6

Using the World Wide Web and Java for Network Service Management

M. C. Maston
Cisco Systems, Inc., WAN Business Unit
1400 Parkmoor Avenue, San Jose CA 95126 USA
408-525-2734 (Voice)
408-525-7414 (Facsimile)
mmaston@cisco.com

Abstract

This white paper discusses the potential applications of World Wide Web technologies to satisfy network service management requirements. Web-based tools can be applied in many areas of network management from simple element to highly sophisticated service management applications. The background issues leading to this approach, examples of solutions and the overall benefits of Web browsers and Sun's Java* are discussed.

Keywords

World Wide Web, HTTP, CGI, Java, network management, wide area

1 INTRODUCTION

One of the most confounding issues faced by network equipment vendors, network service providers and end users today is achieving scaleable, reliable and distributed network management. These problems are most vexing in the wide area networking (WAN) space due to several factors including:

- immature or exceedingly complicated management standards
- extremely complex security considerations created by large global networks
- lack of coordination and compatibility among vendor network management equipment
- a myriad of complex management issues surrounding the expensive bandwidth consumed by management traffic

* Java is a registered trademark of Sun Microsystems, Inc.

A. A. Lazar et al. (eds.), *Integrated Network Management V*

- relatively poor understanding of the management issues across the LAN/WAN boundary
- monumental cost and support burden associated with administering management applications across many different operating systems and hardware platforms.

All of these issues contribute to relatively low quality network management functionality being the norm while the market continues to push the edge of the envelope for even more sophisticated management support. It is usually at this point in a technology cycle that a paradigm shift must occur in order to reset the standards and methodologies in place to a new starting point. From such a point, satisfactory solutions can begin to be delivered. Without this shift, products will continue to fall behind market expectations with little or no hope of catching up.

Presently, network management is no longer considered "icing on the cake" by serious network service providers. Rather, it is perceived as an integral component in a complete network design to assure initial and ongoing success. This change in attitude comes largely from a shift in what *users* have come to demand in the way of services from their service provider. Service providers then naturally drive vendors to add functionality in order to satisfy end-user requirements and allow their service to remain competitive in a cutthroat market. Continuous, reliable and high performance service is considered the minimum for what a service provider must offer and the requirements for more advanced management features escalate from there.

Today's service providers and their users expect features such as virtual private networks, customer network management (CNM), advanced reporting facilities and a plethora of other highly customized network management services. The bar has been raised for all service providers to bring bigger and better services to market faster and more often. More importantly, the expectation has been set that these services are delivered in a simple, "plug and play" way so that customers of the service provider can spend little time *learning* and more time *using* the service. Simply exposing an SNMP MIB for customer use no longer provides a complete solution and will only become less satisfactory in the future.

In order for these services to be delivered and continue to be augmented at the necessary pace, a paradigm shift as described before must take place. This change in the delivery of management services will most likely come from the technologies developed for the World Wide Web. Specifically, the HyperText Markup Language (HTML) and Sun Microsystems Java bring significant advantages to the task of deploying versatile, robust and powerful network management solutions. This white paper will discuss how these tools can be used to provide a fundamental change in the way network management applications are developed and delivered.

2 NETWORK MANAGEMENT ISSUES

The key issues that face most vendors and service providers in delivering network management applications are similar to those faced by most mass market software

producers. As might be expected, network management has a few twists that distinguish it from mainstream software, but in general the same rules apply. The most prominent features and functionality required for management applications include:

- security of the network and of all user traffic on the network
- standards-based solutions; no access to information through proprietary interfaces
- support for all the equipment in today's heterogeneous networks
- a scaleable solution that can be deployed economically across the market continuum from enterprise to carrier networks
- affordable for large and small network service providers
- robust and reliable architecture
- high performance, responsive user interface
- "plug and play" installation, upgrades and maintenance
- ease of use for both the novice and "power user"
- availability on a wide range of operating systems and hardware platforms from desktop PC's to high-end workstations
- user interface that can be adapted to different operational models by the end user

Each of these requirements can be met exceptionally well by a combination of HTML and Sun's Java. By exploiting the strengths of each of these tools, a very sophisticated network management product can be developed. Before explaining how these technologies can be put to work for network management, however, it is necessary to separate network management into three classes: element, network and service management.

Element Management

In the simplest terms, element management is the process of monitoring, controlling and configuring any network entity. An entity may be a workstation, router, backbone ATM (Asynchronous Transfer Mode) switch or other network-connected device. Potential management activities include reading statistical counters, issuing commands to control the flow of network traffic, and configuring parameters such as the IP address of the entity or its network name.

Network Management

Network management is the next step in the hierarchy of management from element management. Where element leaves off in only controlling properties of individual network nodes, network management delivers an overall network view and is able to perform operations within or across the complete breadth of the network. Administering PVC's, defining preferred routing and testing the integrity of trunks between nodes are just some of the duties of a typical network management application.

Typically, network management functionality is delivered by vendors on a network management station (NMS). This allows users a single point of access to a set of information that is common to many or all of the network elements. The management station also serves as a focal point for network-wide operations such as provisioning, diagnostics and statistics collection.

Service Management

Service management embodies the processes required to monitor, command and control the actual services delivered by the managed elements. These services can range from LAN services such as TCP/IP to WAN services such as frame relay, ATM, video and voice.

These services can (and typically do) provide more than the ability to transfer network traffic from one point on the network to another. Service providers deliver guaranteed quality of service, advanced performance reporting, customer network management, and usage-based billing. These are value-added services above and beyond basic network transport access. Many service providers use these types of mechanisms as differentiators from their competition.

3 WORLD WIDE WEB TECHNOLOGIES

With each passing day there are new products being offered to enhance and expand the capabilities of the World Wide Web. Products which allow monetary transactions, database access and on-line information retrieval via the Web are common and continue to appear at a phenomenal rate. For the purposes of this discussion, the basic tools of the Web, HTML and Java, will be highlighted.

HTML

HTML has been proven to be extremely easy to learn for many non-technical individuals. Perhaps even more impressive are the results these same people have produced given a little time and a simple text editor. Rich text content, along with audio, video and other multimedia data can be easily delivered with very little in-depth computer knowledge. All of this information is transmitted to client "browsers" such as Netscape Navigator™ from a Web server using the standards-based HyperText Transfer Protocol (HTTP). In essence, HTTP is simply a connection-oriented TCP/IP-based file transfer mechanism. Typically, a user asks for a particular page and the browser retrieves that page from the server. Once downloaded, the browser scans the contents of the page which may call out other files (such as bitmap images) located on the originating Web server or elsewhere. The browser then requests these files until the entire page content is loaded and rendered on the user screen.

What HTTP was not designed to do, however, is provide a continuous stream of updating information. A file is requested, downloaded and the connection is closed once the file has been retrieved. No further interaction between the browser and the server takes place. Obviously, this makes having any sort of dynamic information on a Web page difficult to accomplish. Fortunately, this is where Java steps in to save the day.

Sun Microsystems Java

Sun Microsystems designed Java with the following criteria in mind:

- Simple, object-oriented and familiar

- Robust and secure
- Architecture neutral and portable
- High performance
- Interpreted, threaded and dynamic

Java is a programming language which is syntactically similar to C++, but has some of the less useful functionality removed. This tool picks up where HTML leaves off by allowing for small applications, called "applets", to be developed. These applets can draw arbitrary shapes and text, create data connections, accept user input and generally perform those functions necessary to produce dynamic, interactive content on Web pages.

Applets are called out in a standard HTML page just like a bitmap graphic or link to multimedia data. Any parameters which are important to the applet are described along with the applet citation. When the page is loaded, the applet is requested and downloaded from the server. Once retrieved, the applet is validated by the browser and begins to execute on the *client* machine. Normally, the applet will continue to run until it exits programmatically or a new page is loaded.

4 NETWORK MANAGEMENT WITH A WEB BROWSER

As previously mentioned, the technologies that have been driven by the explosive growth of the Internet and World Wide Web sites are well suited for network management applications. These tools provide a way for fairly inexperienced computer users to publish electronic information content with some truly impressive results. In this section, two practical applications of these tools are illustrated.

Element Management using HTML

In order to perform element management, two key functions must be possible:

- the ability to read any and all configuration information about the element
- the ability to set the state of any and all configuration parameters for the element

The easiest way to gain this functionality is to leverage the existing network management protocols deployed in the elements. A sample system using SNMP as the managing protocol is depicted in Figure 1.

In this implementation, a Web server is actually integrated into the managed device. This Web server does not significantly differ functionally from the Web servers freely available via the Internet or those sold commercially. In addition to a set of predefined HTML *forms* that contain configuration templates for the managed device, a Common Gateway Interface (CGI) application is installed. The CGI program in this instance is referred to as the Command Processor since its sole function is to accept commands from the user and generate the appropriate configuration messages.

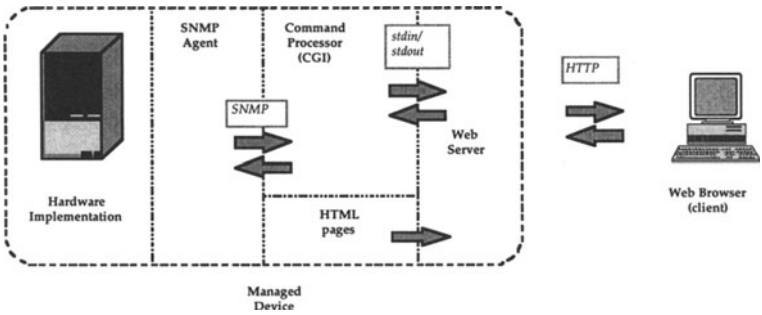

Figure 1 Web CGI-based Element Management

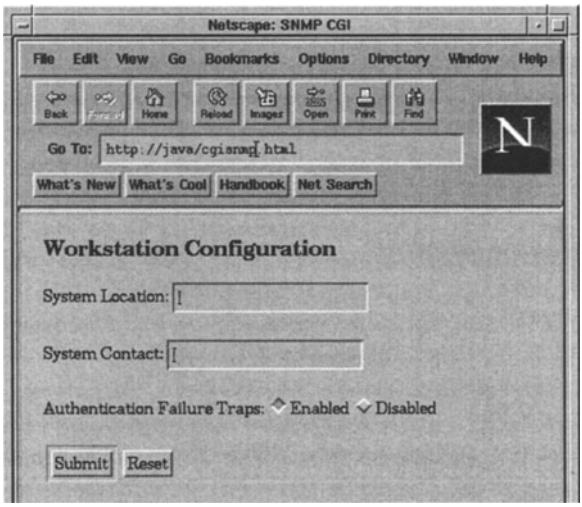

Figure 2 A simple HTML form

In this case, the Command Processor accepts name/value string pairs through the *stdin* logical device. It then parses these strings and generates the appropriate SNMP SET and GET commands. Any result codes and values returned from these commands can then be formatted into HTML and returned to the client browser (via the Web server) through the *stdout* device. The string pairs "posted" by the HTML form are really just the name and value of all the user input fields on the form. Figure 2 shows an extremely simple example of a form created to set some of the variables in a MIB for a UNIX workstation, the *sysLocation*, *sysContact* and *snmpEnableAuthenTraps*. Figure 3 shows the underlying HTML file the browser uses to build this page.

The form shown in Figure 2 allows a user to type in desired strings for the *sysLocation* and *sysContact* field using text input boxes. The *snmpEnableAuthenTraps* field is controlled by radio buttons (mutually exclusive). For the new values to be sent to the Web server and thereby set on the target device, the user must press the "Submit" button at the bottom of the screen. The "Reset" button initializes the form to its original state.

```
<HTML>
<HEAD>
<TITLE>SNMP CGI</TITLE>
</HEAD>
<BODY>
<H2>Workstation Configuration</H2><P>
<FORM ACTION="cgi-bin/cmdproc" ENCTYPE="x-www-form-encoded" METHOD="POST">
System Location:  <INPUT TYPE=text NAME=".iso.org.dod.internet.mgmt.mib-
2.system.sysLocation.0" ><P>
System Contact:  <INPUT TYPE=text NAME=".iso.org.dod.internet.mgmt.mib-2.system.sysContact.0"
><P>

Authentication Failure Traps:
<INPUT TYPE="radio" VALUE="1" NAME=".iso.org.dod.internet.mgmt.mib-
2.snmp.snmpEnableAuthenTraps.0" CHECKED=true> Enabled
<INPUT TYPE="radio" VALUE="2" NAME=".iso.org.dod.internet.mgmt.mib-
2.snmp.snmpEnableAuthenTraps.0"> Disabled <P>

<INPUT NAME="Button" TYPE="submit" VALUE="Submit">   <INPUT NAME="name"
TYPE="reset" VALUE="Reset">
</FORM>
</BODY>
</HTML>
```

Figure 3 HTML source file for form page

In Figure 3, the most important elements of the form are highlighted for clarity. The first line defines what action is to be taken upon the "Submit" button being activated:

```
<FORM ACTION="cgi-bin/cmdproc" ENCTYPE="x-www-form-encoded" METHOD="POST">
```

Here, the form will "POST" the entered data to the Web server using the "x-www-form-encoded" MIME encoding format. The exact details of this encoding are not important for this discussion. The "ACTION" field directs the server to launch a program called *cmdproc* and pass the posted data to it.

The next fields are the actual data entry fields:

```
System Location:  <INPUT TYPE=text NAME=".iso.org.dod.internet.mgmt.mib-
2.system.sysLocation.0" ><P>

System Contact:  <INPUT TYPE=text NAME=".iso.org.dod.internet.mgmt.mib-2.system.sysContact.0"
><P>

Authentication Failure Traps:
<INPUT TYPE="radio" VALUE="1" NAME=".iso.org.dod.internet.mgmt.mib-
2.snmp.snmpEnableAuthenTraps.0" CHECKED=true> Enabled
<INPUT TYPE="radio" VALUE="2" NAME=".iso.org.dod.internet.mgmt.mib-
2.snmp.snmpEnableAuthenTraps.0"> Disabled <P>
```

There are two "text" entry fields and a pair of "radio" buttons. As mentioned previously, each field has a name and a value. In order to allow the command processor to be generic in implementation, the "NAME" field of each input is actually set to be the fully qualified SNMP MIB name of the object to be set. By using this naming convention, the Command Processor application can be almost completely oblivious to the device being managed and can therefore be freely re-used for many products without

modification. The radio buttons, it may be noted, use the same name for each button. This is a common convention of HTML forms so that dependent controls can be grouped together. More importantly, however, each button has a predefined value associated with it (1 and 2, respectively), which correspond with the acceptable values the MIB object can be assigned. These values are not what are shown on the screen, however. Instead, HTML allows more familiar labels to be used ("enabled" and "disabled"), but the numerical values are what actually get submitted to the Command Processor.

Finally, the line,

```
<INPUT NAME="Button" TYPE="submit" VALUE="Submit">  <INPUT NAME="name"
TYPE="reset" VALUE="Reset">
```

is used to create the "Submit" and "Reset" buttons.

Assuming that the user sets the form fields to the following:

System Location: MyOffice
System Contact: MCM
Authentication Failure Traps: Disabled

the following string of characters concerning the form content will be submitted to the Web server and then passed on to the Command Processor:

```
.iso.org.dod.internet.mgmt.mib-2.system.sysLocation.0=MyOffice&.iso.org.dod.internet.mgmt.mib-
2.system.sysContact.0=MCM&.iso.org.dod.internet.mgmt.mib-
2.snmp.snmpEnableAuthenTraps.0=2&Button=Submit
```

Name/value pairs are concatenated and delimited using the "&" symbol so they can be easily parsed. In addition, each name is separated from its corresponding value by an equals sign. The reader may note that the name and value of the "Submit" button was passed as well. This is normal and can (and should) be ignored by the Command Processor.

Element Management using HTML and Java

For situations where only static information is to be displayed, the above example is adequate and versatile. For more sophisticated element management, where information needs to be periodically or asynchronously refreshed, HTML and CGI alone are not sufficient. The first reason for this, as noted previously, is that HTTP does not hold connections to the browser open and therefore has no way of sending updated data when it is available. Secondly, HTML only really permits text and bitmap images to be specified and has no mechanisms for drawing arbitrary graphics such as lines, circles, etc. Together, these limitations make the creation of dynamically updating graphs, charts and text exceedingly difficult and cumbersome

Using a mixture of HTML and Java applets, however, quickly eliminates this dilemma. HTML code can be used to produce the static fields and fixed hyperlinks that would be useful for headers, links to on-line help and other relatively unchanging information. Java

applets, alternately, can be used for a myriad of tasks including context-adaptive forms, real-time graphs and charts as wells as alarm lists. A simple example of a combination of HTML and Java applets can be seen in Figure 4.

Shown here is the shelf-level view of the StrataCom AXIS interface shelf, a high density device for aggregating user traffic into a broadband ATM network. HTML "frames", a way to divide the browser window into logical panes, have been used to provide the user with a toolbar of common functions. Each button on the toolbar can transport the user to different views of this element including alarms, configuration, diagnostics and statistics pages. The buttons themselves are simply bitmap images that are defined as hyperlinks to the appropriate supporting HTML pages.

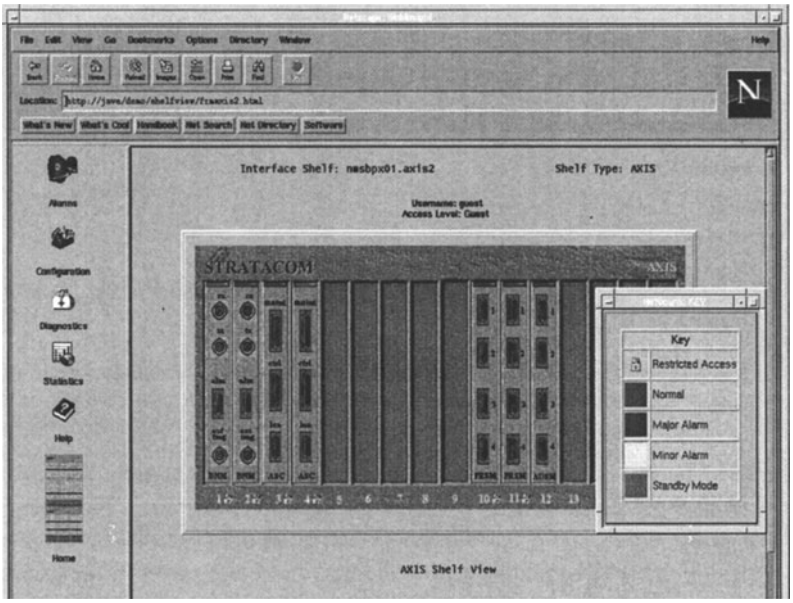

Figure 4 Java and HTML example

A Java applet has been used to build up the actual device view on the right side of the page. The applet scans the SNMP MIB of the AXIS device. It determines how many cards are installed, what slot they are installed in and what type of card they are. Empty slots are shown as blanks. After discovering the cards present, appropriate images of those cards are rendered in the correct positions along with the physical ports and their states. For instance, slots 10 and 11 have four port T1 FRSM (Frame Relay Service Module) cards installed. All four ports are shown in green to indicate they are active and operating normally. Slot 12 has an AUSM (pronounced *awesome*) interface, the ATM UNI Service Module, installed in it, again with all ports showing normal status. When the status of any port or card changes or when a card is inserted or removed, the applet will automatically update the state based on an SNMP trap, changing to red if in major alarm, for example.

Most importantly, each card and port (and even the shelf itself) shown is actually a hyperlink to pages about that object. As the applet builds the view of the card shelf, the page links are automatically created. When the user moves to a new page, the toolbar hyperlinks are updated to represent pages for alarms, configuration, statistics, diagnostics and help for the correct object. In this way, a cohesive, integrated user interface can be presented to user at every level of device management.

Figures 5a-c show the HTML pages responsible for producing this interface.

```
<HTML>
<head>
<TITLE>AXIS Shelf Home page</TITLE>
</head>
<FRAMESET COLS="15%,85%">
<FRAME SRC="buttons.html">
<FRAME SRC="axis.html">
</FRAMESET>
</HTML
```

Figure 5a axisshelf.html - The main page

```
<HTML>
<CENTER>
<A HREF="alarms.html"><IMG BORDER=0 SRC="images/alarm.gif"></A><P>
<h2>Alarms</h2><P>
<A HREF="config.html"><IMG BORDER=0 SRC="images/netconfig.gif"></A><P>
<h2>Configuration</h2><P>
<A HREF="diags.html"><IMG BORDER=0 SRC="images/diags.gif"></A><P>
<h2>Diagnostics</h2><P>
<A HREF="stats.html"><IMG BORDER=0 SRC="images/report.gif"></A><P>
<h2>Statistics</h2><P>
<A HREF="help.html"><IMG BORDER=0 SRC="images/q_mark3.gif"></A><P>
<h2>Help</h2><P>
</CENTER>
</HTML>
```

Figure 5b buttons.html - The toolbar page

```
<HTML>
<CENTER>
<H1>StrataCom AXIS Shelf</H1>
<P>
<APPLET CODE="AXISsnmp.class" CODEBASE=classes WIDTH=800 HEIGHT=300>
<PARAM NAME="LEVEL"
  VALUE="topshelf">
<PARAM NAME="MIBS"
  VALUE="axis.mib">
<PARAM NAME="STATE_NORMAL"
  VALUE="GREEN"
<PARAM NAME="STATE_MAJOR_ALARM"
  VALUE="RED">
<PARAM NAME="STATE_INACTIVE"
  VALUE="BROWN">
<PARAM NAME="UPDATE_BY_EXCEPTION"
  VALUE="true">
</APPLET>
</CENTER>
</HTML>
```

Figure 5c axis.html - The shelf applet page

Figure 5a defines two frames, dividing the page vertically into sections that maintain proportions of 15 and 85 percent of the width of the page. The button toolbar is loaded

into the smaller frame along the left edge of the window. The AXIS shelf applet page is loaded in the larger frame on the right.

Figure 5b shows the HTML code responsible for displaying the five button images and text labels in the toolbar frame page. Each is defined as a hyperlink to a page with more functionality related to each category.

The page shown in Figure 5c has some arbitrary header text and then the applet reference, signified by the "APPLET" keyword. Java applets have several standard parameters including the name of the class being used, where the class is located within the Web server directory and the dimensions of the applet on the page. These parameters are specified using the "CODE", "CODEBASE", "WIDTH" and "HEIGHT" keywords, respectively. There are other standard parameters, but they will not be addressed here. Applets can also have parameters defined for them by the applet designer which end users can then use to adjust the operation of the applet. The shelf Java class used here, *AXISsnmp.class*, has several useful parameters, some of which are called out in Figure 5c. The "LEVEL", "MIBS", "STATE_NORMAL", "STATE_MAJOR_ALARM", "STATE_INACTIVE" and "UPDATE_BY_EXCEPTION" parameters are used to tune the applet's functionality to the user's needs. These parameters are defined as follows:

- LEVEL - Starting view level of the applet when it is first run. In this case, the "topshelf" view is specified, indicating the uppermost shelf-level view should be used. Lower level views of specific cards/slots can also be specified.
- MIBS- The MIB file(s) to be loaded for use with this device. Any SNMP commands will be correlated with these MIBs to generate the appropriate messages. More than one MIB can be specified, multiple file names being delimited by the "|" character.
- STATE_NORMAL, STATE_MAJOR_ALARM and STATE_INACTIVE - These allow the user to set the color to be used when a device is in each of these states. Color names can be specified or RGB values. STATE_MINOR_ALARM also exists but is not shown. If values are not explicitly called out for each value, standard default color mappings are used.
- UPDATE_BY_EXCEPTION - Defines whether the display is updated by exceptions (traps) or if the device is periodically polled. Option POLL_RATE is used to defined the polling interval if this value is set to "false".

5 SERVICE MANAGEMENT WITH A WEB BROWSER

Using Web technology to perform service management functions is fairly straightforward. Most service management systems are actually based on the information collected in the network management layers: elements, ports, trunks, connections and all other manageable parameters. The service management system then manipulates this information and provides service *views* into it.

For example, a common requirement that can be fulfilled by Web technology is that of customer network management. Customers wish to be able to view and, in some cases,

control the service they have purchased from a network service provider. To accomplish this, a service provider must first and foremost deliver an isolated view of the overall network, showing only those items in use by the particular customer. This information is usually available from vendor management stations in some form of database, but the actual delivery of this data to the customer can (and in most cases *has*) become the major stumbling block for most service providers.

Customer service data is typically offered to customers as access to an SNMP proxy MIB. Many customers of these services, however, have little or no interest in spending the time and money to integrate to this interface to get meaningful information. That is not to say no customers want a purely SNMP-based interface; many sophisticated corporate IT groups are willing and able to make the investment necessary to get a collective view of their local network and the purchased service. The vast majority of service purchasers, however, are only interested in running their business efficiently and have no time for generating customer management applications, but still require the information such systems deliver at their fingertips.

The dilemma that most service providers face, therefore, is that each customer wants something different in the way such a system would look and work. Many only want very simple performance charts they can review to assure themselves they have gotten what they have paid for. Others want alarms, topology views, near real-time performance statistics and control capabilities. And, of course, all of these features have to be very customizable, available on every hardware platform from low-end PC's to workstations and have to be "plug and play" for the least experienced users. Given the need to deliver the most cost-effective service, the intensely competitive market and staff shortages at most network operators, these requirements are nearly impossible to meet for the majority of service providers. Only technologies such as HTML and Java can allow many of these goals to be accomplished within the timeframe available.

Service Management using HTML and Java

Many times, aspects of service management require the dynamic qualities provided by Java combined with HTML. As an example, Figure 6 shows a sample page from a fairly sophisticated customer network management system. This page has a connection topology map which shows the state of customer connections between endpoints around the country. The customer also has a form at the bottom of the page that allows the creation of additional connections within the network.

First, the topology map shows five connected cities: Portland, San Jose, Los Angeles, Boston and Atlanta. The fictitious customer has offices in each of these locations and different types of network connections including voice, frame relay and ATM running between them. The service provider actually has many other nodes, but the customer is only allowed to see those in use by their service. Further, there are several intermediate nodes between the endpoints shown (such as might exist between Los Angeles and Atlanta), but only the start and end of each *virtual customer* link is shown. A failure of one of these transitional elements will be shown as a link failure between the endpoints. The operational status of the links carrying these connections and the connections themselves are reflected on this map by the color of the nodes and interconnecting lines.

All of this information is gathered, updated and displayed in real-time by a Java applet communicating with a management server at the service provider location.

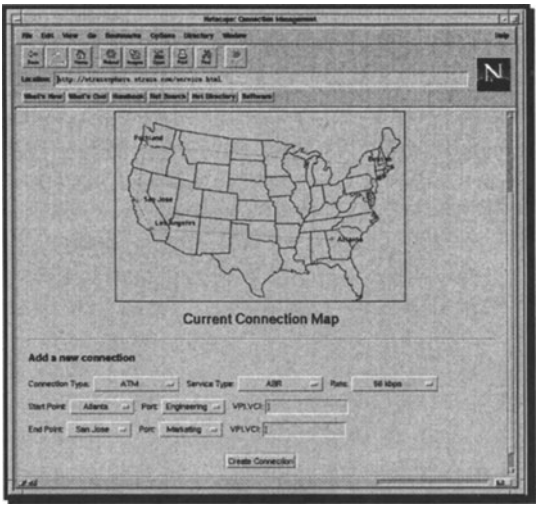

Figure 6 Service management with HTML and Java

Second, the user can add new connections to the service without having to directly involve the service provider by using a context-adaptive provisioning applet. In actuality, the service provider is always in control of the process since no commands issued by the user will be *directly* sent to the service network. Instead, these commands will be handled by an intermediate proxy located at the service provider facility which will determine if a customer has the appropriate privileges to execute any given commands. As can be seen from the figure, the customer can create ATM (CBR, VBR, ABR, etc.) as well as frame relay, voice and other types of connections. When the user chooses a particular connection type, the remainder of the input fields adapt to reflect information needed to complete the operation. In this case, the user need only select the endpoints of the connection, the desired port and an appropriate virtual path identifier (VPI)/virtual circuit identifier (VCI) pair. If this had been a frame relay rather than an ATM connection, the form would have changed to prompt the user or a DLCI (Data Link Connection Identifier) rather than the VPI/VCI needed for ATM.

The endpoints are limited by the applet to the cities predefined by the customer. Ports are given meaningful names such as "Engineering" and "Marketing" and are restricted to those ports on equipment already in use by the customer. By imposing these restrictions the service provider is assured that the customer cannot accidentally attempt to make changes to equipment not assigned to them. Additionally, customers will appreciate these limits since it not only helps reduce mistakes, but also gives them easily remembered names and only a few choices to make rather than having to recall potentially obscure node names and port numbers assigned by the service provider.

Finally, the customer can choose an appropriate data rate from a set of common values. Once all parameters are set to the user's liking, a request to set up the connection can be sent to the service provider by pressing the "Create Connection" push-button below the form. The actual implementation of what happens next is completely up the service provider. The information from the customer request may be sent to a service proxy, checked for validity, converted to the appropriate management protocol messages and sent to the network. Alternately, all of the information could be formatted and sent by electronic mail to a service provider human operator. The operator could then evaluate the request and manually create the network connection. How the actual connection is provisioned is completely within the discretion of the service provider and generally unimportant to the service user as long as the results are reasonably timely.

6 CONCLUSION

Clearly, the technologies driven by the World Wide Web are quite powerful and potentially useful for a variety of tasks, including network element and service management. This is not to say that it should be considered a complete replacement for the huge amount of management software and protocols that exist today. Nor by any means are HTTP and Java the right choice for every application. These tools are, however, uniquely qualified to meet a large number of the requirements of network and service management applications due to their ease of use, low cost, reliance on standards, flexibility, platform independence and security.

Web technology delivers on the promise that a developer can create one set of programs and content material which will run on virtually any machine and operating system. This functionality is what should be exploited by service providers most and wherever possible. Web servers and browsers should not be seen as a competing approach to the existing management protocols, but instead a way to deliver the information gained through such protocols to a wider audience. By building upon all the work that has already been done in the standard management interfaces, service providers can use the power of Web-based management to improve their internal processes and deliver the highest quality services to their customers.

7 BIOGRAPHY

Michael Maston is a network management product line manager within the WAN Business Unit (formerly StrataCom) at Cisco Systems in San Jose, CA. His primary responsibilities currently include defining requirements for device management for Cisco's wide area ATM switches and Web-based network management solutions. Prior to working for Cisco, Mr. Maston spent 8 years as an electrical engineer developing hardware and software for Ford Aerospace, GTE Government Systems and Landis and Gyr Energy Management Systems. He holds a Bachelor of Science in Electrical Engineering from Santa Clara University in Santa Clara, CA.