sockets to communicate from the application to the web server, the applications could be on a different computer talking over the network. This is a huge advantage since the servers can be isolated. If a program were to crash, and start forking, or using up all of the computer memory, the web server would remain stable, also, it lets users run programs that use lots of resources, and not slow down the web server for other uses.

1.5 How it Works

WAPS is made up of a few different parts that all work together to allow for ncurses applications to run over the web. There is a C++ library that is included instead of the normal `ncurses.h`, this library has all the same functions that the standard ncurses library does, but writes commands to a file, so that the next part of the WAPS system can work. The second part is a web-based PHP backend and a Javascript/Ajax frontend to display the GUI that's described by the output file from the C++ library. Figure 1.1 shows how the different parts of WAPS work together (Wikipedia (2007)).

1.5.1 C++ Library

The C++ library, called `webncurses.h` is included in the application instead of the standard ncurses library, it has a version of all the normal ncurses functions. Due to time constraints and the fact that WAPS is just a proof of concept, the webncurses library only implements a subset of the standard ncurses functionality. However, if WAPS was to be taken over by either a company, or an open source project, adding this functionality would be rather simple. The C++ library opens two files, `main.output.win` and `main.input.win` (technically a named pipe) for IO with the client. Whenever the application calls an output function, the C++ library then append the output file with the position and the content of the string.

1.5.2 Web System

There are two parts the web part of WAPS, a PHP backend consisting of `output.php` and `input.php`. The PHP output file merely displays the commands from the file to the Ajax request. It uses a session variable to save how much of the file has been displayed.

The second part is the `WAPS.html` file which has a number of Javascript functions to update the window, and handle user input. Every second, a Javascript function is called which makes an Ajax request to the `output.php` file. If there is text to be output, it adds the text at the right location to the window. If the application wants to input text from the user, is sends a `INPUT [TYPE]` command, and the Javascript makes a form that gets the input and sends it to the `input.php` file using an Ajax request.

1.6 Considerations

Now that both the hand coded and WAPS application are shown to work, it's obvious that WAPS is a powerful tool in the conversion from the desktop to the web environment. However, there are some considerations that need to be taken into account about WAPS, and it's problems and limitations.

1.6.1 Setup and Configuration

To get a WAPS server running, one must simply have a server capable of serving dynamic PHP webpages, and a way to compile the WAPS program to be run, such as g++. To have an application run on the same server as the web server, meerly put the WAPS executable in a web readable directory, and put the WAPS web code (the .php and .html files) in the same directory. Once that's all done, the permissions must be set on the WAP web code so that the web server can read and write to the input and output files, and read the .php files.

If the applications are to reside on a different server, the process is quite similar, but the application server must be able to read and write to the web directory, so it must be mounted by some means, be it NFS, SSH FS, etc... In this manner, the network transactions are tranparent to the application and WAPS, and the server administrator can choose a transport that fits best.

1.6.2 Security

Since WAPS is aimed to be used by companies as well as the individual computer user, there are some security considerations to take into account. None of the security issues are too serious; with careful planning, and a good security policy from the server administrator, WAPS shouldn't provide a security breach. However, if run by someone who doesn't know what they're doing, or doesn't configure everything correctly, WAPS could open a door for crackers into the application and web servers of an organization.

Since WAPS uses sockets to communicate between the application and the server, there is a possibility for a number of problems to arise:

If the application is running on a machine separate from the application server, then there is a possibility of the packets from the application to the server being intercepted. This could either let the attacker read the data that the application is using (potentially confidential), or the attacker could alter or stop the flow of packets, causing erroneous data or a DOS. This is a very serious problem, but it is easily remedied by having the network reasonably secure (internal), using a secure transport between the two servers, or running the applications on the same physical machine as the server.

The other foreseeable security hole is if an attacker were to connect a modified application to the server in place of the one being requested. This would let the attacker get data from the user. To fix this problem, the server would have to have a list of IP addresses that applications can connect from, and have good security on those machines.

The last issue in WAPS is that there are many different POFs compared to a either using the application in a desktop environment or in a hand-coded web application. See table 1.1 below to see a comparison of how a failure would affect each system. The table shows whether the application would run or fail depending on each failure. As can be seen by the first row, if the desktop or server hardware were to fail, then the application wouldn't run.

Failure Type	WAPS	Desktop System	Web Application
Server/Desktop Hardware	Fail	Fail	Fail
Network (User and Server)	Fail	Run	Fail
Network (Application Server)	Fail	Run	Run

Table 1.1: Table of how different failures affect each system

As you can see, the desktop application is the most reliable of the three, however, the current trend in computing is to replace desktop applications with their web-accessible counterparts which can be accessed from anywhere, and from more devices. This comparison also assumes that the applications in the WAPS system are running on a different server than the web server. In the case where the applications reside on the same physical hardware as the web server, then WAPS is just as reliable as a hand-coded web application.

Even though there are many serious security considerations to be taken into account when running a WAPS system, with common sense and some good network security practices, WAPS should be as secure as running a normal web application.

1.6.3 Scaling and Performance Issues

Since the WAPS system runs a new instance of the application for each user, it takes up quite a large amount of system resources for many users. In addition, if the user is running a program that runs using a JIT compiler (i.e. .NET program), or has large libraries, running many instances of the program would have a very large performance impact. WAPS can offset this by spreading out the applications to other remote servers, but by doing that, there are more security considerations.

In this sense, WAPS is an inferior option to writing the application from scratch. However, depending on the size of the intended audience, and the time and effort required to hand code a web application, it might still make sense to use WAPS to reuse existing code.

Another performance issue is that WAPS uses whatever format of storing data the application uses, whereas most web applications use databases which are designed to be fast and allow for multiple reads and writes at once. Since the applications being used through WAPS may use a database, or another format of data storage, it may be much slower. Most databases also perform data consistency checks to ensure that if the server crashes, either the last transaction will be performed, or it won't, meaning that there won't be any data corruption. Most applications that would be used in a WAPS system don't have this safeguard, and would be adversely affected in the effect of any component in the system failing.

1.6.4 External Limits

The most common ncurses applications, `emacs, vim, nano,` etc... all require the use of control sequences, such as control-c, or esc. This works just fine in a terminal, where all the control sequences are passed either to the terminal, or to the application itself, however, the web browser used to run the WAPS application won't forward those on. There is no way for the web part of WAPS to reconfigure the browser so that control-x is sent over Ajax to the server, rather than cutting text. It may be possible to add a text box that is checked for content, and then send the control sequence that way. For example, the user would type 'Cĉ' in the box, and the control sequence would be sent to the application.

1.6.5 GTK WAPS Problems

The last foreseeable problem with WAPS is the increased complexity of creating a WAPS compatible version of other GUI libraries such as GTK and Qt. Ncurses is a very simple library compared to the aforementioned ones, and mostly works with layout, colors and where text can be entered. The other libraries deal with mouse clicks, images, drop-down boxes and other more complicated elements that would both make programming the WAPS library more complicated, but also use more bandwidth to handle all the events introduced by GTK or Qt WAPS programs.

1.6.6 WAPS Proof of Concept Problems

The above considerations are for a polished, more complete version on WAPS, this section will cover the problems that the current proof of concept version of WAPS has in addition to the aforementioned issues. Since there was a time constraint, and the developer's lack of previous knowledge of Ajax and named pipes, the current implementation of WAPS is most likely not as good as it could be. With a team of knowledgable developers, WAPS could prove to be a very useful tool, as currently it is no more than a proof of concept.

Scaling Issues

Due to the fact that the `output.php` script uses sessions to keep track of how much of the GUI has been drawn, if two users were to run the same application, the output would be split among the different users. On the input side, because there is no tracking or validation in the `input.php` script, both users' input would be sent to the application. There are two ways to fix this: when a user first asks for an application to be run, a script makes a copy of the application into a temporary sub-directory, so all the input and output files would be isolated. This could potentially use lots of space, and require lots of copying and erasing. The second plan would be to generate a random number, and set that in the user session, and add it to the output and input filenames. The second approach would require modification to the WAPS code, though it would be a better solution, the former idea would be rather slap-dash and could cause performance issues.

The other scaling issue is if a user wishes to run multiple applications at once, because WAPS uses sessions to store how much of the output has been sent to the user, the two different applications running at the same time could cause the session to be reset. This could be fixed by using the random number, as suggested above, which would help fix both of the scaling problems. Have each user-application instance given a unique random number, and append the session name and IO filenames with that number to partition each user from others, and keep each application from interfering with other applications.

Refresh Rate

In professional Ajax web applications, Ajax requests are made preemptivly, in an attempt to cache as much data as possible, so that the page is more responsive to user clicks. The web part of WAPS was not coded in such an intelligent way, relying on 1.5 second cycle of output updates. Since input is triggered by the output script, it can take 1.5 seconds before the input is asked for, and then the time to transport that data back to the server. This makes WAPS impractical for applications that require speedy input and output, such as games or other real-time applications.

1.7 Where to From Here

Now that WAPS has shown to be a feasible option for web-enabling applications, it's time to see where WAPS should go from its relatively simple start. Currently, WAPS is limited to simple ncurses applications. However, this meager start has shown that WAPS is a promising technology that will help advance the speed of web based application development. What WAPS needs now is a group of dedicated individuals who are willing to work on improving the current WAPS system, and converting other GUI libraries so that they work with WAPS.

1.7.1 Rewards and Challenges for Further WAPS Development

If development were to continue on WAPS, the development team would face many challenges, but there would be enough payoffs to make WAPS a worthwhile project. Below are some of the predictable rewards and challenges facing a team to port other libraries to WAPS.

The challenges of making WAPS work for all programs that use the graphics library are that these libraries are much more complex than the simple ncurses. They have many functions, have bindings to lots of different languages. Since the goal of WAPS is to be able to rapidly deploy any desktop application for web users, the library needs to be able to be flexible enough to work in many different situations, and handle bad programming. It also must be an almost transparent replacement for whatever library is being replaced. Because of this, the WAPS replacement needs to inherit all the quirks and oddities from the old library. The last, and largest challenge is to protect from header rot. To protect from this, the WAPS development team will need to either work with the original authors of the graphics library, or play a constant game of catch up to make sure that the new library is compatible with the new graphics library.

The rewards for creating a port of a major GUI library to WAPS is immense, it would save many man-hours and much money. Rather than rebuilding the entire program to be web accessible, the developers, or any savvy user could change the include statements and recompile. Once this simple process has been done, the new binary can be used in a WAPS setup with no extra configuration.

Glossary

.NET Microsoft's SOA architechure, which lets classes written in different languages communicate. It also provides a powerful class library for developing Web Services

Ajax *Asynchronous Javascript and XML* A technology that lets web pages communicate with the web server without refreshing the page.

beta When software is usable, but may still be buggy, or not have all the features.

C A simple language normally used in systems programming.

C++ A superset of C which adds many new features such as OOP.

DOS *Denial of Service* When the service or server is unreachable by some or all of its users.

GTK A GUI graphics library that is very common in Linux.

GUI *Graphical User Interface* What the user sees and interacts with using both a keyboard and a mouse.

GWT *Google Web Toolkit* A set of libraries written in Java to convert Java GUI applications into web-based applications.

HTML *Hyper Text Markup Language* A language that describes the layout and appearance of web documents.

Java A cross platform, object-oriented computer language.

Javascript A programming language used to create dynamic elements in a web-site. It is executed by the web browser.

ncurses A terminal-based graphics library. It is a revision of the curses library, and is maintained by the GNU project.

OOP *Object Oriented Programming* A new programming mindset, where the programmer makes small 'objects' of code, and then pieces them together.

POF *Point of Failure* A place where an error can occur (server crash, program crash, etc...) and cause the whole system to fail.

Qt A library used to write cross platform GUI applications.

sessions A way for a web server to store data unique to each visitor.

SOA *Service Oriented Architechure* The latest phase in programming, where classes are made web accessible, and applications are made by stringing together classes and functions from many different servers across the internet.

Web Services A web accessible class or set of functions that provide a well defined and documented function.

Bibliography

Google. "Google Web Toolkit Product Overview." 2007. Available from: http://code.google.com/webtoolkit/overview.html.

Wikipedia. "Ncurses — Wikipedia, The Free Encyclopedia." 2007. [Online; accessed 28-April-2007]. Available from: http://en.wikipedia.org/w/index.php?title=Ncurses&oldid=123352423.