Automation Testbed for 3Com Internet Protocol Testing Suites

Adam Fairbanks
Advisor: Professor Mark Claypool

Major Qualifying Project MQP-MLC-AT00
Computer Science Department, WPI
Terms E 2000

Abstract

3Com's Software Quality Assurance group needs automation, especially in the Internet Protocol tests that are part of the SmartBits testing library. We designed and implemented a 3Com Internet Protocol Automation Suite that was created as both an aid for Test Engineers at 3com and as part of an Automated Testing System with other protocols' automated tests. Our two-part automation testbed solution satisfies all of the requirements and goals for the project.

1 Introduction

The Software Quality Assurance (SQA) Group of any networking company tests the software released by the Development group of the company and reports bugs in the software when they run a series of well set-up and documented tests on the hardware the software runs on. These tests are set up in bundles for specific protocols, and are designed to not only show that the protocols works for some of the protocols capabilities, but that they perform under all conditions right up to the limits of what the hardware capabilities are as well. The SQA group is usually divided into teams for the different protocols that run on the specific hardware, regardless of the networking layer they use. Then the Development group that coincides with the group for SQA will work with them to try and resolve the bugs that occur for that protocol or platform. An example of this would be the Open Shortest Path First (OSPF) teams from both Development and SQA working together to resolve a bug in which the OSPF packet coming out of the router does not advertise the right routes of its network.

A problem that arises in SQA testing is that some of the commands and procedures used are both tedious and monotonous, so that it takes up much of a testers day simply typing in commands, waiting for the result, then typing in more commands and so on. Multitasking in testing then becomes hard to do, as you are waiting for a response from an input to continue on one test, but it is not really enough time to setup another test before the response comes back, so the person must wait. Automating these procedures can solve multiple problems and it also allows for new avenues in testing.

The amount of time saved can be so great that it cuts testing from weeks to hours. A script used at 3Com developed by Ben Douglas, an alumnus of WPI, helped 3Com

Employees download the latest version of beta code from the directories they were in onto the hardware. Normally half an employee's day would be taken up if they had 20 or so pieces of hardware to download the new code. The script's main feature is that is dynamic, and can "look" at the individual blades using SNMP, determine what type of hardware it was (a router, a switch, how many ports, etc) and download the appropriate code for them. This takes a maximum of 2 hours, so it saves each individual person around 4 hours (assuming it took 6 originally). Multiplied by the number of people in SQA using the script at the time, which was around 80, which comes to 320 man-hours per beta code release saved. Beta code releases happened every two weeks, and the time saved can be quite substantial and well worth the time to develop the automation.

Multitasking with automation becomes increasingly easy, as you can either have the scripts do multiple tasks, or set off multiple scripts. You can even have scripts that set off scripts, and this is where the opportunities for huge time saving takes place. The time saved by automation can be used for other things such as non-automatable tests (physical network configurations, line switches, etc.) or to create or run other scripts.

1.1Networking

The main focus of the SQA group mentioned earlier is to test the routers and switches and other hardware that will later be used by consumers and businesses to create and maintain connections to the Internet. Networking, and more specifically the Internet, has been around since 1969, when the Advanced Research Projects Agency Network (ARPANET) was first formed by the U.S. government [Ner00]. The idea was to have a decentralized network for electronic communication that would not only allow for remote access to computers in other parts of the country, but could also function the same if parts of it were destroyed, for example, in a nuclear bombing attack. This worked well for the government, and soon a select group of universities were given permission to use the network as well to exchange files for educational use. Eventually this led to more schools using it, and soon non-educational and for-profit agencies began to use the Internet, as protocols and formats for communication such as ftp and the World Wide Web made communication rich and easier to use. Throughout the entire life of the Internet, there have been companies who have created and improved upon the infrastructure of hardware and software that is used. Therefore, having testing procedures and more specifically automation tools that are complimentary to this goal of improvement of hardware help to further the growth of the infrastructure of the Internet.

1.2 3Com Corporation

Currently networking is among the most rapidly expanding industries in the world, with annual sales expected to surpass US$50 billion by 2001 [3c100]. 3Com holds leading market positions in solutions for both consumers and business applications, all with the common benefits of simplicity and reliability. They include fixed-port and stackable Ethernet switches, Ethernet hubs, multi-services access platforms, desktop modems, LAN and modem PC Cards, desktop and server network interface cards (NICs), and handheld organizers. This coverage of solutions gives customers the ability to buy complete networking solutions from 3Com for any network application.

Total network availability is what 3Com strives to produce. Providing customers with reliable networking solutions that support mission-critical voice, video and data E-business applications. 3Com provides users with secure access to those applications, regardless of their physical location.

3Com is a leader in the industry for Ethernet, Fast Ethernet, and Gigabit Ethernet connectivity, specifically NICs, switches, and hubs.

1.3 Approach

A very time consuming testing environment with the potential for streamlined automation is the area of IP packet testing using hardware packet generators to simulate network traffic. IP packet testing requires complicated tests that require many hours of user manipulated functions that are very repetitive.

Besides the ability to increase the speed of testing with automation, there is a need to integrate these tests into the UNIX based Automated Test Simulation environment at 3Com. Basically, many different types of automated tests are joined in this environment consisting of many different types of routers and switches, and they are run over these devices for each release of beta code, making it easier to find some of the more critical bugs in the releases faster. By using software to automate the hardware that generates traffic into the routers and switches, a highly controlled and specific test can be performed for various types of protocols. Having an IP automation Testbed in this environment will be a useful tool to diversify the testing in the automation environment, and help with test coverage over a larger area.

2 Background

The IP protocol is very large and is complex in design, but one that is the basis for much of Internet traffic today. An explanation into how the Internet Protocol works is detailed in section 2.3. However, the IP packet tests that will be automated are the Packet Loss Rate Test and the Throughput Test, both of which are included in Netcom's Smart Applications for the SmartBits 2000 packet generator as stand alone tests for large core networking hardware.

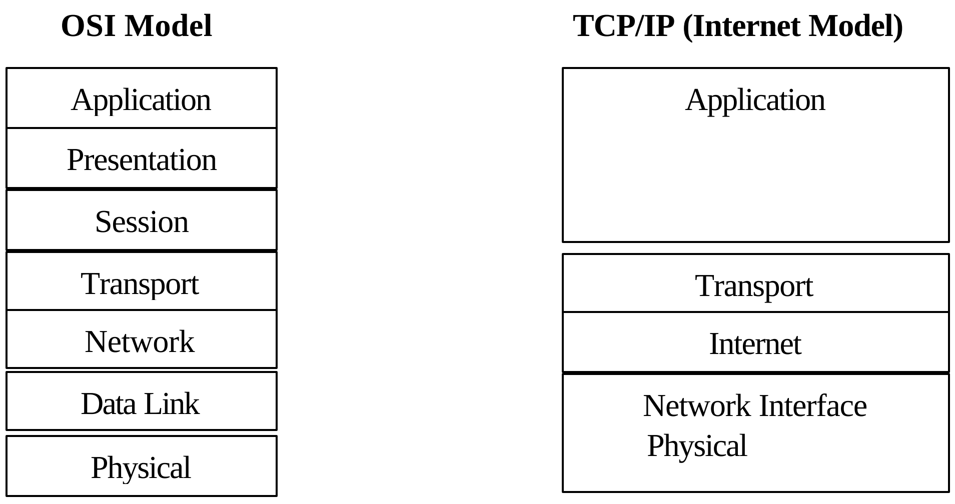
To understand the automation of the hardware, some concepts must first be explained. This section will introduce networking models and hardware, as well as both software that runs the networking hardware and the software that will automate it.

2.1 Models of Networking

Within the world of networking, there are two models that generally describe the scope of the different protocols and the hardware and software that use them.

2.1.1 TCP/IP Model

Figure 2.1: OSI and TCP/IP models



Source: [Tan96]

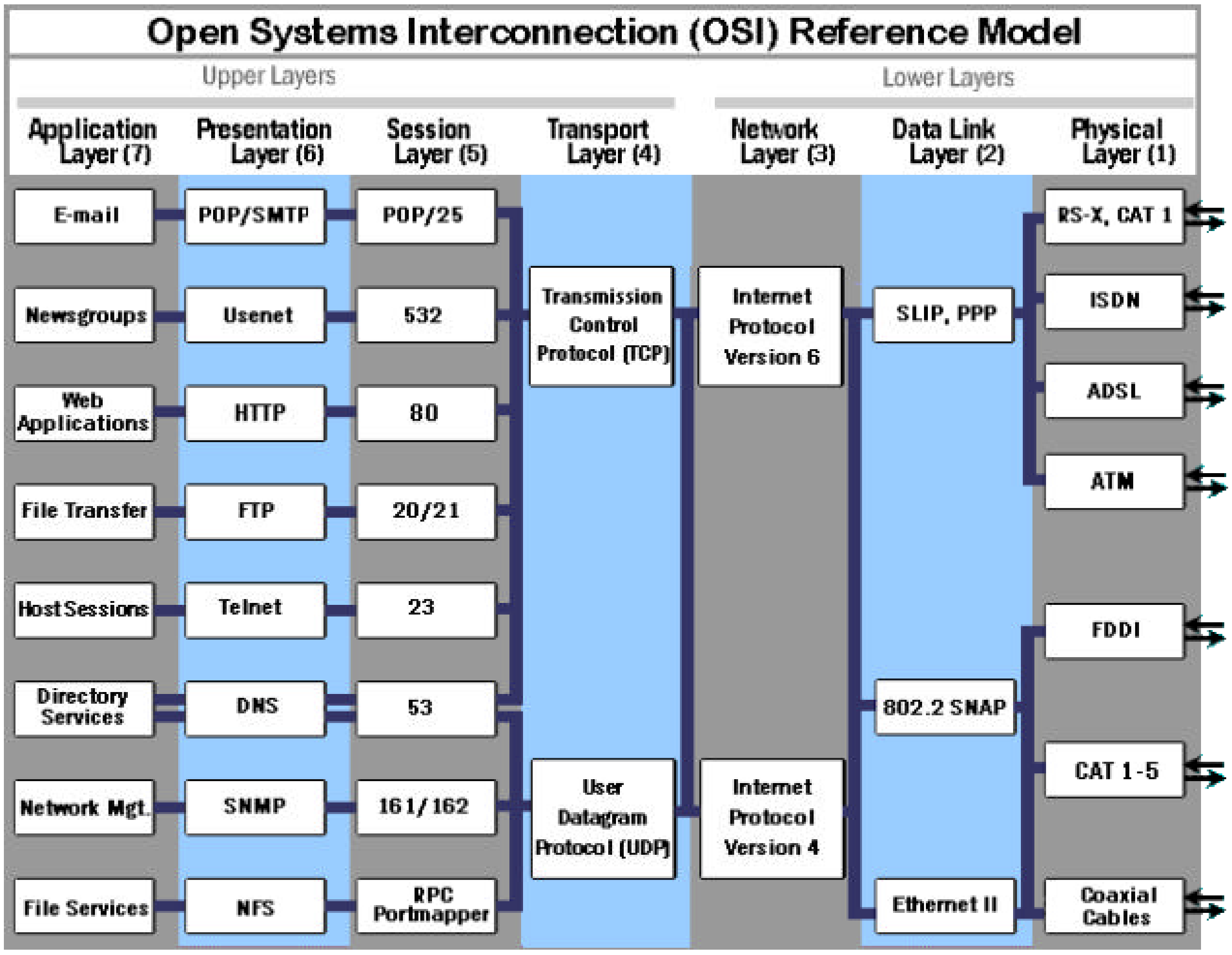
The TCP/IP model was invented to help explain how different virtual layers of networking could be put on top of one-another in order to be useful [Tan96]. The TCP/IP model has there are four distinct layers, representing the major groups of networking. They are virtual because each layer has to go down to the physical layer before it goes out to another piece of hardware, but it will come back up to the same layer that started in order to understand what the first piece of hardware was transmitting. As illustrated in Figure 2.1, the lowest level represents the physical wire and some of the very basic protocols that are used at that level. On top of that is the Internet level, upon which the protocols that drive the Internet such as IP are located. This layer is where the data is transported, and where computers and hardware communicate with each other. The transport layer above it is used to primarily have protocols that manage the ones on the Internet level for such things as managing errors, quality of service (the ability to have connections as at a guaranteed rate), and other such tasks. The final level is the application level, where the computer uses the information from the transport layer to communicate with its programs and present it to the user.

This model, although beneficial in its simplicity, has limited descriptive powers for the complex world of networking, and some protocols simply don't fit right. An example of this would be a protocol called Ethernet at the Network Interface level, which needs a protocol below it to go across a wire. However, since the lower two levels are joined in the TCP/IP model, Ethernet can not have anything below it.

2.1.2 Open Systems Interconnection model

Although more complicated, the OSI model is flexible model for understanding the virtual layers of networking, and where hardware, software and the protocols that make them connect are [Tan96]. OSI stands for Open Systems Interconnection, and it is divided up into seven virtual layers.

Figure 2.2: OSI model with protocols



Source: [Wha00]

As can be seen in Figure 2.2, there are many different protocols within the seven layers. An example that could show some of these would be if there were a TCP connection going on between a computer and a networking hardware, which is the Transport layer. A packet would originate from computer on layer four, but it would be split up into IP packets (Network layer). These packets would then be split up into Ethernet packets (Data Link layer), which would then be split up into any number of Physical layer packet designs, and go out over the wire to the networking hardware. The hardware would then receive these packets, and assemble them to form Ethernet packets, which would assemble to form IP packets, which would finally form TCP packets, which would give the hardware the packets the computer sent out on the original level it was sending on. From both the hardware and computer's perspectives, the layers do not really exist, as a layer 4 packet gets from the computer to the hardware as if it's directly going layer 4 the entire time, even though in reality it's going all the way down to layer 1 before it goes out.

2.2 Routing and Switching

Since the IP protocol is in the Network Layer as can be seen in Figure 2.2, and the fact that there are many different types of hardware that do functions in both Data Link Layer and the Network Layer, it is important to understand specifically what is the function of both routers and switches.

2.2.1 Switches

A switch directs Data Link Layer traffic from different Media Access Control (MAC) addresses to their respective destinations [Dav00]. The job of a switch is fairly simple: it reads the Data Link Layer packet's destination MAC address, looks up the address in a table it has stored on the hardware, and then sends the packet out the port that that address is associated with.

Figure 2.3: Data Link Layer Switching

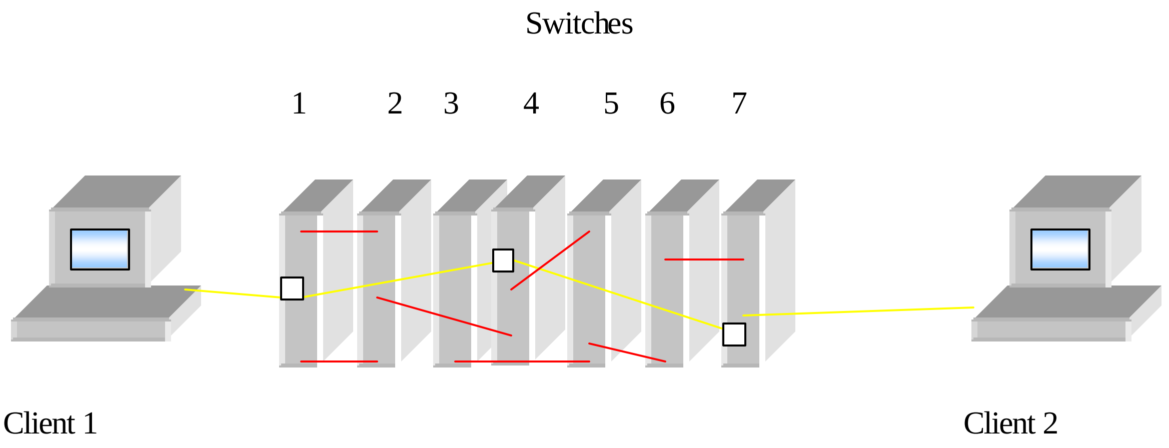
Client 1 Sends:

Source 00:00:00:00:00:01

Destination 00:00:00:00:00:12

Client 2’s MAC:

00:00:00:00:00:12



Switch 1 MAC table:
00:00:00:00:00:03 port 1
00:00:00:00:00:05 port 2
...
00:00:00:00:00:12 port 5
...

Switch 4 MAC table:
00:00:00:00:00:23 port 1
00:00:00:00:00:12 port 3
...

Switch 7 MAC table:
...
00:00:00:00:00:12 port 9
…

Source: [Dav00]

Figure 2.3 shows how the address tables do not necessarily tell the switch where the end point destination of the packet will be, but rather tells it the port to send the packet out, possibly to another switch (each table above is for its own respective switch labeled in the diagram). So following the example above, Client 1 sends out a packet with a destination address of 00:00:00:00:00:12 to switch 1. Switch 1 sees that that address should go out port 5 (look at MAC table for switch 1), and forwards that packet out that port. In the diagram we see that port 5 goes on to switch 4, which determines based on its table that the packet should be forwarded out port 3. Port 3 leads to switch 7, who forwards the packet out port 9 (again, based on its MAC table). This final forward sends it to the second client, the packets final destination. As you can see, this system is simple and effective, as the entire route does not need to be known by every switch, and works very effectively in forwarding the packets to their proper place. Typically, a switch is only concerned with addresses that are inside its own Local Area Network (LAN).

2.2.2 Routers

A router is similar to what a switch would be if it involved Network Layer packet movement, only much more complex (in fact that movement of packets is sometimes referred to as Layer 3 switching). The router's job is to move a Network Layer packet to the destination IP address of the packet [Say00]. This is nowhere near as easy as it seems however, as the IP packets are constrained by more than just their address. To better understand how a router works, the Internet Protocol needs to be better understood.

2.3 The Internet Protocol

The early ARPANET protocols were very slow and subject to crashes, and a new more dependable protocol was needed to remedy the situation [Fei97]. By 1974, a new set of protocols had been developed, and over the next 6 years, hardware that ran the networks were converted over to run the Transmission Protocol and Internet Protocol (TCP/IP). TCP/IP has characteristics that make it an ideal choice for networks to use, as it connects different networks together seamlessly, so that to the end user, it appears as though the entire Internet is one giant network directly connected. The TCP portion of TCP/IP is the Transmission Control Protocol, which is on the Transmission Layer, the layer that connects clients together in peer-to-peer, reliable connection oriented communications. However, this is the Layer above the one that we will test, the Network Layer, on which IP resides. IP is the carrier of most of the traffic on the Internet today and is the basis for many other protocols on levels above and below it.

IP works in many different ways but there are a few general ideas. Unlike the Data Link Layer, which is typically concerned with MAC addresses, the Internet Protocol uses IP addresses, which are used to make connections from one point to another, based on those addresses. For example, if one client with an IP address of 150.1.1.19 connects to one at 200.1.1.14, all the packets from the first client will have a source address in the packet of 150.1.1.19 and a destination of 200.1.1.14. The reverse would then be true of the second client. During communication, packets would go back and forth for different reasons, from transmitting data or reestablishing connections to managing flow as sometimes packets are going too fast for the hardware to handle, so some get dropped and have to be resent. So the packets contain IP addresses in them and varying amounts of data after. But generally speaking, the two IP addresses are what the connection is based on, and determines where the connection is.

3 Approach

Now that the background information has been introduced and explained, this next section discusses the approach that will be used to create the automation, and how the software and hardware will be used.

3.1 Software

The automation is using Tcl/Tk 8.0 as the platform. This scripting language was chosen both for its ease of writing and debugging but also because it will be easy to integrate into the ATS that 3Com has in its current state, as all of the scripts there are currently in Tcl [Rou00].

The scripts for the automation is set up into two different parts, a backend in Tcl and a front-end in Tk. The front-end is in Tk, because it is graphical and has the ability to configure which parts of the scripts will run, the different addresses for the hardware, and other configuration information for the user to set up and run the tests. All of this information is stored in a file that the backend uses. The back-end will be written in Tcl, which is command line oriented, and uses the text file saved by the front-end to determine how to connect, what tests to run, and other configurations told to it by the file. Both of the parts of the automation are bundled together by a shell script that simply runs them in succession with some environment variables set up to allow for multiple users to run the Testbed at once, from the same location.

The two-part approach allows for the back-end to only need the text file written by the front-end to run. Therefore, by simply editing that file for the configuration needed, the back-end can run stand alone without the front-end, and can then be easily integrated into the

Automated Testing System (ATS) at 3Com (which tries to use no graphical configuration, just straight command line scripting).

3.2 Hardware

The hardware used will comprise of three distinct groups of devices: Workstations/Clients, Packet Generators, and Routers/Switches. Each group has different connections physically linked by different types of media.

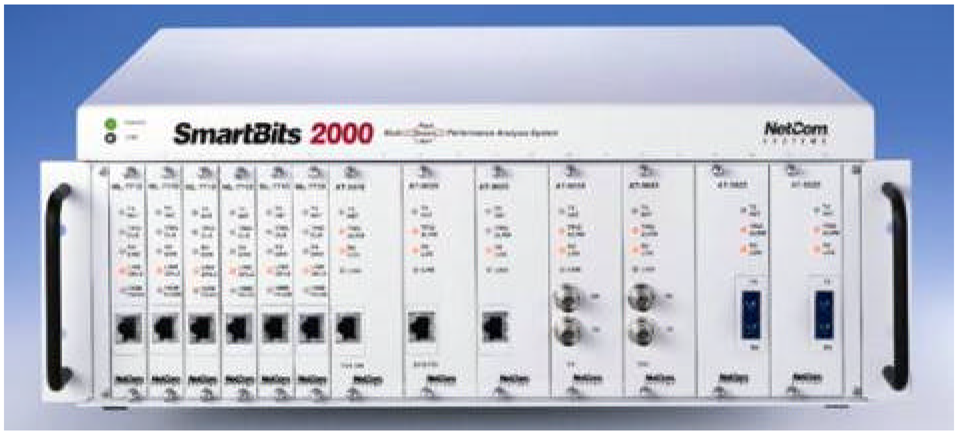
3.3.1 Workstations

There are several different types of computers that could be used to perform the task of setting off the automation, but the platform of choice is UNIX. The specific workstation used is a Sun Microsystems Ultra 60 running Solaris 2.6, which will allow for ease of use with Tcl/Tk.

3.3.2 Packet Generators

The Packet Generators that will be used are the SmartBits 2000 Multi-User Family.

Figure 3.1 SmartBits 2000 Packet Generator



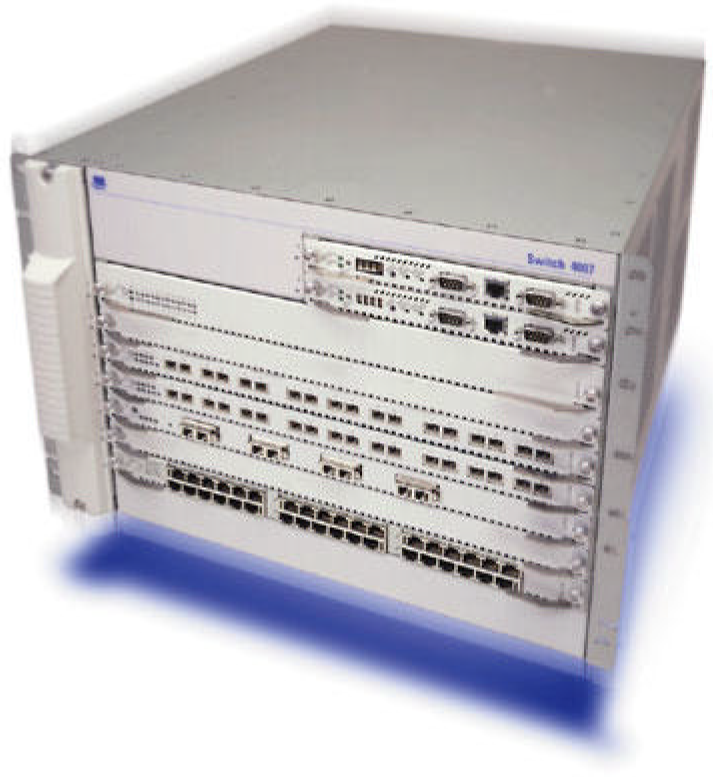
Source: [Net00]

Figure 3.1 shows the generator, which uses multiple slots in which SmartCards slide into and lock in place. Different cards can have different media speeds: Ethernet, Fast Ethernet and Gigabit Ethernet (10, 100 and 1000 Megabits per second respectively). The ability to switch between different cards allows for more flexibility when testing. A console connection or a telnet connection controls the SmartBits Generator, with the generator able to have its own IP address. These features coupled with the SmartBits library, a bundle of software that allows for direct communication between the generator and a Sun Workstation in Tcl or C++ , allow for this hardware to be automated with relative ease.

3.3.3 Routers and Switches

The 3Com hardware that will be tested is fairly complex, as it combines both switches and routers into one self contained unit.

Figure 3.2: 3Com Switch 4007



Source: [3c2 00]

The 3Com 4007 Switch pictured in Figure 3.2 is 7 slot chassis that allows for many different types of modules to be connected into the slots. Each module is actually a router or switch that is controlled by the Enterprise Management Entity (EME), a smaller module that allows for either a console or telnet connection [3c300]. Once connected to the EME, it then allows you to connect to each blade and both configure them and check statistics on the operations the hardware is performing (such as types of packets being forwarded, rate of speed, etc). Not only can these modules use the connections on the front of them, but when they snap into each slot, they also make connections through the back of the chassis, or the backplane, that can also be configured and controlled through a special module called a fabric. The fabric is also controlled by the EME, and allows for configuration similar to any other blade.

3.3.4 Networking Media

For the different connections that will be made, RJ45 connectors (for Ethernet standards) on Category 5 twisted pair copper wire cables is used. They vary in length, but all of them are able to transmit in speeds over 100 Megabits per second. These are used both for the console connection from the Sun Workstation to the SmartBits Packet Generator but also from the SmartBits to the Switch 4007 Network.

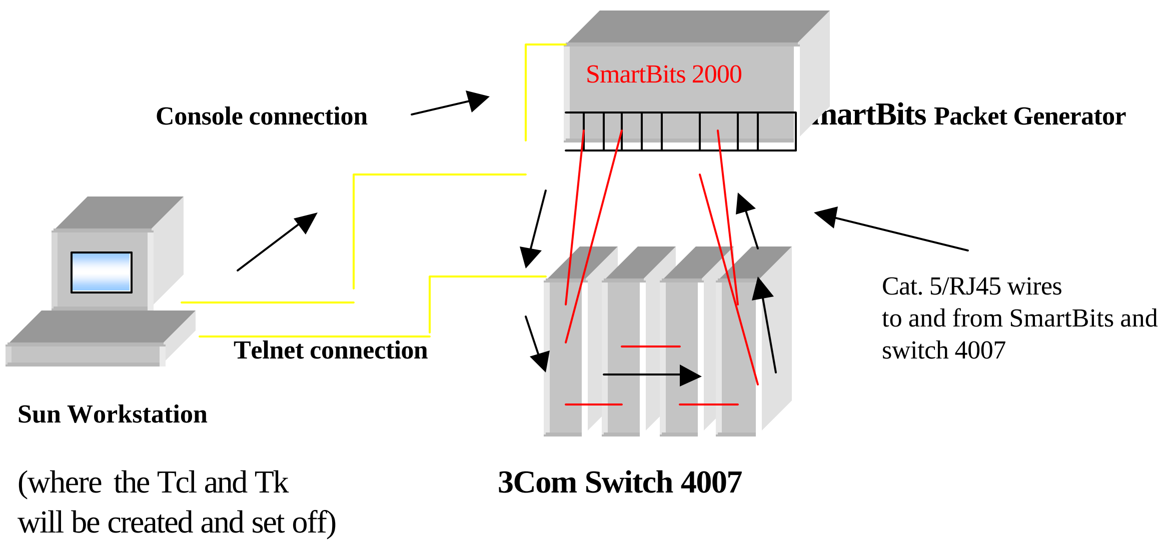
4 Introduction to Methodology

Now that the general groundwork for both the networking protocols and the hardware and software that will be used has been completed, the general way the tests are performed can now be explained.

4.1 Network Setup

Next is a general example of how the test setup will be, including the hardware and software.

Figure 4.1: General Test Network Setup



The flow of the script dynamics can be seen in Figure 4.1. The script connects to the SmartBits Packet Generator from the Sun Workstation, communicate it and starts the packet flow from the SmartBits to the Switch 4007. At the same time, it communicates with the 3Com

Switch 4007 through a telnet connection, and monitor different information from it such as packet loss rate (if any), speed of transmission, if all packets are being received, etc.

4.2 Test Methodology

The following sections introduce how the tests will work, and what limits and boundaries they have for both normal and abnormal behavior for the hardware being tested.

4.2.1 Throughput Test Methodology

The Throughput test finds the fastest rate at which a device can forward packets without dropping any. If a single packet is dropped the test fails and the test is repeated at a lower throughput rate.

The throughput test satisfies the terminology criteria of RFC 1242 and the test methodology specified in RFC 1944. From RFC 1242, throughput is "the maximum rate at which none of the offered packets are dropped by the device." From RFC 1944, the methodology to measure throughput is to "Send a specific number of packets at a specific rate through the Device Under Test (DUT) and then count the packets that are transmitted by the DUT."

If the initial rate is 100%, the first throughput test trial packet rate for a given packet length is the maximum rate at for the topology and speed of the transmitting SmartCard. If the receiving port receives all packets from the transmitting SmartBits port, no further trials are attempted and the maximum packet rate is recorded as the throughput.

If the first trial fails (if even a single packet is lost), the second trial packet rate drops to 20% lower than the rate at which it failed. The third and each subsequent trail use a binary search to determine a rate that is halfway between the failed rate and the last successful rate. The does not finish until the packet loss percent is less than or equal to the resolution value in the test setup.

Results of the test are logged to a file showing the maximum data rate by port and the packet data rate. The percentage of the maximum possible packet data rate is also recorded.

4.2.2 Packet Loss Rate Test Methodology

This test measures the percentage of packets lost by the Device Under Test (DUT) that should have been forwarded.

The Packet Loss Rate test satisfies the terminology criteria of RFC 1242 and the test methodology specified in RFC 1944. From RFC 1242, Packet Loss Rate is the "Percentage of packets that should have been forwarded by a network device under steady state (constant) load that were not forwarded due to a lack of resources." The test allows for varying packet sizes from 64 bytes to 1518 bytes. From RFC 1944, the methodology to measure packet loss is to "Send a specific number of packets at a specific rate through the DUT to be tested and count the packets that are transmitted by the DUT.

Packets are validated by counting only packets generated by the sending port, not any packets sent by the switch. Keep-alive and routing update packets are not counted as received packets.

The Packet Loss Rate test operates by a packet burst being sent out at the maximum possible rate for a user-specified period of time. After all packets are sent, the receiving port is queried to determine how many packets were received. The number of packets not received is determined and the percentage of loss is calculated. The results for this test are recorded in the same manner as the ones for the Throughput test.

5 Execution and Results

The following sections discuss how the task of automation was accomplished, as well as the results and any future enhancements that could be made to the system.

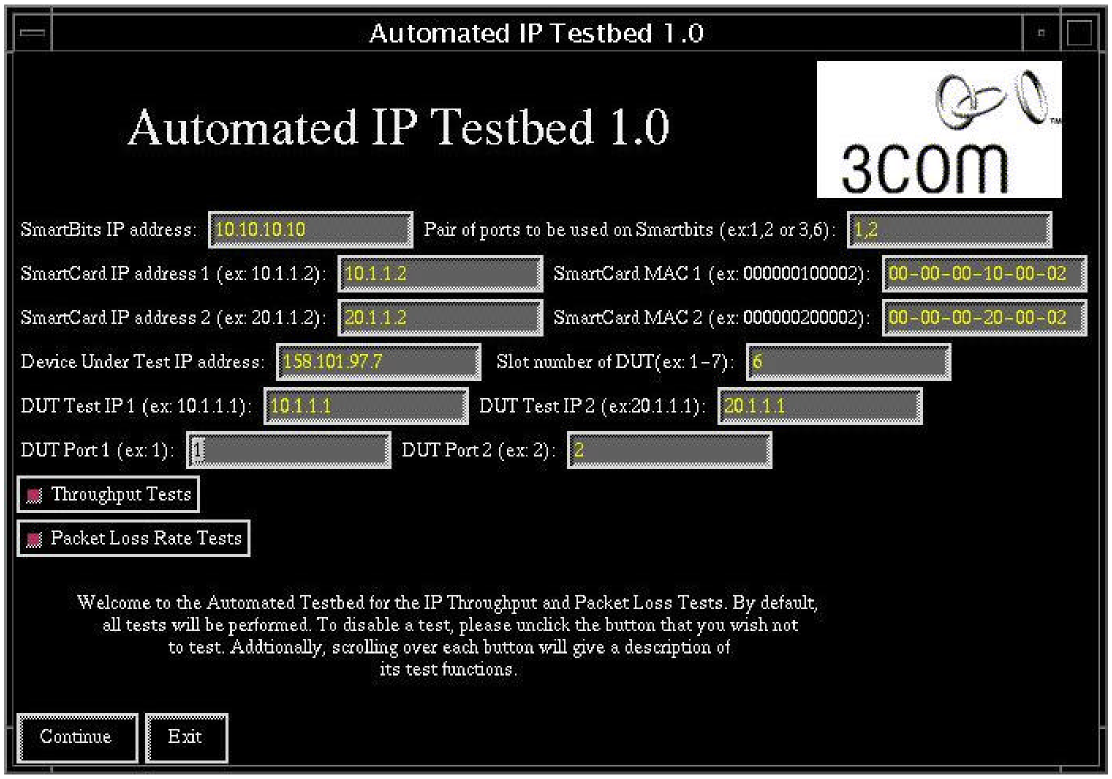
5.1 Interface

The Interface for the automation was made using Tcl and Tk as previously mentioned, with three main scripts run in succession from a shell script. They are explained here in the order that they are run.

5.1.1 Automation Setup Window

The first script run is a setup screen in Tk as pictured in Figure 5.1.

Figure 5.1 Automation Setup Screenshot



Automated IP Testbed 1.0

Automated IP Testbed 1.0

SmartBits IP address: 10.10.10.10
Pair of ports to be used on Smartbits (ex:1,2 or 3,6): 1,2
SmartCard IP address 1 (ex: 10.1.1.2): 10.1.1.2
SmartCard MAC 1 (ex: 000000100002): 00-00-00-10-00-02
SmartCard IP address 2 (ex: 20.1.1.2): 20.1.1.2
SmartCard MAC 2 (ex: 000000200002): 00-00-00-20-00-02
Device Under Test IP address: 159.101.97.7
Slot number of DUT(ex: 1-7): 6
DUT Test IP 1 (ex: 10.1.1.1): 10.1.1.1
DUT Test IP 2 (ex:20.1.1.1): 20.1.1.1
DUT Port 1 (ex: 1): 1
DUT Port 2 (ex: 2): 2

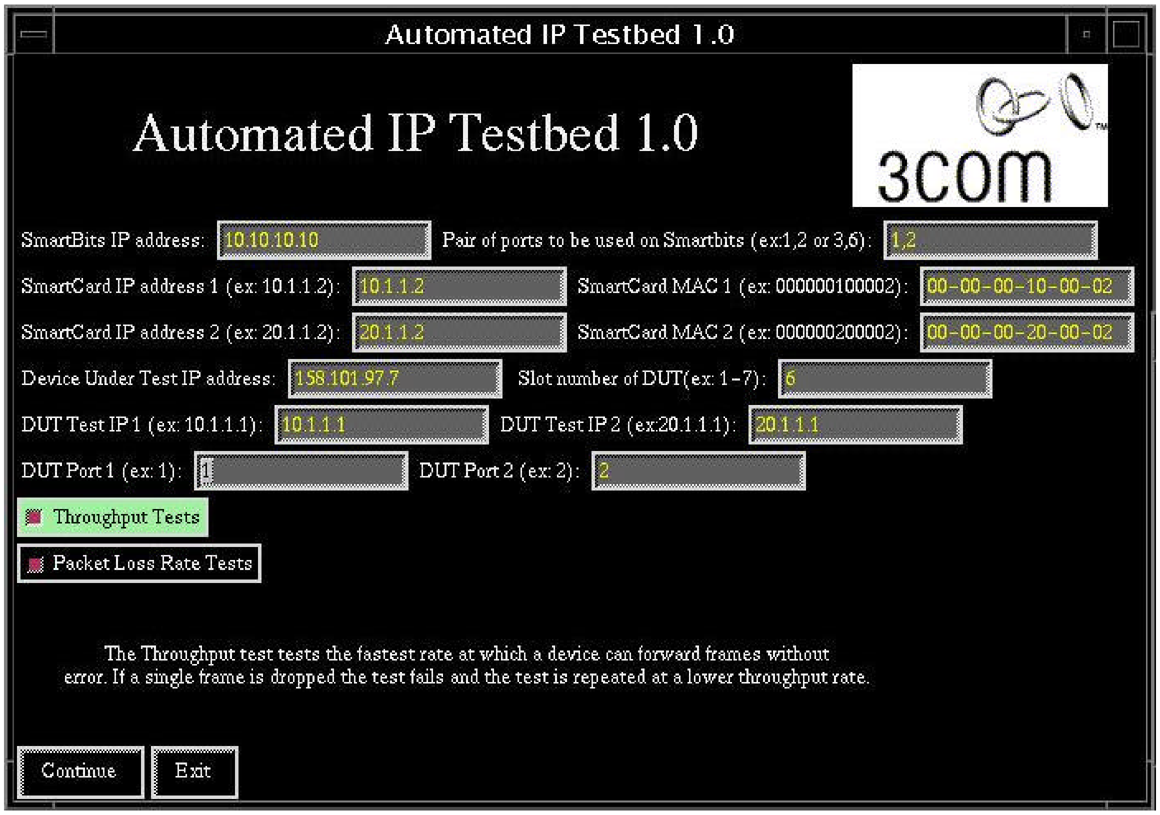
[✓] Throughput Tests
[✓] Packet Loss Rate Tests

Welcome to the Automated Testbed for the IP Throughput and Packet Loss Tests. By default, all tests will be performed. To disable a test, please unclick the button that you wish not to test. Addtionally, scrolling over each button will give a description of its test functions.

Continue Exit

As you can see, the screen is composed of several different parts. First, the title and 3Com logo, to differentiate it from other setup screens that may be used for other programs running on a user's workstation. Next are the text boxes with information such as IP addresses, MAC addresses and other necessary data. The values are loaded from the text files named setup.cfg and DUT_setup.cfg, where the values will be stored again when the user presses the continue button. If the files do not exist, the textboxes default to "????" and when the user presses continue, the files are created with the new information entered. The buttons below the text boxes are the two tests that will be run with information provided from the boxes, and can be selected or deselected depending upon which ones are desired to be performed (they are defaulted to be run). The status of what tests will be run are also saved in the files.

Figure 5.2 Example of scroll-over for test buttons



Automated IP Testbed 1.0

Automated IP Testbed 1.0

SmartBits IP address: 10.10.10.10 Pair of ports to be used on Smartbits (ex:1,2 or 3,6): 1,2

SmartCard IP address 1 (ex: 10.1.1.2): 10.1.1.2 SmartCard MAC 1 (ex: 000000100002): 00-00-00-10-00-02

SmartCard IP address 2 (ex: 20.1.1.2): 20.1.1.2 SmartCard MAC 2 (ex: 000000200002): 00-00-00-20-00-02

Device Under Test IP address: 158.101.97.7 Slot number of DUT(ex: 1-7): 6

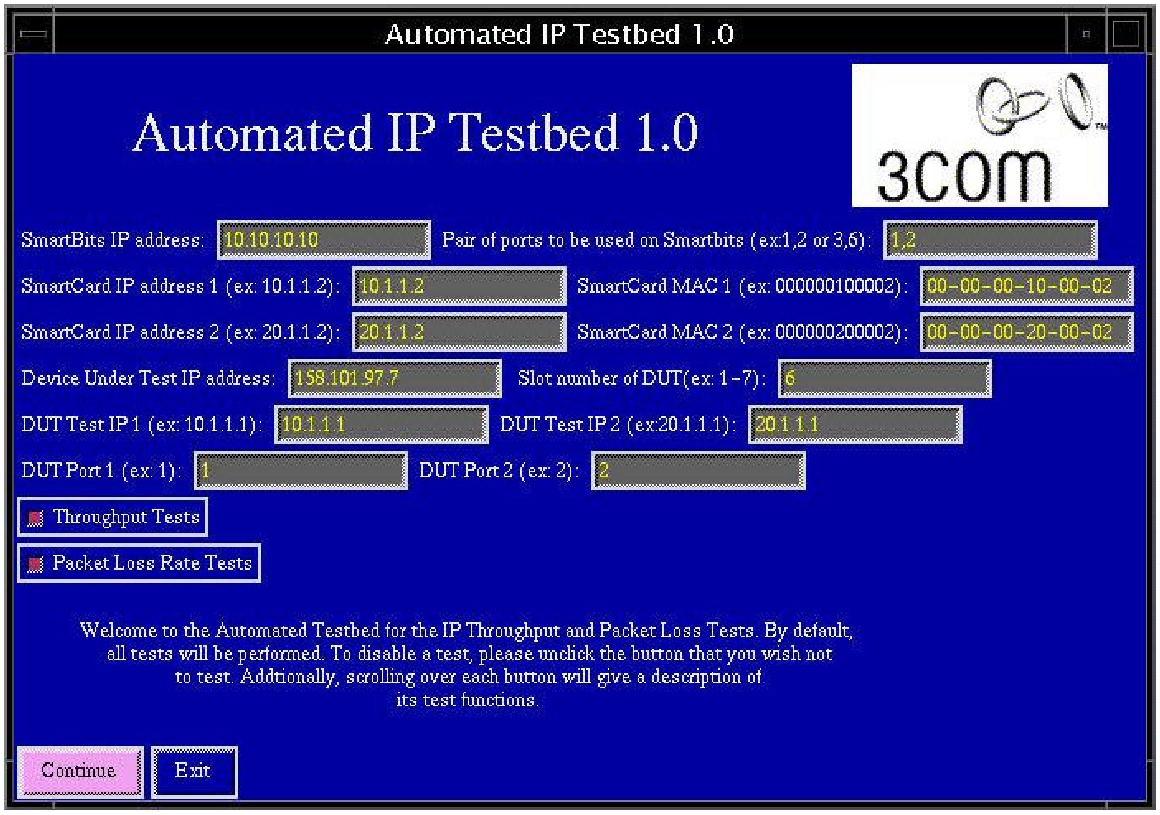
DUT Test IP 1 (ex: 10.1.1.1): 10.1.1.1 DUT Test IP 2 (ex:20.1.1.1): 20.1.1.1

DUT Port 1 (ex: 1): 1 DUT Port 2 (ex: 2): 2

The Throughput test tests the fastest rate at which a device can forward frames without error. If a single frame is dropped the test fails and the test is repeated at a lower throughput rate.

To better help with the understanding of what each of the tests do, scrolling over the buttons will give an explanation of the tests that will be performed, as shown in Figure 5.2. The explanation for the test replaces the regular greeting and general explanation for the setup window. When the mouse moves off of the button, the default explanation returns. There is also the added feature of having different color schemes randomly load each time the automation in run, as Figure 5.3 shows an alternate scheme.

Figure 5.3 Example of Alternate Color Scheme



Automated IP Testbed 1.0

Automated IP Testbed 1.0

3com

SmartBits IP address: 10.10.10.10
Pair of ports to be used on Smartbits (ex:1,2 or 3,6): 1,2
SmartCard IP address 1 (ex: 10.1.1.2): 10.1.1.2
SmartCard MAC 1 (ex: 000000100002): 00-00-00-10-00-02
SmartCard IP address 2 (ex: 20.1.1.2): 20.1.1.2
SmartCard MAC 2 (ex: 000000200002): 00-00-00-20-00-02
Device Under Test IP address: 158.101.97.7
Slot number of DUT(ex: 1-7): 6
DUT Test IP 1 (ex: 10.1.1.1): 10.1.1.1
DUT Test IP 2 (ex:20.1.1.1): 20.1.1.1
DUT Port 1 (ex: 1): 1
DUT Port 2 (ex: 2): 2

Throughput Tests

Packet Loss Rate Tests

Welcome to the Automated Testbed for the IP Throughput and Packet Loss Tests. By default, all tests will be performed. To disable a test, please unclick the button that you wish not to test. Addtionally, scrolling over each button will give a description of its test functions.

This is done for only four different color schemes, but breaks up the monotony of the window if the tester has to run it multiple times in succession.

5.1.2 Setup Information Files

As was mentioned in the last section, two files, DUT_setup.cfg and setup.cfg are where the information from the setup window is stored. There are two files for a reason, as one has the information pertaining to the Device Under Test (DUT_setup.cfg), while the other has the information pertaining to the packet generator, and more specifically SmartBits. DUT_setup.cfg has the following parameters, arranged exactly as they appear in the file:

```
EME_IP='158.101.97.7'             ; export EME_IP
SLOT_NUMBER='6'                   ; export SLOT_NUMBER
DUT_IP1='10.1.1.1'                ; export DUT_IP1
DUT_IP2='20.1.1.1'                ; export DUT_IP2
PORT1='1'                         ; export PORT1
PORT2='2'                         ; export PORT2
IP_ADDRESS1='10.1.1.2'            ; export IP_ADDRESS1
IP_ADDRESS2='20.1.1.2'            ; export IP_ADDRESS2
MAC1='00-00-00-10-00-02'          ; export MAC1
MAC2='00-00-00-20-00-02'          ; export MAC2
```

The different headings are what the script files match on to find the data it wants in the file (ex: EME_IP), and the data it will take from the file is inside the quotes (ex: 158.101.97.7). The ; export followed by the variable is to allow the data to be used by the shell script setting off the Tk and Tcl scripts. This data can then be used in the headings of the windows or in the dialog of the script before it begins. See Figure 5.4's heading; the IP address is taken from the setup.cfg's EME_IP variable.

The file setup.cfg is done in the same manner as DUT_setup.cfg:

```
TEST_CASES='0 1'                  ; export TEST_CASES
SMARTBITS_IP='10.10.10.10'        ; export SMARTBITS_IP
NUM_PORTS='1,2'                   ; export NUM_PORTS
IP_ADDRESS1='10.1.1.2'            ; export IP_ADDRESS1
IP_ADDRESS2='20.1.1.2'            ; export IP_ADDRESS2
MAC1='00-00-00-10-00-02'          ; export MAC1
MAC2='00-00-00-20-00-02'          ; export MAC2
```

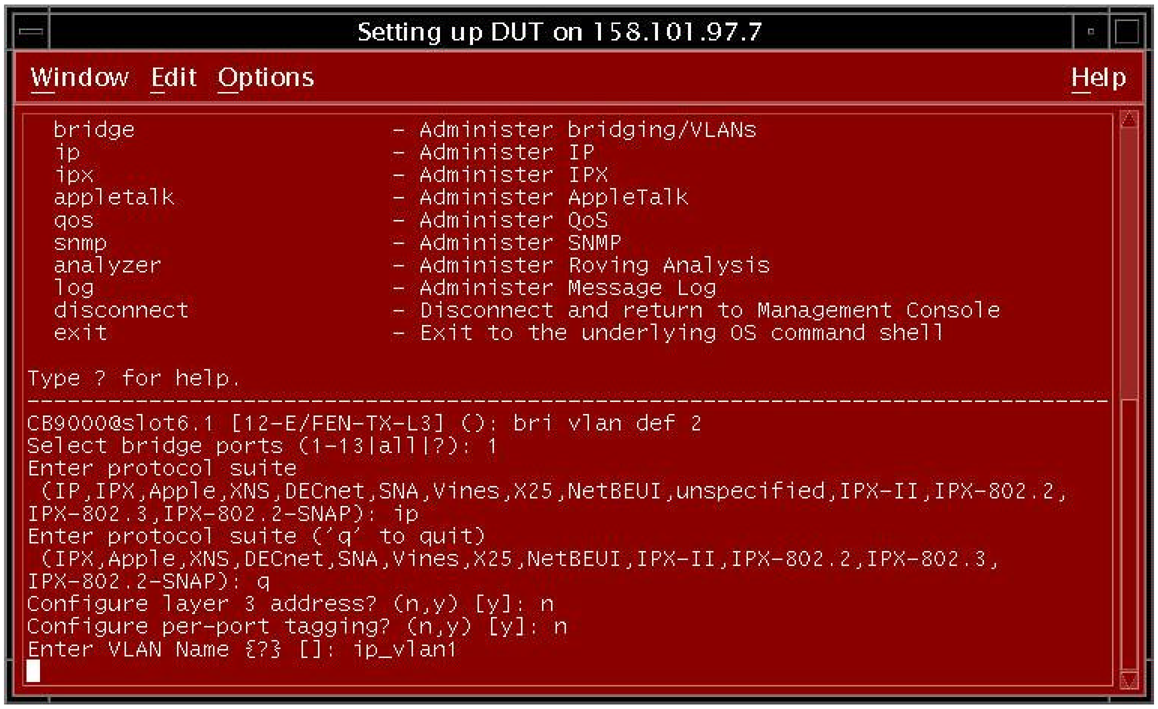
These variables and their corresponding data related to the setting up and running of the SmartBits Automation.

Again, note that these files are needed to run the automation portion of this suite that was created, but the setup window in not required to create them. They can be modified manually and the automation will run with the new information. However, the setup of the text files must be created in exactly the same format.

5.1.3 Device Under Test Automation

After the user has pressed continue on the Tk window, the new data will be written to the files (if there is new data) and the Actual automation in Tcl will begin. The first automation script that runs after the setup screen uses the information stored in the DUT_setup.cfg file to configure the device for the IP test. This means not only setting up the network properly, with IP addresses and other networking protocols in place, but also turning off certain routing protocols that send out advertisement packets (to find other networks they are looking for in hopes one will respond), such as IGMP querying, DVMRP, and Spanning Tree. This is so the statistics of the test will be accurate, as when the test is performed, the amount of packets sent and received would be affected by having other routing packets being sent out along the wires then test in being performed on. Also, the back plane port, which connects the module to the other modules in the chassis, needed to be turned off, as the packets from the other routers can come in through that port on out through the wire being used, again affecting the statistics.

Figure 5.4: Device Under Test Automation window



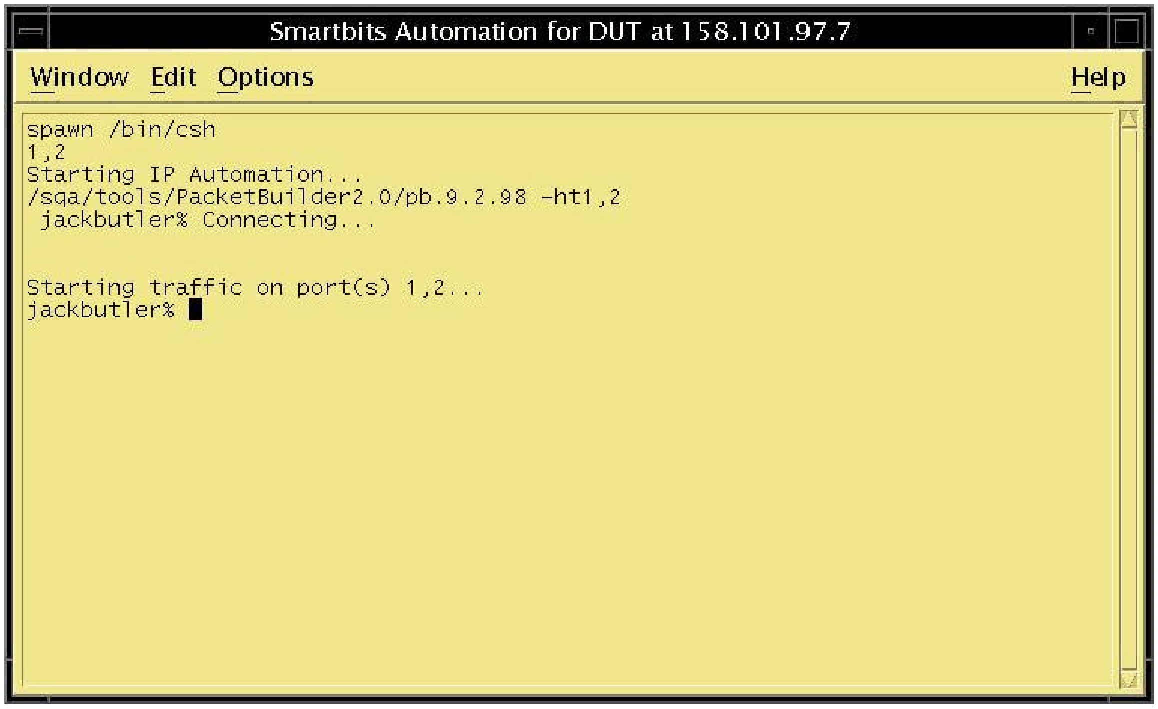
```
  bridge                      - Administer bridging/VLANs
  ip                          - Administer IP
  ipx                         - Administer IPX
  appletalk                   - Administer AppleTalk
  qos                         - Administer QoS
  snmp                        - Administer SNMP
  analyzer                    - Administer Roving Analysis
  log                         - Administer Message Log
  disconnect                  - Disconnect and return to Management Console
  exit                        - Exit to the underlying OS command shell

Type ? for help.
--------------------------------------------------------------------------------
CB9000@slot6.1 [12-E/FEN-TX-L3] (): bri vlan def 2
Select bridge ports (1-13|all|?): 1
Enter protocol suite
 (IP,IPX,Apple,XNS,DECnet,SNA,Vines,X25,NetBEUI,unspecified,IPX-II,IPX-802.2,
IPX-802.3,IPX-802.2-SNAP): ip
Enter protocol suite ('q' to quit)
 (IPX,Apple,XNS,DECnet,SNA,Vines,X25,NetBEUI,IPX-II,IPX-802.2,IPX-802.3,
IPX-802.2-SNAP): q
Configure layer 3 address? (n,y) [y]: n
Configure per-port tagging? (n,y) [y]: n
Enter VLAN Name {?} []: ip_vlan1
```

Figure 5.4 shows the DUT setup in progress. The script uses a telnet connection to log into the chassis by using the IP address of the DUT found in DUT_setup.cfg. It then uses the file again to locate which module to connect to, which it does next. After that, it uses the rest of the information found in the file to setup the DUT as mentioned above, communicating with the Command Line Interface (CLI) of the module and using specific commands in conjunction with the data in the file to setup up the hardware. With that completed, the script logs out of the device, and ends the telnet session.

5.1.4 Packet Generator Automation

The final script in the automation is the Packet generator automation script, which uses a program called PacketBuilder. Figure 5.5 shows the Automation Window with PacketBuilder at work. This program communicates with the hardware and has it create the packets based on the information provided in setup.cfg.

Figure 5.5: SmartBits Automation



```
spawn /bin/csh
1,2
Starting IP Automation...
/sqa/tools/PacketBuilder2.0/pb.9.2.98 -ht1,2
 jackbutler% Connecting...


Starting traffic on port(s) 1,2...
jackbutler% 
```

This information, specifically the MAC addresses and the IP addresses, are then used to put into the packet that is sent out. For both the Throughput and Packet Loss tests each packet is a fixed size. First, 64 byte packets are sent out for 30 seconds at 100% utilization at 100 Megabits per second. If this passes, 128 byte packets are used at the same speed for the same amount of time. If that passes as well, 256 byte packets will be used in the same manner.

Now, for the throughput test, if fewer packets are received than are sent by the SmartBits, the test is repeated at a 20% lower rate. So, for example, if the 100% utilization fails for the 64 byte packets, the test is repeated at 80%. The inter-packet gap, basically the time between the end of one packets transmission and the beginning of another, must be modified for this change in rate to take place. Generally, the more time between packets, the lower the utilization. Appendix 1 shows the calculations required to convert the proper times for 100 megabits per second at 100% speed to 80% speed, and so on. For the packet sizes used in this test, the following numbers were found:

Size of packet in bytes	**Bandwidth Utilization in percentage**	**Inter-packet Gap in millsecs**
64 bytes	100%	.0960
	80%	.272
	60%	.565
128 bytes	100%	.0960
	80%	.4
	60%	.906
256bytes	100%	.096
	80%	.656
	60%	1.589

The Packet Loss test simply sends each of the three packet sizes at 100% utilization and measures the loss.

Both of the tests statistics for packets sent and received, as well as a general pass/fail for each packet size are recorded. The result file's name is IPthroughxxyyzz.txt, and

Ippacketxxyyzz.txt where xx is the day, yy the month and zz the last two digits of the year, and 'through' and 'packet' represent the two different tests.

5.2 Evaluation

The automated Throughput and Packet Loss tests, when compared to the non-automated ones, work in exactly the same way, but at this point, on a much more limited basis. In the non-automated tests, the packet sizes goes all the way up from 64 to 1518 bytes, while, because of a bug in the console connection driver for the SmartBits Tcl libraries (which PacketBuilder uses), the packets can only be a maximum of 500 bytes in the automated environment. This is less than 512 so, that is why only sizes 64, 128 and 256 bytes were used. As far as the Throughput tests' readjusting of percent utilization to figure out the exact maximum throughput, the non-automated test is much more accurate, as there was not enough time to implement the binary search algorithm as described in section 4.2.1. The Throughput Test as it currently is automated will only lower the percent utilization by 20 percent every time. Besides the limitations, the automated test works as it should within the parameters set forth.

6 Conclusions

Automation is a useful tool for network hardware testing, as it allows for the Test Engineer to have tedious and time consuming tests done for them, and allows for increased productivity as they can then do other tests while the automation is running. The 3com IP Automation Testbed was designed to accomplish the goals of automating the Internet Protocol tests for the SmartBits Packet Generator, and was made to help with the tedious way that the Throughput and Packet Loss tests are used. Automation Engineers will also benefit from a setup screen specific to their automation environment. An added benefit would be to have a suite that could be integrated to the test environment.

The automation test environment at this point accomplishes all of those goals. It has a setup screen that allows for configuration specific to a tester's environment, saving the information to files that will be used to run the automation. In addition, the automation only needs those files, and can be integrated into an environment that runs without setup screens (like the ATS system at 3Com). Lastly, it accurately recreates the tests (with some slight limitations), while offering a way to track the results of the test in the files the script produces.

7 Future Work

There are several areas that could be improved upon in the current working version of the automated Internet Protocol Testbed.

First, the Setup screen might be able to be organized better. It would be better if the information that is stored in the two files is separated on the screen so that the user understands better where the information is going. Also, a help button could be useful to determine what each of the different IP and MAC addresses are going to be used for specifically in the automation.

Second, the connection to the SmartBits is currently a console connection because that only way Packet Builder will allow for. A better way to do this would be to write an interface using the SmartBits Tcl libraries that allowed for a telnet connection. This would allow for much more flexibility in terms of where the automation could be performed, because a telnet connection could conceivably allow anyone on the 3Com network to get to the SmartBits if properly connected.

Third, the automation in Tcl could also be accompanied by a status window in Tk that could tell the user generally what is going on. The current system automates and only shows the user what is actually going on, which is very fast and very hard to follow. By giving a user a window to follow, it will allow for assurance to the user that everything is connecting right and working as it should.

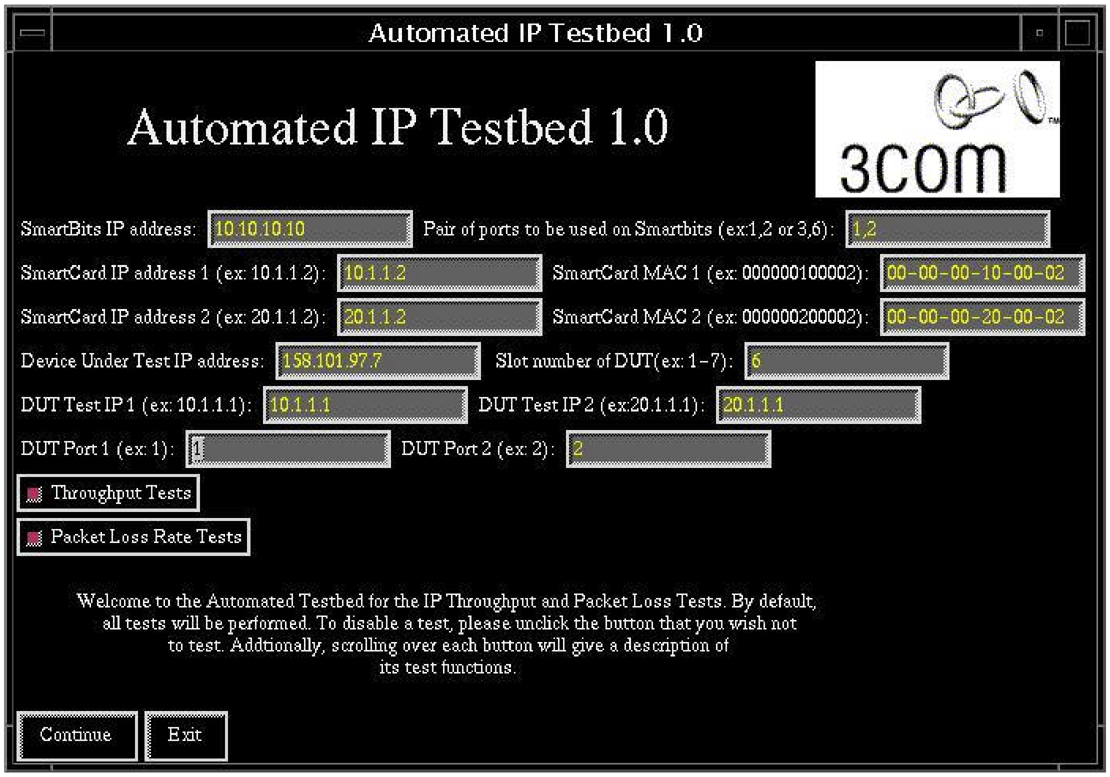
Finally, fixing the current limitations in the automated script, including the SmartBits driver (which only allows a maximum packet size of 500 bytes), and the binary search algorithm for the better approximation of the maximum throughput, would make the Testbed the exact same in testing capability as the stand-alone non-automated tests.

8 Examination of the User Interface

Software Quality Assurance Engineers are familiar with the nature of their environments, and the design of the setup window in Tk is designed specifically for that group of people. However, it is interesting to step back from this perspective for a moment, and examine how other groups of people would perceive this window, and what modifications could be made to improve the usability of the environment for them.

8.1 Current Configuration

Figure 1 shows an example of the current setup window.



Automated IP Testbed 1.0

SmartBits IP address: 10.10.10.10	Pair of ports to be used on Smartbits (ex:1,2 or 3,6): 1,2
SmartCard IP address 1 (ex: 10.1.1.2): 10.1.1.2	SmartCard MAC 1 (ex: 000000100002): 00-00-00-10-00-02
SmartCard IP address 2 (ex: 20.1.1.2): 20.1.1.2	SmartCard MAC 2 (ex: 000000200002): 00-00-00-20-00-02
Device Under Test IP address: 158.101.97.7	Slot number of DUT(ex: 1-7): 6
DUT Test IP 1 (ex: 10.1.1.1): 10.1.1.1	DUT Test IP 2 (ex:20.1.1.1): 20.1.1.1
DUT Port 1 (ex: 1): 1	DUT Port 2 (ex: 2): 2

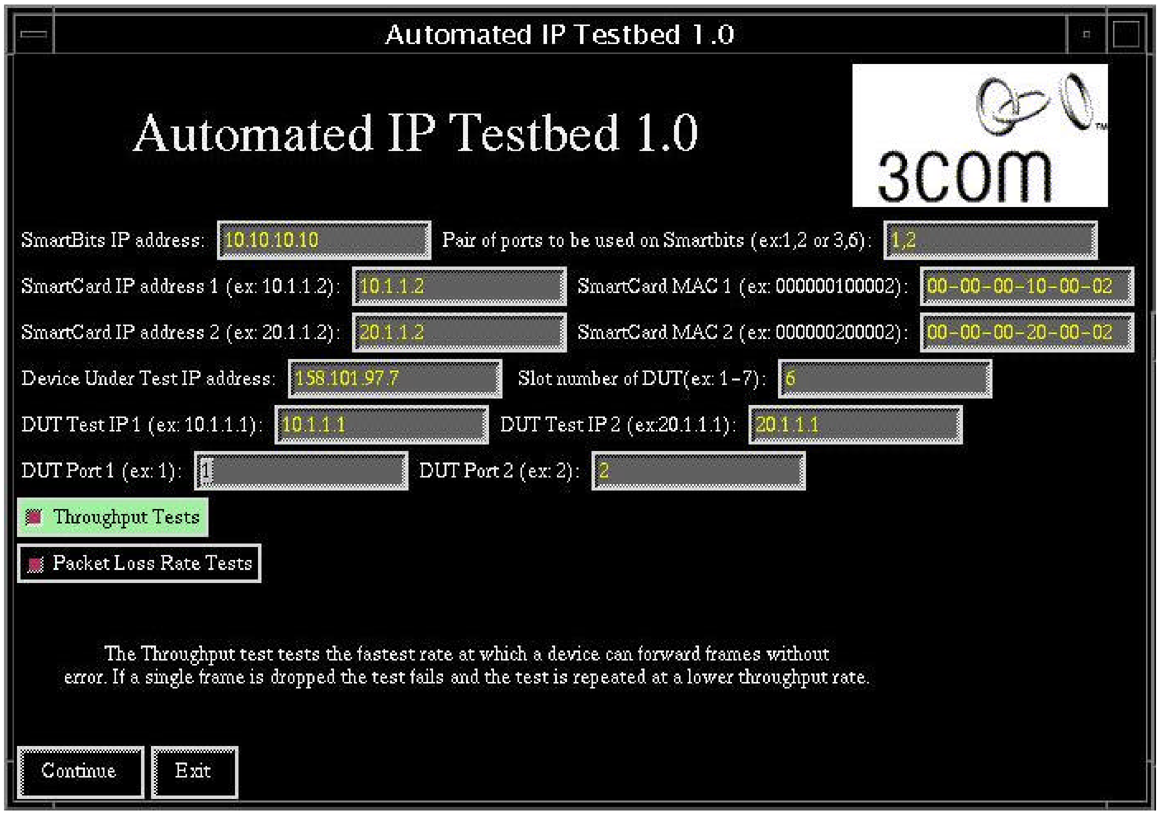
▨ Throughput Tests

▨ Packet Loss Rate Tests

Welcome to the Automated Testbed for the IP Throughput and Packet Loss Tests. By default, all tests will be performed. To disable a test, please unclick the button that you wish not to test. Addtionally, scrolling over each button will give a description of its test functions.

Figure 1: Example of current setup window

The window is comprised of five major sections as shown in the Figure. The first section is the title and version of the test and logo of the company, which is standard for the companies' in-house programs. Next, the different variables and their explanations follow, with the ability for the user to modify them to suit their environments needs. The third part of the screen is the buttons for the particular tests, where clicking them toggles the tests to be performed or skipped. Fourth is the explanation area, where there is a default explanation describing the overall test and other areas of the window, as well as a description of each test when the mouse scrolls over the button for that test. Figure 2 shows both the highlighted button that the mouse is over, as well as the explanation of the test for that button in the space of the default explanation.


Automated IP Testbed 1.0

Automated IP Testbed 1.0

SmartBits IP address: 10.10.10.10

Pair of ports to be used on Smartbits (ex:1,2 or 3,6): 1,2

SmartCard IP address 1 (ex: 10.1.1.2): 10.1.1.2

SmartCard MAC 1 (ex: 000000100002): 00-00-00-10-00-02

SmartCard IP address 2 (ex: 20.1.1.2): 20.1.1.2

SmartCard MAC 2 (ex: 000000200002): 00-00-00-20-00-02

Device Under Test IP address: 158.101.97.7

Slot number of DUT(ex: 1-7): 6

DUT Test IP 1 (ex: 10.1.1.1): 10.1.1.1

DUT Test IP 2 (ex:20.1.1.1): 20.1.1.1

DUT Port 1 (ex: 1): 1

DUT Port 2 (ex: 2): 2

Throughput Tests

Packet Loss Rate Tests

The Throughput test tests the fastest rate at which a device can forward frames without error. If a single frame is dropped the test fails and the test is repeated at a lower throughput rate.

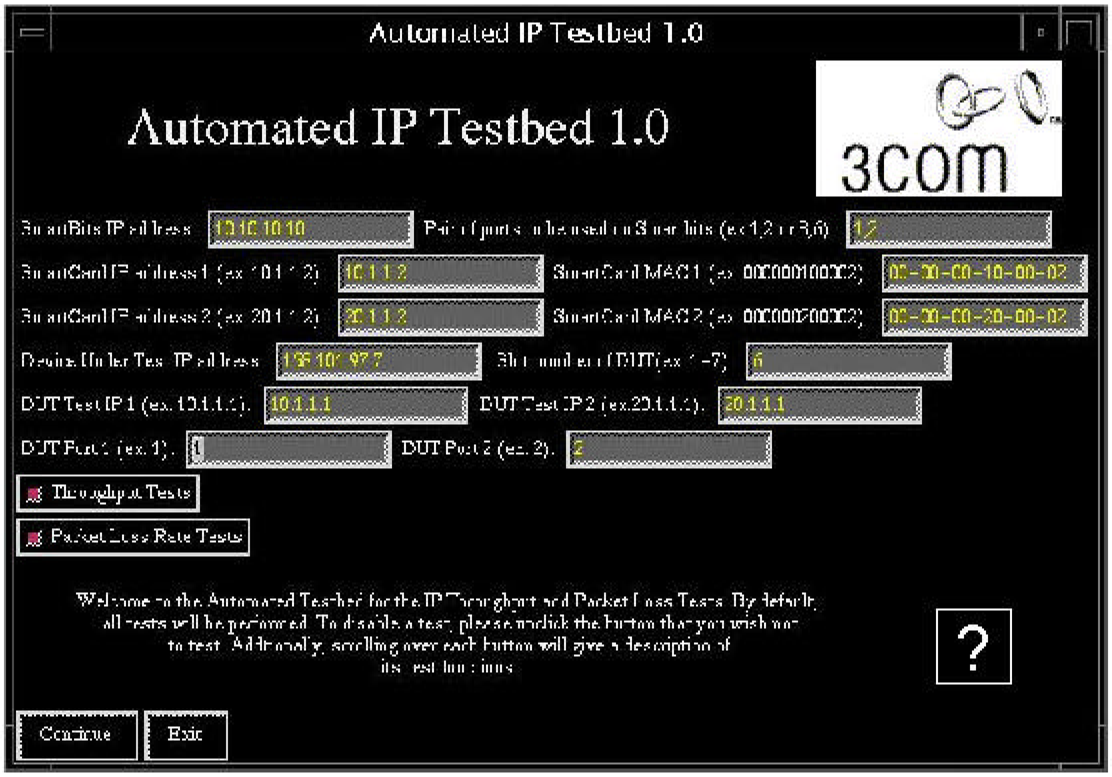
Figure 2: Example of scroll-over buttons and explanation

The final part of the window is the Continue and Exit buttons, leading you out of the window depending upon which button you wish to press.

This current setup is a compromise between experienced users of UNIX and the protocol being tested and those who do not know much about either. The advanced users can deselect certain tests they don't want to do, while inexperienced users can scroll over the buttons to view the explanations for the tests they represent. For the information boxes above the buttons, examples are given for inexperienced users to follow and emulate, but there is still no place where it explains how each variable will be used in the automation.

8.2 Enhancements of features for inexperienced users

To change this window so it would be more accessible to inexperienced users of the protocol, there should be a help button on the lower left-hand corner of the screen that will lead to explanations of the different variables in the test, and how each of the will be used. Figure 3 shows an example of the Help button, as represented by a large Question Mark in a box.



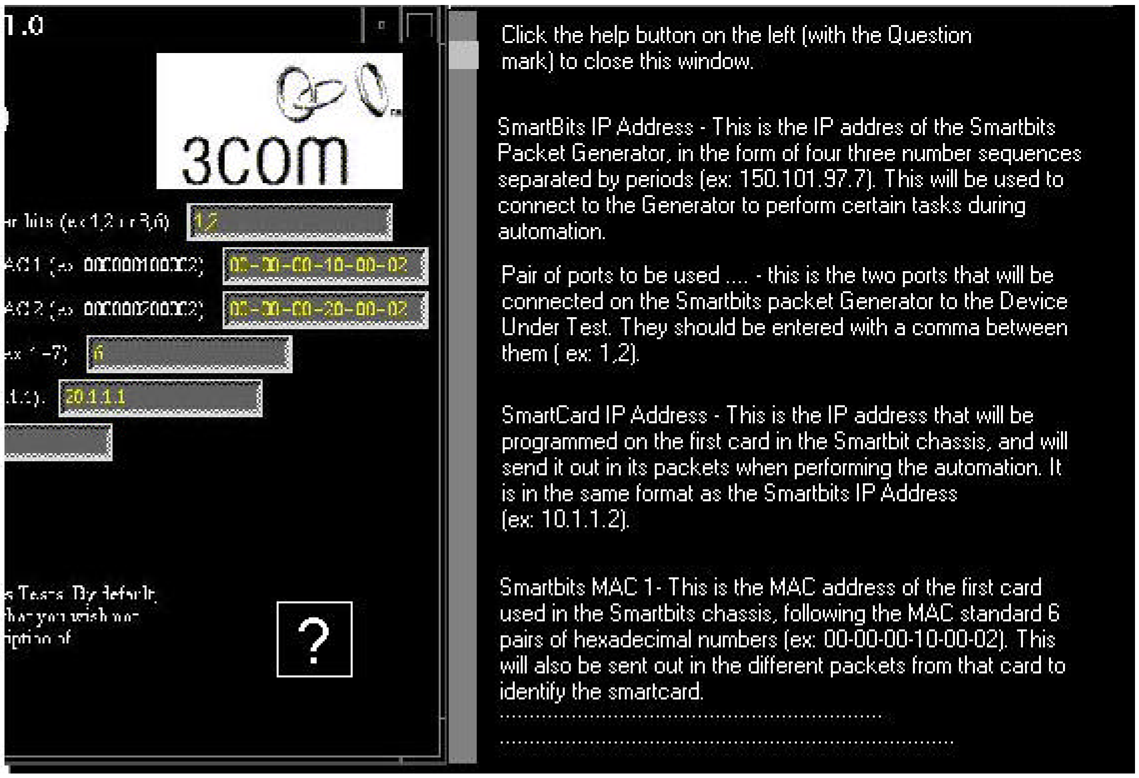
Automated IP Testbed 1.0

SmartBits IP address: 10.10.10.10
Pair of ports to be used on SmartBits (ex 1,2 or 3,6): 1,2
SmartCard IP address 1 (ex 10.1.1.2): 10.1.1.2
SmartCard MAC 1 (ex 000000100002): 00-00-00-10-00-02
SmartCard IP address 2 (ex 20.1.1.2): 20.1.1.2
SmartCard MAC 2 (ex 000000200002): 00-00-00-20-00-02
Device Under Test IP address: 198.100.97.7
Slot number of DUT (ex 1-7): 6
DUT Test IP 1 (ex. 10.1.1.1): 10.1.1.1
DUT Test IP 2 (ex. 20.1.1.1): 20.1.1.1
DUT Port 1 (ex. 1): 1
DUT Port 2 (ex. 2): 2
Throughput Tests
Packet Loss Rate Tests

Welcome to the Automated Testbed for the IP Throughput and Packet Loss Tests. By default, all tests will be performed. To disable a test, please unclick the button that you wish not to test. Additionally, scrolling over each button will give a description of its test functions.

Figure 4: Example of help button

This button would pop up another window, which would go through each of the different text boxes, explaining what each piece of information would be used for. So, for example the “SmartBits IP Address” would be described by “ The SmartBits IP Address is the IP address of the SmartBits Packet Generator. This will be used to connect to that hardware device so that commands can be given to it to execute functions for the automation.” This window would appear from the right hand side of the setup window, as pictured in Figure 5.



Click the help button on the left (with the Question mark) to close this window.

SmartBits IP Address - This is the IP addres of the Smartbits Packet Generator, in the form of four three number sequences separated by periods (ex: 150.101.97.7). This will be used to connect to the Generator to perform certain tasks during automation.

Pair of ports to be used .... - this is the two ports that will be connected on the Smartbits packet Generator to the Device Under Test. They should be entered with a comma between them (ex: 1,2).

SmartCard IP Address - This is the IP address that will be programmed on the first card in the Smartbit chassis, and will send it out in its packets when performing the automation. It is in the same format as the Smartbits IP Address (ex: 10.1.1.2).

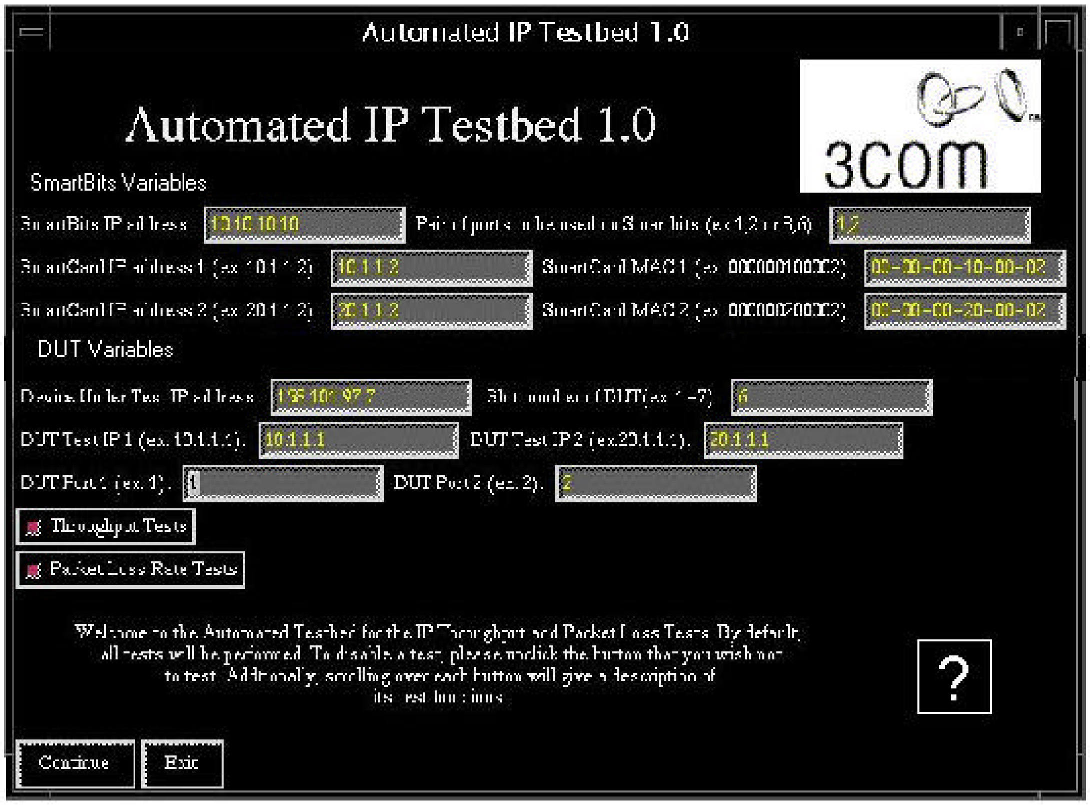
Smartbits MAC 1- This is the MAC address of the first card used in the Smartbits chassis, following the MAC standard 6 pairs of hexadecimal numbers (ex: 00-00-00-10-00-02). This will also be sent out in the different packets from that card to identify the smartcard.

....

....

Figure 5: Example of the pop-up help scroll window

The user simply needs to click on the help button they had pressed before to close this window. To clarify what groups the information on the screen belonged to, it would help to split them up. Since the SmartBits information and the DUT information are already grouped together on the screen anyway, it would be easy to pull them further apart. Figure 6 shows how this would look.



Automated IP Testbed 1.0

Automated IP Testbed 1.0

SmartBits Variables

SmartBits IP address: 10.10.10.10 — Pair of ports to be used on SmartBits (ex 1,2 or 3,6): 1,2

SmartCard IP address 1 (ex 10.1.1.2): 10.1.1.2 — SmartCard MAC 1 (ex 000000100002): 00-00-00-10-00-02

SmartCard IP address 2 (ex 20.1.1.2): 20.1.1.2 — SmartCard MAC 2 (ex 000000200002): 00-00-00-20-00-02

DUT Variables

Device Under Test IP address: 156.100.97.7 — Slot number of DUT (ex 1-7): 6

DUT Test IP 1 (ex. 10.1.1.1): 10.1.1.1 — DUT Test IP 2 (ex.20.1.1.1): 20.1.1.1

DUT Port 1 (ex. 1): 1 — DUT Port 2 (ex. 2): 2

Throughput Tests

Packet Loss Rate Tests

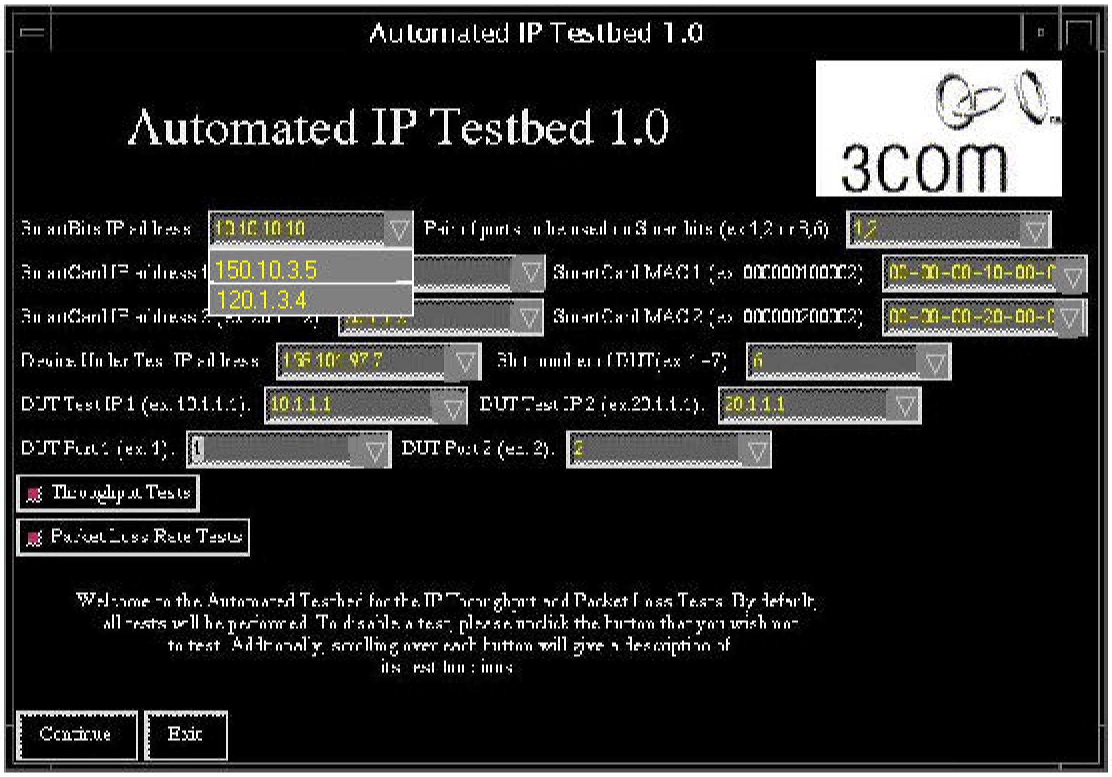
Welcome to the Automated Testbed for the IP Throughput and Packet Loss Tests. By default, all tests will be performed. To disable a test, please unclick the button that you wish not to test. Additionally, scrolling over each button will give a description of its test functions.

Figure 6: Example of separation of variables

These changes have now made the window much easier for people who do not know much about the protocol to use it successfully. The variables are separated into two distinct groups, there is a help section for the explanation of each of the variables and what they will be used for, and the test buttons already had a scroll-over feature to tell what the each of the tests they represented would do. Most important of all, the window is organized in such a way that each of the different parts are clearly defined, so the user will not feel overwhelmed by complexity.

8.4 Enhancements of features for experienced users

The experienced user is one that will understand the nature of the protocols, as well as the nature of the window itself (from a UNIX functionality standpoint). Therefore, complexity is not an issue, and there can be more movement and placement of objects in the window. One of the first things that many experienced users would probably like is a pull down menu for each of the variables, giving the entered data for the past couple of times used (five is a good). so if they were running two test configurations over and over, they would not have to type all the variables in differently every time, they would simply select the variables from the list. Figure 7 shows how this would appear.


Automated IP Testbed 1.0

Automated IP Testbed 1.0

SmartBits IP address 10.10.10.10 | Pair of ports to be used on SmartBits (ex 1,2 or 3,6) 1,2

SmartCard IP address 1 150.10.3.5 / 120.1.3.4 | SmartCard MAC 1 (ex 000000100002) 00-00-00-10-00-0

SmartCard IP address 2 | SmartCard MAC 2 (ex 000000200002) 00-00-00-20-00-0

Device Under Test IP address 198.101.97.7 | Slot number of DUT (ex 1-7) 6

DUT Test IP 1 (ex. 10.1.1.1). 10.1.1.1 | DUT Test IP 2 (ex.20.1.1.1). 20.1.1.1

DUT Port 1 (ex. 1). 1 | DUT Port 2 (ex. 2). 2

Throughput Tests

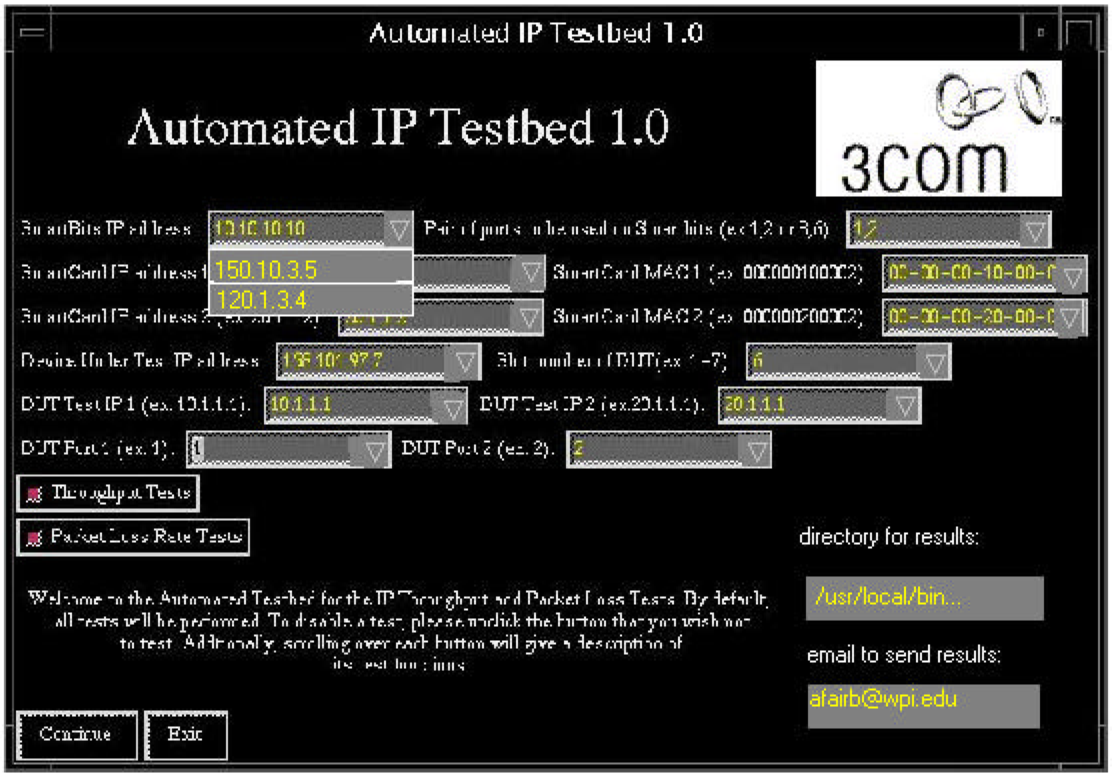
Packet Loss Rate Tests

Welcome to the Automated Testbed for the IP Throughput and Packet Loss Tests. By default, all tests will be performed. To disable a test, please unclick the button that you wish not to test. Additionally, scrolling over each button will give a description of its test functions.

Continue | Exit

Figure 7: Example of pull down menus

This will add some confusion to the screen that inexperienced users would find unnerving, but users familiar with the operating system would find no difficulty in this option, and would probably prefer this for ease of use when doing multiple configurations back to back. Experienced users would also like to have the option of selecting where the results file will go, as well as the option of being e-mailed when the tests are completed. These two features could be added at the bottom right hand side of the screen, as shown in Figure 8.



Automated IP Testbed 1.0

Automated IP Testbed 1.0

SmartBits IP address: 10.10.10.10

150.10.3.5

120.1.3.4

Pair of ports to be used on SmartBits (ex 1,2 or 3,6): 1,2

SmartCard IP address 1:

SmartCard MAC 1 (ex. 0000000100002): 00-00-00-10-00-0

SmartCard IP address 2 (ex. 20.1.1.2):

SmartCard MAC 2 (ex. 0000000200002): 00-00-00-20-00-0

Device Under Test IP address: 136.101.97.7

Slot number of DUT (ex 1-7): 6

DUT Test IP 1 (ex. 10.1.1.1): 10.1.1.1

DUT Test IP 2 (ex. 20.1.1.1): 20.1.1.1

DUT Port 1 (ex. 1): 1

DUT Port 2 (ex. 2): 2

Throughput Tests

Packet Loss Rate Tests

directory for results: /usr/local/bin...

email to send results: afairb@wpi.edu

Welcome to the Automated Testbed for the IP Throughput and Packet Loss Tests. By default, all tests will be performed. To disable a test, please unclick the button that you wish not to test. Additionally, scrolling over each button will give a description of its test functions.

Continue

Exit

Figure 8: Example of added usability for storage and email of results

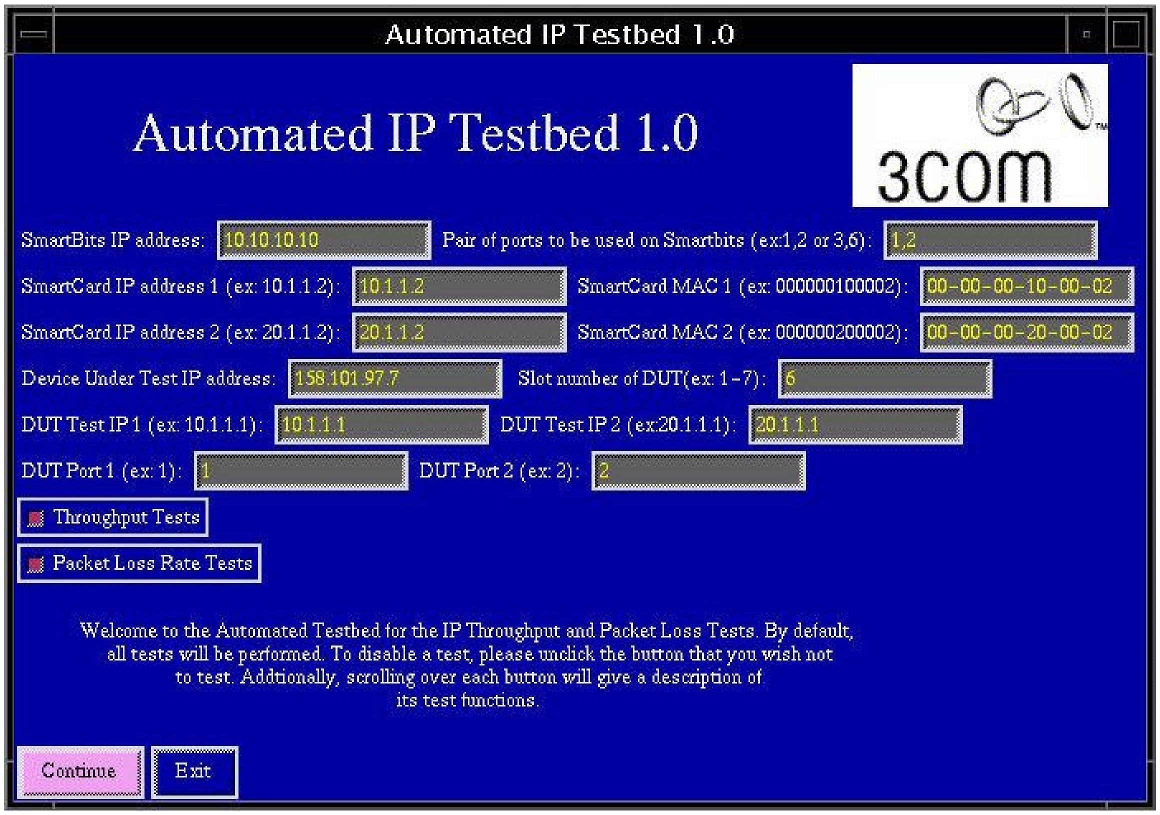
This way, the experienced user gets more control out of the window, and can tailor the way the results are used and stored.

From these two improvements, the experienced user will be much more in control of the elements of the setup window, specifically those tied to the functions of the operating system (UNIX, in this case).

8.5 Color use for different user groups

So far, the features of the window have been the focus of the modifications for both groups, but the color and style of the window may also play a part in how users perceive it's ease of use (or lack thereof).

For inexperienced users, the colors of blue and grey would be most suitable for a window, as they are the most calming and seem friendlier to look at. [someone] This is shown in Figure 9.


Automated IP Testbed 1.0

Automated IP Testbed 1.0

SmartBits IP address: 10.10.10.10 Pair of ports to be used on Smartbits (ex:1,2 or 3,6): 1,2

SmartCard IP address 1 (ex: 10.1.1.2): 10.1.1.2 SmartCard MAC 1 (ex: 000000100002): 00-00-00-10-00-02

SmartCard IP address 2 (ex: 20.1.1.2): 20.1.1.2 SmartCard MAC 2 (ex: 000000200002): 00-00-00-20-00-02

Device Under Test IP address: 158.101.97.7 Slot number of DUT(ex: 1-7): 6

DUT Test IP 1 (ex: 10.1.1.1): 10.1.1.1 DUT Test IP 2 (ex:20.1.1.1): 20.1.1.1

DUT Port 1 (ex: 1): 1 DUT Port 2 (ex: 2): 2

Throughput Tests

Packet Loss Rate Tests

Welcome to the Automated Testbed for the IP Throughput and Packet Loss Tests. By default, all tests will be performed. To disable a test, please unclick the button that you wish not to test. Addtionally, scrolling over each button will give a description of its test functions.

Figure 9: Color scheme for inexperienced users

This is not to say that this would not be a good color choice for experienced users as well, but perhaps a color scheme that is more defined in how the information stands out would be better suited, such as the current color scheme of black background with yellow text. Many other facets of the presentation should be examined as well, such as font size, the font itself, and most importantly placement of information.

Creating windows catered to a specific group is a combination of many different factors, some of which may seem trivial, but can make a difference just the same. For inexperience users, the color, font size and placement of help around the different information (which should not be too cluttered) on the screen, will make it a much more pleasant window for them to use. The experienced users will be less concerned with the presentation, as long as the can find what they are looking for with out too much effort, and are more concerned with the options and availability of information for them to control.

Glossary

Automation – the process of creating a system of software that will do tasks with little to no input from the user that would otherwise have involved manipulation by the user to have it work.

Ethernet protocol – This is the Layer 2 protocol that uses MAC addresses, and is usually used in conjunction with the TCP/IP protocol for the Internet.

Fast Ethernet speed: this is the speed that is defined as up to 100 megabits of data per second. It derives its name from the original Ethernet speed, which was up to 10 megabits of data per second.

IP address – an IP address is used for Layer 3 packets, and is setup in the following way: www.xxx.yyy.zzz, where each letter represents a number. They are used as source and destination addresses in the Layer 3 packet to tell the routers where to forward them, and where they came from.

MAC address – a MAC address is used from Layer 2 packets, and is setup in the following way: 00-00-00-00-00-00, where any of the 0’s could be hexadecimal numbers (0 through F). They are used as source and destination addresses in Layer 2 packets to tell the switches where to forward them, and where they originated from.

OSI model – This model is a complete model for describing the different virtual Layers of networking so that any protocol can be adequately described.

Packet Generator- This is a piece of hardware that creates packets based on information entered by the user. It is usually used to test a router or switches capacities in different areas like bandwidth, speed and transmission quality (if it looses or drops packets).

Router- This is a piece of hardware that primarily concerns itself with Layer 3, specifically IP addresses, and forwards Layer 3 Packets based on the IP information.

Switch- This is a piece of hardware that only concerns itself with Layer 2, specifically MAC addresses, and forwards Layer 2 Packets based on the MAC addresses.

TCP/IP model – this is a model that describes different virtual layers of networking. Although it describes TCP/IP and some other models well, it does not fit all protocols as ideally as would be preferred. This model was used as a stepping stone to the OSI model.

TCP/IP protocol – This is the protocol that uses IP packets (with IP addresses in them), and is responsible for most of the Internet Traffic today.

Workstation – this is a piece of hardware that allows for control over different parts of the setup of the network configurations, and is typically where most of the scripting and automation takes place.

References

[3c100] 3Com Corporation (3c1), http://www.3com.com/inside/overview.html, 6/24/2000.

[3c200] 3Com Corporation, (3c2), http://www.com.com/products/dsheets/400583.html, 7/2/2000.

[3c300] 3Com Corporation, (3c3), http://www.com.com/products/dsheets/400583a.html#4, 7/6/2000.

[Dav00] Davidson, Gary. (2000) Software Quality Assurance Engineer. Interview: 3Com Corporation, Marlboro, Ma. Campus, 6/27/2000.

[Fei97] Feit, Sidnie. (1997) TCP/IP.

[Ner00] Nerds 2.0.1, http://www.pbs.org/opb/nerds2.0.1/, 7/5/2000.

[Net00] Netcom Systems, Inc., http://www.netcomsystems.com/solutions/products/data_sht/0799_0010RevD_SMB-2000.htm, 6/29/00.

[Rou00] Rousseau, Ken. (2000) Software Quality Assurance Principal Engineer. Interview: 3Com Corporation, Marlboro, Ma. Campus, 7/5/2000.

[RF191] Bradner, S. (1991) RFC 1242 Benchmarking Terminology for network interconnection devices.

[RF196] Bradner, S. (1996) RFC 1944 Benchmarking Methodology for Networking Interconnect Devices.

[Say00]Sayaf, Ghassan. (2000) Software Quality Assurance Engineer. Interview: 3Com Corporation, Marlboro, Ma. Campus, 6/25/2000.

[Tan96]Tanenbaum, Andrew S (1996). Computer Networks.

[Wha00]WhatIs?.com, http://www.whatis.com/osifig.htm, 6/15/00.

Appendix A: Calculations for Conversion of Percent Utilization to Inter-packet Gap

1. Calculate the time it takes for a packet to be transmitted, including inter-packet gap and preamble:

Given GAP_{min} = 12 Bytes
Preamble = 8 Bytes

Packet Time (usec) = [PacketSize(Bytes) + Preamble][8bits/Byte][Bit Time (usec)]+ GAP [1]

2. Bit Time = 1/Speed(Bits/Sec) [2]

   Therefore:

   Ethernet = 1/(10 E^6) Bits/Sec = 0.10 E^{-6} Sec/Bit
   Fast Ethernet = 1/(100 E^6) Bits/Sec = 0.01 E^{-6} Sec/Bit
   Gigabit Ethernet = 1/(1000 E^6) Bits/Sec = 0.001 E^{-6} Sec/Bit

3. Next, calculate the minimum inter-packet gap for each speed:

   $GAP_{min-eth}$ = [12 Bytes][8 Bits/Byte][0.10 E^{-6} Sec/Bit] = 9.6 usec
   $GAP_{min-fasteth}$ = [12 Bytes][8 Bits/Byte][0.01 E^{-6} Sec/Bit] = 0.96 usec
   $GAP_{min-gig}$ = [12 Bytes][8 Bits/Byte][0.001 E^{-6} Sec/Bit] = .096 usec

4. To calculate Packets/sec, it is simply the inverse of the packet time [Equation 1]:

   Rate (Packets/Sec) = 1/(Packet Time (Sec)) [3]

5. If we consider Pecent Utilization to be the percent of the maximum packet rate (as SmartBits does), then we are stating:

   Rate(Packets/Sec) = (% utilization (as fraction)[1/Packet $Time_{min}$ (sec)] [4]

6. Since the only control we have to adjust the rate is the inter-packet gap, we can equate [3] to [4] and solve for the gap as a function of packet size:

   1/[(Packet Size+Preamble)(8)(Bit Time)+GAP] = (% utilization)*1/[(Packet Size+Preamble)(8)(Bit Time)+GAP] [5]

7. Thus, we can use equation [5] generically, For example:

   80% utilization, 64 Byte Packet, Fast Ethernet

   Gap(usec)= [[1-0.80][(64+8)(8)(0.01)]+0.96]/0.18 = 2.640 usec

Appendix B: Source Code for Automation Testbed

setup_ip.tk

```
#!/usr/local/bin/expect -f
########################################################################
################################################
#
#        File Name:  begin_ip.tk
# Revision History:  Revision  Date         Author            Remarks
#                    0.1       20 JUN 2000  Adam Fairbanks    First
Version
#                     .2       25 JUN 2000  Adam Fairbanks    made more
input variables
#
#                    1.01      02 JUL 2000  Adam Fairbanks    two
different out
#                                                             put files,
setup.
#                                                             cfg for
smartbits
#                                                             use and
DUTsetup.
#                                                             cfg for
DUT setup
#                    1.05      20 JUL 2000                    Finished
automation for
#                                               DUT_cfg.tcl and
ip_auto.tcl
#                                               Fully Working Version!
#                     Release:  MQP
#      Description:  Automated Test Menu for Internet Protocol Tests
#                     - Packet Loss
#                     - Throughput
#                     - Interacts with DUT to set up environment
#                     - Interacts with Packet Generator (SmartBits),
creates
#                       traffic for tests
```

```
########################################################################
################################################
#
# Files in automation: the following files make up the automation suite

            begin_ip.tk: what we are in now, tk setup window for the
automation
            DUT_cfg.tcl: the Device Under test setup scipt (automation)
            ip_auto.tcl: the Packet Generator automations script
            setup.cfg where the variables form this window are stored
for the ip_auto.tcl
            DUT_setup.cfg where the variables from this window are
stored in for DUT_cfg.tcl
            64IPf.txt: the 64 byte packet( without source and
destination MAC preamble)
                128IPf.txt: the 128 byte packet ( without the source and
destination MACs)
                256IPf.txt:    etc......
                results.txt: where the results of the automation go
#
########################################################################
################################################

source /sqa/test/scripts/co_op_files/common_lib/common.lib

########################################################################
################################################
#
#setup variables

set DUT_setup_file "DUT_setup.cfg"
set setup_file    "setup.cfg"
set test_var      "TEST_CASES"
set title "Automated IP Testbed"

set numports       NUM_PORTS
set smart_ip       SMARTBITS_IP
set sb_ip1         IP_ADDRESS1
set sb_mac1        MAC1
set sb_ip2         IP_ADDRESS2
set sb_mac2        MAC2

set eme_ip_var     EME_IP
set slot_num       SLOT_NUMBER
set DUT_ip1        DUT_IP1
set DUT_ip2        DUT_IP2
set Port1          PORT1
set Port2          PORT2

set notes_dir
/sqa/test/scripts/co_op_files/afairban/multi_firewall/Release_Notes

global 3CL_DEBUG
set 3CL_DEBUG 1
```

```
3CL_getRelNotes $notes_dir

######################################################################
# Setup the window with the colors, and the correct title


proc 3CL_formatWindows {title args} {
    global bg_color;
    global fg_color;
    global select_color;
    global entry_bg_color;
    global act_bg_colors;
    global highlite_colors;
    global font;

    if {![llength $args]} {
      set window .
      set sub_win .
      set border 0
    } elseif {[llength $args] == 1} {
      set window [lindex $args 0]
      set sub_win "$window\."
      set border 0
    } else {
      set window [lindex $args 0]
      set sub_win "$window\."
      set border [lindex $args 1]
    }
    set title_font "-Adobe-Times-Medium-R-Normal--*-320-*-*-*-*-*-*"
    set directory "/sqa/test/scripts/co_op_files/bdouglas/run_files"
    if {![info exists bg_color]} {
      readColors;
    }
    wm title $window "$title"
    $window configure -bg \#$bg_color(0);

    if {$border} {
      frame $sub_win\_top -bg \#$bg_color(0) -borderwidth 5 -relief
groove;
    } else {
      frame $sub_win\_top -bg \#$bg_color(0) -borderwidth 0;
    }
    pack $sub_win\_top -in $window -side top
    label $sub_win\_top.l -font $title_font -bg \#$bg_color(0) -fg
\#$fg_color(0) -text "$title" -width 30 -height 1
    pack $sub_win\_top.l -padx 0 -pady 0 -anchor center -in
$sub_win\_top -fill x -side left

image create photo 3com -file "$directory\/3com_logo.gif"
    label $sub_win\_top.3com -borderwidth 0 -bg \#$bg_color(0) -width
200 -height 99 -image 3com -bd 0
    pack $sub_win\_top.3com -padx 0 -pady 0 -anchor nw -side left -in
$sub_win\_top
}

3CL_formatWindows "$title $v_num"

3CL_getNextColor

wm geometry . +90+5

######################################################################
# every time button is clicked or unclicked, status of tests is shown in
window for debugging use

set widget ".bottom.right"
bind . <Button-1> {+
    set widget %W
    U_onClick
}
bind . <Key> {+
    set widget %W
    U_onClick
}

#####################################################################
#titles, variables and explanations for buttons

set def_explain      "Welcome to the Automated Testbed for the IP
Throughput and Packet Loss Tests. By default,\n all tests will be
performed. To disable a test, please unclick the button that you wish
not\n to test. Addtionally, scrolling over each button will give a
description of\n its test functions."

set test_strings ""
set test_vars ""
set test_explains ""

lappend test_strings "Throughput Tests"
lappend test_vars "TEST_1"
lappend test_explains "The Throughput test tests the fastest rate at
which a device can forward frames without\nerror. If a single frame is
dropped the test fails and the test is repeated at a lower throughput
rate."

lappend test_strings "Packet Loss Rate Tests"
lappend test_vars "TEST_2"
lappend test_explains "This test measures the percentage of frames lost
by the Device Under Test that should\nhave been forwarded."

#***********************************************************************
************************************************
# Procedures
```

```
#**********************************************************************
************************************************



#every mouse click, status of tests for debug
proc U_onClick {} {

    global test_vars
    global num_tests

    for {set i 0} {$i < $num_tests} {incr i} {
      set variable [lindex $test_vars $i]
      global $variable
      puts "$variable == [set $variable]"

    }
}

proc Init {} {

    global test_vars
    global num_tests

    for {set i 0} {$i < $num_tests} {incr i} {
      set variable [lindex $test_vars $i]
      global $variable
      set $variable 1
    }
}
########
#this is where the variables are retrieved from DUT_setup.cfg and
setup.cfg are put into
#the textboxes

set curr_explain $def_explain

if {![file exists $setup_file]} {

    exec touch $setup_file
    exec chmod 777 $setup_file

} else {

    set env_file [open $setup_file r]
    set max_var 0
    while {[gets $env_file line] != -1} {
      if [string match "\[A-Z]*" $line] {
          set var_($max_var) $line
          incr max_var
      }
    }
    for {set i 0} {$i < $max_var} {incr i} {
      set variable($i) [lindex [split $var_($i) =] 0]
      set definition($i) [lindex [split $var_($i) '] 1]
      set $variable($i) $definition($i)
    }
    close $env_file
}


if {[file exists $setup_file]} {

 set $smart_ip [set $smart_ip]

    set $numports [set $numports]
    set $sb_ip1 [set $sb_ip1]
    set $sb_ip2 [set $sb_ip2]
    set $sb_mac1 [set $sb_mac1]
    set $sb_mac2 [set $sb_mac2]
} else {

    set $smart_ip "?????"
    set $numports "?????"
}

if {![file exists $DUT_setup_file]} {

    exec touch $DUT_setup_file
    exec chmod 777 $DUT_setup_file

} else {

    set env_file [open $DUT_setup_file r]
    set max_var 0
    while {[gets $env_file line] != -1} {
      if [string match "\[A-Z]*" $line] {
          set var_($max_var) $line
          incr max_var
      }
    }
    for {set i 0} {$i < $max_var} {incr i} {
      set variable($i) [lindex [split $var_($i) =] 0]
      set definition($i) [lindex [split $var_($i) '] 1]
      set $variable($i) $definition($i)
    }
    close $env_file
}


if {[file exists $DUT_setup_file]} {
  set $DUT_ip1 [set $DUT_ip1]
  set $DUT_ip2 [set $DUT_ip2]
  set $eme_ip_var [set $eme_ip_var]
  set $slot_num [set $slot_num]
  set $Port1 [set $Port1]
  set $Port2 [set $Port2]
} else {
    set $DUT_ip1 "?????"
    set $DUT_ip2 "?????"
    set $eme_ip_var "?????"
    set $slot_num "?????"
}
set num_tests [llength $test_strings]


#this is where the labels and textboxes and buttons are packed into the
tk window for display

frame .top3 -bg \#$bg_color(0)
pack .top3 -side top -anchor w
label .top3.telnetl -bg \#$bg_color(0) -font $font -fg \#$fg_color(0) -
text "SmartBits IP address:"
entry .top3.telnet -bg \#$entry_bg_color(0) -font $font -fg
\#$select_color(0) -textvariable $smart_ip -highlightcolor
\#$highlite_colors(100)
label .top3.numportsl -bg \#$bg_color(0) -font $font -fg \#$fg_color(0)
-text "Pair of ports to be used on Smartbits (ex:1,2 or 3,6):"
entry .top3.numports -bg \#$entry_bg_color(0) -font $font -fg
\#$select_color(0) -textvariable $numports -highlightcolor
\#$highlite_colors(100)
pack .top3.telnetl .top3.telnet -in .top3  -padx 2 -pady 2 -side left -
anchor w
pack .top3.numportsl .top3.numports -in .top3 -padx 2 -pady 2 -side left
-anchor w

frame .top4 -bg \#$bg_color(0)
pack .top4 -side top -anchor w
label .top4.ip1l -bg \#$bg_color(0) -font $font -fg \#$fg_color(0) -text
"SmartCard IP address 1 (ex: 10.1.1.2):"
entry .top4.ip1 -bg \#$entry_bg_color(0) -font $font -fg
\#$select_color(0) -textvariable $sb_ip1 -highlightcolor
\#$highlite_colors(100)
label .top4.mac1l -bg \#$bg_color(0) -font $font -fg \#$fg_color(0) -
text "SmartCard MAC 1 (ex: 000000100002):"
entry .top4.mac1 -bg \#$entry_bg_color(0) -font $font -fg
\#$select_color(0) -textvariable $sb_mac1 -highlightcolor
\#$highlite_colors(100)
pack .top4.ip1l .top4.ip1 -in .top4  -padx 2 -pady 2 -side left -anchor
w
pack .top4.mac1l .top4.mac1 -in .top4 -padx 2 -pady 2 -side left -anchor
w


frame .top5 -bg \#$bg_color(0)
pack .top5 -side top -anchor w
label .top5.ip2l -bg \#$bg_color(0) -font $font -fg \#$fg_color(0) -text
"SmartCard IP address 2 (ex: 20.1.1.2):"
entry .top5.ip2 -bg \#$entry_bg_color(0) -font $font -fg
\#$select_color(0) -textvariable $sb_ip2 -highlightcolor
\#$highlite_colors(100)
```

```
label .top5.mac2l -bg \#$bg_color(0) -font $font -fg \#$fg_color(0) -
text "SmartCard MAC 2 (ex: 000000200002):"
entry .top5.mac2 -bg \#$entry_bg_color(0) -font $font -fg
\#$select_color(0) -textvariable $sb_mac2 -highlightcolor
\#$highlite_colors(100)
pack .top5.ip2l .top5.ip2 -in .top5  -padx 2 -pady 2 -side left -anchor
w
pack .top5.mac2l .top5.mac2 -in .top5 -padx 2 -pady 2 -side left -anchor
w


frame .top2 -bg \#$bg_color(0)
pack .top2 -side top -anchor w
label .top2.telnetl -bg \#$bg_color(0) -font $font -fg \#$fg_color(0) -
text "Device Under Test IP address:"
entry .top2.telnet -bg \#$entry_bg_color(0) -font $font -fg
\#$select_color(0) -textvariable $eme_ip_var -highlightcolor
\#$highlite_colors(100)
label .top2.slotnuml -bg \#$bg_color(0) -font $font -fg \#$fg_color(0) -
text " Slot number of DUT(ex: 1-7):"
entry .top2.slotnum -bg \#$entry_bg_color(0) -font $font -fg
\#$select_color(0) -textvariable $slot_num -highlightcolor
\#$highlite_colors(100)
pack .top2.telnetl .top2.telnet -in .top2  -padx 2 -pady 2 -side left -
anchor w
pack .top2.slotnuml .top2.slotnum -in .top2  -padx 2 -pady 2 -side left
-anchor w


frame .top6 -bg \#$bg_color(0)
pack .top6 -side top -anchor w
label .top6.ip1l -bg \#$bg_color(0) -font $font -fg \#$fg_color(0) -text
"DUT Test IP 1 (ex: 10.1.1.1):"
entry .top6.ip1 -bg \#$entry_bg_color(0) -font $font -fg
\#$select_color(0) -textvariable $DUT_ip1 -highlightcolor
\#$highlite_colors(100)
label .top6.ip2l -bg \#$bg_color(0) -font $font -fg \#$fg_color(0) -text
"DUT Test IP 2 (ex:20.1.1.1):"
entry .top6.ip2 -bg \#$entry_bg_color(0) -font $font -fg
\#$select_color(0) -textvariable $DUT_ip2 -highlightcolor
\#$highlite_colors(100)
pack .top6.ip1l .top6.ip1 -in .top6  -padx 2 -pady 2 -side left -anchor
w
pack .top6.ip2l .top6.ip2 -in .top6 -padx 2 -pady 2 -side left -anchor w


frame .top7 -bg \#$bg_color(0)
pack .top7 -side top -anchor w
label .top7.port1l -bg \#$bg_color(0) -font $font -fg \#$fg_color(0) -
text "DUT Port 1 (ex: 1):"
entry .top7.port1 -bg \#$entry_bg_color(0) -font $font -fg
\#$select_color(0) -textvariable $Port1 -highlightcolor
\#$highlite_colors(100)
```

```
label .top7.port2l -bg \#$bg_color(0) -font $font -fg \#$fg_color(0) -
text "DUT Port 2 (ex: 2):"
entry .top7.port2 -bg \#$entry_bg_color(0) -font $font -fg
\#$select_color(0) -textvariable $Port2 -highlightcolor
\#$highlite_colors(100)
pack .top7.port1l .top7.port1 -in .top7  -padx 2 -pady 2 -side left -
anchor w
pack .top7.port2l .top7.port2 -in .top7 -padx 2 -pady 2 -side left -
anchor w



frame .mid -bg \#$bg_color(0)
pack .mid -side top -anchor nw

label .mid.explain -textvariable curr_explain -bg \#$bg_color(0) -font
$font -fg \#$fg_color(0) -width 90 -height 6

for {set i 0} {$i < $num_tests} {incr i} {

    set string   [lindex $test_strings $i]
    set var [lindex $test_vars $i]

    3CL_getNextColor
    checkbutton .mid.[string tolower $var] -text $string -variable $var
-bg \#$bg_color(0) \
      -highlightcolor \#$highlite_colors(100) -font $font -fg
\#$fg_color(0) -activebackground \#$act_bg_colors(100)

    bind .mid.[string tolower $var] <Enter> {
      set var [lindex [split %W .] 2]
      set variable_index [lsearch -exact $test_vars [string toupper
$var]]
      set curr_explain [lindex $test_explains $variable_index]
    }
    bind .mid.[string tolower $var] <Leave> {
      set curr_explain $def_explain
    }

    pack .mid.[string tolower $var] -in .mid -side top -padx 2 -pady 2 -
anchor nw

}

pack .mid.explain -in .mid -side top -padx 2 -pady 3

Init

frame .footer -bg \#$bg_color(0)
pack .footer -side top -anchor nw

3CL_getNextColor
#########
```

```
##this is where the new information entered by the is stored back in the
two setup files to be
# used by the scripts

button .footer.continue -text "Continue" -bg \#$bg_color(0) -
highlightcolor \#$highlite_colors(100) -font $font -fg \#$fg_color(0) \
    -activebackground \#$act_bg_colors(100) -command {
      if [file exists $setup_file] {
          exec rm $setup_file
      }
      exec touch $setup_file
      exec chmod 777 $setup_file

      set temp_file [open $setup_file w]

      set tests_to_run ""
      for {set i 0} {$i < $num_tests} {incr i} {
          if [set [lindex $test_vars $i]] {
            lappend tests_to_run $i
          }
      }
      puts $temp_file "[set test_var]='[set tests_to_run]'\t\t;export
[set test_var]"
        puts $temp_file "$smart_ip=\'[set $smart_ip]\'\t\t\t; export
$smart_ip"
      puts $temp_file "$numports=\'[set $numports]\'\t\t\t; export
$numports"
        puts $temp_file "$sb_ip1=\'[set $sb_ip1]\'\t\t\t; export
$sb_ip1"
      puts $temp_file "$sb_ip2=\'[set $sb_ip2]\'\t\t\t; export $sb_ip2"
        puts $temp_file "$sb_mac1=\'[set $sb_mac1]\'\t\t\t; export
$sb_mac1"
      puts $temp_file "$sb_mac2=\'[set $sb_mac2]\'\t\t\t; export
$sb_mac2"
        close $temp_file
            if [file exists $DUT_setup_file] {
          exec rm $DUT_setup_file
      }
      exec touch $DUT_setup_file
      exec chmod 777 $DUT_setup_file
      set temp_file2 [open $DUT_setup_file w]
      puts $temp_file2 "$eme_ip_var=\'[set $eme_ip_var]\'\t\t\t; export
$eme_ip_var"
        puts $temp_file2 "$slot_num=\'[set $slot_num]\'\t\t\t; export
$slot_num"
        puts $temp_file2 "$DUT_ip1=\'[set $DUT_ip1]\'\t\t\t; export
$DUT_ip1"
        puts $temp_file2 "$DUT_ip2=\'[set $DUT_ip2]\'\t\t\t; export
$DUT_ip2"
        puts $temp_file2 "$Port1=\'[set $Port1]\'\t\t\t; export $Port1"
        puts $temp_file2 "$Port2=\'[set $Port2]\'\t\t\t; export $Port2"

        puts $temp_file2 "$sb_ip1=\'[set $sb_ip1]\'\t\t\t; export
$sb_ip1"
```

```
      puts $temp_file2 "$sb_ip2=\'[set $sb_ip2]\'\t\t\t; export $sb_ip2"
        puts $temp_file2 "$sb_mac1=\'[set $sb_mac1]\'\t\t\t; export
$sb_mac1"
      puts $temp_file2 "$sb_mac2=\'[set $sb_mac2]\'\t\t\t; export
$sb_mac2"
        close $temp_file2
        exit 1
      destroy .

 }

3CL_getNextColor
button .footer.exit -bg \#$bg_color(0) -font $font -fg \#$fg_color(0) -
text "Exit" -highlightcolor \#$highlite_colors(100) \
    -activebackground \#$act_bg_colors(100) -command {
      exit 0
      destroy .
    }

pack .footer.continue .footer.exit -in .footer -padx 2 -pady .5 -side
left
pack .footer -side top -anchor sw

focus .footer.continue
```

DUT_cfg.tcl

```
#!/sqa/usr/local/bin/expect/ --

#environment variables
global env
spawn $env(SHELL)
set setup_file    "DUT_setup.cfg"

#variables to be used to store information from the DUT_setup.cfg file
set slot_num       SLOT_NUM
set eme_ip_var     EME_IP
set slot_num       SLOT_NUMBER
set dut_ip1        DUT_IP1
set dut_ip2        DUT_IP2
set port1          PORT1
set port2          PORT2
set ip_address1    IP_ADDRESS1
set ip_address2    IP_ADDRESS2
set mac1           MAC1
set mac2           MAC2
########################################################################
#####
###
# start to get information from file either stored in DUT_setup.cfg
###
```

```
########################################################################
#####
if {![file exists $setup_file]} {

    exec touch $setup_file
    exec chmod 777 $setup_file

} else {

    set env_file [open $setup_file r]
    set max_var 0
    while {[gets $env_file line] != -1} {
      if [string match "\[A-Z]*" $line] {
          set var_($max_var) $line
          incr max_var
      }
    }
    for {set i 0} {$i < $max_var} {incr i} {
      set variable($i) [lindex [split $var_($i) =] 0]
      set definition($i) [lindex [split $var_($i) '] 1]
      set $variable($i) $definition($i)
    }
    close $env_file
}

if {[file exists $setup_file]} {
 set $eme_ip_var [set $eme_ip_var]
 set $slot_num [set $slot_num]
    set dut_ip1 [set $dut_ip1]
    set dut_ip2 [set $dut_ip2]
    set port1 [set $port1]
    set port2 [set $port2]
    set mac1  [set $mac1]
    set mac2  [set $mac2]
    set ip_address1 [set $ip_address1]
    set ip_address2 [set $ip_address2]

} else { puts "setup file does not exist!!!!"
}



puts "Starting IP Automation..."

#check to see if DUT responds, login to DUT and connect to appropriate
blade
send "ping [set $eme_ip_var]\r"
expect " [set $eme_ip_var] is alive"
send "telnet [set $eme_ip_var]\r"
expect "Login:"
send "CB9000DEBUG\r"
expect "Password:"
send "\r"
expect ">"
```

```
send "connect [set $slot_num].1\r"

#setup vlan for ip interface one
expect ":"
send "bri vlan def 2\r"
expect "(1-13|all|?):"
send "$port1\r"
expect "):"
send "ip\r"
expect "):"
send "q\r"
expect "(n,y)"
send "n\r"
expect "(n,y)"
send "n\r"
expect "Name {?}"
send "ip_vlan1\r"

#setup vlan for ip interface two
expect ":"
send "bri vlan def 3\r"
expect "(1-13|all|?)"
send "$port2\r"
expect "):"
send "ip\r"
expect "):"
send "q\r"
expect "(n,y)"
send "n\r"
expect "(n,y)"
send "n\r"
expect "Name {?}"
send "ip_vlan2\r"

#setup ip interfaces on DUT for use with Smartbits packets
expect ":"
send "ip int def [set dut_ip1] 255.255.255.0 vlan 2\r"

expect ":"
send "ip int def [set dut_ip2] 255.255.255.0 vlan 3\r"

# setup static arps for smartbits and DUT to communicate through
expect ":"
send "ip arp static 1 [set ip_address1] [set mac1]\r"
expect ":"
send "ip arp static 2 [set ip_address2] [set mac2]\r"

expect ":"
send "ip routing enable\r"

#set eth portspeed at 100 full duplex (fast ethernet)
expect ":"
send "eth portmode a\r"
expect "(n,y)"
```

```
send "y\r"
expect ":"
send "100full\r"

#disable autonegotiation since port speed already at fast ethernet 100m
full
expect ":"
send "eth autoneg a dis\r"

#disable spanning tree packets (for stats to not count other packets)
expect ":"
send " bri spann stpState dis\r"

#disable dvmrp (for statistics to not count other packets)
expect ":"
send " ip multi dvmrp int mod a dis dis\r"

#disable igmp querying (for statistics to not count other packets)
expect ":"
send "ip multi igmp query dis\r"

#disable backplane port
expect ":"
send "eth portState 13 dis\r"

#logout of blade and DUT
expect ":"
send "dis\r"
expect ">"
send "logout\r"
expect "bash"
```

ip_auto.tcl

```
#!/sqa/usr/local/bin/expect/ --

global env
spawn $env(SHELL)
```

```
set result_file   "results.txt"
set setup_file    "setup.cfg"
set numports       NUM_PORTS
set slot_num       SLOT_NUM
set eme_ip_var     EME_IP
set smart_ip       SMARTBITS_IP
set slot_num       SLOT_NUMBER
set mac1           MAC1
set mac2           MAC2

########################################################################
#####
###
# start to get information from file either stored by begin_ip.tk or
manual
###
########################################################################
#####
if {![file exists $setup_file]} {

    exec touch $setup_file
    exec chmod 777 $setup_file

} else {

    set env_file [open $setup_file r]
    set max_var 0
    while {[gets $env_file line] != -1} {
      if [string match "\[A-Z]*" $line] {
          set var_($max_var) $line
          incr max_var
      }
    }
    for {set i 0} {$i < $max_var} {incr i} {
      set variable($i) [lindex [split $var_($i) =] 0]
      set definition($i) [lindex [split $var_($i) '] 1]
      set $variable($i) $definition($i)
    }
    close $env_file
}

if {[file exists $setup_file]} {
 set $smart_ip [set $smart_ip]
     set $numports [set $numports]
    set $mac1 [set $mac1]
    set $mac2 [set $mac2]
} else { puts "setup file does not exist!!!!"
}

####################################################################
#start throughput test

####################################################
#starts at 64 bit packet size, 100% util (defaults at 96 usec)
```

```
puts "[set $numports]"
puts "Starting IP Automation..."
send "/sqa/tools/PacketBuilder2.0/pb.9.2.98 -i\r"
expect "pbCommand>"
send "pb -l\r"
expect "pbCommand>"
send "pb -ht1 -n100000 -vs[set $mac2] -vs[set $mac1] -f 64IPf.txt\n"
expect "pbCommand>"
send "pb -ht1\n"
expect "pbCommand>"
sleep 10
send "pb -ht2 -scr"

if {
expect "RX Packet count for slot 2 is 100000 packets"
} {
    if [file exists $results_file] {
          exec rm $results_file
      }
      exec touch $results_file
      exec chmod 777 $results_file

      set temp_file [open $results_file w]

        puts temp_file "Passed 64 byte test at 100% utilization\n"
    puts temp_file "100000 packets sent from [set $mac1] to [set $mac2]
with .0960 interpacket gap"
        close $temp_file

#################################################################
#if first trial fails at 100%, tries again at 80 (-g272 for 272 usec
gap)
} else {
send "/sqa/tools/PacketBuilder2.0/pb.9.2.98 -i\r"
expect "pbCommand>"
send "pb -l\r"
expect "pbCommand>"
send "pb -ht1 -g272 -n100000 -vs[set $mac2] -vs[set $mac1] -f
64IPf.txt\n"
expect "pbCommand>"
send "pb -ht1\n"
expect "pbCommand>"
sleep 10
send "pb -ht2 -scr"
      if {expect "RX Packet count for slot 2 is 100000 packets"} {

if [file exists $results_file] {
          exec rm $results_file
      }
      exec touch $results_file
      exec chmod 777 $results_file

      set temp_file [open $results_file w]
```

```
        puts temp_file "Passed 64 byte test at 80% utilization\n"
    puts temp_file "100000 packets sent from [set $mac1] to [set $mac2]
with .0272 interpacket gap"
        close $temp_file

} else {
#finally tries at 60% ( 565 usec gap)
send "/sqa/tools/PacketBuilder2.0/pb.9.2.98 -i\r"
expect "pbCommand>"
send "pb -l\r"
expect "pbCommand>"
send "pb -ht1 -g565 -n100000 -vs[set $mac2] -vs[set $mac1] -f
64IPf.txt\n"
expect "pbCommand>"
send "pb -ht1\n"
expect "pbCommand>"
sleep 10
send "pb -ht2 -scr"
      if {expect "RX Packet count for slot 2 is 100000 packets"} {

if [file exists $results_file] {
          exec rm $results_file
      }
      exec touch $results_file
      exec chmod 777 $results_file

      set temp_file [open $results_file w]

        puts temp_file "Passed 64 byte test at 60% utilization\n"
    puts temp_file "100000 packets sent from [set $mac1] to [set $mac2]
with .565 interpacket gap"
        close $temp_file

} else { puts " You're router stinks....it can't even do 60%
utilization!!!!\r Now exiting......"
sleep 5
exit 0
  }
}
}



####################################################################
#continue with 128 byte packets

####################################################
#128 byte packet size, 100% util (defaults at 96 usec)
puts "[set $numports]"
puts "Starting IP Automation..."
send "/sqa/tools/PacketBuilder2.0/pb.9.2.98 -i\r"
expect "pbCommand>"
send "pb -l\r"
expect "pbCommand>"
```

```
send "pb -ht1 -n100000 -vs[set $mac2] -vs[set $mac1] -f 128IPf.txt\n"
expect "pbCommand>"
send "pb -ht1\n"
expect "pbCommand>"
sleep 10
send "pb -ht2 -scr"

if {
expect "RX Packet count for slot 2 is 100000 packets"
} {


      exec touch $results_file
      exec chmod 777 $results_file

      set temp_file [open $results_file a]

        puts temp_file "Passed 128 byte test at 100% utilization\n"
    puts temp_file "100000 packets sent from [set $mac1] to [set $mac2]
with .0960 interpacket gap"
        close $temp_file

#################################################################
#if first trial fails at 100%, tries again at 80 (-g400 for 400 usec
gap)
} else {
send "/sqa/tools/PacketBuilder2.0/pb.9.2.98 -i\r"
expect "pbCommand>"
send "pb -l\r"
expect "pbCommand>"
send "pb -ht1 -g400 -n100000 -vs[set $mac2] -vs[set $mac1] -f
128IPf.txt\n"
expect "pbCommand>"
send "pb -ht1\n"
expect "pbCommand>"
sleep 10
send "pb -ht2 -scr"
      if {expect "RX Packet count for slot 2 is 100000 packets"} {

      exec touch $results_file
      exec chmod 777 $results_file

      set temp_file [open $results_file a]

        puts temp_file "Passed 128 byte test at 80% utilization\n"
    puts temp_file "100000 packets sent from [set $mac1] to [set $mac2]
with .4 interpacket gap"
        close $temp_file

} else {
#finally tries at 60% ( 906 usec gap)
send "/sqa/tools/PacketBuilder2.0/pb.9.2.98 -i\r"
expect "pbCommand>"
send "pb -l\r"
expect "pbCommand>"
send "pb -ht1 -g906 -n100000 -vs[set $mac2] -vs[set $mac1] -f
128IPf.txt\n"
expect "pbCommand>"
send "pb -ht1\n"
expect "pbCommand>"
sleep 10
send "pb -ht2 -scr"
      if {expect "RX Packet count for slot 2 is 100000 packets"} {
      exec touch $results_file
      exec chmod 777 $results_file

      set temp_file [open $results_file a]

        puts temp_file "Passed 128 byte test at 60% utilization\n"
    puts temp_file "100000 packets sent from [set $mac1] to [set $mac2]
with .906 interpacket gap"
        close $temp_file

} else { puts " You're router stinks....it can't even do 60%
utilization!!!!\rNow exiting....."
sleep 5
exit 0
}
}
}


####################################################################
#continue with 256 byte packets

####################################################
#256 byte packet size, 100% util (defaults at 96 usec)
puts "[set $numports]"
puts "Starting IP Automation..."
send "/sqa/tools/PacketBuilder2.0/pb.9.2.98 -i\r"
expect "pbCommand>"
send "pb -l\r"
expect "pbCommand>"
send "pb -ht1 -n100000 -vs[set $mac2] -vs[set $mac1] -f 256IPf.txt\n"
expect "pbCommand>"
send "pb -ht1\n"
expect "pbCommand>"
sleep 10
send "pb -ht2 -scr"

if {
expect "RX Packet count for slot 2 is 100000 packets"
} {


      exec touch $results_file
      exec chmod 777 $results_file
      set temp_file [open $results_file a]

        puts temp_file "Passed 256 byte test at 100% utilization\n"
    puts temp_file "100000 packets sent from [set $mac1] to [set $mac2]
with .096 interpacket gap"
        close $temp_file

#################################################################
#if first trial fails at 100%, tries again at 80 (-g656 for 656 usec
gap)
} else {
send "/sqa/tools/PacketBuilder2.0/pb.9.2.98 -i\r"
expect "pbCommand>"
send "pb -l\r"
expect "pbCommand>"
send "pb -ht1 -g656 -n100000 -vs[set $mac2] -vs[set $mac1] -f
256IPf.txt\n"
expect "pbCommand>"
send "pb -ht1\n"
expect "pbCommand>"
sleep 10
send "pb -ht2 -scr"
      if {expect "RX Packet count for slot 2 is 100000 packets"} {

      exec touch $results_file
      exec chmod 777 $results_file

      set temp_file [open $results_file a]

        puts temp_file "Passed 256 byte test at 80% utilization\n"
    puts temp_file "100000 packets sent from [set $mac1] to [set $mac2]
with .656 interpacket gap"
        close $temp_file

} else {
#finally tries at 60% ( 1589 usec gap)
send "/sqa/tools/PacketBuilder2.0/pb.9.2.98 -i\r"
expect "pbCommand>"
send "pb -l\r"
expect "pbCommand>"
send "pb -ht1 -g1589 -n100000 -vs[set $mac2] -vs[set $mac1] -f
256IPf.txt\n"
expect "pbCommand>"
send "pb -ht1\n"
expect "pbCommand>"
sleep 10
send "pb -ht2 -scr"
      if {expect "RX Packet count for slot 2 is 100000 packets"} {
      exec touch $results_file
      exec chmod 777 $results_file

      set temp_file [open $results_file a]

        puts temp_file "Passed 256 byte test at 60% utilization\n"
    puts temp_file "100000 packets sent from [set $mac1] to [set $mac2]
with 1.589 interpacket gap"
        close $temp_file

} else { puts " You're router stinks....it can't even do 60%
utilization!!!!\rNow exiting....."
sleep 5
exit 0
}
}

########################################################################
##
######Packet Loss test
#########currently does it for 64 byte packet at 100 % utilization
send "/sqa/tools/PacketBuilder2.0/pb.9.2.98 -i\r"
expect "pbCommand>"
send "pb -l\r"
expect "pbCommand>"
send "pb -ht1 -n100000 -vs[set $mac2] -vs[set $mac1] -f 64IPf.txt\n"
expect "pbCommand>"
send "pb -ht1\n"
expect "pbCommand>"
sleep 10
send "pb -ht2 -scr"
    if {expect "RX Packet count for slot 2 is [set $packet1] packets"} {

$final_packet= $packet1-100000

      exec touch $results_file
      exec chmod 777 $results_file

      set temp_file [open $results_file a]

        puts temp_file "Packet Loss Test for 64 bytes:\n"
    puts temp_file "100000 packets sent from [set $mac1] to [set $mac2]
at 100% utilization: "
puts temp_file " [set $final_packet] dropped by DUT\n
        close $temp_file

exit 0
}
}
```

DUT_setup.cfg

```
EME_IP='158.101.97.7'               ; export EME_IP
SLOT_NUMBER='6'               ; export SLOT_NUMBER
DUT_IP1='10.1.1.1'                  ; export DUT_IP1
DUT_IP2='20.1.1.1'                  ; export DUT_IP2
PORT1='1'               ; export PORT1
PORT2='2'               ; export PORT2
IP_ADDRESS1='10.1.1.2'              ; export IP_ADDRESS1
IP_ADDRESS2='20.1.1.2'              ; export IP_ADDRESS2
MAC1='00-00-00-10-00-02'                  ; export MAC1
MAC2='00-00-00-20-00-02'                  ; export MAC2
```

setup.cfg

```
TEST_CASES='0 1'        ;export TEST_CASES
SMARTBITS_IP='10.10.10.10'                ; export SMARTBITS_IP
NUM_PORTS='1,2'               ; export NUM_PORTS
IP_ADDRESS1='10.1.1.2'              ; export IP_ADDRESS1
IP_ADDRESS2='20.1.1.2'              ; export IP_ADDRESS2
MAC1='000000100002'                 ; export MAC1
MAC2='000000200002'                 ; export MAC2
```

64Ipf.txt

```
0800450000320000000040047AC90A0101021401010200000000000000000000000000000
00000000000000000000000000000000
```

128Ipf.txt

```
0800450000720000000040046F860A010102140101020000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000
```

256IPf.txt

```
080045000 0F20000000040047A09000A0101021401010200000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
```