Web Based Management with Lua

Edison Ishikawa
ishikawa@ime.eb.br
Departamento de Engenharia de Sistemas – IME
Praça Gen. Tibúrcio, 80, Praia Vermelha
22290-270 — Rio de Janeiro — RJ

Noemi Rodriguez
noemi@inf.puc-rio.br
Departamento de Informática — PUC/Rio
Rua M. S. Vicente 225, Gávea
22453-900 — Rio de Janeiro — RJ

Abstract

This paper describes an environment, based on the programming language Lua, for the development of web based management applications. The paper discusses the main features of the environment and presents an example web based application, LuaWebMan. LuaWebMan provides a set of example management tools, but its more interesting feature is support for dynamic extensibility; the user can redefine and extend the functionality of the tool through the management application itself.

keywords:

web based management, scripting languages, extensibility

1 Introduction

Management applications typically run on machines with powerful graphical and processing facilities. While this makes sense in the context of large corporate networks, it may be too high a demand on smaller environments. With the growth of network use in small companies and groups, lighter management solutions have become a necessity. The world wide web is a natural place to look for a solution; it offers a standard graphical interface on any platform where a HTML browser is available. The web paradigm separates processing (done at a server machine) from visualization (conducted through a browser), allowing any machine on a network to act as a management console.

However, development of web based network management applications is not trivial. Conversion of reports or graphs to HTML format is simple, but that does not, in general, answer the needs of administrators; often, direct access to the managed devices and services is required. This requirement can be met through the use of dynamic web pages.

Network management needs are highly specific to groups or corporations. Different goals, such as maximum performance or maximum security, may imply in different requirements as to consolidated data and statistics. Once again, this may be no problem for large corporations, who may have a specific application built according to their needs. Besides being expensive, this requires a very precise idea of the managed network and of the management needs in the intended lifescope of the management tool. For a more exploratory approach, it is important to have tools which allow easy construction of specific management applications. If an interpreted language is used as a basis for this construction, it is also possible to dynamically extend the management application with new functionality. This approach, which allows a basic application to be designed and distributed and specific facilities to be added as needed, has been used in several platforms for management application development [Sch97, RLMS98, RM95].

In this paper, we present an environment for the development of web based management applications. This environment is based on the interpreted language Lua [IFC96], and uses only standard web technology. This allows resulting applications to be used through any standard web browser. The paper also presents a sample application, LuaWebMan, which was built to evaluate the development environment. One of the main goals of this evaluation was to verify whether a web based application could provide a degree of extensibility similar to what can be provided by a management application with a conventional architecture. So, besides integrating some common management facilities, LuaWebMan can be customised by the user (typically a system administrator), with the dynamic integration of new scripts and menu items. Many system administrators have some programming background. This is acknowledged in LuaWebMan by allowing extensions to be developed in the same environment that is used for managing the network.

Section 2 discusses the different kinds of web based network management applications currently available. In section 3, we present our development environment. Section 4 describes LuaWebMan. Finally, section 5 contains some final remarks and points for future work.

2 Web Based Management Architectures

Current web based network management applications can be classified according to their functionality, which is closely related to their implementation [Ste97, Ish98]. This classification leads to the following groups:

Group 1 HTML formatting applications

Group 2 Embedded applications

Group 3 Gateway applications

Applications in the first group do not have direct access to the source of management information (typically, SNMP agents); basically, they convert information made available through the use of conventional management tools to HTML format. After that, this information can be accessed through a browser. This kind of aplication is very simple, but useful, however they suffer with the lack interactivity with management agents. Examples are management platforms providing tools for HTML report generation. Also in this category are applications that dynamically access databases generated by management applications.

The second group refers to systems where each managed device runs its own HTTP server. The HTTP server acts as an agent, directly generating the requested (MIB) information. In this category falls all web based management applications that comes embeded in a network equipment, such as a router. Nowadays most vendors offer this functionality as a plus. For instance, an application that falls in this category is ClickStart [Com97], developed by Cisco to configure and monitor its own line of ISDN routers. This kind of application is good to manage one equipament, once per time. To manage two or more equipaments as a set is necessary an application that's able to comunicate with other management agents than only itself.

In the third group of applications, the HTTP server acts as a gateway between agents (SNMP or other) and HTTP clients. In this case, the HTTP server runs scripts which access interactively arbitrary agents, one or more per time, typically using SNMP. LuaWebMan is an example of third group aplication. A good related work to cite is IntraSpection. Like LuaWebMan, it's a third group aplication, developed by AsanteŤecnologies [Tec97]. However, IntraSpection portability is restrict to Windows NT plataform and is not extensible to support private MIBs or other functionalities, rather than agregating personality modules written in C for third part vendors.

SNMP/HTTP gateways

A SNMP/HTTP gateway could be implemented in a number of ways, for example, extending a conventional network management application to act as a HTTP server. However, the use of standard CGI programs offers a series of advantages, such as the portability of the applications across several implementations of the HTTP server. This is the reason why ISAPI (Internet Server Application Program Interface) from Microsoft and others proprietary tecnologies wich extends the HTTP server were not considered to implement LuaWebMan. The goal in this work is to be close to standards. Using standards the application has more chance to get portability across hardware and software plataforms.

The architecture of a management application based on CGI programs is shown in Figure 1. A standard WWW browser acts as a management application window, and accesses a HTTP server (using HTTP). According to the client's selection, the HTTP server activates a specific CGI program, which, if necessary, sends requests to SNMP agents using the SNMP protocol.

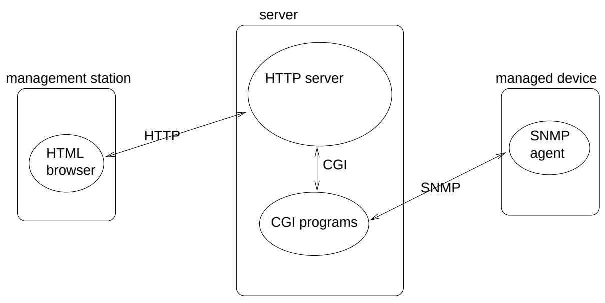


Figure 1: Management application based on a SNMP/HTTP gateway

As in conventional management applications, in this architecture the computation related to management activity is concentrated in a single machine, the HTTP server. However, the management activity itself can take place anywhere on the network, through a standard HTML browser. The standard graphical facilities made available with WWW can be used directly in the management application, with no need to generate specific graphical interfaces. Moreover, this approach takes advantage of all the existing SNMP support.

3 The Lua Management Platform

Lua [IFC96, IFC] is a general purpose configuration language developed at PUC-Rio. Lua integrates in its design data-description facilities, reflexive facilities, and familiar imperative constructs. On the traditional side, Lua is a procedural language with usual control structures (*whiles*, *ifs*, etc.), function definitions with parameters and local variables, and a Pascal-like syntax. On the less traditional side, Lua provides functions as first order values, and dynamically created associative arrays (called *tables* in Lua) as a single, unifying data-structuring mechanism. Garbage collection is also provided.

Tables are values that can be indexed not only by integers, but by strings, reals, tables, and function values. As an example, the sequence of commands

```
sysUpTime = {}
sysUpTime{"ObjectId"} = 3
sysUpTime{"Type"} = "TimeTicks"
sysUpTime{"Access"} = "ReadWrite"
```

```
sysUpTime{"Description"} = "The time since the network
  management portion of the system was last reinitialized"
```

creates an empty table and adds arbitrary fields to this table. Tables can also be created directly with some values, as in:

```
mib2={"system", "interfaces", "at", "ip", "icmp", "tcp", "udp", "egp",
        "transmission","snmp"}
```

(this implicitly associates the strings “System”, “Interfaces”, etc., to indexes 1, 2, ...). Syntactic sugar is provided to allow the programmer to use the familiar `sysUpTime.Type` syntax instead of `sysUpTime{"Type"}`.

We are currently using Lua as a basis for the development of network management applications [MILR98, RLMS98]. One very interesting implication of the use of an interpreted language in this context is the possibility of incorporating new functionality with no need for recompilation.

The choice of Lua, instead of the traditional Tcl/Tk for network management or Perl in Web arena, was motivated by several points. In the first place, it is possible to use Lua with many different levels of sophistication. Because of its simple syntax, Lua can easily be used by novice users for simple configuration tasks, such as automatic determination of fonts and window sizes. On the other extreme, the language offers some very powerful programming mechanisms [IFC96], which can be used by more technically advanced programmers. In second place, it is very easy to provide new libraries for Lua, due to the facilities available for interfacing it with C. This feature was important to us, since it was necessary to develop a management library, LuaMan [RLMS98], providing access to SNMP, DNS, and ICMP. LuaMan is briefly described in section 3.1. In third place, a number of relevant libraries are already available for Lua; these include, for example, LuaOrb [ICR98], which provides access to CORBA objects, CGILua [HBI98], which supports the construction of dynamic web pages in Lua, and DBLua [dbl], which provides access to ODBC databases. Section 3.2 presents a brief overview of CGILua, which was used in the development of our web based application.

3.1 LuaMan

The construction of network management applications requires access to several relevant network services. Access to SNMP (Simple Network Management Protocol) operations allows management applications to interact with SNMP agents and so manipulate the management information stored in their MIBs. Access to ICMP provides mechanisms which can be used for managing devices which do not support SNMP. ICMP messages are also used in the classical algorithms for tracing routes and IP network discovery [SL93]. Another facility which is also extremely useful in a network management environment is access to DNS (Domain Name System) services.

Access to SNMP and other relevant network services was achieved in Lua through the development of LuaMan, a library which provides Lua with a network management API. Several features of LuaMan were based on Tnm, the Tcl Extensions for Network Management implemented in Scotty [Sch97].

The current implementation of LuaMan is based on the Carnegie Mellon CMU SNMP Library. The LuaMan library can be ported to a wide range of platforms. It is currently running on Linux, Solaris, SunOS, IRIX and AIX. Work is under way for porting LuaMan to Windows NT.

3.2 CGILua

CGILua [HBI98] is a tool which supports dynamic web page generation by decoding data sent to forms and simplifying dynamic HTML page construction. Two kinds of files are supported by

CGILua: pure Lua Scripts and mixed HTML files.

A pure Lua script is simply a Lua program; when this file is activated, the Lua program is executed, and its output is interpreted as a HTML document. Figure 2 presents an example of pure Lua script, which uses LuaMan functions to implement a classical *MIB walk*.

```
write("Content-type: text/html\n\n")
write("<HTML>")
write("<HEAD>"
write("<TITLE>CGILua example: MIB walk</TITLE>")
write("</HEAD>")
write("<BODY>")
write("<H1>Objects in the system subgroup -  MIB-II</H1>")
write("<HR>")
write("<TABLE>")
snmp_session = snmp_open{peer="imperio.telemidia.puc-rio.br",
root="system"
varBind = {object_name=root}
repeat
   varBind = snmp_getnext(snmp_session, varBind)
   if varBind then
      if not strfind(varBind.object_name, root) then
         mibEnd = true
      else
         write("<TR><TD>",varBind., "</TD></TR>")
      end --if not strfind
   end  --if varBind
until (not varBind) or mibEnd
snmp_close(snmp_session)
write("</TABLE>")
write("</BODY>")
write("</HTML>")
```

Figure 2: CGILua pure Lua script

A mixed HTML file is a template: a HTML document with escape marks which indicate fields that are handled by the CGILua preprocessor. The HTML document can be manipulated with any HTML editor, allowing designing concerns to be separated from programming. Three kinds of fields are supported by CGILua. *Statement* fields, surrounded by the escape marks `<!--$$` and `$$-->`, contain Lua chunks of code which are executed as normal Lua scripts. To generate any values to the final page, they must explicitly write these values. *Expression* fields, surrounded by the escape marks `$|` and `|$`, contain Lua expressions which are evaluated by the preprocessor to obtain a value to be substituted for this field in the final page. *Control* fields indicate parts of the document to be conditionally repeated. Figure 3 presents an example which uses the three kinds of fields. This example describes a page showing all objects in subgroup *IP* of MIB-II.

The `LOOP` construct is analogous to C's *for* statement, causing repetition of all the code between the `LOOP` field and the matching `ENDLOOP`. Fields `start`, `test`, and `action` allow the programmer to control the repetition.

In dynamic web page generation, security is always an important issue. The architecture of CGILua, together with some features of the language itself, allows the server's administrator to define *protected* environments for the execution of CGILua programs [HBI98]. CGILua has two main modules: a *kernel*, written in C, and a *configuration script*, written in Lua. When

```
<HTML>
<HEAD><TITLE>CGILua template example: iterating through group IP</TITLE></HEAD>
<BODY> <H1>traversal of group IP</H1>
<!--$$
function next_object()
     name, errn = mib_fullname(vb.oid)
     still_ip = strfind(name,"ip")
     if (still_ip) then
       vb,err,ind = snmp_getnext(s1,vb)
     end
end
s1,err = snmp_open{peer="200.20.120.198", timeout=10}
$$-->
<HR>
<TABLE BORDER=1 WIDTH=100%>
<TR>
   <TD>ObjectId</TD>
   <TD>ObjectName</TD>
</TR>
<!--$$ LOOP start="vb, err,ind = snmp_getnext(s1,"ip")",
            test=" (err == SNMP_NOERROR) or (still_ip)",
            action="next_object()"   $$--!>
<TR>
    <TD>$|vb.oid|$</TD>
    <TD>$|name|$</TD>
</TR>
<!--$$ ENDLOOP $$--!>
</TABLE>
<!--$$ snmp_close(s1) $$-->
</BODY></HTML>
```

Figure 3: CGILua mixed HTML file

a CGILua page is accessed, the HTTP server activates the kernel, which prepares an execution environment and then activates the configuration script. Among other initialization activities, the kernel creates a facility that allows a Lua program to erase a global function while keeping private access to it. This facility may be used in the configuration script to redefine functions which are considered insecure for remote execution. Redefinition is supported by Lua's treatment of functions as first-class values.

4 LuaWebMan

This section describes *LuaWebMan*, a web based management application developed using CGILua and the LuaMan library. Section 4.1 describes the implemented management tools, and section 4.2 describes LuaWebMan's facilities for dynamic extension.

4.1 Basic Functionality

One of the goals of this work was to evaluate the ease of development of classic management tools using the environment described in section 3. To this end, we selected a basic set of management functions which explore some diversity in implementation. These functions include:

- network discovery and visualization
- traffic monitoring
- MIB browsing
- trap log

Figure 4 shows a LuaWebMan screen, which is organized in three regions. The left column shows the list of currently available commands or menus. The uppermost region is reserved for messages, and the central region is dedicated to network visualization. The network visualization region allows the user to select a device, which will then be regarded as the *current* device by the other management tools.

Network discovery and visualization

Network discovery, registration and visualization is supported by items in submenu "network" of LuaWebMan. The idea is to maintain a set of *known* networks, which have been previously registered by the user, and to allow management operations to take place either on one of these previously known networks or on a new network. Figure 4 shows the network submenu. The user may ask to see previously registered networks or to register a new network. Networks are registered in a nested hierarchy.

The `discovery` menu item allows the user to discover the configuration of a new network and have it registered as part of the persistent hierarchy. Discovery is implemented by using function `icmp_netecho`, from the LuaMan library, which, given a network address and a mask, returns the addresses of active machines on the given network.

Other menu options are available for manually editing the network hierarchy or specific networks, as well as for entering information to be displayed about specific devices.

The selection of a workstation or other device in the displayed network triggers the execution of a script. This script marks the selected device as the *current* device, upon which other applications will operate. Moreover, it tries to get some information from the SNMP agent at the selected device. When successful, the information returned by the agent is displayed in the message area. An example can be seen in Figure 4.

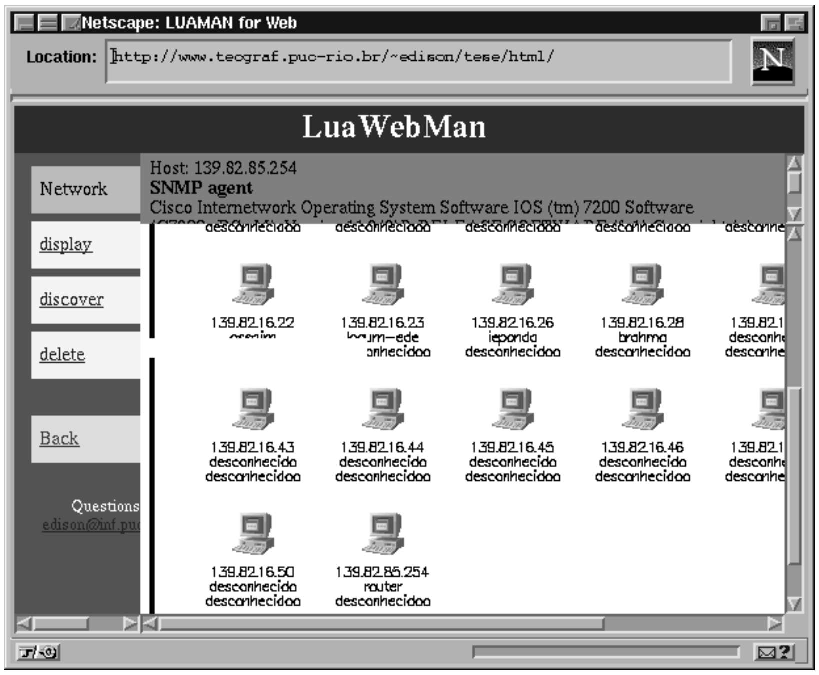
The graphical display of dynamically determined network configurations as *clickable* areas, allowing selection of an icon to trigger a script, is implemented with the help of *ImageMaker* [Spo97], a tool which dynamically generates GIF images and sensitive maps[1] The use of this tool allowed LuaWebMan to obtain a degree of interactivity normally associated to applets using only standard HTML features.

Other Applications

The three other basic applications incorporated into LuaWebMan are grouped under the "Tools" submenu.

The "Traffic" tool monitors incoming and outgoing traffic at a given IP address (as monitored by a SNMP agent at that address). Using HTML's *refresh* facility, a dynamic bar chart is built, showing the total number of received and transmitted bytes in the last 50 cycles (a *refresh* takes place every second). Figure 5 shows an example of this chart. The same procedure could trivially be adapted to show ICMP or SNMP traffic, errors, etc.

[1]Besides network maps, *ImageMaker* also supports the dynamic construction of bar and pie charts as sensitive maps.



Host: 139.82.85.254
SNMP agent
Cisco Internetwork Operating System Software IOS (tm) 7200 Software

Figure 4: LuaWebMan: network display

The bar chart is built using a standard HTML table. This is in contrast to standard display of graphs in GIF or JPEG format. The use of a HTML table greatly reduces the amount of information transmitted from server to browser.

The "MIB Browser" tool is a standard MIB browser, again using HTML tables to display information. A form is used to allow the user to browse through the tree and to request object values.

The "Trap" tool displays a new window with a log of the last traps generated. A separate process is responsible for receiving the traps and logging them on a file. It would be more interesting to have some asynchronous mechanism to allow the tool to generate an alarm whenever a trap occurs. This is difficult to obtain in the standard client-server web environment, as will be discussed in section 5.

4.2 Extending LuaWebMan

The kind of web based feature discussed in the previous section is currently available on a number of tools. The distinguishing feature in LuaWebMan is the possibility of being dynamically extended, and the fact that extension takes place in the same environment which is used for network management. It is not necessary to switch to a different environment in order to build new management functionality.

Two levels of programming are offered to the user for the creation of new functions. On the first level, the user may browse, create, and modify CGILua scripts, as well as explicitly order their execution. On this level, the user may write Lua scripts using any of the available libraries and interactively test these programs. This provides the flexibility of an interactive console; the programs, however, are always executed on the server. LuaWebMan maintains all scripts created by a specific user in a separate area (directory), allowing the user to create new persistent scripts.

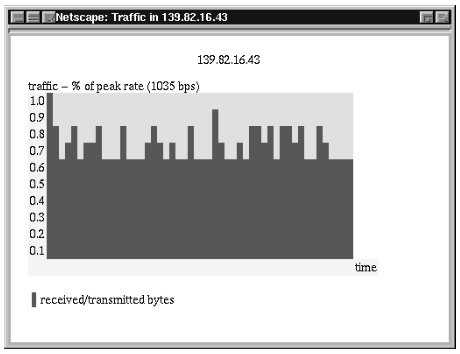

Netscape: Traffic in 139.82.16.43
139.82.16.43
traffic – % of peak rate (1035 bps)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
time
received/transmitted bytes

Figure 5: LuaWebMan: traffic monitoring

This functionality is offered by the items in menu “Scripts”. On a second level, the user may associate existing scripts to existing or new menu items. This functionality is available through submenu “Interface”.

In a typical scenario, to create a new extension, the user would first create and test a new script, and then associate this script to a new menu item. We will now follow this procedure step by step, using as an example the creation of a script which implements *ping*, based on the use of function `icmp_echo`.

```
<HTML>
  <HEADER><TITLE>PING</TITLE></HEADER>
  <BODY BGCOLOR="#FFFFFF">
    <!--$$
        campos={ {txt="ping address: ", type="text", name="ip",
                                        value="0.0.0.0"},
               {txt=nil, type="submit", value="Ok"}
             }
    $$-->
    <FORM METHOD="post" ACTION="ping.lua">
      <!--$$ LOOP start="i=1", test="campos[i]", action="i=i+1" $$-->
          <P> $|campos[i].txt|$
          <INPUT TYPE="$|campos[i].type|$" NAME="$|campos[i].name|$"
                              VALUE="$|campos[i].value|$">
      <!--$$ ENDLOOP $$-->
    </FORM>
  </BODY>
</HTML>
```

Figure 6: New HTML file for data input

To create this function, the user would initially go to the “Scripts” menu and activate the “New” script option and create a script `ping.html` (in this case, a simple HTML form). Figure 6 shows the script. Next, the user would again choose the “New” script option and create the script `ping.lua` (see field `ACTION` in Figure 6), shown in Figure 7. Finally, the user would go to menu “Interfaces”, create a new menu item in menu “Tools”, called “ping”, and associate the selection of this item to the `ping.html` script. Figure 8 shows the dialog for the creation of this association. After this step, menu “Tools” will contain a new item, as shown in Figure 9. The figure shows

```
write("Content-type: text/html\n\n")
dofile("../../cgilua/config.lua")
write("<HTML><HEADER><TITLE>PING</TITLE></HEADER>\N")
write('<BODY BGCOLOR="#FFFFFF>"')
icmp_init()
TripTime, ErrEcho = icmp_echo(cgi.ip)
if ErrEcho == ICMP_NOERROR then
   write("<P>"..cgi.ip..": Time = "..TripTime.."s\n")
else
   write("<P>..cgi.ip..": not responding.")
end
icmp_close()
write("</BODY></HTML")
```

Figure 7: New Lua script for *ping* function

the result of selecting the new item “ping” in menu “Tools”.

5 Final Remarks

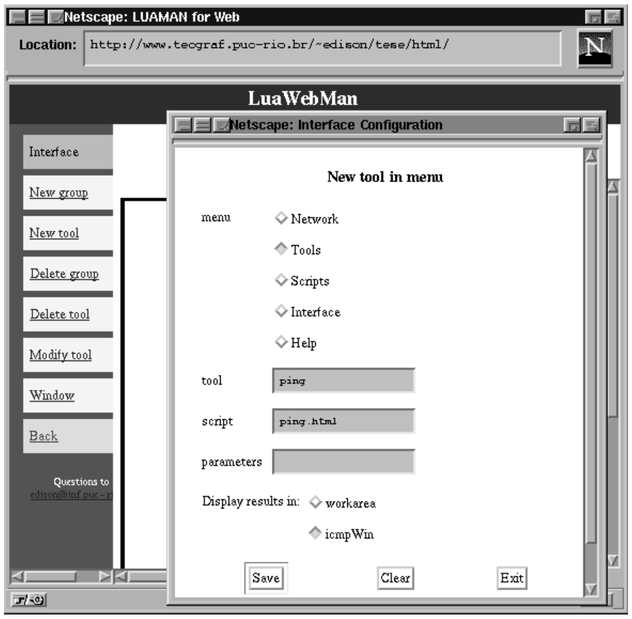
The goal of this work was to evaluate our management application development environment as regards web based management, and to study in what measure we could overcome the limitations imposed by web based management.

Development of LuaWebMan was simple and met with no major problems. A point which makes the ease with which the tool was developed specially relevant is that the programmer of LuaWebMan had no previous experience with either Lua or LuaMan. Although the set of predefined tools in LuaWebMan is small, we believe it fulfils the role of demonstrating the environment's flexibility. Incorporation of other predefined management facilities would be straightforward.

One limitation of the tool is its inability to deal with asynchronous events. A common management facility is alarm generation, in which the system warns the user about newly received traps (in our application, the user must explicitly ask to see the trap log). The behaviour of “warning” the management application is compatible with a producer-consumer communication model, and not with the standard client-server environment model supported by the web.

The extensibility of the tool is certainly its most important feature. To our knowledge, LuaWebMan is the only web based management tool which allows the user to use the tool itself to dynamically create extensions to interface and behaviour. It would be, in principle, possible to implement this facility on any interpreted language with support for network management. However, we believe the idea to be specially useful in the light of Lua's simplicity and flexibility.

The architecture used in LuaWebMan, shown in Figure 1, concentrates all the management information processing in the HTTP server. We are now studying alternatives for distribution. One idea would be to allow the CGI/SNMP gateways to be organized in hierarchies. In this scheme, which requires the CGI scripts to have access to HTTP, the HTTP server shown in Figure 1 would both be accessible directly by a browser, as it is now, and by other CGI/SNMP servers. A *master* HTTP server in a given network would be able to retrieve data directly from SNMP agents, through the CGI/SNMP gateway, and also from *slave* HTTP servers, possibly responsible for collection and consolidation of subnetwork management statistics.



New tool in menu

menu: Network [✓ Tools] Scripts Interface Help

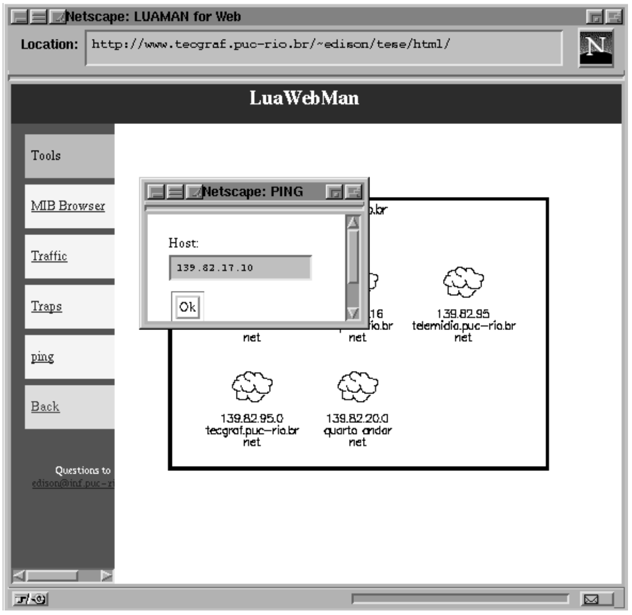
tool: ping

script: ping.html

parameters:

Display results in: workarea [✓ icmpWin]

Figure 8: Dialog for associating a new menu item to a script



Host: 139.82.17.10

Figure 9: New *ping* tool

References

[Com97] Cisco Company. Cisco - clickstart, 1997. http://www.cisco.com/warp/public/728/clickstart/index.shtml.

[dbl] DBLua library. http://www.tecgraf.puc-rio.br/manuais/dblua.

[HBI98] A.M. Hester, R. Borges, and R. Ierusalimschy. Building flexible and extensible web applications with Lua. In *WebNet 98 – World Conference of the WWW, Internet and Intranet*, Orlando, FL, 1998.

[ICR98] R. Ierusalimschy, R. Cerqueira, and N. Rodriguez. Using reflexivity to interface with CORBA. In *International Conference on Computer Languages 1998*, Chicago, 1998. IEEE.

[IFC] R. Ierusalimschy, L. Figueiredo, and W. Celes. The programming language Lua. http://www.tecgraf.puc-rio.br/lua/.

[IFC96] R. Ierusalimschy, L. Figueiredo, and W. Celes. Lua - an extensible extension language. *Software: Practice and Experience*, 26(6), 1996.

[Ish98] E. Ishikawa. Gerência de redes baseada em *web*. Master's thesis, Depto de Informática, PUC-Rio, 1998.

[MILR98] A. Moura, E. Ishikawa, M. Lima, and N. Rodriguez. Aplicações de gerência extensíveis. In *SBRC'98*, Rio de Janeiro, Brasil, 1998.

[RLMS98] N. Rodriguez, M. Lima, A Moura, and M. Stanton. A platform for the development of extensible management applications. In *INET'98*, Geneva, Switzerland, july 1998.

[RM95] M. Rose and K. McCloghrie. *How to manage your network using SNMP*. Prentice-Hall, 1995.

[Sch97] J. Schonwalder. Scotty - Tcl Extensions for Network Management Applications, 1997. http://wwwsnmp.cs.utwente.nl/~schoenw/scotty/.

[SL93] J. Schonwalder and H. Langendorf. How to keep track of your network configuration. *Proceedings 7th Conference on Large Installation System Administration (LISA VII)*, 1993.

[Spo97] Flavio Spolidoro. Imagemaker for network management, 1997. Projeto Final de Curso. Depto de Informática, PUC-Rio. Descrição disponível por www em http://www.tecgraf.puc-rio.br/~spol/imagemaker/.

[Ste97] Steve Steinke. Network management meets the web. *Network*, 12(12):44–50, 1997.

[Tec97] Asante Technologies. Intraspection, 1997. http://www.intraspection.com.