SNMP Smart Chart

By

Nathan D. Smith

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Engineering Technology

University of Cincinnati
College of Applied Science

May 2001

SNMP Smart Chart

By

Nathan D. Smith

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment of the Requirements
for
the Degree of Bachelor of Science
in Information Engineering Technology

Nathan Smith	Date
Prof. Shannon McClintock	Date
Prof. Lawrence Gilligan	Date

Dedication

To Sharon

Table of Contents

List of Figures

List of Figures (continued)

Abstract

Network management protocols have been developed to automate the process of network management. Simple Network Management Protocol (SNMP) is an application layer protocol offering network management services in the Internet Protocol suite. The *SNMP Smart Chart* application is used to gather statistics from SNMP capable networking devices, such as a router, switch or server. The tool stores the collected data within a MySQL database, which can then be used to create graphs. The graphs are created in real time by a Java applet. The applet is viewed on an HTML formatted file that is made available on an Apache Web server running on Linux. *SNMP Smart Chart* incorporates a polling feature written in Perl, which allows statistics to be gathered over extended periods of time. Once configured correctly, the tool will collect data and supply a live visual representation of network performance. The network engineers within the data communications department at Provident Bank will be the users of the *SNMP Smart Chart* application. The engineers currently do not have an accurate and reliable method for measuring network performance. The application will aid network engineers in gathering network statistics and performing capacity planning.

SNMP Smart Chart

1. Statement of Need

The following sections will build a case for implementing the *SNMP Smart Chart* network management application at Provident Bank.

1.1 Provident Bank's Growing Network

The *Data Communications (Data Comm.)* department at Provident Bank maintains a metropolitan area network (MAN), which consists of over 4000 nodes and spans over 100 locations. Upgrading the network infrastructure is an on-going process. The following discussion will present some facts about Providence's current on-going network infrastructure upgrade patterns.

According to Provident Bank's *McCafee* ticket tracking software and the *Change Control* department database, Providence's network infrastructure undergoes changes on an average time basis as listed below.

- A ***small*** network infrastructure change occurs approximately every 14 days. A *small* network change is defined as a hub, router, or switch configuration change which fixes a problem generated via a *McCafee* trouble ticket. Provident Bank uses ticket-tracking software called *McCafee* that is maintained by Providence's *Help Desk* department. The *Data Comm.* department records all small configuration changes within *McCafee* that are made to the network infrastructure.
- A ***critical*** network infrastructure change occurs every 3 months. A critical network change is defined as a pre-planned change to the network, which must be conveyed to the *Change Control* department before the change is implemented. Providence's *Change Control* department ensures that proper authority exists before the change is made, and that communication to all effected employees is properly performed. A *critical* network change will occur at off-hours (i.e. weekend at midnight) to minimize the impact of possible network downtime.

- A ***massive*** network infrastructure upgrade occurs approximately every 6 years. A *massive* network change also must be correctly routed through *Change Control*. But, a *massive* upgrade will ultimately effect the entire Bank's information flow. These upgrades are broken down into smaller tasks. The entire upgrade will span many months to a year. The following is a brief list of examples of massive network infrastructure upgrades Provident has undergone in the past 12 years. The planned backbone upgrade for next year has also been included.

  1988 Provident upgrades all main building connections from individual 128K ISDN lines to *Cincinnati Bell's 16Mb Token Ring (802.5)* service.

  1995 Metropolitan Backbone upgraded from *Cincinnati Bell's 16Mb Token Ring (802.5)* service to *Cincinnati Bell's 100Mb FDDI* service.

  2001-2002 Provident will begin metropolitan backbone upgrade from *Cincinnati Bell's 100Mb FDDI* service to switched 1Gb Ethernet over *CINergy's* leased dark fiber service.

1.2 Network Upgrades Increase Performance

The reason that Provident's network is upgraded is strictly performance based. As user needs grow, the network becomes busier. The previous section described Provident's plans to undergo a massive network infrastructure upgrade to the metropolitan backbone before 2002. The decision to upgrade the backbone was solely performance based. Currently more bandwidth is needed than the network backbone can support. Unfortunately, the need for this costly upgrade was recognized via a slowdown in network performance. Currently the *Data Comm.* department does not have any way to measure network performance.

1.3 Measuring Network Performance Will Save Money

Between the years of 1995 and 2000 Provident has spent an average of over $300,000 a year on network infrastructure capital equipment. This includes only items purchased under cost center 434 (data comm.). If the *Data Comm.* department could measure network performance, then upgrades could be targeted to where it is most needed. The *Data Comm.* department could take a pro-active approach to upgrades, instead of the current reactive approach.

Accurate data showing network performance, could change where and how much money is spent. The order in which upgrades are performed could be changed to receive more performance per dollar spent. Achieving the maximum benefit for our expenses is only attainable with a means to measure network performance.

Currently the *Data Comm.* department does not have an accurate and reliable method for measuring network performance. What is needed to measure network performance is a formal *Network Management* system.

2. Supporting Resources for Network Management

Network management can formally be defined as "the process of controlling a complex data network as to maximize its efficiency and productivity", according to Fang and Leinwald. In the late 1970's, computer networks had grown from a simple layout of small, separate networks that were not connected to each other, to larger networks that were interconnected. The larger these networks became the more difficult they became to manage. It soon became evident that a means to manage these networks needed to be developed. Thus, "Network Management" was born.

The International Organization for Standards has defined five key areas of network management. These are: fault management, configuration management, security management, performance management, and accounting management. For the purpose of Provident Bank's network management issues, the *SNMP Smart Chart* solution only addresses the area of performance management.

Network management protocols have been developed to automate the process of network management. The various network management protocols available will be explored in the following section.

2.1 Network Management Protocol Choices

There are many network management protocols available. The two mainstream protocols however are SNMP (Simple Network Management Protocol) and CMIP (Common Management Information Protocol). Generally, SNMP works under the TCP/IP (Transport Control Protocol/ Internet Protocol) communication stack and CMIP works under the OSI (Open Systems Interconnection) communication stack. Currently Provident Bank utilizes the TCP/IP stack.

There are two main advantages to using SNMP as the network management protocol at Provident Bank over a different protocol such as CMIP. The first reason is because of the simple design of SNMP, which will be discussed in more detail within section 2.2. The streamlined architecture of SNMP does not place a significant amount of stress on a network. The second reason to choose SNMP is because of its wide use today. SNMP became popular when no other network management protocol implementations were available. It soon became the defacto standard of network management. The result is that almost all-major vendors of internetwork hardware, such as bridges and routers,

design their products to support SNMP, making it easy to implement. Provident's Cisco-powered network already fully supported SNMP. No upgrades would be necessary to the existing hardware. CMIP is a well-designed network management system that improves on many of SNMP's weaknesses. But, in fixing SNMP's problems CMIP became such a large and complex management system that only the best equipped networks initially could afford to run it. As a result, CMIP is not highly supported by most network hardware and software manufacturers of today. SNMP is highly supported by almost all-major network manufacturers.

The advantages of SNMP's large support base largely outweighed that of considering CMIP. For this reason, it was clear that Provident Bank should choose SNMP as its network management protocol. Some specifics of SNMP will be explored in more detail within the following section.

2.2 Introduction to SNMP

Network management protocols have been developed to automate the process of network management. Simple Network Management Protocol (SNMP) is an application layer protocol offering network management services in the Internet Protocol suite. SNMP is defined in several Request for Comments (RFCs) published by the Internet Engineering Task Force (IETF). SNMP defines a client/server relationship. The client program, called the network manager, makes a virtual connection to a server program, called the SNMP agent, which is running on a remote network device. At Provident Bank, this remote network device is typically a router, hub, or switch. The SNMP agent controls a device resident database referred to as the SNMP Management Information

Base (MIB). *Figure 1*, illustrates a typical example of a network manager connecting to a remote SNMP MIB agent.

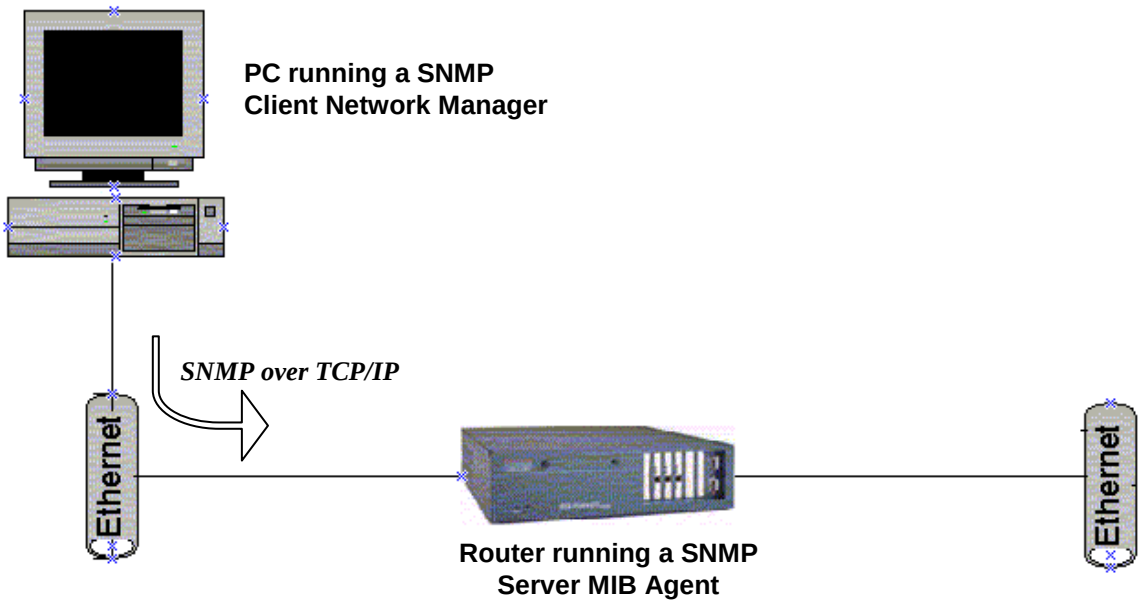


Figure 1. A Network Manager communicating with a SNMP MIB Agent

The data within the MIB can be anything, but is usually a standard set of statistical and control values related to the networking device. SNMP allows networking vendors to extend the standard MIB values with values specific to a particular agent through the use of private MIBs.

Requests issued by the network manager to an SNMP agent consist of the identifier of the SNMP variable, referred to as MIB object identifier (OID) or MIB variable. Also instructions to either "get" the value for the identifier, or "set" the identifier to a new value is supplied. The MIB is structured like a tree. Each "leaf" (or node) within the MIB tree has an OID number associated with it. To get or set data with the SNMP network manager, a valid OID number must be supplied that is associated with the server MIB agent. The subsequent *Figure 2* illustrates a typical MIB tree. Almost all-valid

OIDs in use by modern networking devices fall under the branch of **1.3.6.1.**, indicated by the darker shaded blue boxes at the top of the MIB tree.

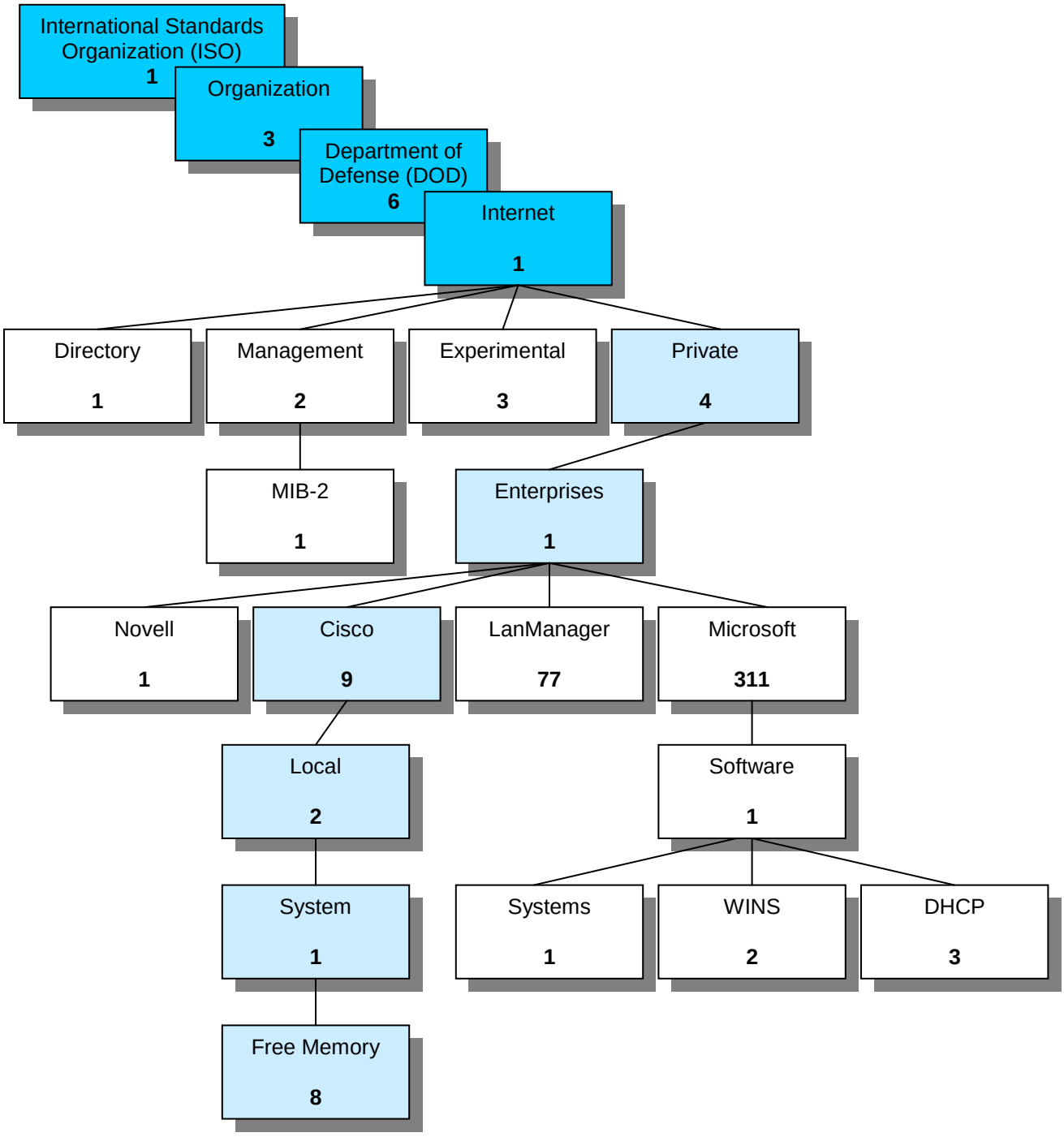


Figure 2. Example of a SNMP Agent MIB Tree

The OID naming scheme is governed by the IETF. The IETF grants authority for parts of the name space to individual organizations such as Microsoft, Novell or Cisco. For example, Cisco has the authority to assign the OIDs that can be derived by branching downward from the node in the MIB name tree that starts with **1.3.6.1.4.1.9**. Microsoft's OIDs branch down from **1.3.6.1.4.1.311**. You can see this structure in the *Figure 2*.

SNMP uses the OIDs to identify objects on each network element (i.e. router/computer) that can be managed using SNMP. For example, in order to get information about the free memory from a Cisco Router the network management station makes a request to the network element using the fully qualified OID that represents the variable containing the number representing the free memory. By Cisco's standard, that would be **.1.3.6.1.4.1.9.2.1.8**. Note how this number matches all the shaded blue nodes within the MIB tree in *Figure 2*. Finding the OIDs available on a particular SNMP device agent can sometimes be tricky. Documentation for standard MIBs can be obtained from RFCs. Private MIB schemes are published independently by each networking device vendor.

3. Product Description and Intended Use.

The *SNMP Smart Chart* application is used to gather statistics from SNMP capable networking devices. The tool stores the collected data within a SQL compatible database, which can then be used to create graphs. The graphs are created in real time by a Java applet. The applet is viewed on an HTML formatted file that is made available on a Web server. *SNMP Smart Chart* incorporates a polling feature, which allows statistics to be gathered over extended periods of time. In this way, the *SNMP Smart Chart* can run with few to no configuration changes. Once configured correctly the tool will collect data and supply a live visual representation of network performance.

The *SNMP Smart Chart* gathers data via SNMP. The information that SNMP can attain from a network device is defined and stored within a MIB which is made available by an SNMP agent, as described previously in section 2.2.

3.1 How the SNMP Smart Chart will be used

The *SNMP Smart Chart* will be used by the network engineers of the data communications department at Provident Bank. The tool will aid network engineers in gathering network statistics and performing capacity planning. The tool will also allow a network engineer to make important network infrastructure design and upgrade decisions. For example, by monitoring and graphing the bytes/second which are forwarded through certain router ports the network engineer will be able to make informed decisions about where upgrades to the network should occur.

The data communications department currently does not have an accurate and reliable method for measuring network performance. The *SNMP Smart Chart* will allow the data communications department to take a pro-active approach to upgrades, instead of the

current reactive approach. Accurate data showing network performance should change where and how much money is spent on the network infrastructure of Provident Bank

3.2 User Profile

The employees within the data communications department of Provident Bank are the intended users of the *SNMP Smart Chart*. The data communications department falls within the *Networking Services* division of Provident Bank. The data communications department is responsible for maintaining Provident's corporate wide area network (WAN), metropolitan area network (MAN), and many local area networks (LANs). Maintaining Provident's network encompasses installing, configuring, and troubleshooting networking hardware. Networking hardware traditionally is a router, switch, or hub. The data communications department is also responsible for planning, designing, and recommending changes and upgrades to the network. This is accomplished by following network management techniques such as statistics gathering and metrics reporting on availability and performance.

Currently the telephone network of Provident Bank is not integrated with the data network. The voice communications department maintains the telephone network and will not need to use the *SNMP Smart Chart*.

The data communications department at Provident Bank is currently made up of five network engineers and one department manager. The five network engineers are currently either titled as an *Associate Network Engineer* or a *Senior Network Engineer*. An associate level network engineer will have a minimum of two years of formal technical related schooling, and at least three years of data communications experience. A senior level network engineer will have a minimum of four years of formal technical

related schooling and at least five years of data communications experience. A senior level network engineer will also often have career specific certifications such as a Cisco Certified Network Associate (CCNA), or a Cisco Certified Network Professional (CCNP).

The *SNMP Smart Chart* is intended to be used by an individual that possesses the minimum requirements of an associate level network engineer at Provident Bank. A user at this level will have a solid foundation in the necessary networking knowledge to understand the *SNMP Smart Chart*. A beginning to intermediate level understanding of TCP/IP and SNMP is required to effectively use the *SNMP Smart Chart.*

3.3 Design Protocols

The subsequent *Figure 3* is a logical representation of how the *SNMP Smart Chart* will work. The figure is color-coded according to autonomous systems of communication (i.e. HTTP, SNMP). The following explains the applied color-coding scheme.

- *Green* represents HTTP communication between the Web server and Web browser. Perl scripts, utilizing the CGI.pm module, generate dynamic HTML from the CGI-BIN directory of the server.
- *Violet* represents the MySQL database communication between Perl scripts and the Java Server daemon using DBI.pm and JDBC respectively
- *Orange* represents the SNMP communication between the *SmartChartCollector.pl* daemon and the SNMP MIB agent on the networking device.
- *Blue* represents the Java ICMP communication between Java Applet Chart and the Java Server daemon.

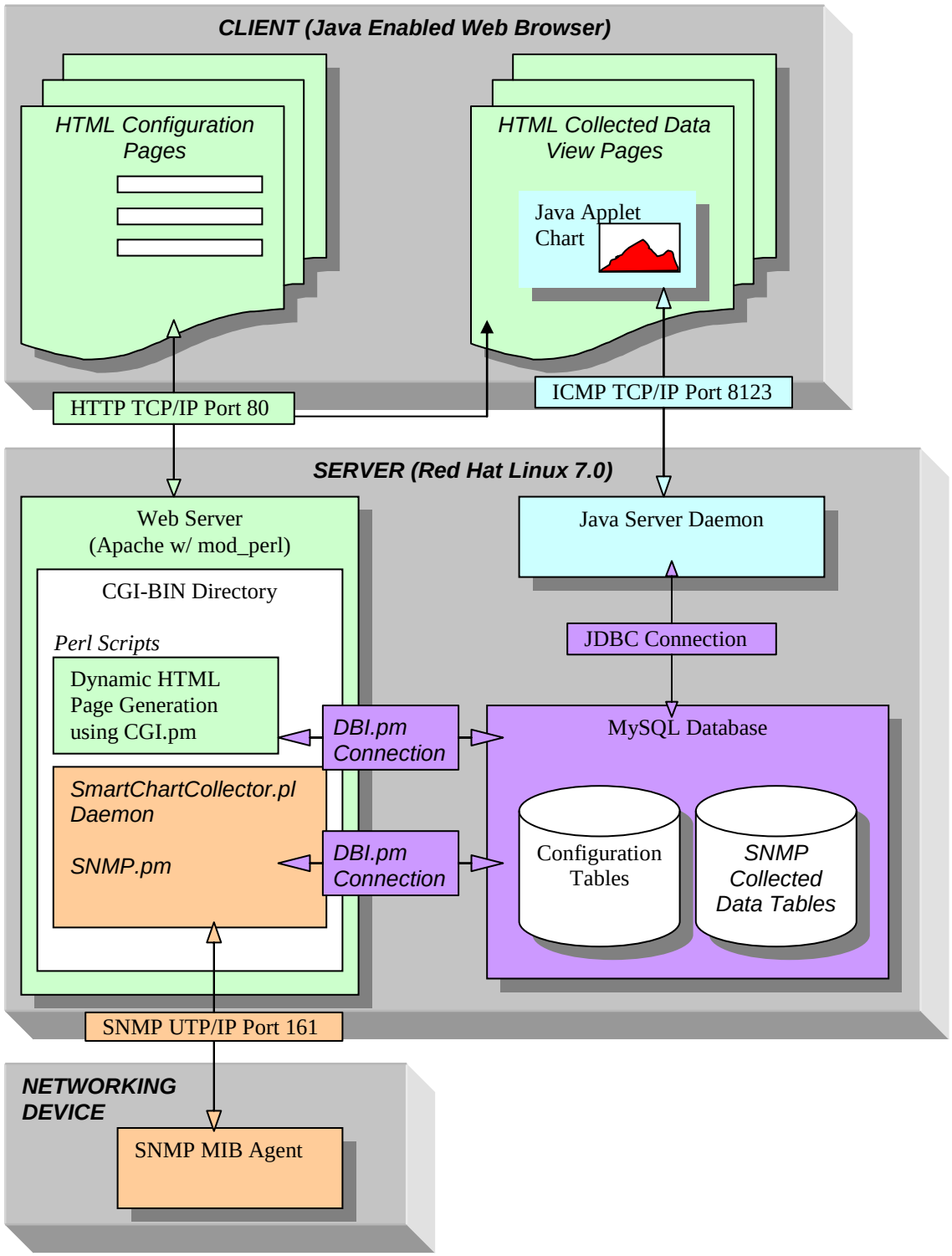


Figure 3. Logical *SNMP Smart Chart* Communication Overview

3.3.1 HTML Configuration Pages

The first step in collecting data from SNMP enabled networking devices is to tell *SNMP Smart Chart* exactly what data it is that you want to collect. This is accomplished via the HTML configuration pages.

As opposed to having only one configuration page which would allow the user to input the needed information, the SNMP *Smart Chart* application is divided into many logical objects which when correctly associated with one another allow for data to be collected. Initially this may appear intimidating and complex, but by incorporating an object model the user gains flexibility, and power. Many devices can be setup and monitored fairly quickly and configuration changes quick and straightforward. Other SNMP monitoring applications have opted for a simpler but far less comprehensive setup interface. These applications often sacrifice power and flexibility for simple. The *SNMP Smart Chart* application was designed to function in small and large networks. Large networks often have a complex array of SNMP networking devices to be monitored. The *SNMP Smart Chart* configuration process is build for flexibility and power.

The following is a list with of all the *SNMP Smart Chart* logical objects. There is nearly a one to one relationship between the logical objects and the database configuration tables that will be discussed later in *section 6.2*.

- OID Units
- OID Types
- OID Nodes
- Communities
- Devices
- Charts
- Web Pages

Each object is represented by a link on the Menu Bar located at the top of each HTML configuration page. The Menu Bar is the central point of navigation for the application.

Figure 4. Menu Bar located on each HTML Configuration Page

In the following sub-sections, each object will be explained and shown how it is set up via the configuration HTML pages.

3.3.1.1 OID Units

An OID Unit represents the unit of measurement that is associated with the data collected via an SNMP MIB node. For example **bits/sec** might be associated with a node that reports the number of bits forwarded out of a router interface each second. The following figures show the HTML pages used to view/edit, add or delete OID units.

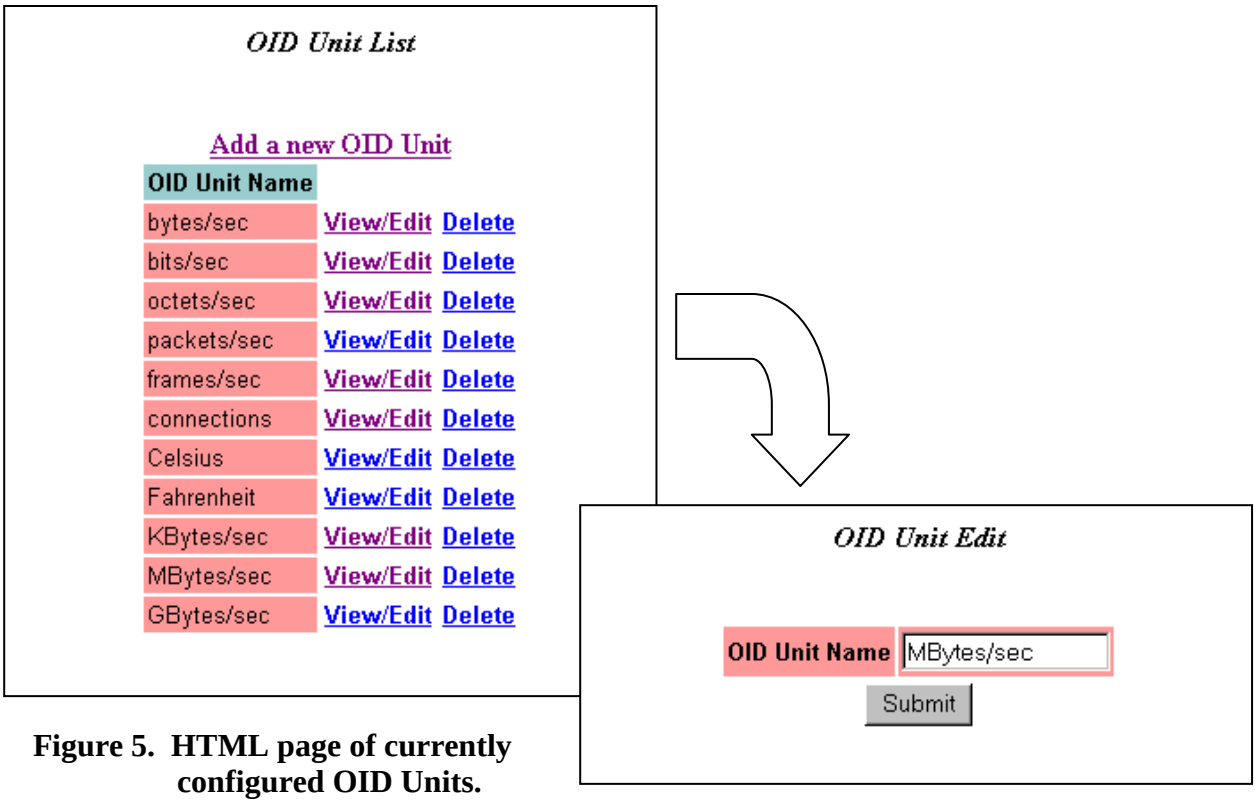


Figure 5. HTML page of currently configured OID Units.

Figure 6. HTML page to view/edit or add an OID Unit.

3.3.1.2 OID Types

OID Types are an optional object to configure. OID Types allow the user to convert the collected SNMP data using a simple conversion string. For example string "**/8"** would represent a string to divide the raw MIB data by 8 before displaying it. This might apply when the MIB agent reports the data in bytes but you prefer to view it in bits. The following figures show the HTML pages used to view/edit, add or delete OID Types.

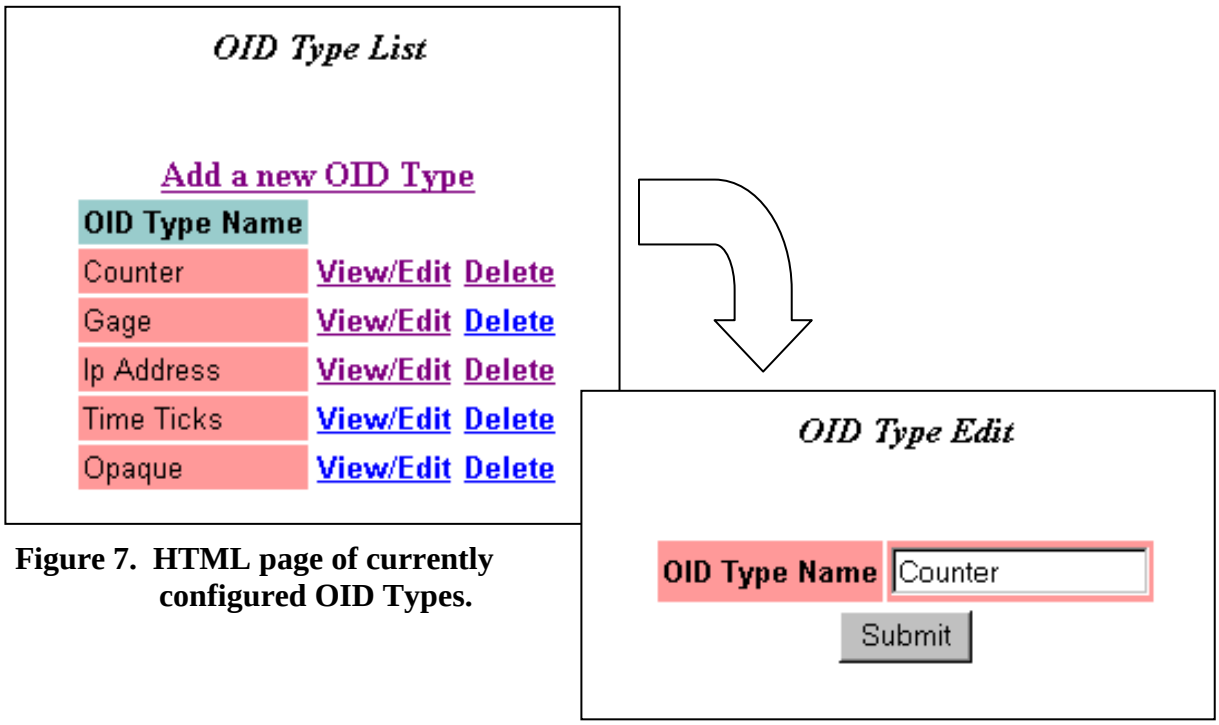
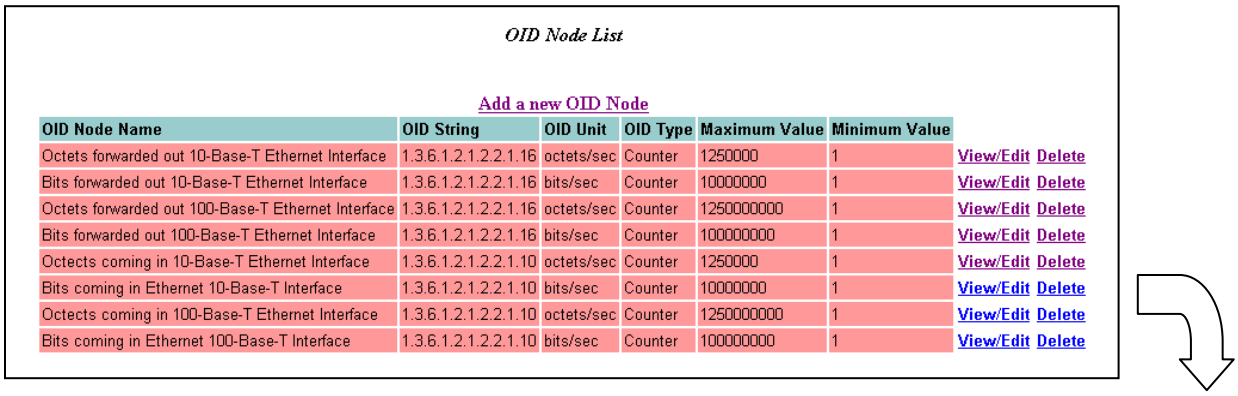


Figure 7. HTML page of currently configured OID Types.

Figure 8. HTML page to view/edit or add a OID Types.

3.3.1.3 OID Nodes

OID Nodes are abstract objects, which are associated with an OID description, a fully qualified OID string, OID Type, OID Unit, Maximum Value and Minimum Value. The OID description is a human friendly explanation of what the OID string represents. A typical OID description might be **"System Name on 3-Com Hub"**. An example of an OID string might be **.1.3.6.1.4.1.16.1.1.1.4.1**. This is the fully qualified value used to read the system name on a 3-COM device. The maximum and minimum values tell the boundaries that the collected data will fall within. Descriptions of the OID Unit and OID Type where discussed earlier. The following figures show the HTML pages used to view/edit, add or delete OID Nodes.

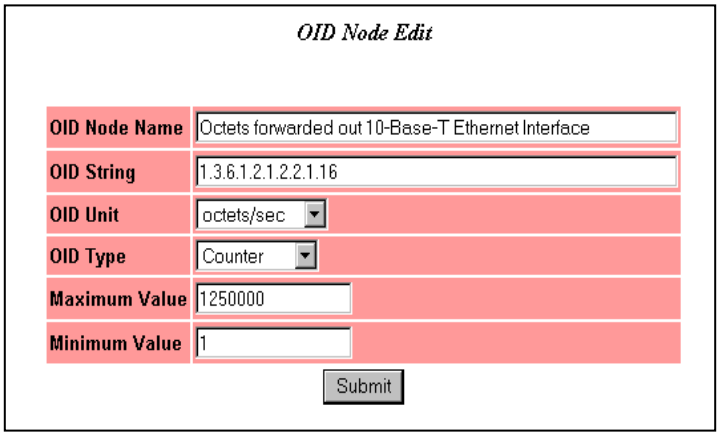


OID Node List

Add a new OID Node

OID Node Name	OID String	OID Unit	OID Type	Maximum Value	Minimum Value	
Octets forwarded out 10-Base-T Ethernet Interface	1.3.6.1.2.1.2.2.1.16	octets/sec	Counter	1250000	1	View/Edit Delete
Bits forwarded out 10-Base-T Ethernet Interface	1.3.6.1.2.1.2.2.1.16	bits/sec	Counter	10000000	1	View/Edit Delete
Octets forwarded out 100-Base-T Ethernet Interface	1.3.6.1.2.1.2.2.1.16	octets/sec	Counter	1250000000	1	View/Edit Delete
Bits forwarded out 100-Base-T Ethernet Interface	1.3.6.1.2.1.2.2.1.16	bits/sec	Counter	100000000	1	View/Edit Delete
Octects coming in 10-Base-T Ethernet Interface	1.3.6.1.2.1.2.2.1.10	octets/sec	Counter	1250000	1	View/Edit Delete
Bits coming in Ethernet 10-Base-T Interface	1.3.6.1.2.1.2.2.1.10	bits/sec	Counter	10000000	1	View/Edit Delete
Octects coming in 100-Base-T Ethernet Interface	1.3.6.1.2.1.2.2.1.10	octets/sec	Counter	1250000000	1	View/Edit Delete
Bits coming in Ethernet 100-Base-T Interface	1.3.6.1.2.1.2.2.1.10	bits/sec	Counter	100000000	1	View/Edit Delete

Figure 9. HTML page of currently configured OID Nodes.



OID Node Edit

OID Node Name	Octets forwarded out 10-Base-T Ethernet Interface
OID String	1.3.6.1.2.1.2.2.1.16
OID Unit	octets/sec
OID Type	Counter
Maximum Value	1250000
Minimum Value	1

Submit

Figure 10. HTML page to view/edit or add an OID Node.

3.3.1.4 Communities

A Community represents the values needed to communicate via SNMP to a particular group of networking devices. The Community is given a Name, SNMP Port Value, and a SNMP Public Community String. The SNMP Port value represents the logical UDP/IP port value that the networking device SNMP service agent will be listening on. **161** is the standard value used, but any valid port number is acceptable. The SNMP Public Community String represents a custom string to gain read-only access to the device's MIB values. The string "public" is a common default value on many networking devices. This Community String is often used as a password, although SNMP does not implement any security techniques on the wire to hide the string from snooping eyes. Changing the SNMP Public Community String on all associated devices is as simple as updating one field value within the Community. Situations such as this demonstrate the flexibility and power of the SNMP Smart Chart object style interface. The following figures show the HTML pages used to view/edit, add or delete Communities.

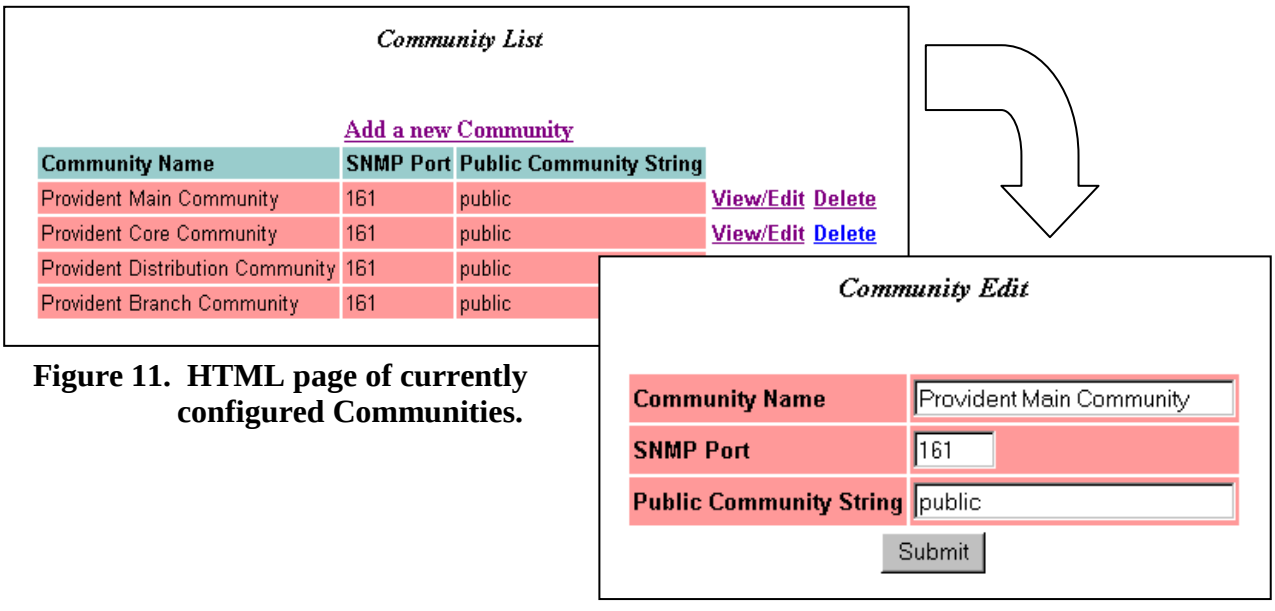


Figure 11. HTML page of currently configured Communities.

Figure 12. HTML page to view/edit or add a Community.

3.3.1.5 Devices

Devices represent a unique physical SNMP enabled networking device (I.E a router or a server). A Device is given a Name, DNS Name or IP address, and is associated with a Community. More often than not a single router will possess many IP addressees or DNS names. Since such devices still internally run only one single SNMP service agent, any valid IP address associated with the device will suffice. Examples of valid values might be **192.1.1.1** or **Router_Cincy**. The associated Community provides the remaining device specific information. The following figures show the HTML pages used to view/edit, add or delete Devices.

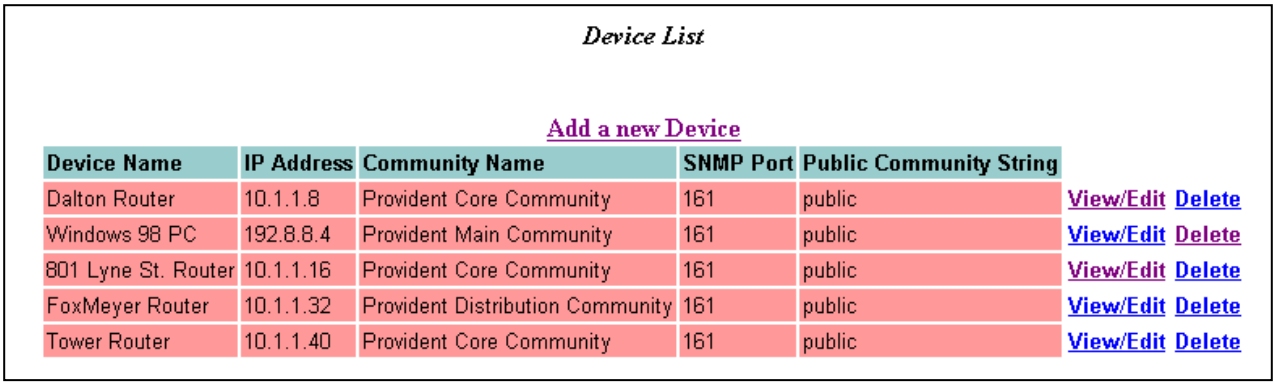


Figure 13. HTML page of currently configured Devices.

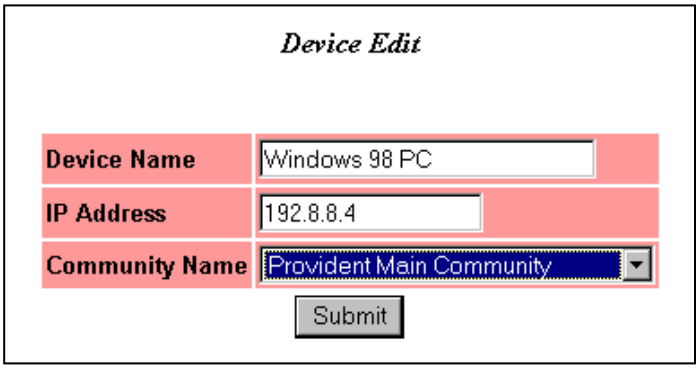


Figure 14. HTML page to view/edit or add a Device.

3.3.1.6 Charts

Charts are used to represent the collected SNMP data within a real time graphical form. Each Chart object contains settings that are associated with displaying data via a line graph. A Chart is given a Name, a Time Interval Value, a Time Unit, an OID Node, and a Device association. The following figures show the HTML pages used to view/edit, add or delete an OID unit.

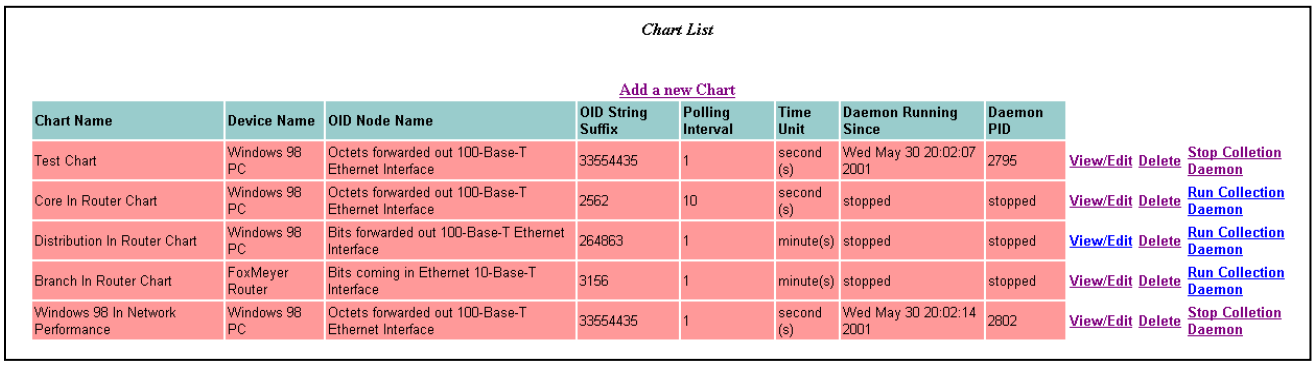


Chart List

Add a new Chart

Chart Name	Device Name	OID Node Name	OID String Suffix	Polling Interval	Time Unit	Daemon Running Since	Daemon PID			
Test Chart	Windows 98 PC	Octets forwarded out 100-Base-T Ethernet Interface	33554435	1	second (s)	Wed May 30 20:02:07 2001	2795	View/Edit	Delete	Stop Colletion Daemon
Core In Router Chart	Windows 98 PC	Octets forwarded out 100-Base-T Ethernet Interface	2562	10	second (s)	stopped	stopped	View/Edit	Delete	Run Collection Daemon
Distribution In Router Chart	Windows 98 PC	Bits forwarded out 100-Base-T Ethernet Interface	264863	1	minute(s)	stopped	stopped	View/Edit	Delete	Run Collection Daemon
Branch In Router Chart	FoxMeyer Router	Bits coming in Ethernet 10-Base-T Interface	3156	1	minute(s)	stopped	stopped	View/Edit	Delete	Run Collection Daemon
Windows 98 In Network Performance	Windows 98 PC	Octets forwarded out 100-Base-T Ethernet Interface	33554435	1	second (s)	Wed May 30 20:02:14 2001	2802	View/Edit	Delete	Stop Colletion Daemon

Figure 15. HTML page of currently configured Charts.

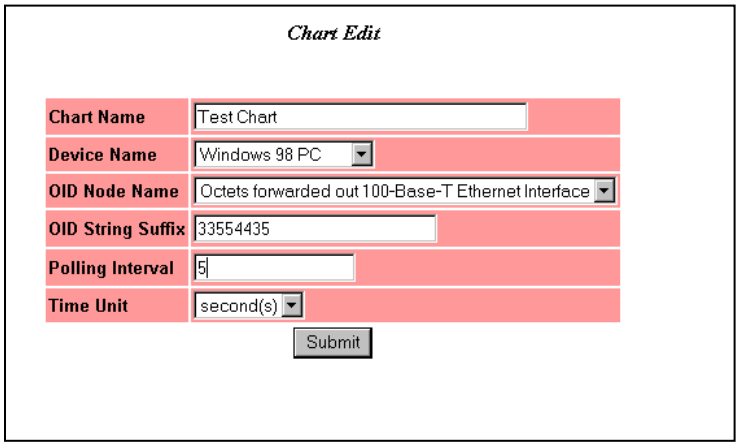


Chart Edit

Chart Name	Test Chart
Device Name	Windows 98 PC
OID Node Name	Octets forwarded out 100-Base-T Ethernet Interface
OID String Suffix	33554435
Polling Interval	5
Time Unit	second(s)

Submit

Figure 16. HTML page to view/edit or add a Chart.

3.3.1.7 Web Pages

Web Pages are custom configured HTML views, which display the SNMP collected data via a Java Applet Chart. A Data view page is given a Name, and Chart association. Many Web Pages can be associated with a single Chart. The following figures show the HTML pages used to view/edit, add, delete and run Web Pages. All other settings, such as changing the chart settings, are performed upon the configured Web Page itself

Figure 17. HTML page of currently configured Data View Pages.

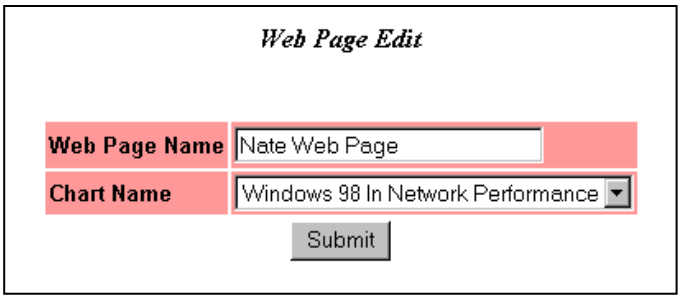


Figure 18. HTML page to view/edit or add a Data View Page.

4. Deliverables

1. The *SNMP Smart Chart* will collect MIB variable data from networking devices using the Simple Network Management Protocol (SNMP).
2. The *SNMP Smart Chart* will have the ability to poll networking devices with SNMP "get" requests over an extended period of time.
3. The data returned from the SNMP enabled networking devices will be saved within ASCII text files.
4. The SNMP polling module will be written in PERL.
5. The *SNMP Smart Chart* will have an HTML statistics output page.
6. The user will have the ability to save the configuration settings of the HTML statistics about page.
7. A Java applet will reside on the HTML statistics output page.
8. The Java applet will communicate with a server side Java class to query the ASCII data files.
9. The Java applet will produce a line graph to represent the SNMP collected data.
10. All Web pages and Java applets will follow a consistent color theme.

5. Development

The following sections will show the Timeline, Budget, Software and Hardware used to develop the *SNMP Smart Chart*.

5.1 Timeline

Date	Accomplishment
12/10/00	Complete Final Proposal Report and presentation
1/7/01	Finish build of test lab
1/21/01	Investigate and obtain necessary SNMP object libraries
1/28/01	Using Perl programming, successfully return variable information from a SNMP MIB agent on a SNMP agent simulator
2/4/01	Complete final proposal report and presentation
2/18/01	Using Perl programming, successfully store SNMP collected data in an ASCII text file
2/15/01	Complete the rough draft Design Freeze report
2/22/01	Program Java server class to receive a data stream
3/1/01	Program Java server class to query the data ASCII text files
3/2/01	Complete final Design Freeze report
3/12/01	Program Java applet data retrieval functionality
3/12/01	Program Java applet line graph production functionality
3/15/01	Complete Design Freeze presentation
3/22/01	Program Perl module to periodically poll SNMP devices over an extended period of time.
3/29/01	Program Perl module daemon to write collected data to ASCII text files
4/5/01	Complete configuration console HTML functionality
4/12/01	Complete CGI-Bin configuration console Perl modules
4/19/01	Place a "Run" and "Stop" polling daemon control on the configuration console Web page.
4/26/01	Debug Java modules
5/3/01	Debug Perl modules
5/10/01	Incorporate a consistent color theme into HTML pages & Java applet
5/20/01	Fine tune application to be deployed within a production environment (i.e. Ensure interface is intuitive and functional)
6/1/01	Complete final Project Report and presentation

5.2 Budget

Server PC (1GHz Pent, 40GB HD, 256MB Ram, 17" Mon.)	$	2,000.00
Client PC (500MHz Pent, 10GB HD, 128MB Ram, 17" Mon.)	$	1,000.00
Cisco 1600 Series Router (For test lab purposes)	$	1,139.00
Linux 2.2 Operating System (Red Hat 7.0 distribution)		Free
Apache 1.3 Web Server		Free
MySQL 3.23 Database Server		Free
Java 2 Development Kit (SDK 1.3)		Free
Perl 5.6		Free
Misc. Cabling (Cat 5 patch)	$	20.00
Total	$	3,959.00

5.3 Software

The *SNMP Smart Chart* application has been solely developed and implemented using open-source software strategies. Open-source software is a distributed program that includes the source code, and allows for re-distribution of the program in source code as well as compiled form. One popular avenue for protecting open-source software is by committing the software to the terms of the GNU Public License (GPL). The GPL ensures that code that begins free remains free. All of the open-source software used by the *SNMP Smart Chart* is protected by GPL. There are, of course, software projects, which would not be well suited to the open-source approach. But it was decided that a network centric programming project, such as the *SNMP Smart Chart* application, was an excellent fit for an open-source approach for the following reasons

- SNMP in itself is an open standard, which is not confined to a commercial license. SNMP versions 1 and 2 standards are maintained and controlled by an independent body called the Internet Engineering Task Force (IETF) within the RFC repository.
- SNMP development tools and components tend to be open-source (I.E. the SNMP.pm Perl module available via CPAN)

- Open-source software is free so the cost of application development and implementation is generally lower than commercial software.
- Open-source software has very strongly tested and seasoned history within the infrastructure/back-end side of the software spectrum.
- The open-source software used within the *SNMP Smart Chart* has been ported to many platforms, which increases the portability of the underlying Smart Chart application. (I.E. Perl, Java, MySQL, and Apache are all available for Windows, Linux and many Unix vendors)
- The open-source software approach allowed me the opportunity to improve my skill set.

The following is a list of the open-source software components implemented within the *SNMP Smart Chart* application. All of the components listed are freely available and downloadable from the Internet.

5.3.1 Platforms

- Linux 2.2 Operating System (Red Hat 7.0 distribution)
- Apache 1.3 Web Server
- MySQL 3.23 Database Server

5.3.2 Languages

- Perl 5.6 (Practical Extraction and Report Language)
- Java 2 (SDK 1.3)

The Java API is owned by Sun Microsystems and is *not* open sourced. But Java is highly favored in many open-source development strategies because Sun has elected to freely distribute the Java SDK (Software Development Kit) and JRE (Java Runtime Environment). This free distribution of Java helps to contribute to the creation of open-source software.

5.3.3 Modules

- mod_perl (Apache module which allows Perl scripts to be interpreted from within a cgi-bin directory)
- CGI.pm (Perl module which allows server side dynamic HTML creation)
- SNMP.pm (Perl module which allows SNMP communication over TCP/IP)
- DBI.pm (Perl module which allows independent database connectivity)
- DBD::MySQL (DBI driver for MySQL connectivity)
- JDBC (Java class which allows independent database connectivity)
- MM.MySQL (JDBC driver for MySQL connectivity)

5.4 Hardware

The hardware used to implement the SNMP Smart Chart application was a single PC consisting of the following.

- Pentium 100 MHz Processor.
- 48 Mbytes of Memory.
- 1 Gbyte Hard Drive.
- 10-Base-T NIC.

6. Proof of Design

The following sections will discuss how the implemented *SNMP Smart Chart* application meets the original objectives of the project, specified in the *Deliverables* section 4.

6.1 Web Implementation

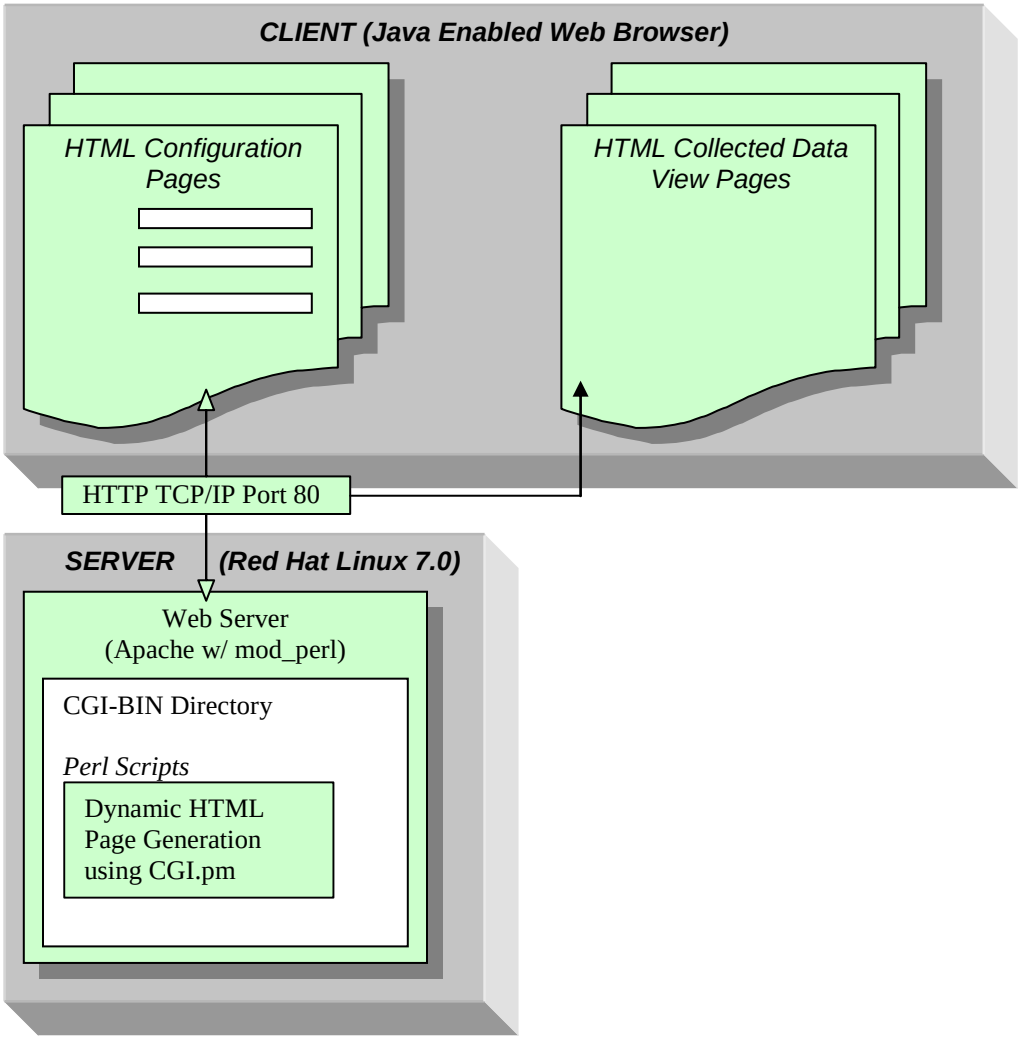


Figure 19. Logical Web Implementation

Figure 19 is a subset of *Figure3*. This represents the HTTP communication between the Web server and Web browser. Perl scripts, utilizing the CGI.pm module, generate dynamic HTML from the CGI-BIN directory of the server. The configuration, and

collected data view pages provide the user interface to the *SNMP Smart Chart* application. The configuration pages are used to populate the configuration tables within the MySQL database. The data view pages are used to view the collected data from the collected data tables. The following section will examine the HTML configuration pages.

6.1.2 Web Server Specifics

An Apache 1.3 Web server, set to listen on TCP/IP port 80, handles all HTML formatted text delivery to the client browser and receives user input. All HTML that is delivered to the client browser is dynamically produced via Perl scripts within the CGI-BIN (Common Gateway Interface) directory in-conjunction with the mod_perl module. Mod_perl is an additional module which is compiled within the Apache HTTP daemon to embed a persistent Perl interpreter within the Web server. This feature allows Perl scripts to be executed by the HTTP daemon while avoiding the overhead of starting an external interpreter and the penalty of Perl start-up time. In short, mod_perl allows an Apache Web Server to directly execute Perl code with in the cgi-bin directory. The root web and cgi-bin directories are located at /var/www/html and /var /www/cgi-bin respectively on the Linux file system.

6.2 Data Storage Implementation

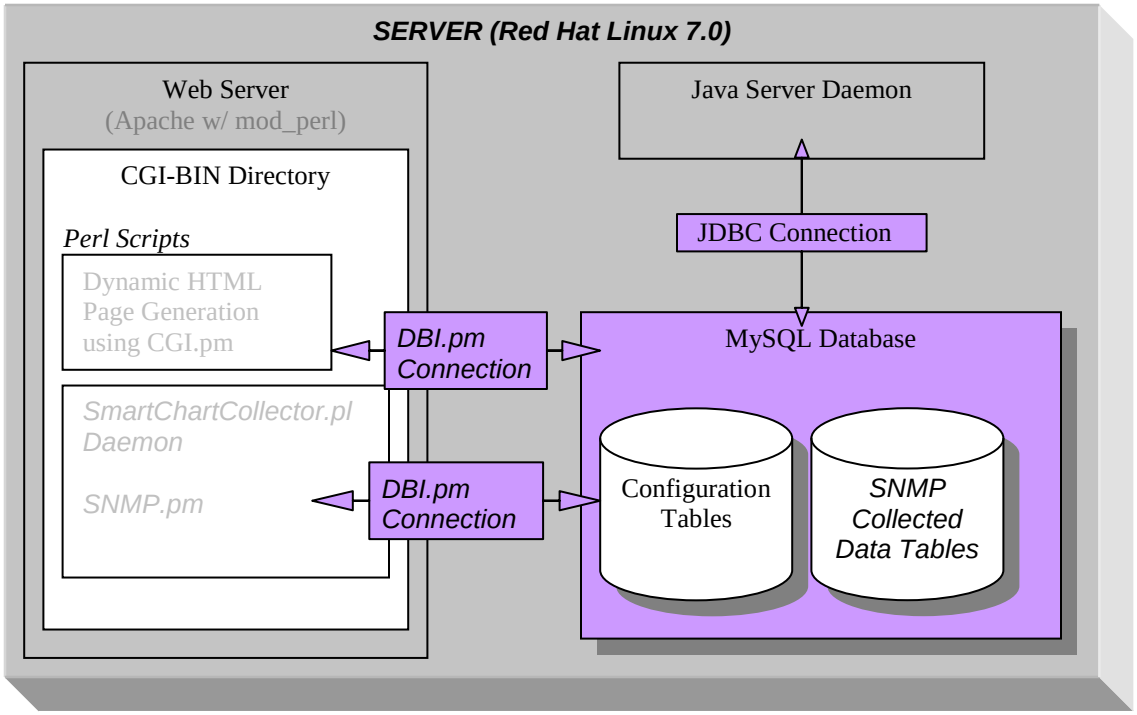


Figure 20. Logical Data Storage Implementation

Figure 20 is a subset of *Figure 3*. This represents the MySQL database communication between the Perl scripts and the Java Server daemon using DBI.pm and JDBC respectively.

6.2.1 Perl DBI Connection to MySQL Database

DBI, which stands for Database Interface, is a database access Application Programming Interface (API) for the Perl Language. The DBI API specification defines a set of functions, variables and conventions that provide a consistent database interface independent of the actual database being used. Perl module DBI.pm contains the DBI specification. This file is included in each Perl script, which accesses the MySQL database. The DBI module also requires a driver module used to interface with a specific database. DBD::MySQL is the driver used to interface between the Perl DBI

API and the MySQL relational database API. Below is an example of the Perl scripting used to make a connection to the MySQL SNMP_Database and initiate a standard SQL SELECT statement.

```
use DBI;
my $dsn = 'DBI:mysql:SNMP_DataBase:localhost';
my $db_user_name = 'root';
my $db_password = 'secret';
my $dbh = DBI->connect($dsn, $db_user_name,
$db_password);

my $sth = $dbh->prepare("SELECT * FROM OID_units");
$sth->execute();
```

6.2.2 Java JDBC Connection to MySQL Database

Java JDBC, which stands for Java Database Connectivity, is very similar in concept to the Perl DBI interface. JDBC exposes an API, which allows access to virtually any tabular data source from the Java programming language. It provides cross-DBMS connectivity to a wide range of SQL databases. Because the JDBC API is database independent, as DBI is, developers can take advantage of Java's "Write Once, Run Anywhere capabilities". JDBC also requires an intermediate driver, called appropriately a JDBC driver. This driver serves as an interface between the JDBC API and MySQL's relational database API. Below is an example of the Java code used to make a connection to the MySQL SNMP_Database and to initiate a standard SQL SELECT statement.

```
// Load the MySQL database driver
  Class.forName( "sun.jdbc.mysql.JdbcOdbcDriver" ) ;
// Get a connection to the SNMP_Database
  Connection conn = DriverManager.getConnection(
  "jdbc:mysql:SNMP_Database" );
  Statement stmt = conn.createStatement() ;
  ResultSet rs = stmt.executeQuery( " SELECT * FROM
  OID_units " ) ;
```

6.2.3 MySQL Database Table Schema

The following *Figure* 21, shows the tables, fields, and relationships created within the MySQL database server. The database is named SNMP_Database, and during the design of the table layout, a high priority was set on achieving the 1st and 2nd rules of normalization. This objective was achieved, while still making the database straightforward and efficient from a developer's perspective. There is nearly a one-to-one relationship between the table names in the SNMP_Database and the user interface objects discussed earlier within section 3.3.1, regarding the HTML Configuration Pages. Logically, the tables within the SNMP_Database can be further divided into those that store the configuration settings, and those that store the collected SNMP data. *Figure 21*, illustrates this logical relationships of the tables. The SmartChartCollector.pl daemon and the JavaServer.class daemon are discussed within sections 6.3 and 6.4 receptively.

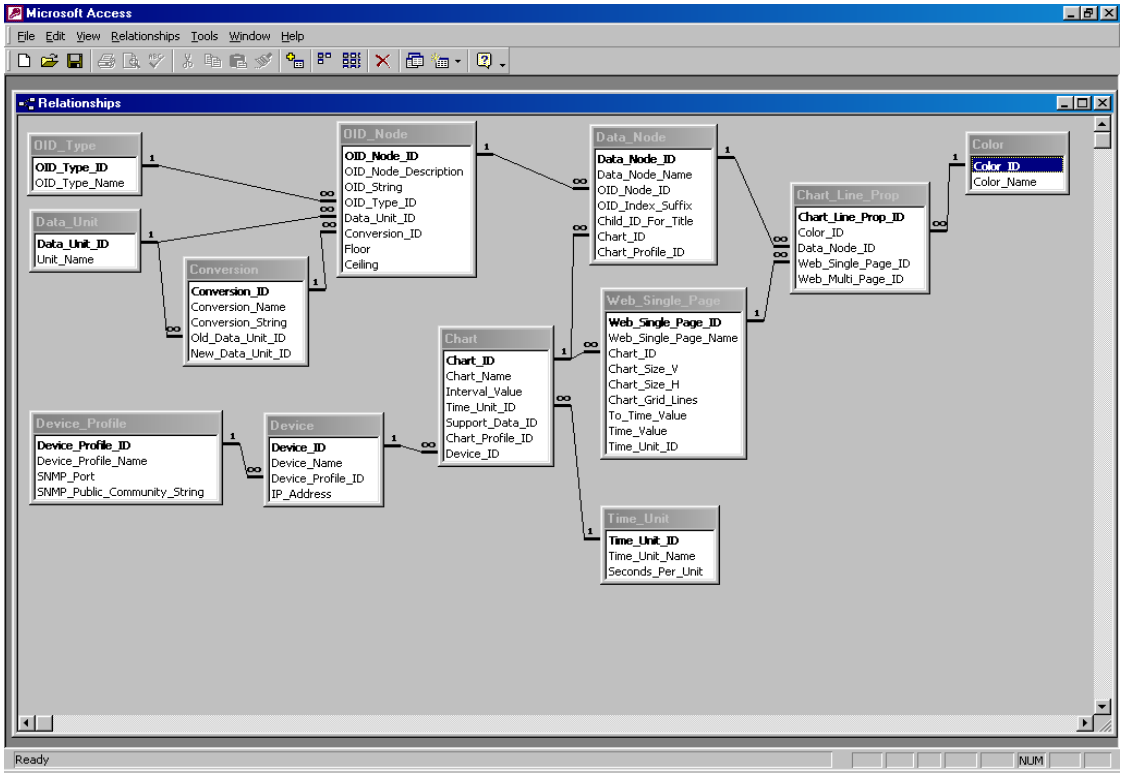


Figure 21. MySQL Database Table Relationships

6.3 SNMP Implementation

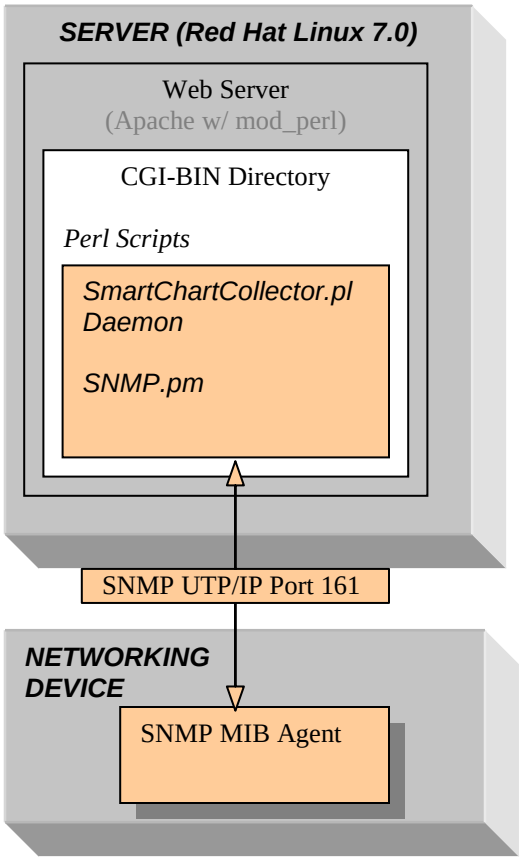


Figure 22. Logical SNMP Implementation

Figure 22 is a subset of *Figure 3*. This represents the SNMP communication between the *SmartChartCollector.pl* daemon and the SNMP MIB agent on the networking device. SNMP communication is implemented within the *SNMP Smart Chart* application from within the Perl scripting language. PERL is an interpreted scripting language, in which is highly supported by many operating systems. CPAN (the Comprehensive Perl Archive Network) is a central repository for Perl modules. The SNMP Smart Chart utilizes the SNMP.pm Perl module from CPAN. This Perl module is a library of utilities for configuring and monitoring SNMP based devices. This library requires the UDP port of SNMP. To use the SNMP.pm module, You must first do a "use" statement to pull in the

library. Then the SNMP object can be created. The following is an example of Perl scripting used to create and the SNMP object.

```
#!/usr/local/lib/Perl
use SNMP.pm;
use SNMP::Util;

$snmp = new SNMP::Util(-device => $IP,
                       -community => $community,
                       -timeout => 5,
                       -retry => 0,
                       -poll => 'on',
                       -poll_timeout => 5,
                       -verbose => 'off',
                       -errmode => 'return',
                       -delimiter => ' ', )
```

The SNMPSmartChart.pl Perl module runs as a daemon on the server computer. A daemon is a process which can run in background and is not dependent upon a parent process, such as a user's login shell process. This is accomplished by forking an additional process from the Unix/Linux command shell when executing the Perl script. On Unix/Linux based systems this is accomplished by proceeding the command with an **&** (ampersand).

```
perl SNMPSmartChart.pl &
```

The command is executed via a Perl script initiated from the cgi-bin directory of the Web server. Since the SNMPSmartChart.pl script contains an infinite looping structure, the SNMPSmartChart.pl daemon remains in memory and can collect data over an extended period of time.

6.4 Applet Chart & Java Sever Daemon Implementation

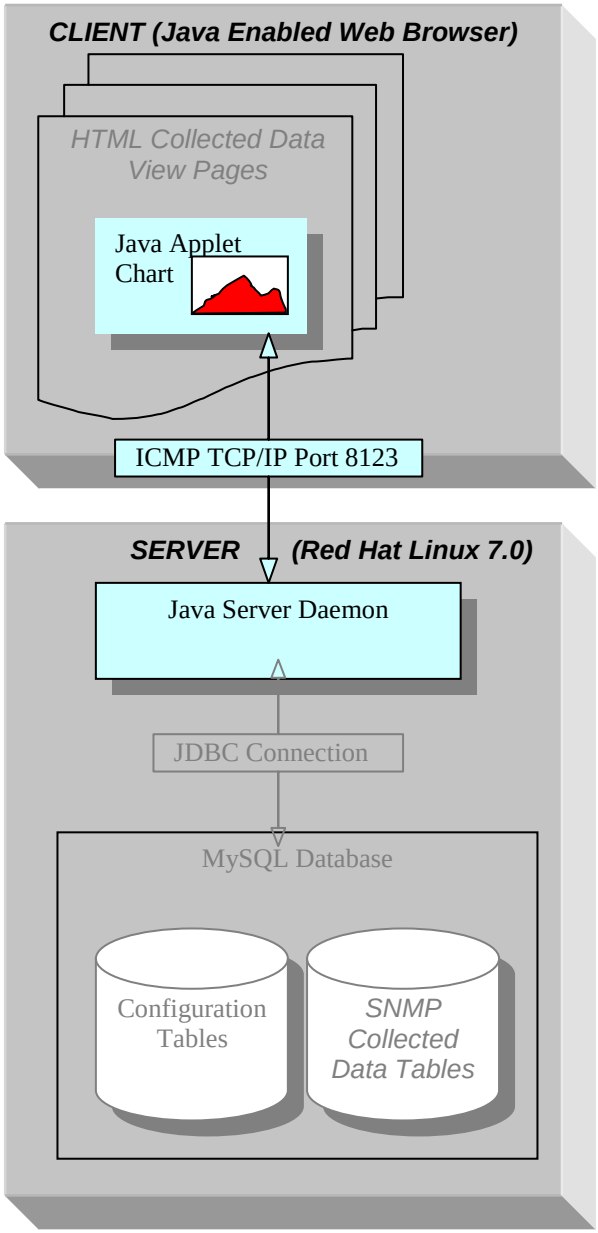


Figure 23. Logical Applet Chart & Java Server Daemon Implementation

Figure 23 is a subset of *Figure 3*. This represents the Java ICMP communication between Java Applet Chart and the Java Server daemon.

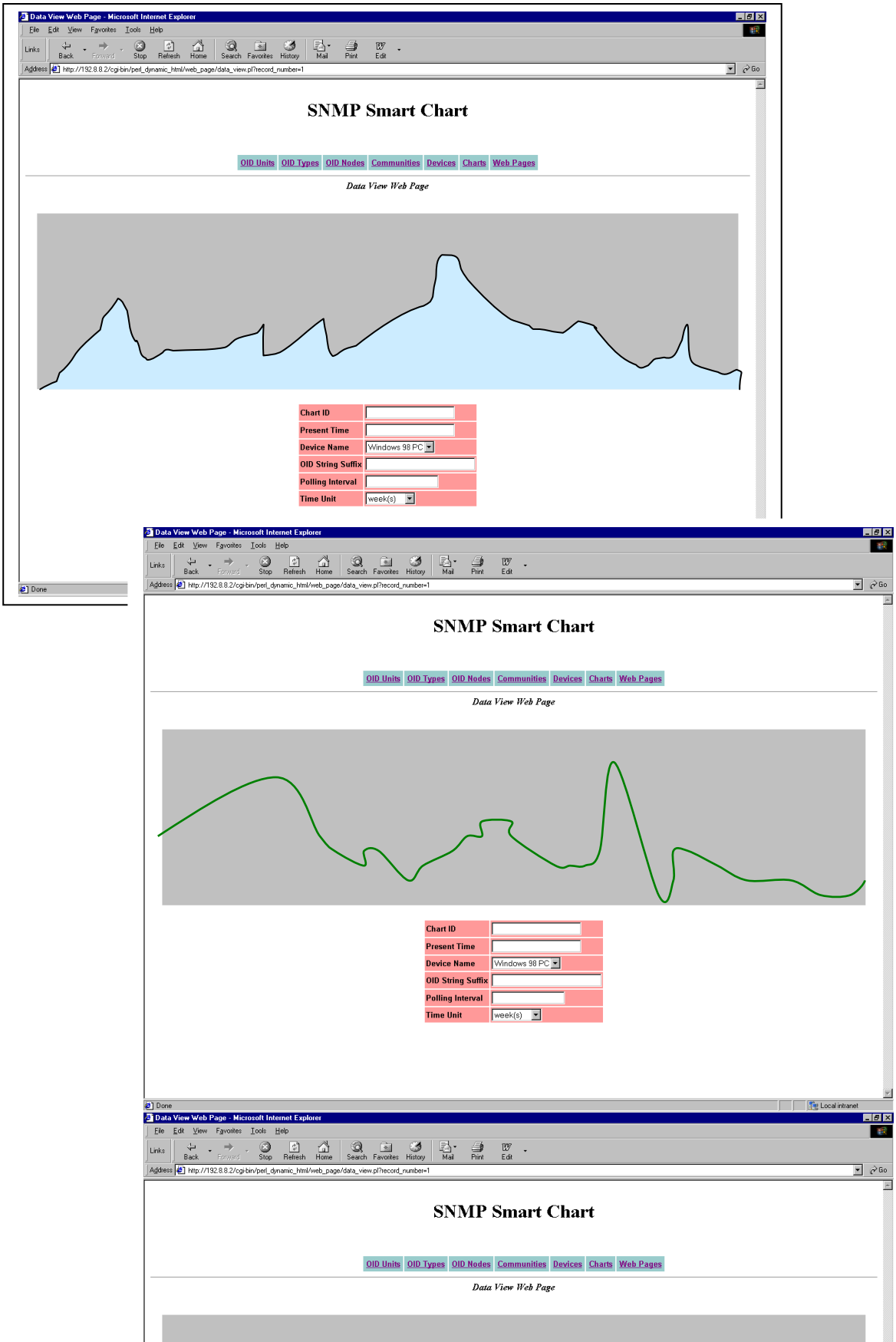
Below is an example of the Java code to implement Java applet network connection to the Java Server.

```
        // Load the MySQL database driver
        Socket connectToServer = new Socket("10.1.1.1", 8123);

        BufferedReader isFromServer = new BufferedReader(new
        InputStreamReader(connectToServer.getInputStream()));
        PrintWriter osToServer = new
        PrintWriter(connectToServer.getOutputStream
        (), true);
        osToServer.println("C|"+ jtfCollectionID.getText() +"|"+
sToVal
        ue +"|" + jtfFromValue.getText() +"|"+ jtfFromUnit.getText() );
        sRecieve = isFromServer.readLine();
```

Below is an example of the Java code within the Java server used to to receive the network connection request from the Java applet.

```
      // Create a server socket
      ServerSocket serverSocket = new ServerSocket(8123);

      while (true)
      {
        // Listen for a new connection request
        Socket connectToClient = serverSocket.accept();

        // Create a new thread for the connection
        ThreadHandler thread = new ThreadHandler(connectToClient, i);

        private Socket connectToClient; // A connected socket
        BufferedReader inDataFile = null;

        public ThreadHandler(Socket socket, int i)
        {
      connectToClient = socket;
      // Create data input and print streams
      BufferedReader isFromClient = new BufferedReader(
        New InputStreamReader(connectToClient.getInputStream()));
      PrintWriter osToClient =
        new PrintWriter(connectToClient.getOutputStream(), true);

      String sRecieve = isFromClient.readLine();
      StringTokenizer stRecieve = new StringTokenizer(sRecieve,"|");
      String sToken = stRecieve.nextToken();
```

The following figures show examples of the Java Applet Chart embedded within the HTML data view page.



7. Conclusions and Recommendations

Currently the *Data Comm.* department does not have an accurate and reliable method for measuring network performance. What is needed to measure network performance is a formal *Network Management* system. Network management protocols have been developed to automate the process of network management. SNMP is an application layer protocol offering network management services in the Internet Protocol suite. The *SNMP Smart Chart* application is used to gather statistics from SNMP capable networking devices, such as a router, switch or server. The tool stores the collected data within a MySQL database, which can then be used to create graphs. The graphs are created in real time by a Java applet. The applet is viewed on an HTML formatted file that is made available on an Apache Web server running on Linux. *SNMP Smart Chart* incorporates a polling feature written in Perl, which allows statistics to be gathered over extended periods of time. Once configured correctly the tool will collect data and supply a live visual representation of network performance. The network engineers within the data communications department at Provident Bank will be the users of the *SNMP Smart Chart* application. The application will aid network engineers in gathering network statistics and taking a proactive approach in performing upgrades and design decisions.

The *Proof of Design* section 6, as shown how the implemented *SNMP Smart Chart* application has met nine of the ten original objectives for the project that are specified within the *Deliverables* section 4. The following will explain how one objective was not completely adhered to, because of a clear opportunity in producing a more favorable result.

Deliverable No. 3 for the *SNMP Smart Chart* states, "the data returned from the SNMP enabled networking devices will be saved within ASCII text files. I originally choose to use ASCII text files, as opposed to a formal database system, as an attempt to simply the requirements needed to run the application. If the application did not require a separate database to run, then the application would be easier to re-install and would be easier to port to other platforms. I opted to store both the configuration settings and the collected SNMP data within ASCII text files. Not to far into the develop of the application it became apparent to me that to make the application flexible and powerful for large networks, such as Provident Bank's, I would need to store the configuration settings within a relational data model. This could possibly have still been done using text files, but the coding required to maintain the files have been difficult. Adding, editing, and deleting records from multiple related tables within a SQL compatible database requires far less code overhead, with the programming components available today, than attempting to manipulate text files in a similar fashion. Since the deliverables did not specify where the configuration settings would be stored, I opted to use the open-source MySQL database for the storage and management of this data. Once the application became dependent upon an external database service it seemed quit efficient and at no benefit to continue storing the SNMP collected data within text files. For this reason, the data returned from the SNMP enabled networking devices is not saved within ASCII text files but instead within a MySQL database. Even though this is not in accordance with deliverable No. 3, the application is far more efficient and robust for using a formal database system.

For me, the implementation of the *SNMP Smart Chart* application was as much about learning new technologies as it was about the underlying SNMP data collection. I had only minimal applied experience using Perl and Java, and had never used Linux, the Apache Web server or the MySQL database. Getting all of these different technologies to work together proved to be a great challenge for me. I feel my skill set has been significantly improved as a result.

References

1. Scoggins, Sophia and Adrian Tang. *Open Networking with OSI*. Toronto: Prentice-Hall, 1992.

2. Fang, Karen and Allan Leinwald. *Network Management: A Practical Perspective*. Don Mills: Addison Wesley, 1993.

3. Rose, Marshall. *The Simple Book*. New Jersey: Prentice Hall, 1994.

4. SNMP. "*SNMP, Network Management Software*". http://snmp.cs.utwente.nl/software/select_obj.php3. 1999.

5. Stallings, William. *SNMP, SNMPv2, and CMIP*. Don Mills: Addison-Wesley, 1993.