Network Management: Open Source Solutions to Proprietary Problems

Shane O'Donnell
OpenNMS
975 Walnut St., Suite 242
Cary, NC 27511
1.919.368.7415
shane@opennms.org

1. ABSTRACT

The deployment of network management technologies in most college and university environments is similar to network management deployments in large enterprises in the private sector; most contain some remnants of a failed deployment of one of the "framework" tools. However, in academia, there is usually a follow-on deployment of some set of open source network management tools which support daily operations.

This paper will provide a comparative survey of the more popular open source tools, addressing their strengths and weaknesses, and discuss some of the "next generation" open source network & systems management tools. For a definition of "open source", please see [1].

1.1 Keywords

network management, HP, OpenView, IBM, Tivoli, CA, Unicenter, Cabletron, Spectrum, Aprisma, snmp, UCD, CMU, gxsnmp, mrtg, router, performance, monitor, scotty, tkined, network map, cheops, cheops-ng, tcpdump, snort, ethereal, iptraf, libpcap, snmp sniff, rrd tool, rrdtool, event monitor project, emonitor, mon, big sister, opennms, bluebird, synthetic transactions, synthetic transaction, availability, reporting, Java, Swing, XML, XSL, JDBC, Postgres, Oracle, scalable, scalability, distributed, master station, distributed poller.

2. THE PROBLEM

It's a pretty common occurrence. The one person that knows the network inside and out, built it from scratch, and tended to it like it was their progeny, just gave their notice. The company down the street wooed them with promises of higher salaries and shorter working hours. And what's worse, this common occurrence is happening in private industry. Education, with the lowest average compensation by industry for IT professionals (including not-for-profits), doesn't stand a chance in an industry where 80% of employees leave for better pay [2].

But high turnover is a known problem in the industry as well as the educational sector—this is not news. However, one of the situations it creates is at the root of our problem. Infrastructure managers responsible for network availability and reliability, when stymied by loss of key employees, are often provided budgets that allow them to at least partially address their problems. The resulting scenario leaves a responsible manager with a significant budget and a worthy goal, a dependable network. This manager has just made themself a sitting target for the "framework" vendors. Sales personnel from HP, IBM, CA, and Aprisma (formerly Cabletron's network management tool division) [3] sell their tools as "plug and play" solutions, and despite their high price tags, the noble manager pursues the option presented: an integrated solution to provide availability and performance information, event correlation, and automation with the addition of no new personnel roles.

A lofty goal indeed, but according to the Gartner Group, 70% of enterprise management packages are neither fully implemented nor meeting user needs and expectations within three years of their deployments [4]. An analyst from the Hurwitz Group argues in [4] that the number of successful deployments in this same timeframe more closely "approximates zero". Regardless, our example manager's laudable efforts have likely been rewarded with a fragmented, partial solution, while the software support bills continue to arrive. In an effort to save face, the initial target is inevitably lowered to providing basic network management functionality, and in many cases, this involves either augmenting or replacing the costly "framework" solution with no-cost alternatives readily available on the web, or skunk-works projects that provide quick and dirty, temporary solutions to the problem at hand.

One is likely to pose the question, "Why weren't these options pursued up front?". In all likelihood, they probably were, but with some of the inherent drawbacks in the current tools themselves, and a "framework" sales person making regular visits to assure the manager that "you get what you pay for", they were eventually dismissed.

However, we've recently entered a period in which new open source tools have begun to emerge and existing tools are beginning to re-invent themselves, addressing the needs of the larger network installations.

3. THE TOOLS

The existing open source tools can be broken up into five subgroups: SNMP agents, data collection/presentation tools, network mapping tools, network protocol analysis tools, and network and systems monitors

3.1 SNMP Agents

As SNMP is the de facto standard in network management [5], one of the first challenges for deploying an SNMP-centric management system is proliferating SNMP agents to systems which may not provide one by default. In most cases, enterprise-class network hardware ships with some level of SNMP support, however, as many systems are now serving infrastructure roles, whether a departmental Linux-based router or an Intranet server, the deployment of SNMP agents to these boxes is critical to incorporate them in an enterprise management solution.

3.1.1 CMU SNMP

A project originally born at Carnegie-Mellon University by Steve Waldbusser, one of the original authors of SNMPv2 [6], the CMU SNMP project [7] sees little active development. Most of the current efforts come from the Linux CMU SNMP Project, a focused effort to port the CMU SNMP libraries to Linux, led by Jürgen Schönwälder and Erik Schönfelder [8].

The early work of the CMU SNMP project was truly groundbreaking, as it provided the first commercial-grade SNMPv2 libraries under an open source license. However, with the advent of other projects to extend that initial work, the CMU SNMP project has evolved into the AgentX project [9], focused on building a reference implementation of an RFC 2741-compliant SNMP agent [10]. Even with this new focus, the project has been overshadowed by other efforts and has found little interest within the development community.

The latest release from the CMU team was released in October 1998, while Schönwälder and Schönfelder have released updated and more feature-rich versions as late as July 1999.

The CMU SNMP project, its follow-on AgentX project, and the off-shoot Linux CMU SNMP project all focus on delivering SNMPv1 and SNMPv2-compliant libraries and agents, written in C, and ported to most Unix and Unix-like platforms, as well as Windows NT.

3.1.2 UCD-SNMP

The current leader for product functionality, standards-compliance, and adoption speed for new technologies and standards, the UCD-SNMP project's code was initially based on version 2.1.2.1 of the CMU SNMP project [11]. However, since that initial release, the code has been greatly enhanced, with several significant features added, including support for SNMPv3, several command line SNMP tools, and a graphical MIB browser, as described in [11].

The UCD-SNMP project is today the choice of network managers for open source SNMP agents and libraries, and has announced a new release as recently as May 5, 2000.

The UCD-SNMP project, much likes the CMU SNMP project, delivers its RFC 2741-compliant agent and libraries as C source, available under an open source license, and is available for most Unix and Unix-like platforms, as well as both Windows NT and Windows 9x.

3.2 Data Collection/Presentation Tools

It is likely that the area of data collection is where the greatest amount of skunk-works projects begin. This likelihood is most easily argued because data collection in SNMP environments is relatively simple and is easily automated. The UCD-SNMP project provides a command-line tool to execute an SNMP GET transaction, which allows the user to request a specific stored value. Additionally, Perl modules exist for both the UCD -SNMP and CMU SNMP libraries which provide a very easy way to script and automate data collection.

The differences in SNMP data collection with open source tools come in two primary areas: their threading capabilities and their means of storing the data once retrieved. Both of these aspects will be accentuated in the following discussion of data collection tools.

3.2.1 MRTG & RRDTool

MRTG, or the Multi-Router Traffic Grapher, was originally built as a set of Perl scripts by Tobias Oetiker in 1995 [12]. It provides the capability to not only collect data from SNMP-manageable devices, but also to graphically represent that data in an HTML page, so it can easily be web-accessible. MRTG almost single-handedly introduced the network management community to open source tools, and it (and its derivative packages) still reigns as one of the most popular tools deployed in college and university network operations centers.

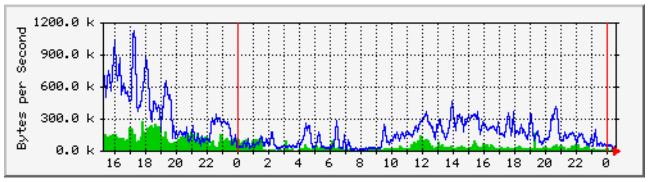


Figure 1. MRTG Graph Example

Due to its original architecture in Perl (pre-threads) and associated performance problems, MRTG suffered from crippling performance problems. Soon after its initial release, Dave Rand emerged as a worthy co-author, and converted the time-sensitive code to C, improving performance by a factor of 40.

However, even with the performance optimization, the MRTG still suffered from basic architectural flaws which have hampered its ability to scale, including its lack of support for threads, its ability to only retrieve and store two pieces of data from each device, and its data storage requirements, which stymies scalability with extremely inefficient reads and writes.

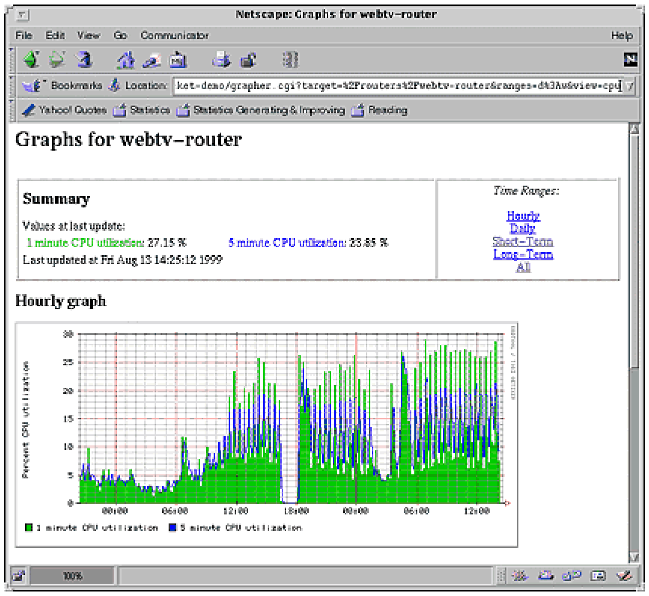
Fully aware of these limitations, Oetiker temporarily suspended development on the next version of MRTG (MRTG 3.0) while work was completed on a new underlying data store for the product, known as RRDTool [13]. RRDTool, an acronym for Round Robin Database tool, provides a fixed-size database with automatic data consolidation, or the packing of time-series data to allow space for new data being acquired. It also provides support for storing more than two values per device (originally intended for input bandwidth/output bandwidth in MRTG), as well as a float data type versus MRTG's support for integer alone [14].

With the introduction of RRDTool and the development of MRTG 3.0 suspended until RRDTool's completion, other tools entered in the marketplace that improved on MRTG and leveraged RRDTool, exactly what Oetiker had hoped to do with MRTG 3.0.

RRDTool is written in C and is available for most Unix and Unix-like platforms, as well as with instructions for building on Windows platforms.

3.2.2 Cricket

Jeff R. Allen's Cricket has proven to be the most popular of the tools built to front-end the RRDTool [15]. A result of Allen's work at WebTV, Cricket emerged due to a combined dependence on MRTG for network information and early signs that MRTG wasn't going to scale to WebTV's rapidly growing network.


Netscape: Graphs for webtv–router
File Edit View Go Communicator Help
Bookmarks Location: ket-demo/grapher.cgi?target=%2Frouters%2Fwebtv-router&ranges=d%3Aw&view=cpu
Yahoo! Quotes Statistics Statistics Generating & Improving Reading
Percent CPU utilization
30
25
20
15
10
5
0
00:00
06:00
12:00
18:00
00:00
06:00
12:00
🟩 1 minute CPU utilization
🟦 5 minute CPU utilization
100%

Graphs for webtv–router

Summary

Values at last update:

1 minute CPU utilization: 27.15 % 5 minute CPU utilization: 23.85 %

Last updated at Fri Aug 13 14:25:12 1999

Time Ranges:

Hourly
Daily
Short–Term
Long–Term
All

Hourly graph

Figure 2: Cricket Web Page Example

While Cricket is certainly a worthy tool for review, it really only duplicates MRTG's basic functionality and leverages the benefits of RRDTool, which itself provides a 1-2x performance improvement over the I/O inefficiencies in MRTG's original database [14]. However, in duplicating MRTG's functionality, Cricket also inherits some of the design flaws that prevent true scalability, including a single-threaded data collector and implementation in Perl, which while impacting performance, pales when compared to the impacts of wait times in a single-threaded collector mechanism. Due to its innovative tree approach to configuration and the ability for multiple instances of Cricket to use differing sub-trees for their configuration source, Cricket can be paralleled in an effort to overcome its threading limitations [14]. This has been thought out by the Allen and his design team, but its actual implementation is still a workaround for a lack of native threading within the application.

If selecting an open source tool for data collection and representation of SNMP-based values, Cricket is today's obvious choice, with some other worthy alternatives mentioned at [15]. The key to all of these tools, though, is the underlying technology provided in Oetiker's RRDTool.

3.3 Network Mapping Tools

Inherently, network mapping tools are fraught with scalability problems, as there is only limited "real estate" on an operator's monitor and as nodes are added and iconically represented, their size, recognizability, and label readability must decline. Additionally, most map-based tools, including the commercial "framework" tools, mandate some amount of operator-intervention for map maintenance at potential costs of hundreds of thousands of dollars [16], or the addition of even more commercial software to automate this role.

Network mapping tools that map based upon the topology of the network also introduce an interesting question: which OSI layer's topology is the right one to use? With the advent of switched networks and Layer2/3/4 switches, the question about topology-based mapping has only gotten cloudier. Most tools that provide topology-based maps, including the "framework" tools, have settled on Layer 3 topology, but even these tools have experienced technical problems and product false-starts in attempting to support Layer 2 topology mapping.

3.3.1 Scotty/Tkined

Written by Leonid Furman, Konrad Zufelde, and Jürgen Schönwälder (referenced earlier as developer behind [7]), Scotty is arguably a data collection tool, as in its own right, it provides a shell with Tcl extensions to access various TCP/IP network related information [17]. It is one of the few open source tools with the ability to query network-related services such as DNS, NTP, and others, as well as being intentionally designed for scripted use with Tcl. However, in its typical deployment, it is coupled with Tkined, or the Tk Independent Network Editor.

Scotty/Tkined together provide the ability to discover and map TCP/IP networks, as well as providing a Layer 3 topological layout, troubleshooting and network monitoring tools. Quite functional in smaller, static networks, Scotty/Tkined enjoys a significant user base.

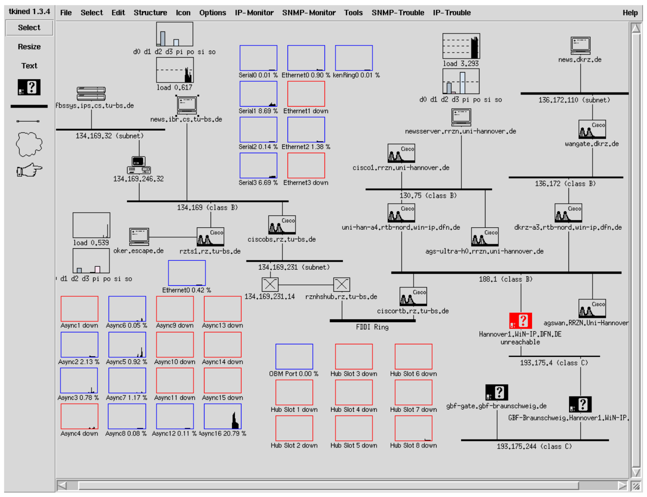

tkined 1.3.4
File Select Edit Structure Icon Options IP-Monitor SNMP-Monitor Tools SNMP-Trouble IP-Trouble Help
Select
Resize
Text
FDDI Ring

Figure 3: Scotty/Tkined Map Example

However, due to critical design flaws, future work on Tkined has been abandoned pending a complete redesign [18], and unfortunately, it appears that the redesign effort has been tabled due to lack of resources.

Current development efforts on Scotty are exclusive to the Tnm Tcl extensions, and the primary focus of the team is to add SNMPv3 support.

Scotty is built in Tcl and requires Tcl 7.6 or higher (Tkined requires Tk 4.2 or higher). It is currently available for most Unix and Unix-like platforms, with limited support for Windows NT.

3.3.2 Cheops

Originally written by Mark Spencer and with any current development being attributed to Adtran, Inc., Cheops claims to be the network "swiss army knife", providing mapping functions on par with HP's OpenView [19]. Cheops provides mapping based on the results of traceroute and mtr [20] and also does limited host identification based on QueSo. With network port scans and host identification built into the product, it is arguably as much a network security tool as a network mapping tool, and is often considered a hacker tool. Spencer acknowledges this use of the product in its accompanying documentation [19], while decrying its use in this way.

Alternative information sources indicate that development on Cheops may have been abandoned [21], and a follow-on project named "Cheops-NG" ("Cheops-Next Generation") is underway [22].

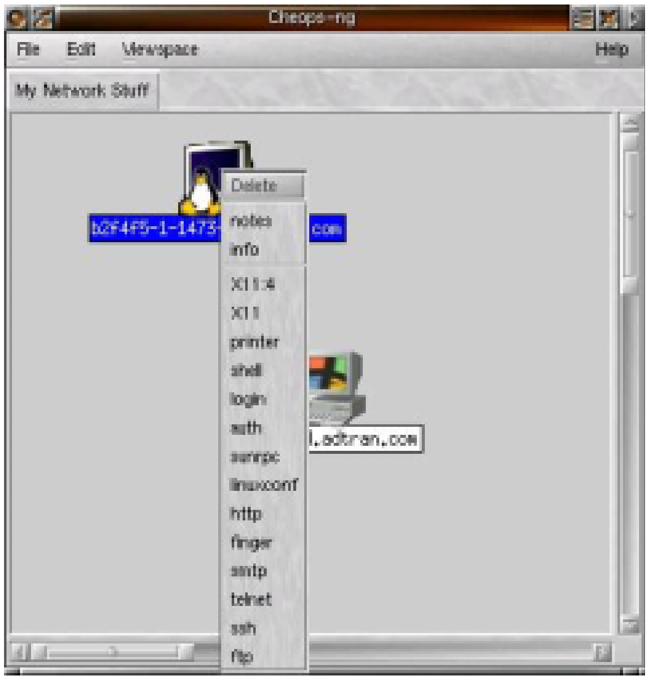


Figure 4. Cheops-NG Example, showing output from default port scan of target device.

3.4 Network Protocol Analysis Tools

The options for open source protocol analysis tools are amazingly broad and as such, several of the more common, relevant, or high quality tools will be mentioned here with brief descriptions, but a detailed comparison within this category is outside the scope of this paper and will not be provided.

3.4.1 tcpdump

Arguably the most common network capture tool, tcpdump is included with most of the major Linux distributions. Written and maintained by the Network Research Group (NRG) of the Information and Computing Sciences Division (ICSD) at Lawrence Berkeley National Labs (LBNL) [23].

Tcpdump is somewhat erroneously named, in that when executed, it listens promiscuously on the default network adapter for not only TCP traffic, but any IP traffic. It also supports command line options to store and replay captures, filter at capture or replay time, and select the adapter to listen on. It is quite configurable, but also quite simple to use for brute force network analysis.

Tcpdump's most important contribution is less the tool itself, and moreso the accompanying libpcap, the protocol capture library for Linux which most other protocol analysis tools for that platform leverage, as referenced in [23].

3.4.2 SNORT

Snort, by Martin Roesch, is a libpcap-based network capture tool, but with intelligence built-in that it not only does packet capture, but also packet logging for advanced TCP/IP network analysis, and even security-focused intrusion detection [24].

Snort's intrusion detection capabilities can be used while capturing in real-time or in post-analysis of previous captures, including captures taken natively by tcpdump.

3.4.3 Ethereal

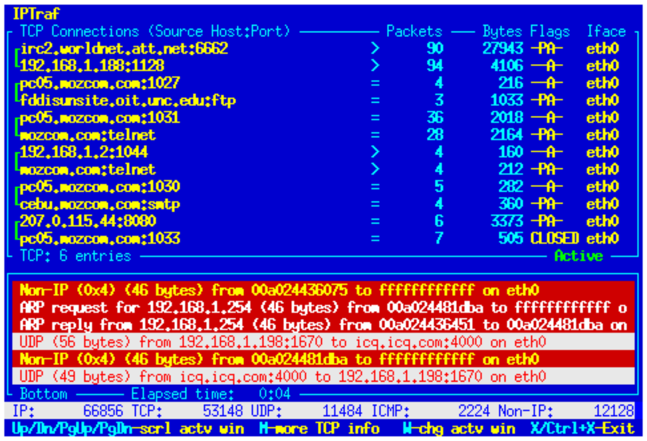
Probably the most feature-rich of the open source protocol analyzers, Ethereal was initially written by Gerald Combs, with ongoing development provided by a sizable team [25].

Ethereal too uses the libpcap library, so in turn, users can unleash its powerful display capabilities against previous tcpdump captures.

3.4.4 iptraf

Gerard Paul Java's iptraf is a statistical analysis tool with a clean and very usable curses interface for Linux [26]. Since iptraf was built to leverage some of the built-in networking features in the Linux kernel, its portability is limited compared to some of the other tools (which are constrained by the porting of libpcap to other platforms).

Iptraf provides statistical information on conversation pairs, protocols, and packets, and is better suited for a higher-level network analysis than true packet captures. However, if used correctly, iptraf is a very powerful and easy to use tool.



```
IPTraf
┌ TCP Connections (Source Host:Port) ──────────── Packets ─── Bytes Flags  Iface ┐
┌irc2.worldnet.att.net:6662                  >        90      27943 -PA-   eth0
└192.168.1.188:1128                          >        94       4106 --A-   eth0
┌pc05.mozcom.com:1027                        =         4        216 --A-   eth0
└fddisunsite.oit.unc.edu:ftp                 =         3       1033 -PA-   eth0
┌pc05.mozcom.com:1031                        =        36       2018 --A-   eth0
└mozcom.com:telnet                           =        28       2164 -PA-   eth0
┌192.168.1.2:1044                            >         4        160 --A-   eth0
└mozcom.com:telnet                           >         4        212 -PA-   eth0
┌pc05.mozcom.com:1030                        =         5        282 --A-   eth0
└cebu.mozcom.com:smtp                        =         4        360 -PA-   eth0
┌207.0.115.44:8080                           =         6       3373 -PA-   eth0
└pc05.mozcom.com:1033                        =         7        505 CLOSED eth0
└ TCP: 6 entries ──────────────────────────────────────────────── Active ──┘

 Non-IP (0x4) (46 bytes) from 00a024436075 to ffffffffffff on eth0
 ARP request for 192.168.1.254 (46 bytes) from 00a024481dba to ffffffffffff o
 ARP reply from 192.168.1.254 (46 bytes) from 00a024436451 to 00a024481dba on
 UDP (56 bytes) from 192.168.1.198:1670 to icq.icq.com:4000 on eth0
 Non-IP (0x4) (46 bytes) from 00a024481dba to ffffffffffff on eth0
 UDP (49 bytes) from icq.icq.com:4000 to 192.168.1.198:1670 on eth0
└ Bottom ─────── Elapsed time:    0:04 ─────────────────────────────────────┘
 IP:        66856 TCP:       53148 UDP:       11484 ICMP:        2224 Non-IP:       12128
Up/Dn/PgUp/PgDn-scrl actv win  M-more TCP info   W-chg actv win  X/Ctrl+X-Exit
```

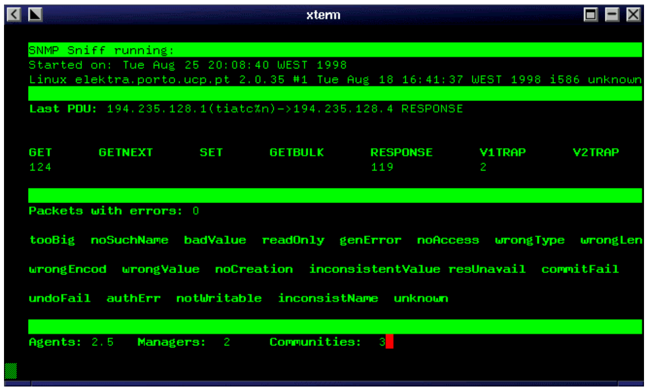
Figure 5. Iptraf Traffic Monitor Example

3.4.5 SNMP Sniff

The final in our list of protocol analyzers, SNMP Sniff's functionality is quite well focused and for network managers with SNMP-related problems, it saves the step of having to manually define a capture filter for another protocol analysis tool.

Written by Nuno Leitao, SNMP Sniff's obvious focus also provided the author with the ability to do better PDU decodes than many of the other packages described here [27], which makes this tool ideal for the network manager. Often, this tool is deployed in network operations centers even if an expensive "framework" tool has been deployed as well, as none of the "frameworks" include a tool with similar functionality. Even if used only for troubleshooting other tools, it is well worth an administrators time and effort to download and build this tool, adding it to their respective "bag of tricks".

SNMP Sniff has dependencies on libpcap as well as CMU SNMP. And with the appropriate curses libraries and Perl modules installed, SNMP Sniff can also serve as a statistical monitor for SNMP, as shown in Figure 6.



```
SNMP Sniff running:
Started on:  Tue Aug 25 20:08:40 WEST 1998
Linux elektra.porto.ucp.pt 2.0.35 #1 Tue Aug 18 16:41:37 WEST 1998 i586 unknown

Last PDU: 194.235.128.1(tiatc%n)->194.235.128.4 RESPONSE


GET       GETNEXT      SET       GETBULK       RESPONSE      V1TRAP      V2TRAP
124                                            119           2


Packets with errors:  0

tooBig  noSuchName  badValue  readOnly  genError  noAccess  wrongType  wrongLen

wrongEncod  wrongValue  noCreation  inconsistentValue resUnavail  commitFail

undoFail  authErr  notWritable  inconsistName  unknown


Agents:  2.5    Managers:   2     Communities:   3
```

Figure 6. Example of snmpstat from the SNMP Sniff package

3.5 Network and Systems Monitors

This subgroup of network management is considerably larger than the others, if you consider sheer numbers of entrants. This is mainly due to the plethora of system administration tools that contain some sort of alerting functionality. If a tool's primary focus is systems administration and not monitoring, it will not be discussed in depth in this paper. However, several tools in this group merit mention, including the following:

Table 1. Overview of Open Source Systems Administration Tools

Tool Name	Description
PIKT	The "Problem Informant/Killer Tool" is a sysadmin-focused scripting environment with auto-execute alarm capabilities [28]. A powerful tool, but can be quite labor-intensive to deploy.
GAP	The "GNU Administration Project" provides a CORBA-based distributed architecture to allow execution of system admin functions on multiple platforms [29].
Linuxconf	A local/remote administration tool for Linux, this tool is the de facto standard in the popular distributions today, but only handles administration on a machine-by-machine basis [30].

The remainder of the products discussed in this section will have significant portions of their architecture dedicated to monitoring remote network devices, systems, and/or services.

3.5.1 GxSNMP

GxSNMP, under the guiding hand of Jochen Friedrich and a small team of developers, is probably the closest any existing open source application has come to rivaling the appearance of one of the "framework" tools [31]. It has a very rich set of GUIs, but its underlying network monitoring/polling components are "under construction" [32], which is unfortunate, given the projects otherwise promising, if aggressive, goals.

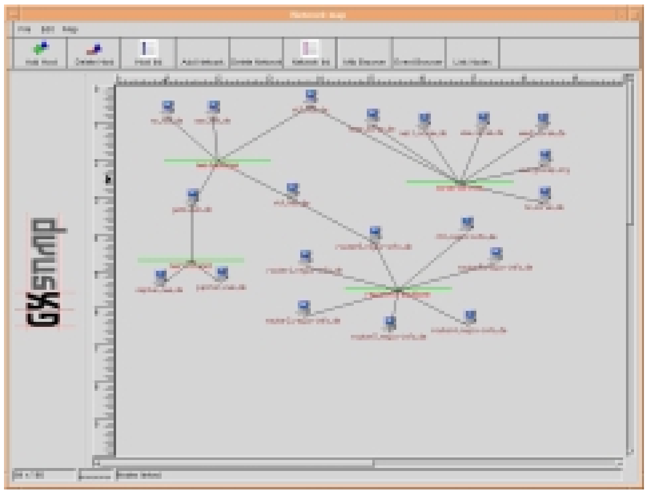


Figure 7. Example of GxSNMP's Map-based GUI

The development efforts of the GxSNMP team have been very typical of unfunded, large-scope open source development efforts—the development is slow and irregular, but the resulting code is sound. Most of the recent releases of GxSNMP have been bug-fix releases and the last significant development initiative was begun in April 1998, with work beginning on a network discovery algorithm.

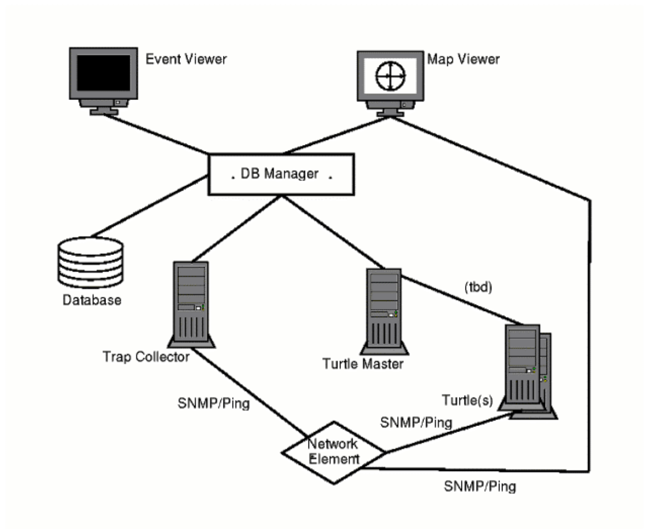


Figure 8. An Architecture Diagram from the GxSNMP Project

This project has a strong core team with knowledgeable developers, but in an effort to deliver a basic level of functionality, distributed network monitoring has been pulled out of the development until some later date, as noted in [31]. This will likely result in an awkward, retrofit implementation of distributed monitoring, as has happened with HP and IBM's "framework" tools.

Additionally, the project's dependence on a map-based GUI will introduce an unnecessary degree of maintenance in large enterprise environments, offsetting many of the benefits of a network management platform, as described in [16].

3.5.2 The Event Monitor Project (Emonitor)

The Event Monitor Project, a one-man effort led by Juan Casillas (half of the Gnome Administration Project team, referenced in Table 1), is an agent-based approach to monitoring remote systems [33]. While the architecture and plans call for system-centric network monitoring capabilities as well as other features, the only agent piece that exists in production today is the emdskagt (disk monitoring functionality). Despite this, the architecture for server-agent communications is in place and is reasonably extensible.

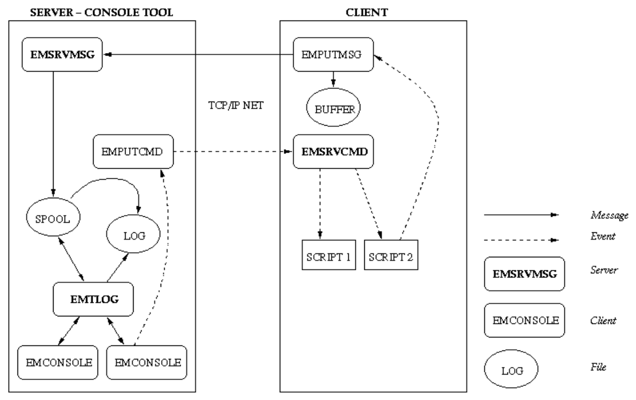
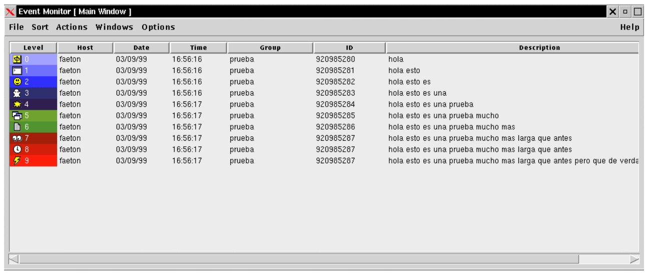


Figure 9. Architectural Overview of the Emonitor Project

One of the more interesting aspects of the Event Monitor Project is that it takes an event-centric approach to the user interface level, providing an event browser as the only graphical interface. This "browser only" approach, while arguably better than the "map only" approach of other tools, will find difficulties in environments where the event frequency is higher than a user can address the issues. These tools are also likely victims of "event storms", a situation in which a misconfigured device can cause itself of other devices to generate large amounts of messages to a management platform.



Level	Host	Date	Time	Group	ID	Description
0	faeton	03/09/99	16:56:16	prueba	920985280	hola
1	faeton	03/09/99	16:56:16	prueba	920985281	hola esto
2	faeton	03/09/99	16:56:16	prueba	920985282	hola esto es
3	faeton	03/09/99	16:56:16	prueba	920985283	hola esto es una
4	faeton	03/09/99	16:56:17	prueba	920985284	hola esto es una prueba
5	faeton	03/09/99	16:56:17	prueba	920985285	hola esto es una prueba mucho
6	faeton	03/09/99	16:56:17	prueba	920985286	hola esto es una prueba mucho mas
7	faeton	03/09/99	16:56:17	prueba	920985287	hola esto es una prueba mucho mas larga que antes
8	faeton	03/09/99	16:56:17	prueba	920985287	hola esto es una prueba mucho mas larga que antes
9	faeton	03/09/99	16:56:17	prueba	920985287	hola esto es una prueba mucho mas larga que antes pero que de verda

Figure 10. The Event Browser from the Event Monitor Project

The Event Monitor Project is focused specifically on Unix and Unix-like platforms, and in turn, is written in C and leverages Tcl/Tk for the server-side GUIs. This limitation, coupled with the absence of an underlying database (it leverages log files) and potential security concerns with server-agent communications will likely stymie an headway this otherwise feature-rich tool might make into the enterprise marketplace.

3.5.3 Mon

No discussion of open source tools in monitoring would be complete without mention of mon. A widely-used tool for service monitoring and notification, mon is described as a "service monitor daemon" by its primary author, Jim Trocki [34]. Mon has both client and server components, a web interface, and product independent monitors which are simply invoked by the client.

Due to its reliance on Perl alone, both the client and server components are very portable, and mon ships with several notification scripts, which when coupled with other open source tools, provide the ability for email, alphanumeric page, and SNMP trap-based notification.

Mon is a very effective product in its niche, but most enterprises require operator interfaces, and mon's web interface was not designed to provide this functionality. Although, the possibility of coupling mon with another tool to leverage its focused functionality could make mon a significant player in an enterprise NOC.

3.5.4 Big Sister

Annoyed by the limitations and licensing requirements of Big Brother, a quasi-commercial monitoring tool [35], Thomas Aeby built a project around replacing and extending its functionality.

Big Sister is a Perl-based service monitor with a web-server front-end and a client-server architecture, which Aeby describes in his architecture as *agent* (client) and *status collector* (server) [36]. The Big Sister agent can run on any platform with Perl 5.002 or later and when configured, will check the status of a variety of services, with user extensibility possible. The agent serves three basic functions in Big Sister's architecture: to provide a distributed monitoring capability, to provide updates to the status collector, and to provide easier extensibility for platforms running services that are not monitored by default.

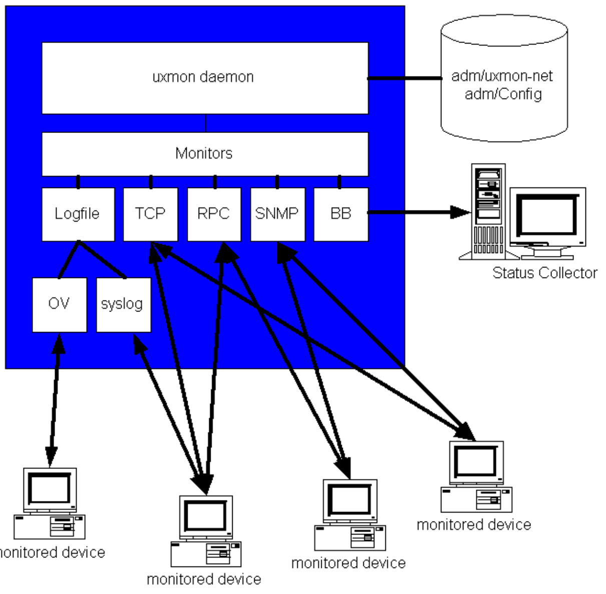
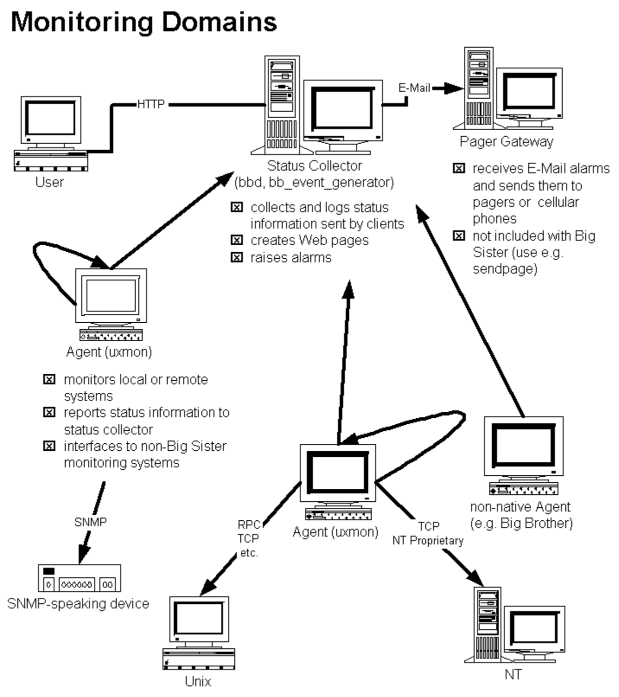


Figure 11. Big Sister's Agent Architecture

The status collector, Big Sister's server component, is responsible for maintaining the log of events from each of the agents, as well as interfacing to the notification system (which is not included in the distribution.



Monitoring Domains
User
HTTP
E-Mail
Status Collector
(bbd, bb_event_generator)
collects and logs status information sent by clients
creates Web pages
raises alarms
Pager Gateway
receives E-Mail alarms and sends them to pagers or cellular phones
not included with Big Sister (use e.g. sendpage)
Agent (uxmon)
monitors local or remote systems
reports status information to status collector
interfaces to non-Big Sister monitoring systems
SNMP
SNMP-speaking device
RPC
TCP
etc.
Agent (uxmon)
TCP
NT Proprietary
non-native Agent
(e.g. Big Brother)
Unix
NT

Figure 12. Big Sister's Distributed Architecture

Much like mon before it, Big Sister is constrained almost solely by its user interface, which is vastly improved over mon's.

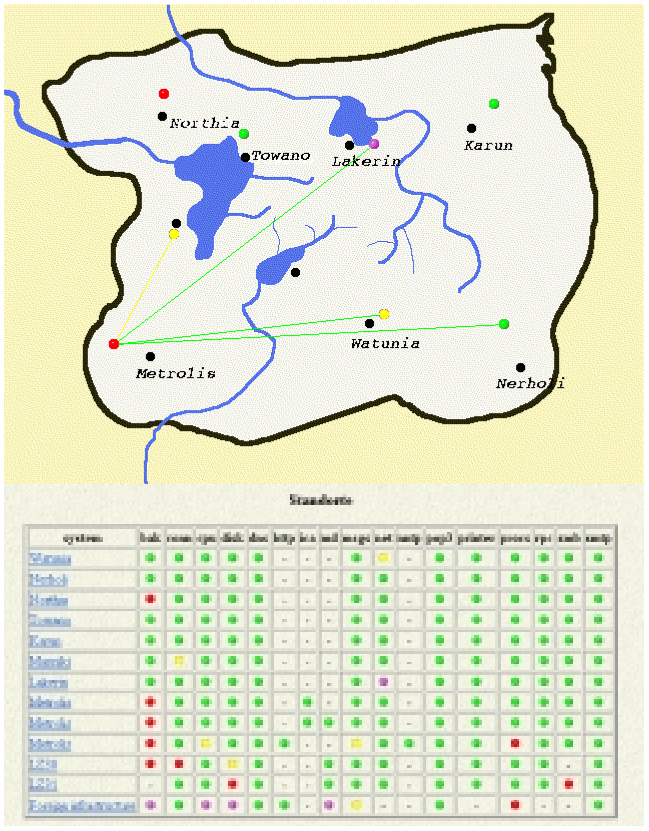


Figure 13. Big Sister's Web Interface

Big Sister's web interface only provides snapshot views, with no ongoing view of the network as status changes occur without constant refreshes. The absence of an event browser will likely prevent Big Sister from progressing into the enterprise marketplace, but like mon, is very effective in its niche role as well as in smaller NOCs.

4. Next-Generation Tools

The tools described above, and combinations thereof, can and do provide core network management functionality in many of today's college and university networks. But with the ongoing growth in demand for IT resources in educational environments, yesterday's boutique tools are not well-positioned to provide the necessary scalability and distribution demands placed on today's university IT departments.

New technologies are emerging in network management, amongst which is the inevitable XML, positioned for potential broad-sweeping changes across many disciplines, as discussed in [5]. None of the tools mentioned above have addressed the potential for XML's introduction into network management, although many network element providers have already announced strategic plans to include the technology [37].

And with no exceptions, due to a reliance on C and early Perl, multi-threaded applications are not to be found in our discussion. Yet, given the nature of network management processing and the excessive amount of CPU time left idle due to network transmission-induced wait states, threads are the last best answer for scalability and performance improvement.

The next-generation applications to emerge, whether closed or open source, must address these needs. And within the open source fray, there is only one such project that promises to address these needs—OpenNMS' Bluebird Project.

4.1 OpenNMS' Bluebird Project

Originally architected by Steve Giles and implemented by Brian Weaver, the Bluebird Project has aggressive plans and lofty goals. As stated in their design goals, there are two primary areas where the project intends to differentiate itself from the field: it will leverage emerging technologies, including Java, XML, and XSL, and it will leverage what Giles has termed "synthetic transactions", or a lightweight test to exercise the protocols related to services that a platform might offer, such as FTP, SMTP, et al. [38].

Table 2. Bluebird Project First Release Functionality

Functionality	Description
Network Discovery	ICMP-based discovery of TCP/IP networks
Capability Checking	An interface-level analysis of services provided by the host, augmented by SNMP information, if a configured agent is present.
Status Polling	ICMP-based polling for interface reachability
Service Polling	Protocol-level polling of HTTP, SMTP, DNS and FTP services
Distributed Architecture	Full distribution capabilities, including *distributed pollers* and *master station(s)*.
Java-based User Interface	A real-time user interface to reflect network changes as soon as they are realized to the master station.
Data Reporting	Report data available in XML with XSL capabilities.
Business Views	Topological maps have been eschewed in deference to a view of the network that groups devices into customer-defined groups.
Graphical Rule Builder	Java-based tool with drag-and-drop metaphor to allow users to build rule sets by which *Business Views* are defined.
Configuration Panels	Java-based tools to configure both the master station and distributed pollers. All configurations are then stored in XML to allow manipulation by any XML editor/tool.
Ranging/ Filtering	Ability to restrict discovery to specific IP ranges, as well as to restrict the monitored nodes by IP range or rule set.
Scheduling	Ability to define one-time or recurring periods of planned outage.
Event Subsystem	SNMP trap receiver as well as listeners to receive XML events, each of which are processed/correlated and populated to event browser and/or an automated action is invoked.

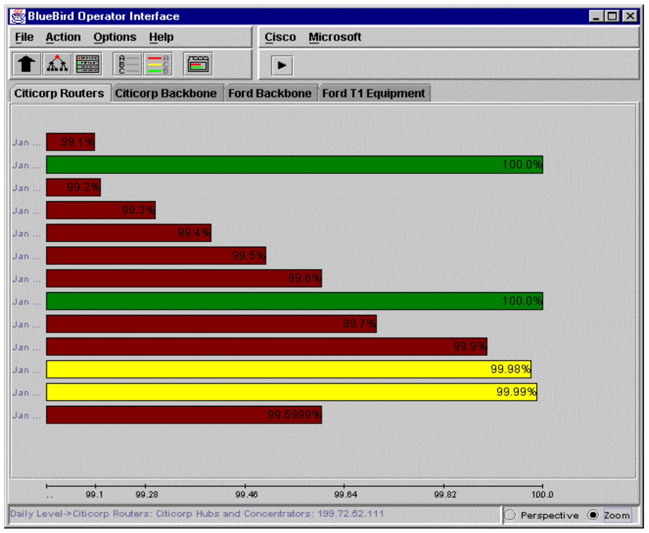
The team at OpenNMS has also taken an architectural approach to scalability and distribution, arguing that the ultimate scalability cannot be introduced after the fact and must be incorporated from the earliest design phases.

Initially targeting an initial production release in 4Q00, OpenNMS has identified a list of basic functionalities that are planned for the initial release, including the elements outlined in Table 2.

4.1.1 User Interface

One of the most obvious differences between the Bluebird Project and other tools is its use of histograms representing devices/services in lieu of a map-based interface. This accomplishes three basic goals for the design team: the administrative overhead associated with maps is avoided, the topology argument is mute, and the visual representation can more closely approximate the end user's view of the network (or in many cases, management's view of the network).

Additionally, each histogram is actually the top-most point in a tree of histograms, each successively representing specific devices, time frames, and eventually, an event browser specific to a device over some given time period.



Daily Level->Citicorp Routers: Citicorp Hubs and Concentrators: 199.72.52.111

Figure 14. The Bluebird Project's Real-Time Console

4.1.2 Distributed Architecture

The Bluebird Project has embraced the concept of monitoring networks with remote sites, and has architected their solution to address the possible bandwidth constraints that often accompany these remote sites. In their architecture, a distributed poller, or a system responsible for discovery and status/service polling of network-attached devices, can be deployed in a remote site so that all discovery and polling-related traffic remains local to that network and need not traverse the WAN segment.

Communications between the distributed poller and the master station, or the system responsible for maintaining the centralized database and providing reporting and end-user interfaces functions, can be via "push" or "pull" mechanisms. With this option, distributed pollers are being built to allow for up to one day with no contact between between the poller and the master station, to allow for potential network outages.

Additionally, the design team at OpenNMS has applied a degree of social conscience to their use of bandwidth. With their concept of “bandwidth trolls”, the Bluebird Project software can be self-regulating in its use of network resources, and users can control by percentage of overall bandwidth and time period (repeating schedule) how much traffic a distributed poller is allowed to generate.

4.1.3 *Synthetic Transactions*

Relying on ICMP for availability information creates an inevitable fallibility; if a system responds to an ICMP echo request (e.g., ping), it has validated nothing beyond Layer 3 reachability, not availability of any services. The Bluebird Project addresses this with a concept of *synthetic transactions*, or a lightweight implementation of a protocol client that actually exercises a services protocol. For example, an SMTP-based synthetic transaction might include the establishment of a TCP socket connection to port 25, the receipt of an SMTP banner, the issuance of a HELO command, and the receipt of a 250 “Hello”.

While this does not actually generate and force delivery of an email, it does prove that a server exists on port 25 and is capable of receiving and responding to standard SMTP transactions. This is well-beyond the capabilities of any of the ICMP-centric tools discussed earlier.

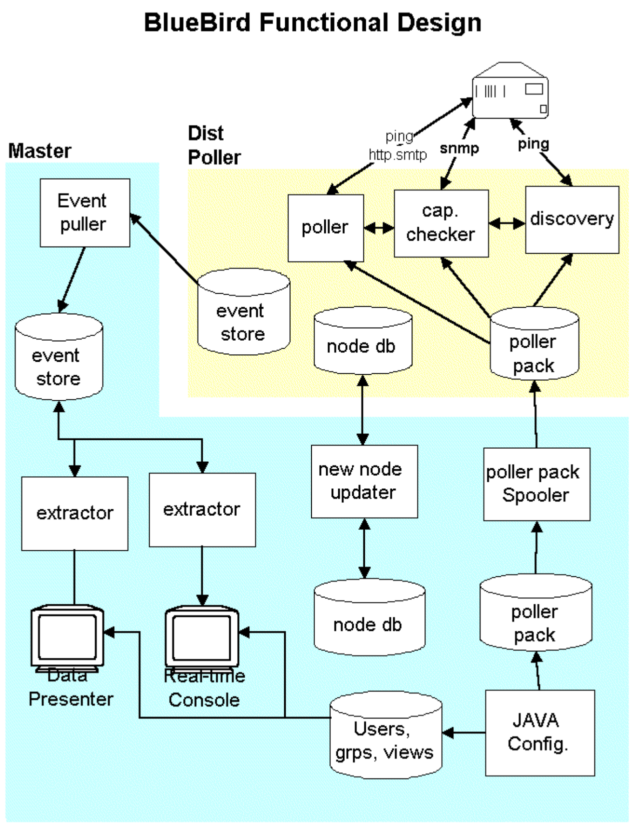
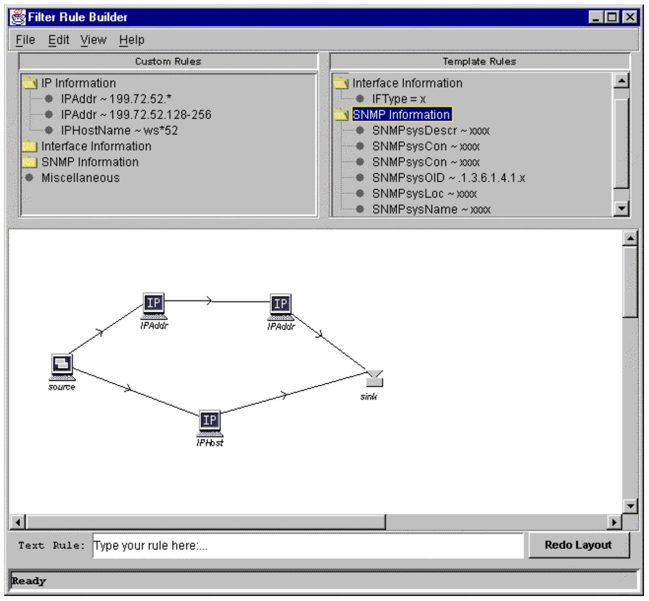


Figure 15. The Bluebird Project’s Functional Design

4.1.4 *Graphical Administration Capability*

A duality in configuration options is almost an unspoken requirement in enterprise network management packages today. Complex configurations are often simplified with graphical tools, yet large-scale configurations often need to be built by script or export. The Bluebird Project has addressed both of these needs with their configuration panels.

The “Administrator Mosh Pit” is a blank canvas which allows users to add their own icons which in turn invoke user-defined applications. The “Mosh Pit” ships with the Bluebird administrative tools defined, including the Graphical Rule Builder, to allow for definition of the business views. This tool simplifies the often confusing Boolean logic rules that are necessary in defining rule sets. However, the tool also provides a text entry box, allowing knowledgeable administrators to simply enter their own rule. And once the configuration is saved, it is built in XML, so any future updates (or a complete replacement) can be built manually or by script.



Custom Rules:
- IP Information
  - IPAddr ~ 199.72.52.*
  - IPAddr ~ 199.72.52.128-256
  - IPHostName ~ ws*52
- Interface Information
- SNMP Information
- Miscellaneous

Template Rules:
- Interface Information
  - IFType = x
- SNMP Information
  - SNMPsysDescr ~ xxxx
  - SNMPsysCon ~ xxxx
  - SNMPsysCon ~ xxxx
  - SNMPsysOID ~ .1.3.6.1.4.1.x
  - SNMPsysLoc ~ xxxx
  - SNMPsysName ~ xxxx

Text Rule: Type your rule here:...

Figure 16. The Bluebird’s Project Graphical Rule Builder

4.1.5 *Underlying Technologies*

Due to the nature of network management tools, much of the possible processing time is wasted in wait states, blocking for receipt of a message from a network device. The best way to address this is through the use of threads, which allows the system to process more than one transaction seemingly at the same time, as well as to distribute the processing associated with those transactions over multiple processors.

Following an aborted attempt to build threaded libraries for C++, the project team converted to Java2, using Sun’s JDK 1.2.2. This provided three key pieces to the puzzle: native threads, platform independence (or restrictions only on platforms with a Java2 run-time environment), and a shorter development cycles with Java’s rich development environment.

The team has also been to leverage work from the Apache project [39] and IBM [40] to help shorten development cycles, specifically in their respective work to provide hooks between Java and XML.

4.1.6 *Risks Associated with Open Source*

The biggest risk that any open source project runs is to bite off too much for an initial deployment, allowing the window of opportunity to pass while working toward first release. To avoid this, OpenNMS has sought institutional funding for the Bluebird Project, in an effort to staff a full-time development team focused on bringing the initial release to market.

4.1.7 Code Releases

The Bluebird Project currently houses its development code in a CVS instance on the projects CVS server [41]. Additionally, the project has already released some important subsets of code.

4.1.7.1 End User Interfaces

In April 2000, the Bluebird Project released the initial release of their user interfaces. This was targeted as the first major milestone to allow the community time to work with and familiarize themselves with the histogram-based metaphor.

4.1.7.2 JSNMP Libraries

In June 2000, the project team announced the release of the first commercially viable SNMPv1 and SNMPv2 libraries for Java. Since its initial announcement, the jSNMP libraries have been downloaded over 300 times and the team has been contacted by multiple commercial organizations seeking LGPL licensing, to allow for commercial use of the currently GPL product [42].

4.1.8 Project Status

The project team has fluctuated in size with different development phases, but has been as large as 12. A permanent project manager has been added to the team, who is currently responsible for communications with the community of over 2000.

There are approximately 50 community members who are active contributors in design, architecture, development, documentation, or testing.

5. REFERENCES

[1] http://www.opensource.org/osd.html

[2] http://salaryadvisor.informationweek.com/cgi-bin/ibi_cgi/ibiweb.exe?IBIF_ex=ind00&level=STAFF&submit=Submit

[3] http://www.aprisma.com/ournews/1999/dec/12-15.html

[4] http://www.techweb.com/wire/story/TWB19980218S0012

[5] http://www.nwfusion.com/newsletters/nsm/0920nm2.html

[6] http://www.ietf.org/rfc/rfc1441.txt

[7] http://www.net.cmu.edu/groups/netdev/software.html

[8] http://www.gaertner.de/snmp/

[9] http://www.net.cmu.edu/groups/netdev/agentx/

[10] http://www.scguild.com/agentx/

[11] http://ucd-snmp.ucdavis.edu/FAQ.html#What_is_it_

[12] http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/mrtg.html#HIST

[13] http://ee-staff.ethz.ch/~oetiker/webtools/rrdtool/index.html

[14] http://www.usenix.org/events/neta99/full_papers/allen/allen_html/index.html

[15] http://ee-staff.ethz.ch/~oetiker/webtools/rrdtool/frontends/

[16] http://www.ops.com/prodinfo2/amerigo/amgo_cs/amgo_cs.html

[17] http://wwwhome.cs.utwente.nl/~schoenw/scotty/

[18] http://www.ibr.cs.tu-bs.de/projects/scotty/

[19] http://www.marko.net/cheops

[20] http://www.bitwizard.nl/mtr/

[21] http://freshmeat.net/news/1999/06/09/928940947.html

[22] http://cheops-ng.sourceforge.net/

[23] http://www.tcpdump.org

[24] http://www.snort.org/

[25] http://ethereal.zing.org

[26] http://cebu.mozcom.com/riker/iptraf/

[27] http://www.linuxave.net/~nunol/snmpsniff/

[28] http://pikt.uchicago.edu/pikt/index.html

[29] http://www.gsyc.inf.uc3m.es/~assman/gap/

[30] http://www.solucorp.qc.ca/linuxconf/concept.hc

[31] http://www.gxsnmp.org/

[32] http://www3.scram.de/gxsnmp/msg00864.html

[33] http://www.gsyc.inf.uc3m.es/~assman/em/

[34] http://www.kernel.org/software/mon/

[35] http://maclawran.ca/bb-dnld/

[36] http://bigsister.graeff.com/

[37] http://www.nwfusion.com/newsletters/nsm/1004nm2.html

[38] http://www.opennms.org/goals.html

[39] http://xml.apache.org/

[40] http://alphaworks.ibm.com/aw.nsf/frame?ReadForm&/aw.nsf/techmain/F62DB5F8684DCF6A8825671B00682F34

[41] http://www.opennms.org/cache/9.html

[42] http://www.opensource.org/license