EARLY CONTROL SYSTEM APPLICATIONS USING PDAS AT TRIUMF

J.J. Pon, E. Klassen, K.S. Lee, M.M. Mouat, T.M. Tateyama, P.J. Yogendran
TRIUMF, Vancouver, BC, Canada

Abstract

Several software applications are being developed that use a personal digital assistant (PDA), a wireless Ethernet infrastructure, and the Central Control System (CCS) of TRIUMF's 500 MeV cyclotron. These applications are primarily for use in the field rather than at a console and allow for monitoring and controlling of cyclotron parameters from a pocket-size appliance. This paper describes the initial applications, the preferred form of deployment, and the development environment. Also discussed are several potential pitfalls that may arise from the adoption of this technology.

INTRODUCTION

When a device in the cyclotron needs to be monitored or controlled, operators can usually do it remotely from a console. However, there are cases in which devices need some form of local intervention but an X terminal is not in close proximity. In situations like these, the standard procedure is for an operator to stay in front of a console and a second operator to walk over to where the device is located. While the device is handled, the two operators give each other feedback and support via two-way radio.

An early attempt at facilitating remote access was not completely adequate. The objective was for the operator to carry a laptop computer into the field, connect it to the TRIUMF's wireless network, operate on the device in question and get feedback directly from the laptop. But more often than not, devices are handled locally for a very short time only. It is impractical to carry a heavy laptop and wait several seconds for it to boot up and connect to the network. This proposed solution was deemed to be more inefficient than the problem itself.

With the recent emergence of personal digital assistants (PDAs) capable of connecting wirelessly to an Ethernet network, we are presented with another opportunity to improve on field operations. As opposed to laptops, PDAs are light, easy to carry, and boot up instantaneously. This initiative is proposing direct interaction with cyclotron devices through the handheld's web browser [1]. In essence, TRIUMF's Central Control System (CCS) is experimenting with a truly portable console for field operations.

APPLICATION DEPLOYMENT

Several options for application deployment were considered. They ranged from serving existing CCS programs via X Window System to writing native PDA programs. In the end, we chose the web server/client model, where a web server delivers HTML and CGI script applications to the PDA's web browser: NetFront v3.0 [2]. NetFront gave us a very good first impression. It handles fairly sophisticated web pages including those containing tables, frames, cascading style sheets, and even JavaScript code. In addition, the CCS already had a web server in place and some staff members had previous experience with web pages. In other words, the cost of prototyping a web application for the PDA was affordable and could be accomplished in a relatively short time. Incidentally, web pages are extremely portable and can be deployed in a multitude of platforms with little modification.

DEVELOPMENT ENVIRONMENT

Since this project is still in its infancy, we have tried to keep the development environment as modest as possible. For the most part, we made use of existing equipment and kept exclusivity to a minimum.

Development Infrastructure

An existing HP AlphaServer DS10 system [3] running OpenVMS is used as both a web server and also as a development machine. Web server software is HP Secure Web Server (CSWS) [4]. Editing is done using the OpenVMS text editor EVE. Web pages are written in HTML (with embedded JavaScript code when necessary). CGI scripts are done in DEC command language (DCL) or C. For debugging, we use Mozilla's [5] JavaScript console on a Windows 2000 computer. The only item specifically acquired for this project was the Sony Clie PEG-NX70V PDA [6] with wireless LAN card [7]. The PDA runs Palm OS v5.0 [8] and we installed the web browser that came with it: NetFront v3.0. The wireless network in the CCS is using protocol 802.11b, also known as Wi-Fi [9].

Development Cycle

First, an HTML page is typed up using EVE. To view the HTML page properly, two style sheets are created: One for NetFront and the other for PC-based web browsers. We will eventually use more appropriate web page developing tools, but for now it is all being done with a simple text editor. Next, the page is invoked from NetFront, Mozilla, and Internet Explorer to get an idea of how the final product will look. Once satisfied with the page layout, we create a CGI script, which when invoked by a browser generates the HTML page dynamically. Our original HTML page can now be discarded. If the script is relatively simple, it is written in the form of a DCL command procedure. Otherwise it is written in C. Since our web applications display changing data, the web page generated by the CGI script contains a command to call itself every second or so to refresh the data. There is also embedded JavaScript code in the page to select the right style sheet based on the calling browser (see Fig. 1).

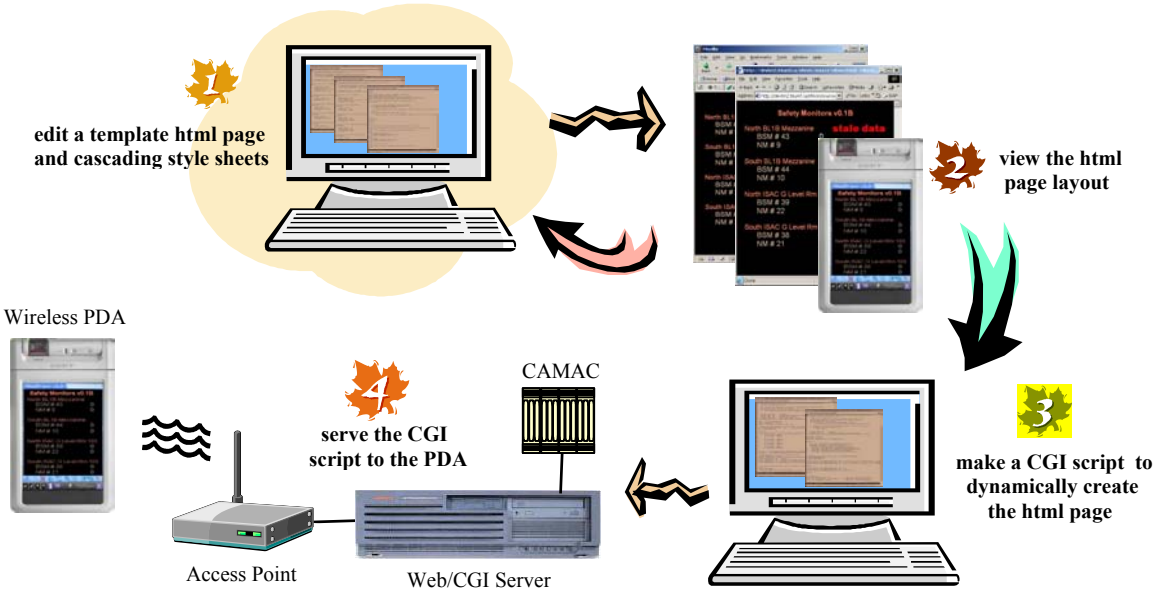

Figure 1: Development Cycle

Design Considerations

Screen size is the major design consideration when developing applications for a PDA. It is a balancing act trying to fit information in a single page as completely as possible while still maintaining a reasonable level of readability. Next, since we are dealing with dynamic data, the page is refreshed at regular intervals. It is of no use to create a scrollable page because the next refresh will always bring up the top left corner of the page no matter which part was being displayed. In addition, since we are refreshing the same page continuously, the "Back" button of the browser is rendered useless. Tapping on the "Back" button would bring up an earlier version of the currently displayed page. In short, our PDA web applications are constrained to a very small, single, un-resizable, un-scrollable window that, for all intents and purposes, will not recall previous pages.

INITIAL APPLICATIONS

Safety Monitors

On a weekly basis, an operator goes around the TRIUMF site testing radiation monitors by bringing an active source near them. In the past, a second operator stayed at the console and would confirm that the monitor was functioning properly. The PDA application SFMon facilitates this routine. SFMon is a CGI script that displays the current readings of safety monitors on NetFront. Now, along with the source, a field operator carries a PDA, calls up SFMon from NetFront and records the values of the safety monitors without the need for a second operator.

Tank Levels and 1A Triplet Pressure

Adjusting tank pressure levels is another activity that requires local manipulation away from a console. To facilitate these field procedures, we are currently testing two CGI-scripts: Tank Levels and 1A Triplet Pressure. These applications display real-time water and gas capping pressure levels on NetFront while an operator works the valves. In the future, there will be no need for a second operator to stay at the console and communicate the readings to the field operator.

WRC

WRC is a project that is under initial development. WRC stands for Web Remote Console and its purpose is to replicate the behaviour of physical consoles. Unlike our other programs that have been developed with a very specific use in mind, WRC is a general-purpose web application in which a user can select almost any device in the CCS. WRC will not only display the current values of a device, but will also let users send command actions to it. Because of the ambitious requirements for WRC, initial versions contained a lot of JavaScript code. This has resulted in unacceptably long page loading times and very slow response from NetFront.

PROBLEM AREAS

We identified certain issues that need to be addressed before running wireless applications in production mode.

Security and User Authentication

Until now, the existing login/password system in the CCS has been a reasonable and adequate form of user authentication. But when applications are invoked from a stateless web browser environment, the issue of security becomes more prevalent. Should we encode at the source,

decode in the target, and authenticate every single request coming from a browser? This would clearly take a toll on performance. On the other hand, we cannot allow wide-open access to the system for obvious reasons.

Wireless Issues

The following are just a few questions that we have yet to fully address. What is the optimum location for a given access point? If users wander away from the nearest base station, how can they tell that their signal is weakening? And if the signal is lost, how do we tell that data is stale or that commands did not take effect?

Limitations of NetFront

With simple web applications, NetFront's performance far exceeded our expectations. But as our requirements started getting more ambitious, we began recognizing a few drawbacks. First, although NetFront can interpret JavaScript, we found that it executes the code very slowly. This could degrade overall performance of more sophisticated web applications. As PDAs become faster, hopefully this issue will be resolved. It is also unfortunate that the browser does not support Java code. It would have been nice to have Java applets in our web programs. And Lastly, NetFront does not come with any debugging tools at all. It is hard to find out what is wrong with a web page that is not behaving as expected. In our case, the work around has been to debug the page using Mozilla, a PC-based web browser that offers adequate debugging features.

Web Server Performance

Performance of the CSWS web server was markedly different based on the type of document being served. We noticed that the web server used a lot of CPU resources when serving CGI scripts. The heavy load was not much different whether the script was a DCL procedure or a C program. On the other hand, the footprint of serving plain HTML or text pages was barely noticeable. The lesson here is to minimize the use of CGI scripts. Unfortunately, when data on a page needs to be refreshed on a constant basis, there is little choice but to invoke CGI scripts. With only a few users to serve, big payloads from the web server are still tolerable. But should CGI scripting become too taxing on the CPU, we would need to look at other ways of serving CGI like mod_perl [10] or FastCGI [11].

SUMMARY

The TRIUMF's CCS has recently had the chance to experiment with PDAs that can connect to the Ethernet network wirelessly. This new technology has allowed us to explore the possibility of deploying CCS applications into a very small, highly portable device. Our choice of application deployment was determined mainly by two factors: Time and cost. We needed to come up with proof of concept within reasonable time while at the same time keeping the cost of training, development, hardware, and software as economical as possible.

Currently, there are three applications that are being field-tested. These programs display readback data of devices like safety monitors and tank level gauges. They provide a function that would normally require support from a second operator. A fourth application, under initial development, will allow field operators not only to read data but also to send commands and control CCS devices.

A few problem areas have been identified. With the proliferation of wireless technology, user authentication and the overall issue of security have become more prevalent. Also, there is a need for some kind of mechanism to ensure that data and commands arrive at their intended receivers in a safe and timely fashion. And of course, we could always benefit from better performing PDA browsers and more efficient methods of serving CGI scripts.

REFERENCES

[1] J.J. Pon et al., "Preliminary Use of PDAs in TRIUMF's Central Control System", these proceedings.
[2] http://www.access-us-inc.com/products/cewb_nf.asp
[3] http://h18002.www1.hp.com/alphaserver/ds10/
[4] http://h71000.www7.hp.com/openvms/products/ips/apache/csws.html
[5] http://www.mozilla.org/
[6] http://www.sonystyle.ca/commerce/servlet/ProductDetailDisplay?storeId=10001&langId=-1&catalogId=10001&productId=165023&navigationPath=45520n45521
[7] http://www.sonystyle.ca/commerce/servlet/ProductDetailDisplay?storeId=10001&langId=-1&catalogId=10001&productId=167483&navigationPath=45520n46100n46100
[8] http://www.palmsource.com/includes/palmos5.pdf
[9] http://www.wirelessethernet.org
[10] http://perl.apache.org/
[11] http://www.fastcgi.com/