Robotic Telescopes: An Interactive Exhibit on the World-Wide Web

Mr. Mark J Cox, Dr. John E F Baruch
Department of Industrial Technology,
University of Bradford, England.

Abstract

The World-Wide Web can be thought of as a distributed multimedia museum. Museums try to attract a wide audience by making their information as interesting as possible. One way they accomplish this and provide a complete learning experience is to use interactive exhibits to challenge and stimulate visitors. This is also true for information on the World-Wide Web, where interactive applications excite users and gain media attention. These novel applications are a form of virtual reality that seamlessly connects users with remote applications and simulations around the world.

The University of Bradford have a prototype low-cost autonomous telescope using a World-Wide Web gateway to provide access for schools, amateurs and professionals. This seamlessly integrates the telescope into the rest of the Web allowing user manuals, live reports, and astronomy lessons all to be accessible from the same interface. The telescope is controlled robotically using a simple form interface where jobs are submitted in advance, or remotely for real-time events.

Using the World-Wide Web makes an interface to the telescope available on most computer platforms and hence widely accessible. In this paper the design and implementation is discussed. It is found that some modifications to the protocols to aid remote control of equipment are useful but not essential: It is recognised that it is important not to over-complicate the protocols by adding specific features for every new innovative application.

Introduction

The World-Wide Web initiative is designed to bring a global multimedia information universe into existence with present technology [Bern92]. When researchers were asked to predict the future of the Internet they said that it would support on-line multimedia collaborations, sophisticated information retrieval and remote control of expensive or unique scientific instruments via tele-experimentation [Hoke94]. The Internet of today is not far from this goal; the World-Wide Web gives access to a wealth of distributed hypertext documents. Extensions already being planned by organisations will support collaboration [Glic94], and attempts at tele-operation are also now being performed on the Web using remotely controlled machinery.

In this paper we look at the details of how and why we connected a Robotic Telescope to the Web, we uncover some problems with the compatibility of browsers and then make suggestions for future changes to the protocols.

Interactive Exhibits on the Web

Before examining our contribution to the Web of a remote and robotically controlled telescope it is worth looking at what types of interactive exhibit are possible and at examples of what has already been achieved. We can currently find four categories of interactive exhibits on the web: computer simulation, remote sensing, simple remote operation and complete real-time control. Computer simulations are the most common interactive exhibit currently found on the Web and there are many examples including an interactive dissection of a virtual frog[1], and a gallery of interactive on-line geometry[2].

[1] URL:http://george.lbl.gov/ITG.hm.pg.docs/dissect/info.html
[2] URL:http://www.geom.umn.edu/apps/gallery.html

The simplest form of hardware connected to the Web is that used for remote sensing. Readings from a machine or sensor are made available periodically or in real-time. There are many examples of this on the Web: weather sensors[3], radiation detectors[4], traffic flow sensors[5], and fun exhibits such as a video camera looking at an iguana[6].

The next category of exhibits are those machines which the user has a limited ability to interact with: the Web user makes some change to the state of the hardware. Interaction is limited to the user sending a single request and receiving a single response. Examples of this are the soft drinks machine[7] that can dispense cans for users with accounts, and a speech synthesiser connected at a research student's house that lets a Web user talk to his cat[8].

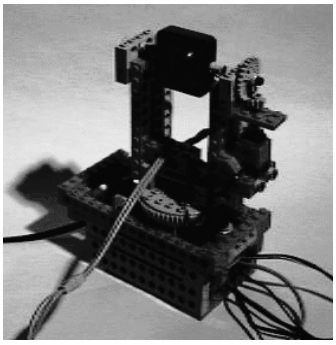

Figure 1 Remote Camera

There are also some examples of complete real-time control of machinery over the Web. One example is a remotely controlled camera from Børre Ludvigsen in Norway. A small video camera is mounted on a base unit that can move the camera in two dimentions as shown in Figure 1. An authorised Web user can control the movement of the base by using simple controls[9]. Pictures from the camera are not relayed via the Web but instead can be watched in real time by using another application called CU-SeeMe. CU-SeeMe is a program for both the PC and Macintosh that shows live video pictures sent over the Internet in a small window [Unkn94]. Another example is from the University of Southern California where a robotic arm with an attached camera and air jet can be controlled. Using the Web[10] users can take turns at controlling the robot and excavating dirt to solve a collaborative puzzle [Gold94].

The Bradford Robotic Telescope

The University of Bradford has been working for a number of years on the development of low-cost robotic and remote telescopes [Baru93]. A fully robotic telescope can decide when conditions are good enough and make observations of the sky by itself: an astronomer does not need to be present and waste time waiting for clear weather. Robotic telescopes are also useful in education where students can send observations to the telescope from their classroom and pick up the results the next day.

Our prototype robotic telescope is situated on the Pennines in West Yorkshire, England. The telescope computer systems are connected to the rest of the world using the global Internet and a large number of environmental, safety and security sensors enable it to operate completely autonomously. The telescope is currently acting as a proving vehicle to assess the feasibility of a global network of similar automated telescopes. The network would be used not only by professionals, but also by amateurs and in education to increase public understanding of science.

Figure 2 shows a picture of the telescope inside its custom-built building. The telescope is a 46cm Newtonian reflector with an alt-az mounting and a cooled CCD camera. Four PCs interface with the telescope and its instruments, and a block diagram of their connections is given in Figure 3. A control computer handles job scheduling, system logging and communication with the University computers. Environmental and security sensors are monitored by a weather computer and when conditions are good enough to make observations the sliding roof is opened. The telescope's CCD camera is connected to an image computer that takes the pictures of the sky, processes, compresses and analyses them. This computer also has to control the selection of viewing filters and to make sure that the telescope is correctly focused. The telescope positioning is controlled by a forth PC that is responsible for accurately finding a position in the sky and tracking the stars as they move during a night. Communication between the telescope and the user is handled by a Sun workstation at the University, ten miles from the telescope building.

---

[3] URL:http://cougarxp.princeton.edu:2112/bpd/webweather.html
[4] URL:http://gargravarr.cc.utexas.edu/geiger-server-ii/geiger.html
[5] URL:http://www.scubed.com:8001/caltrans/transnet.html
[6] URL:http://iguana.images.com/dupecam.html
[7] URL:http://www.csh.rit.edu/proj/drink.html
[8] URL:http://queer.slip.cs.cmu.edu/cgi-bin/talktocat
[9] URL:http://www.ludvigsen.dhhalden.no/webdoc/pt/pt_dummy.html
[10] URL:http://www.usc.edu/dept/raiders/

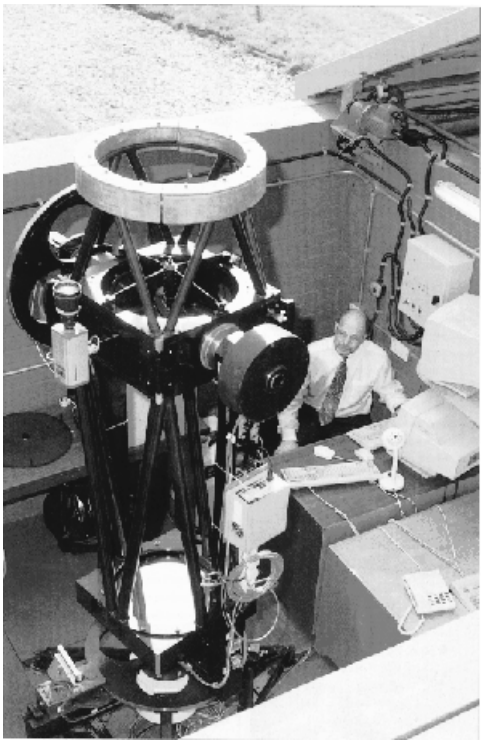

Figure 2 Bradford Robotic Telescope

The telescope normally operates in a "Robotic" mode. Users of the telescope can request observations and these requests are scheduled and uploaded to the telescope based on a priority. The telescope works by itself during the night and returns processed results, operational logs, statistics and weather details the next morning. The telescope can also be controlled in real-time, in its "Remote" mode. In this mode the user submits a request in a similar way but the job is processed immediately and the results are available for analysis or viewing as soon as the telescope has finished the observation. In both modes the telescope is always operating robotically and this ensures that no damage can be caused to the telescope and that it is safe to leave unattended. The remote control facilities are reduced but this makes operation much easier since there is no need for the user to learn complex operating procedures. For effective use of the telescope's resources there is an ability to eavesdrop, where multiple users can watch the results of the telescope being controlled by a single operator.

The telescope is designed to be used by a wide range of people with different backgrounds and so the selection of a good user interface is of critical important to the success of the project. The interface needs to be available over a wide range of machines, should preferably be graphical and must be able to integrate the telescope control with on-line educational and instructional material.

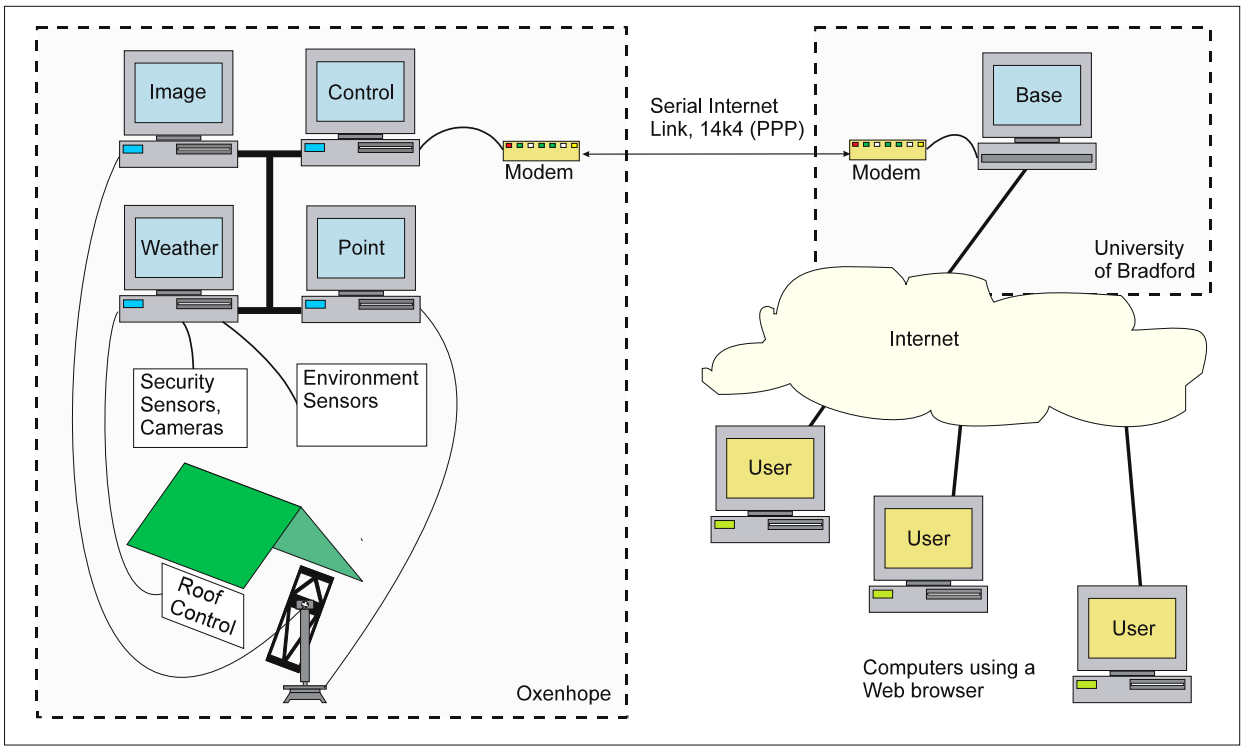

Figure 3 Block diagram of the system

Why use the Web?

The telescope is connected to the global Internet making it accessible to a large number of people. Remote and robotic telescopes in the past have used either custom graphical interfaces limited to a particular computer platform, or simple electronic mail gateways. Producing a custom graphical interface for our telescope would allow great flexibility in the operations that could be performed but would be a huge task, since applications for many different computer platforms and operating systems would have to be written and maintained.

We estimated that there were four platforms used by the majority of people interested in controlling the telescope: PCs using Windows, Macs, UNIX Workstations and text based terminals. Our early developments were with a custom PC interface, and this was used to allow a group of UK school children to interact with the system to generate instructions for using a telescope on Mount Hopkins, Arizona.

It is important to have a consistent user interface over different platforms and operating systems. Users must invest their time in learning a new interface and this time is wasted unless the skills they build up can be easily transferred [Brow94]. The World-Wide Web was considered as a way of providing an interface to the telescope which was independent of the platform and operating system: Web browsers have been written for a range of text based and graphical environments and there are an increasing number of users and applications. The Web also has mechanisms for supporting fill-in forms and virtual documents that let browsers act as remote display interfaces for hardware.

With a Web interface, the telescope can become part of a seamless virtual reality linking in not only its documentation, technical reports and astronomy lessons but also any other relevant work done by anyone else around the world. Users need only a standard Web browser to be able to submit observing requests, find out about our telescope[11] and compare it with the large Apache Point remote telescope[12]. They can look at an image taken from an observation with our telescope and compare it with one taken from a standard star database held at NASA[13].

Telescope Operation on the Web

A Web interface was written for the telescope and can be accessed with URL **http://www.eia.brad.ac.uk/rti/**. The main menu is shown in Figure 4. Users can read an on-line user guide about the telescope, find out technical details of the hardware and software, learn more about stars and galaxies, read weather reports, and control the telescope - all from the same interface.

Job submission and observation results are password protected. On entering the correct username and password a user can look at their jobs' progress from the database, as shown in Figure 5. New jobs can be submitted by filling in a simple form; users specify what object or area of sky they wish to examine and a few other details. Once checked for errors the job is scheduled based on the user's priority ready for processing by the telescope. Just before it turns dark the jobs for the night are collected and sent up to the telescope. The next morning the completed jobs are returned and users can look at their results and download any images.

Reports of telescope performance and the previous day's weather details are also returned from the telescope and made generally available. Since the English weather tends to be poor, there is often a delay of several nights waiting for good conditions and so users are notified automatically via electronic mail when their jobs have been completed.

Anyone on the Internet with a compatible Web browser can submit a low priority job to the telescope as a guest user by using the username *"guest"* and the password *"astro"*. During our testing phases these will be completed as time allows. Details will be available in the future about becoming a registered telescope user.

---

[11] URL:http://www.eia.brad.ac.uk/rti/guide/
[12] URL:http://www.apo.nmsu.edu/
[13] URL:http://skview.gsfc.nasa.gov/

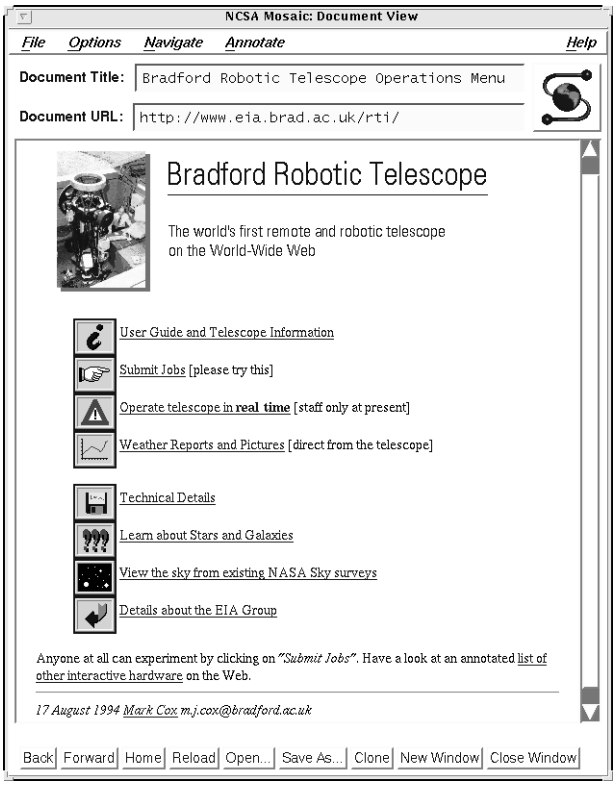

Figure 4 Telescope main page

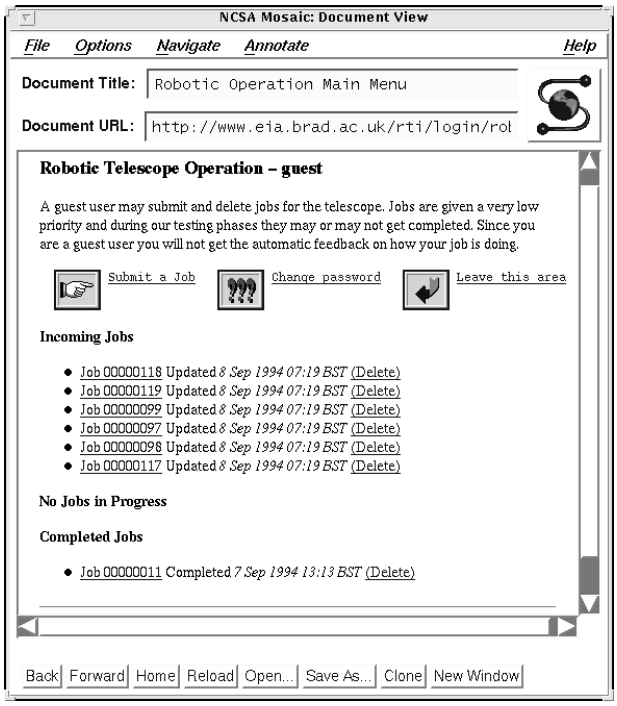

Figure 5 Examining observation requests

Server Implementation

The Web server runs on the Sun workstation and uses a slightly modified version of NCSA httpd[14]. A collection of programs are used to interface the Web to the telescope and these are written in a combination of C and Perl. The C programs are slowly being rewritten in Perl which will make them easier to maintain and keep consistent. User authentication is provided by the standard HTTP basic authentication protocol.

We use three ways of creating the documents for use with the Web interface. The user however is not aware of the mechanism used to create a particular document.

Static Documents

These are pages that do not change automatically: each page is created by hand or by conversion from another format and they are updated manually. This category of document includes the majority of pages on the World-Wide Web.

Periodically Updated

These are pages that are automatically generated or changed periodically by external programs. Our observatory reports, weather summaries and other log files are changed daily by a number of programs.

Dynamic

With a dynamic document each request from a browser causes the server to run an external program. The output of this program is used to create a "virtual page" of information that is then returned to the user. Programs are interfaced using the Common Gateway Interface (CGI) [McCo93]. Such pages are often called "Active Pages" and they are essential for interactive exhibits [Houc94].

Figure 6 shows how the server communicates with the various types of document. For robotic operation scripts are used to accept and process submitted jobs, and add them to the job database ready to be scheduled. Remote control

[14] URL:http://hoohoo.ncsa.uiuc.edu/docs/Overview.html

is more complex with dynamic pages returning updated status information from the telescope. Various additional scripts also have Web interfaces: adding users, assigning priorities and other telescope administration. Other programs are used to periodically update pages, such as the weather summary and operational reports.

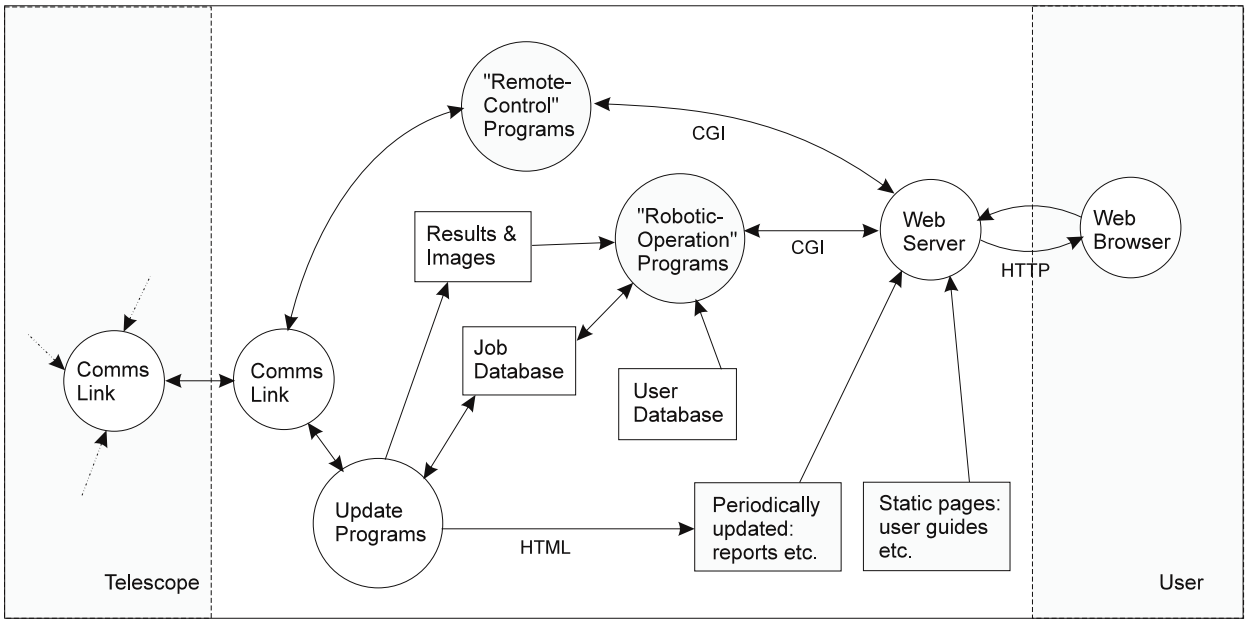

Figure 6 Dynamic, updated and static pages

Forms

One important change we made after initial implementation was a way of improving fill-in forms for the users without any additional changes to the browsers or protocols. Initially a dynamic program would run to check the form was filled in correctly. At the first mistake it would return an error message and expect the user to be able to use the "back" option on their browser to go back to the form and correct their entry. The submission-correction process would continue until a user had entered correct data into all the fields: this is the way many of the forms found on the Web operate. A first improvement can be made by returning all the error messages at once. This lets the user correct all the incorrect fields, reducing the number of times the form has to keep being submitted and returned. A second improvement can be made by returning a copy of the original form along with the error messages back to the user. The server can fill in the user's previous entries in the form when it is returned so that the user can avoid using their browser to go back. This was particularly useful because it was found that some PC and text-based browsers would forget all the data entered into a form after it had been submitted, and so corrections would involve re-typing the whole form each time.

The Perl language is ideal for writing these programs: being an interpreted language it allows the form and processing to be changed without the need for re-compilation. The same script handles both the data submissions (POST) and the generation of the initial form (GET) so that only one program has to be updated when the form needs to be changed.

Graphs

In the same way that text documents on the Web can be dynamic, graphs can also be generated "on-the-fly" based on the user's request. Applications that use this ability have been in operation for some time, the Xerox world map viewer being a good example [Putz94]. It is useful when submitting jobs to the telescope to get an idea of the weather conditions, and data is collected for this purpose from many weather and security sensors at the telescope site[15]. The user can choose to look at graphs of the data from any sensor on any date using a simple form interface. A script checks the form and returns a page that references dynamic in-line images. When the browser requests these in-line images graphs are created from the data archives and returned. A library of C routines, GD [Bout94],

[15] URL:http://www.eia.brad.ac.uk/rti/reports/weather/archive.doit

is used to render the graph onto a bitmap ready to be returned as an in-line image. To reduce load on the server a cache containing the most recently requested images is kept and these are returned where possible.

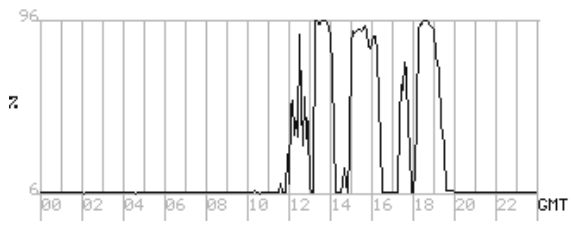

Figure 7 In-line rain

The graphs can then be added into any documents as required. For example, including the following in a document would insert an in-line image showing a graph of the times when it was raining on the 6 September 1994, as shown in Figure 7.

```
<img src="http://www.eia.brad.ac.uk/
rti/reports/weather/archive.doit?
graph=1&date=060994&asens=Rain+
Average">
```

Real-time Control Problems

When exhibits are controlled in real-time it is useful to be able to continually monitor the status of the hardware being used. However, the Web is designed to return a document only when a user requests it and so continuous updating of information cannot be easily performed. The easiest solution is to give the user an "update this page" link which they have to periodically select. Another solution is to write a program that tells a browser to periodically re-load a particular page using the "remote-control" feature of some browsers. Both these solutions have their limitations and a possible alternative is to write a completely separate application for a number of platforms to provide constant communication and display.

Another problem with real-time control is keeping state information: knowing what the user is doing and making sure you can only have one operator at a time. Solutions are currently possible by using application programs running on the server to keep track of users' movements through the system. It is easy for a user to become confused however by saving references to the documents or by using the browser's "Reload" or "Back" options.

Server Security

The basic method of HTTP password security is used to authenticate users and this provides a simple but fairly low level of access protection: passwords are transmitted un-encrypted and are vulnerable to detection by IP monitoring programs. Users can only perform limited functions anyway, and since the telescope runs robotically there is nothing that can be done to cause damage to the equipment or render the telescope unsafe. Electronic mail confirmation is sent when passwords are changed or jobs are completed, so users will become quickly aware of any unauthorised use of their account. The telescope administration programs for updating user records, changing software on the telescope computers and so on, also have Web interfaces. At the moment these are protected by host-based authentication. Using secure encrypted transfers for these types of documents is desirable and several organisations are currently working on additions to the Web protocols to do this[16].

Browser Problems

There are now more than a dozen different types of browser that connect to the telescope server daily. The browsers come from a range of platforms and operating systems and they all have different sets of features. Differences in the implementations of the browsers have caused some problems for our project as it is difficult to tell exactly which browsers support which functions. To a non-technical user these differences are not immediately apparent and the compatibility problems can cause confusion. The simple solution to this is to make sure the telescope application only uses the functions found in all browsers, but this places limits on the system and would over-complicate its design.

The system can identify what browser is currently being used by the user most of the time. We use this information to look up a browser's capabilities in a table. When the user makes a request to submit a job a dynamic page informs them if their browser has the right support and if not what other browsers on their platform will work instead. This is not ideal as new browsers and versions are being written all the time and we have to keep track of which ones support which functions. In the future compliancy testing of browsers might make users more aware of

[16] URL:http://info.cern.ch/hypertext/WWW/Shen/ref/shen.html

the compatibility problems. There are two main problems with some browsers that affect us: the caching of dynamic documents and the built-in basic authentication.

Caching of Dynamic Documents

A very common problem shared by nearly every browser available at the moment is that they incorrectly cache dynamic documents. Many browsers cache all documents so that when a user requests a page they have recently seen the browser does not need to make another connection to the server to get it again. However, dynamic documents can produce different output each time the same page is requested. It is important that these pages are not cached by the browser so that the user always obtains the latest version.

The "Expires" header in the HTTP protocol [Bern94] is designed to stop browsers and proxy servers from caching frequently changing pages. Unfortunately there are very few browsers that take notice of this header and so a work-around has to be used on the telescope pages. A simple and effective way of stopping documents being cached is to append a different string to the URL each time a dynamic page is referenced (encoding a string from the current date and time works well). This fools browsers into thinking that the document is different each time.

Basic Authentication

Fill-in forms and authentication are both moderately new additions to the Web protocols. Some browsers support forms and some support authentication. To submit a job to the telescope a browser needs to support both. At the time of writing there were no PC browsers available that could submit forms into authenticated areas. A possible solution is to not use the built in authentication but instead get the user to enter their username and password each time they ask to see protected document or submit a form. However this is very inconvenient for the user. We do not do this, instead a user with an incompatible browser is told of the problem when they try to submit a job and it is suggested that they use an alternative browser. In the future compatibility problems will slowly disappear as competition between browsers increases.

Suggested Protocol Additions

Other interactive Web applications have found problems with the Web protocols: some need to keep track of users by using state information [Ibra94], others find that the interface itself is limiting [Klem94]. Whilst writing the various gateway programs to interface the telescope to the Web we have thought of a number of changes and additions to the protocols. These changes would also be useful for other interactive applications and so they are briefly outlined in this paper. Whilst the changes would increase the usefulness of the telescope exhibit they are not essential and it is important that the protocols are not made over-complicated by new additions for every innovative application. Browsers should be kept as simple as possible, since adding more and more complexity to the protocols will put off potential authors. New protocol additions would also have to be implemented in all the current browsers. One solution to this would be to have small applications that can be integrated with browsers as proposed by Gutfreund [Gutf94]. These applications would have a defined way of talking to a browser and allow many new possibilities such as real-time status displays for remote-control applications.

We propose three possible additions. In-line vector graphics are a useful way of increasing the speed of producing graphical output; a permanent image cache would allow users with slow Internet links to still see the attractiveness of Web exhibits; and a progress indicator would be useful during searches and other processes that take a long time to complete.

In-line Vector Graphics

Images displayed in-line in Web documents are currently transferred as compressed bitmaps (GIF format). Vector based information like graphs or block graphics have to be rendered onto a bitmap and encoded before they are returned to the browser. This process is not only slow for encoding and decoding but it also limits the resolution of the image: lines and boxes become aliased and small text becomes unreadable. Adding a vector based language into to protocol would allow a picture to be scaled based on the resolution of the display being used. For our exhibit in its real-time mode a line drawing of the telescopes current position could be encoded in only a few bytes.

There are frequent requests for large numbers of other image formats to be supported such as JPEG images which are smaller but take a long time to decode. It is unlikely that new formats will be added to the specifications as

every browser writer would then have to include and maintain code to render or decode the new format images; with only one format currently widely supported this task is quite manageable.

Permanent Local Image Cache

Many of the documents found on the Web have in-line images of some sort and size. For example menu pages have small coloured "bullet" icons, sound samples are referenced by a standard small picture of a speaker, and University home pages have their collection of logos. Over a slow Internet link (such as a telephone connection) these images can take a long time to download and so users will often tell their browsers not to load them at all. Some of the icons are very popular and can be found in use at many sites.

In-line images could have the option of being assigned a unique identifier. When a browser is about to download an image it would first look in its permanent cache. If the browser does not already have a copy of the image locally it would then request it from the server and the user would have the option of adding it to the permanent cache for the future. A standard pack of the icons most commonly found on the Web could be distributed with all browsers, and institutes such as Universities could install their logos and other graphics on all campus machines. A standard pack of graphics and icons for our telescope could be downloaded and installed in advance and our application would be visually attractive even if used over a slow serial line. The format and assigning of the unique identifier is left for further discussion, although it would be reasonable to use the work being done on the Universal Resource Name (URN) scheme [Soll94].

Progress Indication

Static Web documents require minimal server processing time and any delays in displaying such a document are normally due to the time it takes to transfer the document over the network. Dynamic documents used for searching or other interactive applications can take much longer to complete on the server. Current browsers have a "bytes-downloaded" or percentage indicator to give an idea how long a document being sent from the server will take to arrive. It is fairly trivial to add to this indicator so that interactive applications can display a message about their current progress at any time whilst they are running on the server. Patches for two of the popular browsers have already been written to allow this [Perr94] and the implementation does not affect existing servers or browsers. A certain type of dynamic document (a non-parsed header CGI script) can output an extra "Progress" header as many times as required before the main body of the page. This is incredibly useful for the telescope as some scripts that involve connecting to the telescope site and retrieving data can take several minutes to run. An example is given below of what the header of a dynamic document would look like after the initial HTTP headers.

```
Progress: Starting to dial remote site (pause)
Progress: Connection established, sending command (pause)
Progress: Command accepted, retrieving data (pause)
Content-type: .. (rest of html document)
```

Future of the Telescope

The telescope has been available on the Web since November 1993 and it is still evolving. At the moment only simple observations are able to be performed but this will soon be altered to allow more complex observations and the tracking of the planets and comets. More detailed astronomy lessons, telescope manuals and technical details are also being added.

In the future we will be working on the robustness and ease of use of remote operation to allow control of the telescope by students and other World-Wide Web users around the world whilst it is dark in England. Another addition will allow a number of observers to eavesdrop while the telescope is being controlled remotely by a single operator. The ability to see or hear the telescope moving in these situations is a great benefit and this will involve the connection of our security cameras to a frame capture board or possibly a CU-SeeMe system.

Conclusion

We have found that the World-Wide Web can act as an excellent interface to the Bradford Robotic Telescope. Reports, user manuals, observation results and control of the telescope can all be integrated together and with other

Web exhibits. Web browsers exist on many platforms and this makes our application operating system and platform independent.

The World-Wide Web interface to a robotic telescope has demonstrated that it is possible to revolutionise astronomy for users such as school students, undergraduates and amateurs. For professional astronomers it opens up whole new areas of astronomy based on the rapid reaction to events and the continuous observation of objects. The interface allows any Internet user to explore real objects and events and make them available to all interested parties - a truly planetary museum.

References

[Baru93] Baruch, John E.F., "Robots in Astronomy," *Vistas in Astronomy*, vol. 35, 1993, pp. 399-438.

[Bern92] Berners-Lee, Tim, Robert Cailliau, Jean-Francios Groff and Bernd Pollerman, "World-Wide Web: The Information Universe," *Electronic Networking: Research, Applications and Policy*, vol. 1, no. 2, Westport CT, Spring 1992

[Bern94] Berners-Lee, Tim, "HTTP: A protocol for networked information," An electronic document available via WWW http://info.cern.ch/hypertext/WWW/Protocols/HTTP/HTTP2.html, 1994.

[Gold94] Goldberg, Ken, Michael Mascha, Steve Gentner, Juergen Rossman, Nick Rothenberg, Carl Sutter and Jeff Wiegley, "Beyond Cyberspace: Excavating the Real World via Mosaic," *Proceedings of the Second International Conference on World-Wide Web*, Chicago, October 17-20 1994.

[Bout94] Boutell, Thomas, "Techniques for Server-Side Dynamic Document Generation," *Proceedings of the Second International Conference on World-Wide Web*, Chicago, October 17-20 1994.

[Brow94] Brown, Lin, "Human-Computer Interaction and Standardization," *StandardView*, vol. 1, no. 1, September 1994, pp. 3-8.

[Glic94] Glicksman, J., G A Kramer and N P Mayer, "Internet Publishing via the World Wide Web," *Proceedings Groupware '94*, San Jose, August 1994, pp. 431-442.

[Gutf94] Gutfreund, Yechezkal-Shimon, "WWWinda: An Orchestration Service for WWW Clients and Accessories," *Proceedings of the Second International Conference on World-Wide Web*, Chicago, October 17-20 1994.

[Hoke94] Hoke, Franklin, "New Internet Capabilities Fueling Innovative Science," *The Scientist*, vol. 8, no. 10, May 16 1994.

[Houc94] Houch, Henry, Chris Lindblad and David Wetherall, "Active Pages: Intelligent Nodes on the World Wide Web," *Proceedings of the First International Conference on World-Wide Web*, Geneva, May 25-27 1994.

[Ibra94] Ibrahim, Bertrand, "World-wide algorithm animation," *Proceedings of the First International Conference on World-Wide Web*, Geneva, May 25-27 1994.

[Klem94] Klements, Anders, "The Design and Implementation of a Media on Demand System for WWW," *Proceedings of the First International Conference on World-Wide Web*, Geneva, May 25-27 1994.

[McCo93] McCool, Rob,"The Common Gateway Interface," An electronic document available via WWW http://hoohoo.ncsa.uiuc.edu/cgi/, National Center for Supercomputer Applications, 1993.

[Perr94] Perry, Bill, "Status Information - Summary?," An electronic message posted to the www-talk mailing list (Message ID: 8133*phillips@cs.ubc.ca) This message is available from the archives, http://gummo.stanford.edu/html/hypermail/www-talk-1994q2.messages/364.html, 28 April 1994.

[Putz94] Putz, Steve, "Interactive Information Services Using World-Wide Web Hypertext," *Proceedings of the First International Conference on World-Wide Web*, Geneva, May 25-27 1994.

[Soll94] Sollins, K. and L. Masinter, "Requirements of Uniform Resource Names," An Internet Draft (draft-sollins-urn-00) available via FTP ftp://ana.lcs.mit.edu/pub/uri/draft-sollins-urn-00.txt, March 26 1994.

[Unkn94] "The CU-SeeMe Frequently Asked Questions," An electronic document available via FTP. ftp://gated.cornell.edu/pub/video/CU-SeeMe.FAQ.7-6.txt, 1994.

Author Biographies

Mark Cox

Mark Cox is currently studying for a PhD in the Department of Industrial Technology at the University of Bradford, England. His research focusses on the use of and interfaces to robotic and remotely controlled telescopes and his interests include public accesss to computer systems and World-Wide Web technologies. Mark is also the author of a large number of programs, libraries and drivers both freeware and commercial used in PC applications. Mark holds a BEng in Electronics, Communications and Computer Systems Engineering also from the University of Bradford which he obtained in 1992.

John Baruch

Dr John Baruch has been working on the development of robotic telescopes and the remote operation of telescopes for over ten years. He has been active in the parallel development of technologies with industrial partners. The objective has been to develop technologies that will support astronomical research and innovation in industry. The robotics programme for astronomy has a range of industrial spin-offs ranging from Internet developments to distributed information systems for manufacturing. His current concerns are the process of innovation and the optimisation both of University research and wealth creation for the industrial partners.

Electronic Mail Contact Address

Mark Cox, m.j.cox@bradford.ac.uk