Information Technology Journal 3 (1): 79-92, 2004
ISSN 1682-6027

A Practical Approach to Network Performance Characterization

[1]Khaled Shuaib and Andrew Brooks
[1]College of Information Technology, United Arab Emirates University, P.O. Box 17555, Al-Ain, UAE
Équipe Communications Corporation, 100 Nagog Park Acton, MA 01720, UAE

Abstract: With the great expansion of the Internet in the past few years, many new service providers and venders of networking equipment evolved to compete against one another. This melting pot has called for a comprehensive, universal and practical approach to the performance evaluation of devices running the most widely used technologies and applications. In this paper we address testing methodologies considering in most cases an Asynchronous Transfer Mode (ATM) device and a network of such devices. The proposed methodologies addresses network performance and characterization based on gained industrial practical experience from two perspectives: the service providers and vendors of switching/routing equipment. These procedures although mostly specific to ATM can be easily generalized to other technologies.

Key words: ATM, network performance, testing methodologies

INTRODUCTION

The explosive growth of multi-service networks has placed the focus on the scalability and performance issues of the current network infrastructure. Internet Service Providers already having large networks or fast growing networks in size and customer base are more vulnerable to network failures due to lack of proper design and well defined network performance and scalability characterization. The proper design of a network based on performance measures and desired scalability is the key element to preventing network failures and to control and limit the consequences of any failure once it happens.

In this document we outline a strategic direction by which to empirically tackle the quantification of the general areas of network performance. Conceptually, one can take a look at the topics of performance and scalability from a function-by-function perspective. The approach outlined in this document discusses performance in a holistic approach across the switch/router and it's applications. The reasoning behind this approach is that there are many aspects that can be measured or that compromise performance that are independent of the particular system component. CPU, utilization, memory consumption and the efficiency or lack thereof in implementing software algorithms such as search or storage functions are examples of metrics of concern common to most applications. Figure 1 illustrates the way the performance approach cuts across multiple disciplines focusing on each area as a quantifiable user of system resources.

In order to fully understand the complexities of the network element not just as it is fully built out, but as it is deployed in a network, it is necessary to take a multi-disciplinary approach to quantifying key performance and scalability metrics. Figure 2 highlights graphically the approach taken. In an ideal world, we would be able to have an instantly provisionable several hundred node network for each engineer to run through what-if scenarios to best emulate deployed networks in the field. Practically speaking, this is not feasible. In lieu of exclusively building out large networks, an approach which combines real network configurations with much larger emulated networks allows a much faster turnaround for test results of what if scenarios and a more feature rich set of benchmarking experiments that can ideally characterize even the largest networks. This is simply another approach and set of tools that is an enabler of real answers to some of the most difficult performance and scalability questions.

One of the most important aspects in creating a successful performance and scalability engineering effort is to have a well-defined understanding as to where it fits into the product development and deployment lifecycle. The earlier the concepts of performance and scalability are addressed in the cycle, the more impact it will have on the product. Likewise, without collective history of what was done right and what was done wrong from a performance and scalability perspective, the more likely the same mistakes will be made on new hardware and software projects. Figure 1-3 highlights the process by which performance and scalability becomes an integral part of the product lifecycle.

In this document we apply the general strategy outlined in this section to a more focused tactical approach meant to fully understand the metric being

Corresponding Author: Dr. Khaled Shuaib, College of Information Technology, United Arab Emirates University, P.O. Box 17555, Al-Ain, UAE

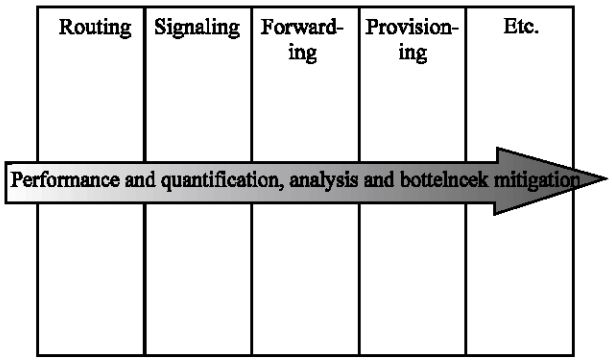

Fig. 1: Cross-application approach to performance and salability analysis

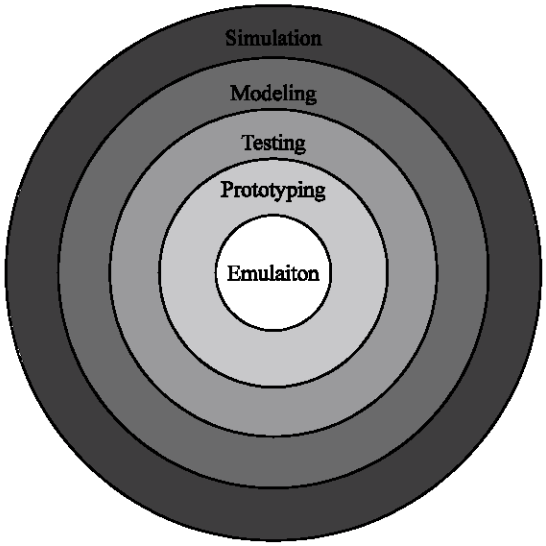

Fig. 2: Cross-functional approach to performance and scalability analysis

evaluated from a performance perspective. The objectives of the strategies and methodologies outlined in this document are many folds and serve many uses in Engineering, Marketing, Services, Customer Engineering and others. The following examples illustrate some of the specific applications where the performance testing can apply:

Release validation

- Ensure release to release performance and scaling metrics are met. This is mainly done to benchmark the performance of any new features, to discover any degradation in the system performance due to the additional code modules and to characterize any crossover performance relationships between code related modules.

Benchmarking

- Aid in network design and architecture to ensure reliability and scalability.
- SW and HW design and architectural tradeoffs. The tradeoffs are made so that neither performance nor reliability is compromised and so that a balance between the two is achieved.

Aid in simulation and modeling

- Raw data collected from limited and controlled laboratory performance testing of certain scenarios can be used to simulate and model scenarios of real networks running in the field.
- Validation of simulations and modeling results. Many ISPs can not afford to conduct laboratory experiments to test or emulate a certain network scenario at large scale. The results obtained from smaller real networks emulating the larger networks can be used to validate the results obtained via simulation or modeling.

Customer specific studies

- Understand topology specific behaviors before deploying these topologies carrying live traffic of real customers.
- To ensure resiliency, fault tolerance and backward compatibility that will minimize service disruption during the network migration or expansion.

Stress is the system/network behavior when either the performance or the capacity/scalability is exceeded. We believe that stress testing should be handled separately from performance testing and therefore it is not in the scope of this document.

This document serves to illustrate a starting point for each focus area. The actual simulation/modeling / testing/ prototyping/ emulation (Fig. 2) will build significantly off of these example test cases to fully characterize the Devise Under Test (DUT) not only as a network element but also as a constellation of network elements. The remainder of this document is organized as follow: in section three, testing classification with respect to the DUT is outlined. Section four, discusses in details the performance metrics and testing methodologies while considering an ATM switch as the DUT; however, most of these methodologies can be generalized to other switching and routing technologies. Metrics used for network performance are discussed in section five. Tools that could be used to enhance and facilitate comprehensive testing both at a single device and network level are discussed in section six. The conclusion of the paper is presented in section seven.

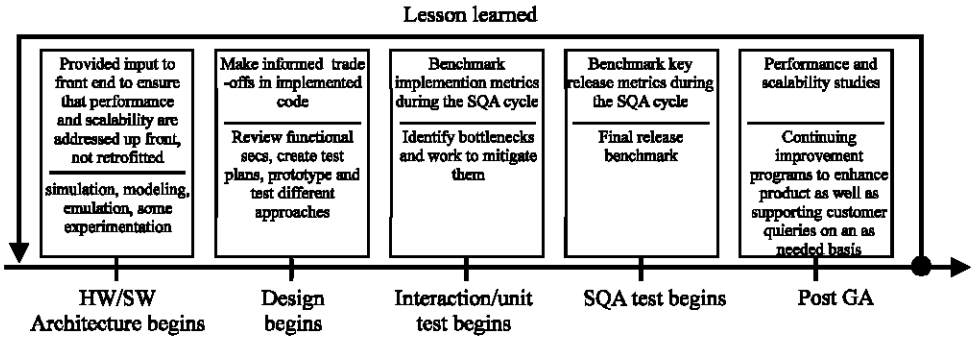

Fig. 3: Performance and scalability lifecycle process

Testing classification: Testing of a specific network element or a collection of elements is usually done by two parties: the vendors who sell these products and the customers who use them. The testing done by the vender is mostly done to debug and discover any potential software or hardware pitfalls or what is called bugs or defects and limitations. This is usually an elaborate testing which is done during and after the product production cycle. This testing is done using both the white box and black box approaches which are detailed below. The testing done by the vendors most likely to be generic and not tailored toward a specific customer especially when the customer base of the product is large.

However, the testing done by customers is usually limited in nature and tailored toward their own applications and specifications. This testing is of the black box type only and defects or limitations found mostly apply to a specific scenario or application.

White box testing: White box testing is done primarily by equipment vendors who have detailed architectural information that give them insight not necessarily available without access to specifications and source code. This type of testing will allow the tester to:

- Look into areas of potential concern or weakness that they might expose and mitigate
- Quantify limitations to help customers architect around them
- Baseline behavior that could be fed into models to predict the impact of future enhancements to software and hardware
- Access proprietary information about the tested software and hardware
- Test in a debug mode environment with access to source code
- Verify reported defects and proposed solutions at a faster base.

Black box testing: Black box testing is done by anyone using the product of interest to understand and characterize the behavior without knowledge of specifications and source code. Black box testing is extremely useful in:

- Allowing vendors to test a box in a more general way typical of a customer test or deployment scenario
- Allowing customers to "reverse engineer" the behavior of a box without access to proprietary information
- Controlled lab environment testing of specific feature or application
- Interoperability testing with other network devices to ensure compatibility and reliability of a network that might consist of equipments of several vendors or a mixture of the same vendor equipment.

Performance metrics and testing methodologies: When considering ATM performance metrics[1-7] it is important to consider the different connection types that exist and the unique metrics associated with each type. The three main types of connections on an ATM switch are PVCs, SVCs and SPVCs. The differences between these connection types are best understood from the perspective of the DUT. Table 1 summarizes the connection types and the dynamic of their setups. Below we briefly provide a description of each type.

PVCs: From the DUT's point of view, a Permanent Virtual Circuit (PVC) endpoints must be explicitly defined. In other words, an explicit interface and VPI (PVPC) or VPI/VCI (PVCC) must be defined for each circuit endpoint.

SVCs: From the DUT's point of view, a Switched Virtual Circuit (SVC) endpoints are signaled via the User to

Table 1: ATM connection types and their set-up mechanism

Connection Type	Originating EP	Interswitch EP	Interswitch EP	Terminating EP
PVC	Static	NA	NA	Static
SPVC	Static	Dynamic	Dynamic	Static
SVC	Dynamic	Dynamic	Dynamic	Dynamic

Network Interface (UNI) protocol. In other words, the circuit endpoints are negotiated between the attached device and the DUT. Likewise, in the inter-node case the Private Network to Network Interface (PNNI) endpoints are negotiated between the nodes connected with a PNNI link.

SPVCs: From the DUT's point of view, a Soft PVC's (SPVC) endpoints must be explicitly defined. In other words, an explicit interface and VPI (PVPC) or VPI/VCI (PVCC) must be associated with a particular ATM address, which is defined for each circuit endpoint. Like with SVCs, the PNNI endpoints are negotiated between the nodes connected with a PNNI link.

Element node/card stand alone performance

System and card boot time: One of the essential performance metrics that distinguishes vendor devices is the boot time of the device. It is critical to service providers to maintain a short boot time for a device or a card on it running in a production network. Minimum service disruption time upon a network failure is what customers are looking for when buying their services from an ISP.

The amount of time it takes a DUT or one of its cards to boot or fail over may or may not be related to a wide variety of metrics including: number of cards, card types, number of virtual interfaces, number of circuits, number of configured adjacencies, the size of the routing database and routing table, etc. In order to understand the contribution of each component to boot time one metric at a time should be varied while holding the rest of the metrics constant.

The results of this set of tests should be bimodal. If the metric being measured has no effect on boot time, the boot time would be flat across all data points. If the metric being measured does have an impact, the impact would most likely be linear, although in some special cases an exponential type of increase might occur. The following metrics should be applied against boot /fail over time for a DUT as a minimum.

Number of cards: Vary the number of cards of a given type between 1 and n where n is the maximum number of cards of a given type that can be supported by the DUT and measure the time it takes to reboot the DUT as well as one of the cards.

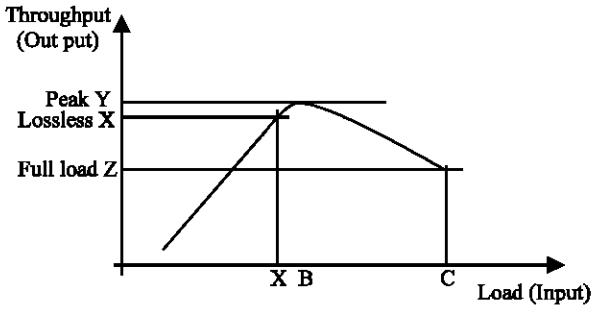

Fig. 4: Peak, lossless and full-load throughput

Card type: Put only one card of a given type in the chassis at a time and compare the boot time of each different card type.

Number of virtual interfaces: On a given card type, measure in fixed increments from 1 to n (where n = the maximum number of supported virtual interfaces) the time it takes a card to boot. Once the maximum has been reached, configure additional cards in the same manner (i.e. also with the maximum number of virtual interfaces) from 1 to n and measure the system boot time.

Number of circuits: On a given card type with one or more virtual interfaces, measure in fixed increments from 1 to m (where m = the maximum number of supported circuits) the time it takes a card to boot. Once the maximum has been reached, configure additional cards in the same manner (i.e. also with the maximum number of circuits) measure the system boot time.

Redundant/resilient failover: The intention of this section is to quantify the performance of DUT in the case of resilient and redundant failover. This should be done for all of the card types in the system which can be configured in a redundant/resilient mode and metrics appropriate for each card type should be applied.

I/O processor card fail over

I/O processor card failover performance can be broken down into two main categories

- 1:X (where X = 1 or N) Protection scheme fail over, this done at the card level where a card would be configured as a backup card for 1 or more cards incase of a failure.

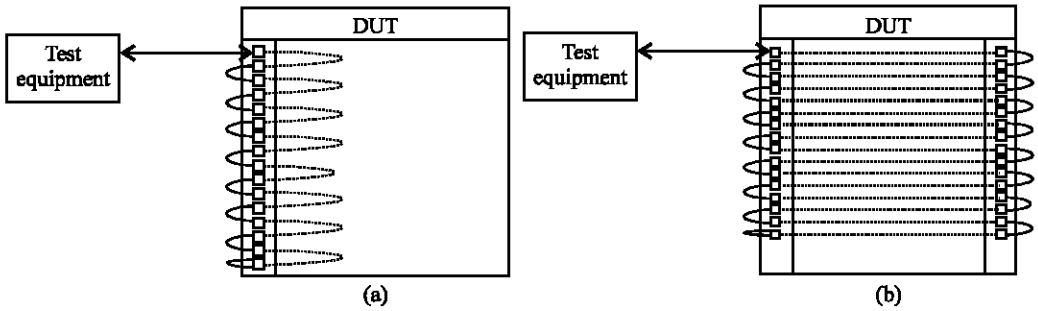

Fig. 5a) Serialized throughput test setup (intracard), b) serialized throughput test setup (intercard)

- Automatic Protection Switching (APS) switchover. As defined in Anonymous[7], linear APS is a standardized specification that allows interoperability among multiple vendor switches. APS at the line layer fully protects all connections within the line. There are two types of linear APS: APS 1:N and APS 1+1. The APS 1:N specification provides one backup or protection line to protect every N (where N is greater than or equal to 1) working lines. The APS 1+1 specification provides one protection line to act as a backup for one working line. In either APS 1:N or APS 1+1, should the working line experience a failure, the protection line automatically takes over and restores data flow to the network.

Both of these categories can be further broken down by the type of outage measured. Two main types of outage are:

Data plane: This is defined as the outage time in processing user packets. Performance testing should be conducted in APS, 1:1 and 1:N test cases using the same methodologies applied towards all other performance metrics. Single point measurements can often be misleading and dose not provide the context by which to determine the behavior. Data plane outage time can be measured by injecting cells across a configured SVC at a fixed interval and measuring the outage time. The error of this measurement is +/- the interval time. The maximum outage time will be ascertained by making the measurement have the lowest restoral priority.

Control plane: In addition to the data plane outage characterization outlined in the previous section, another important measure of system availability will be to characterize the ability of the DUT to be able to process new call setup messages post failover. The ideal methodology would be to measure the availability of the control plane in such a way as to set a time-stamped flag when the control plane becomes unavailable and another when availability resumes. These timestamps should be used in a way that would be least likely to perturb the very performance we are looking to measure. A good corroboration to this technique would be to do it with a fixed level of periodicity send call setup attempts and measure the outage during the failover period. A balance would have to be struck between the precision of the measurement using this technique and the additional perturbation such a test would have on the failover time due to CPU loading.

Software load time: One of the important features that service providers look for in a switch/router is how long would a software upgrade take. A test should be done first to measure the time it takes to reboot a node on the same software release, then measure the time it takes to reboot a node with a new software release. The time measured here should be from the moment the reboot takes place to the moment that all services are restored on the switch/router. With newer software releases that might contain extra code modules, it is often expected an extra delay in boot time upon an upgrade from a previous software release. In addition it is always recommended that a similar test is done for software down grade from a newer release to an older one.

Throughput benchmarking: Throughput is possibly one of the most commonly measured yet commonly miss-measured of all performance metrics. The most common misinterpretation usually made is the significant difference between lossless and lossy (henceforth referred to as full-load throughput) throughput. Lossless throughput can be described as the maximum achievable packet-forwarding rate beyond which packets are queued and/or discarded. Full-load throughput can be defined as the maximum achievable packet-forwarding rate based on a given input rate with no regard to the maximum achievable lossless rate of the DUT. In other words, the technique consists of measuring packet-forwarding capacity on an interface

Table 2: QoS class priority comparison matrix

Stream A QoS class	Stream B QoS class
UBR	UBR
UBR	ABR
UBR	VBR-NRT
UBR	VBR-RT
UBR	CBR
ABR	ABR
ABR	VBR-NRT
ABR	VBR-RT
ABR	CBR
VBR-NRT	VBR-NRT
VBR-NRT	VBR-RT
VBR-NRT	CBR
VBR-RT	VBR-RT
VBR-RT	CBR
CBR	CBR

card based on an input rate of 100% port speed capacity. For example, if a given card has 4 ports of OC-48 and when forwarding 64 byte IP packets the maximum achievable lossless throughput is 90% of line rate. If we conduct our test to blast 4x100% OC-48 64 byte streams at this card and measure the intact packets seen at the analyzer, the intact or good packet rate can be referred to as goodput. In this example we are measuring the goodput of a full-load performance test. Goodput beyond full-load is considered from the standpoint of this document as stress testing and will not be covered. An interesting point to make is that maximum achievable packet-forwarding rate does not necessarily occur with lossless throughput, X as defined in Fig. 4. refers to this condition as peak throughput Y. In other words, in some rare cases peak goodput rates could exceed lossless throughput rates. Figure 5 shows two possible examples of how the intra-card and inter-card throughput or goodput tests can be done on a DUT.

Latency and delay jitter characterization: Latency and delay jitter are two important QoS parameters that are often measured as part of any performance study of a DUT or a network of such devices. When evaluating a particular DUT, it is crucial to understand how to measure each of these factors accurately and in a practical manner. In the two sections below we discuss the way these two factors should be measured for performance testing of a DUT. The delay and delay jitter a cross a network of devices will consist of different segments and would vary from network to network depending on the network configuration, topology, load, size as well as the devices in the network[8].

Cell Transfer Delay (CTD): When measuring the latency across a DUT, the latency measurement technique that should be used is according to the method described by Anonymous[1] which is Message In Message Out

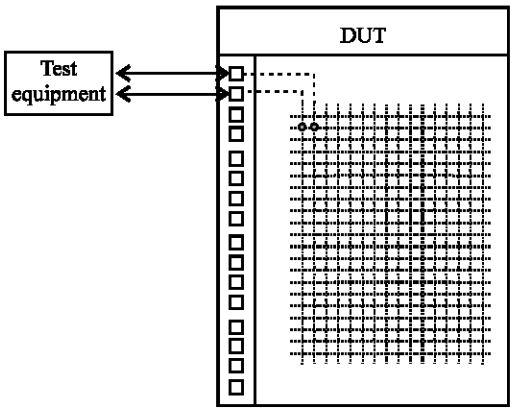

Fig. 6: CTD measurement a cross the DUT

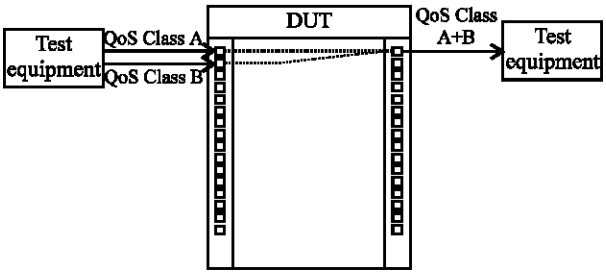

Fig. 7: QoS class priority comparison test setup

(MIMO). One of the best ways by which to measure the MIMO latency of a DUT, would be by inserting time stamps within the cell payload as done by many available test equipments. For practical purposes as well as the fact that the ATM Forum only suggests certain input rates, the following rates should be used: 1, 50, 75 and 100% of full foreground load (FFL). Minimum, maximum and mean CTD should be reported for all rates based on timed runs of at least 5 minutes to ensure that queues are not building up due to any system bottlenecks. If queue depth is increasing for a given input rate, this rate is beyond the capacity of the DUT and should not be considered in the CTD characterization.

CTD should be measured as shown in Fig. 6. Taking an initial run at each rate with the test equipment looped back to itself and subtracting the mean value from the measured results normalizes for the test equipment delay. It is assumed that the generators and analyzers are synchronized to the same clocking device.

Cell Delay Variation (CDV): When speaking of CTD it is not only the absolute delay that is important. The variation between measured CTD values or Cell Delay Variation (CDV) is used in conjunction with the GCRA[3] algorithm to police cell

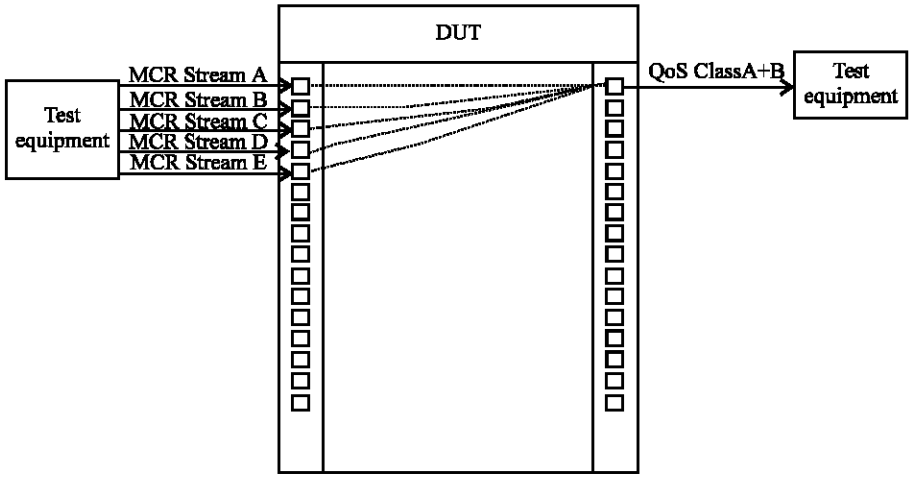

Fig. 8: MCR priority comparison test setup

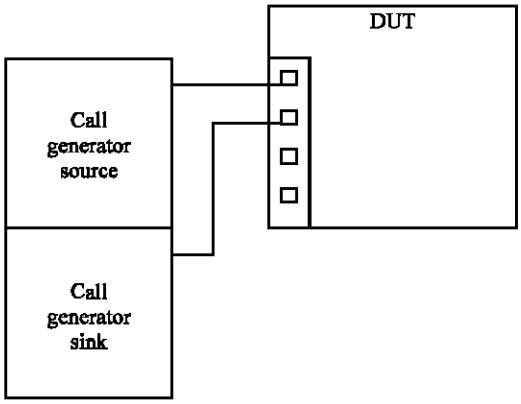

Fig. 9: SVC performance intracard test setup

streams to ensure that they are honoring their traffic contracts. The most common measurement of CDV is what is referred to as peak-to-peak CDV. Peak-to-peak CDV is simply the delta between CTD_{max} and CTD_{min} within the same media stream. Another definition for CDV is the inter-cell delay variation which measures differences against the nominal inter-cell spacing within the same media stream[8,9]. CDV can have a drastic effect on the QoS of certain applications especially real-time ones, like video conferencing or audio[10].

When testing the performance of a DUT, CDV will be measured using one of the mentioned definitions similar to the way the CTD is measured.

Priority/fairness: Priority can be described as the relationship between two or more streams of traffic as they vie for system resources (such as shared queue capacity and output port servicing) between QoS classes. In ATM, for example, the comparisons from a priority perspective deal with the relationship between QoS classes (CBR, VBR-RT, VBR-NRT, UBR, ABR) as shown in Table 2. Different priority schemes are used by different vendors. These schemes include: Strict priority, weighted round robin, weighted fair queuing, etc.

Priority and fairness as it pertains to the DUT or any other switch can be quantified in terms of not only throughput, but also latency. For the setups shown in Fig. 7 and 8 the impact of one QoS class on another in terms of CTD and CDV should also be measured.

Operational queue depth validation: Depending on the architecture of the DUT, it might have a shared ingress or egress buffer pool available to all QoS classes of size m Kcells, which might be shared equally amongst all cards and then further subdivided per QoS classes which are given different priorities. It is important to understand the relationship of buffering to both QoS class as well as how fairly the buffer pool is shared between configured circuits, that is per VC queuing.

Per QoS class: One way to validate the Queue depth per a QoS class is by sending a traffic stream at the highest priority destined for a given output port at 100% of line rate (assuming that a port can achieve 100% without queuing). A burst of cells at lower or equal priority can be sent to the same port. Since the input to the queue and the servicing of that queue are equal, then finding the maximum burst that can be achieved before cells are discarded is the effective queue depth of that port for the QoS class being tested.

Virtual per VC Queue depth as a function of provisioned circuits: For architectures that employ a buffer pooling scheme, it is important to verify that the buffer pool is

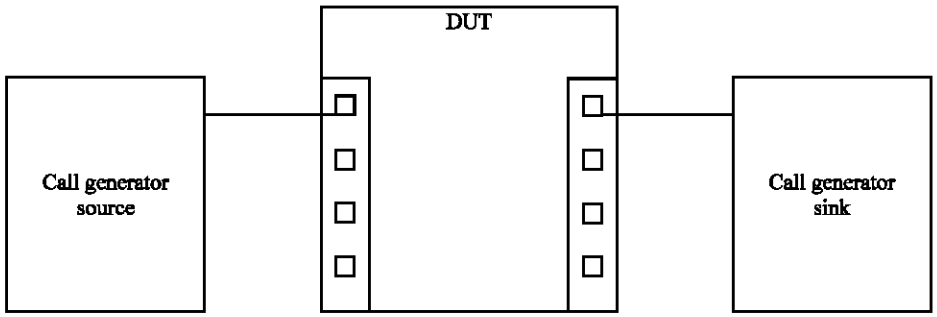

Fig. 10: SVC performance intercard test setup

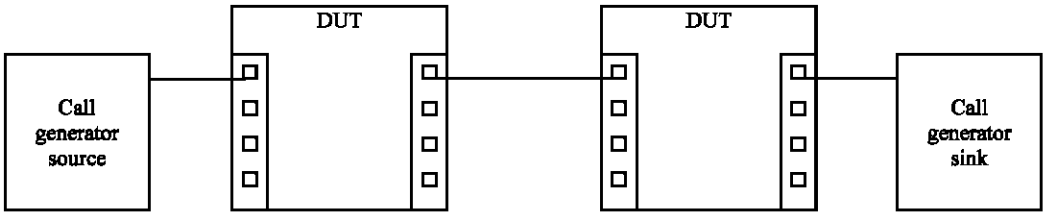

Fig. 11: SVC performance interswitch test setup

being shared equally between streams. To accomplish this, a traffic stream should be sent at the highest priority destined for a given output port at 100% of line rate (assuming that a port can achieve 100% without queuing). Two or more bursts of cells at lower or equal priority can be sent to the same port. Since the input to the queue and the servicing of that queue are equal, then finding the maximum burst that can be achieved before cells are discarded is the effective queue depth of that port for the QoS class being tested. If the number of streams is n, then the overall queue depth should be split equally between the lower priority streams.

Switched Virtual Circuit (SVC) performance: Switched Virtual Circuits are similar in structure to PVCs, but SVCs are provisioned on demand by the Customer's Premise Equipment (CPE). The CPE signals the ATM cell relay network to set up and tear down logical connections. The network will respond to the signals by provisioning a virtual connection across the network based on quality of service parameters requested, provided that sufficient network resources are available to establish the connection. SVCs utilize Constant Bit Rate, Variable Bit Rate or Unspecified Bit Rate.

There are a number of different methods of measuring SVC performance. We will cover some of the more typically used metrics. An important note to make is that performance is usually measured to determine a maximum rate that the DUT can handle before processing overload conditions which would elicit such additional work as call rejections.

For each test setup shown in Fig. 9 through 11, a number of different call profiles should be run. Each profile will look at a different aspect of call loading and call latency that will be important to how network designers layout their networks. Below we specify and detail these different call profiles and different measurements methods as they apply.

Burst setup rate: This is the measurement of the maximum call setup arrival rate the DUT can handle. For this test, the call duration is set to infinite (i.e. greater than the test duration). Different sized burst should be used and the maximum burst rate for each setup burst size should be measured. A successful test run is considered when all of the connect messages are received back at the calling test equipment within the prescribed test interval with no cause codes issued.

Method 1:
Avg. Burst Setup Rate (burst avg.) is: $\frac{n}{t_{connect\_n} - t_{setup\_1}}$

where n = the number of calls in the burst, $t_{connect\_n}$ = the time of the last connect and t_{setup_1} = the time of the first setup

Method 2:
Avg. Burst Setup Rate (call avg.) is: $\frac{n}{\sum a_1 a_2 a_3}$, where a

is the setup to connect time for a given call and n is the number of calls specified.

Burst teardown rate: This is the measurement of the maximum call release arrival rate the DUT can handle. This test begins with a fixed number of calls setup. For that fixed number of calls. A burst of release messages should be sent from the test equipment to the DUT. This test should be repeated varying the amount of calls to be torn down. The maximum burst teardown rate for each burst size should be measured. A successful test run is considered when all of the release complete messages are received back at the calling test equipment within the prescribed test interval with no cause codes issued.

Method 1: Avg. Burst Teardown Rate (burst avg.) is: $\frac{n}{t_{release-complete\_n}-t_{release\_1}}$, where n = the number of calls in the burst, $t_{release\_complete\_n}$ = the time of the last release complete and $t_{release\_1}$ = the time of the first release

Method 2: Avg. Burst Teardown Rate (call avg.) is: $\frac{n}{\sum a_1 a_2 \ldots\ldots a_n}$, where a is the release-to-release complete time for a given call and n is the number of calls specified.

Per-call setup latency: This is either the average or the explicit measured setup to connect time for a given call. It is important to note that due to batched processing of messages, the average call setup latency within a burst can often be lower than the average setup to connect latency of a given call. Several different measurement techniques can be employed to better understand the latency behavior. Three common methods are as follows:

Method 1: Avg. Call Setup Latency within a Burst is: $\frac{t_{connect\_n}-t_{setup\_1}}{n}$, where n = the number of calls in the burst, $t_{connect\_n}$ = the time of the last connect and t_{setup_1} = the time of the first setup

Method 2: Avg. Call Setup Latency is: $\frac{\sum a_1 a_2 \ldots\ldots a_n}{n}$, where a is the setup to connect time for a given call and n is the number of calls specified.

Method 3: Avg. Call Setup Latency is: $\frac{\sum a_1 a_2 \ldots\ldots a_n}{n}$, where a is the average setup to connect time within a sample period and n is the number of samples in a test run.

Per-call teardown latency: As with per-call setup latency, per-call teardown latency is the measured release-to-release complete time for a given call. It is important to note that due to batched processing of messages, the average call release latency within a burst can often be lower than the average release-to-release complete latency of a given call. The difference is as follows:

Method 1: Avg. Call Release Latency within a Burst is: $\frac{n}{t_{release-complete\_n}-t_{release\_1}}$ where n = the number of calls in the burst, $t_{release\_complete\_n}$ = the time of the last release complete and $t_{release\_1}$ = the time of the first release.

Method 2: Avg. Call Release Latency is: $\frac{\sum a_1 a_2 \ldots\ldots a_n}{n}$, where a is the release-to-release complete time for a given call and n is the number of calls specified.

Per-call setup to disconnect latency: This latency is measured as the time period from the time a setup message is sent to the time the test equipment sees a release complete message with a zero duration call.

Maximum continuous loading rate: This test consists of discovering the maximum zero call duration rate a switch can run at. This test should be run overnight (at least 8 h) to ensure that no memory leaks or other resource impairment exist that would affect the system performance.

Maximum loading with five-minute call durations: Five minute call durations are a metric that is commonly used to benchmark calls in legacy voice networks. This test should be run overnight (at least 8 h) to ensure that no memory leaks or other resource impairment exist that would affect the system performance.

To illustrate the performance of a DUT, consider Fig. 12a which shows a theoretical example of what would be typically seen if the DUT had the ability to queue calls that were not processed due to lack of resources. In our example plot, this behavior is indicative of a system that could handle the offered rate of 1000 setups/sec for up to 3 sec. This says that given our terminal rate (rate at which a DUT can process setups up to its capacity) of 680 setups/sec, we can assume at a 3 sec burst we are queuing: O_p-T_p

Where O_p is equal to the number of offered calls in the period and T_p is equal to the terminal rate performance for that same period. This is not an absolute measure of calls queued since different DUT signaling subsystem architectures will process calls differently.

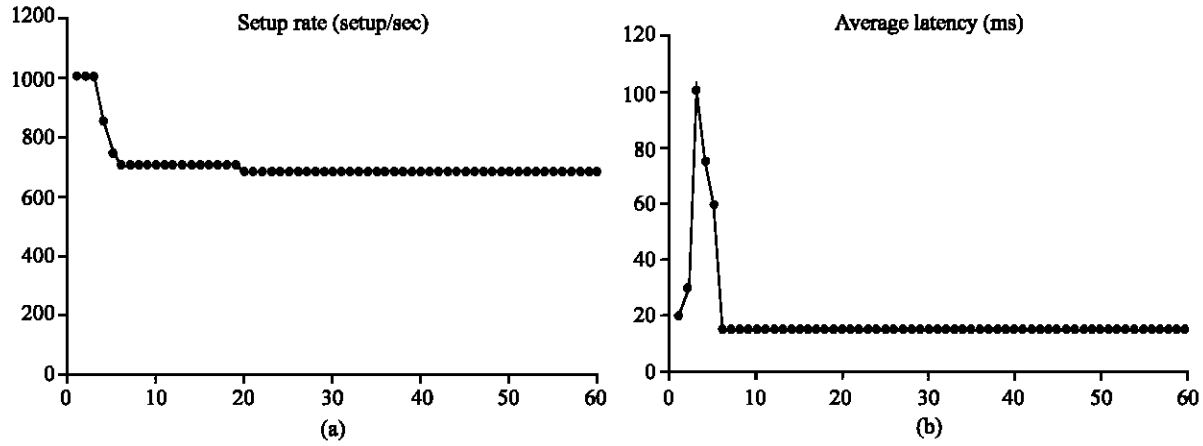

Fig. 12: Typical call setup rate performance graph

The queuing of these calls is reflected in Fig. 12b. Note that the average call latency is the average latency for a given call and that due to concurrent processing; more than one call is processed at a time in most DUTs.

Signaling (UNI) code path as bottleneck: User to Network Interface (UNI) provides the signaling mechanisms for dynamically establishing, maintaining and clearing ATM connections at the ATM User-Network Interface[11].

The test configurations shown in Fig. 9 through 11 will focus on the UNI signaling performance in three basic configurations. Intracard; where the calls are being sourced and sinked on the same I/O processor card. Intercard; where the calls are being sourced and sinked on different I/O processor cards. And Interswitch; where the calls are being sourced and sinked on different I/O processor cards on different switches with the added dimension of PNNI links connecting between these switches. However, in third case the PNNI link or Links between the two switches must be able to handle the SVC testing easily without being overloaded so that they do not act as the bottleneck. The limits obtained in the tests specified in the previous sections are due to the UNI code path limitations. In other words the limitations of the signaling path used for SVCs.

Private Network to Network Interface (PNNI): PNNI[7] is a routing protocol that is link state based and used in ATM networks. The function of PNNI routing protocols include the discovery of neighbors and link status, database sychnchronization, construction of routing hierarchy and flooding of PNNI topology state elements (PTSEs). In PNNI reachability information consists of addresses and address prefixes which describe the destinations to which calls might be routed. Topology updates are advertised using PTSEs in the PNNI routing domain.

PNNI performance can best be broken down into two major components; PNNI signaling and PNNI routing.

PNNI signaling: PNNI signaling is simply the internode compliment to UNI signaling that allows user (outside of the DUT) initiated calls to setup through and between intermediate nodes. Measuring PNNI call setup performance is most effectively measured indirectly by setting up UNI calls over the PNNI links and measuring such things as:

- Call setup rate
- Call teardown rate
- Sustainable call rate
- Call crankback/reroute rate

Further dimensions can be given to this testing such as varying number of hops between the two end points, topology (such as inter/intra peer group, hierarchy, degree of connectivity, etc.) and number of calling/called endpoints

Figure 13 shows that an increased number of I/O processors originating and terminating the calls over the same link as compared with Fig. 11. This will allow the cards hosting the PNNI link to act as the bottleneck so that the effective call setups limitations of the PNNI link can be determined.

PNNI routing: Since PNNI routing is a topic that has network wide ramifications and is best discussed in that context.

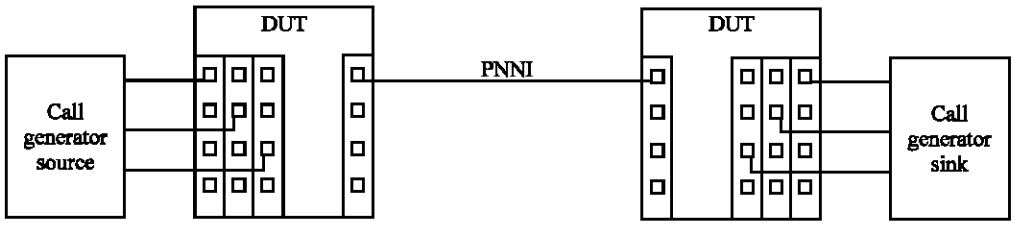

Fig. 13: SVC performance interswitch test setup, PNNI link as the bottleneck

Network performance metrics: There are several metrics that one can consider when evaluating the performance of a network. In this paper we will only consider several of them, based on what we think most important. These metrics are explained below in detail.

Network size and hierarchy: The performance metric here is the balance between the size, load and hierarchy of a network. There are several network design considerations that one must consider when building a network or expanding on an existing one. Internet Service Providers with large networks or fast growing networks in size and customer base are more vulnerable to network failure due to lack of proper design of their networks. The proper design of a network is the key element to preventing network failures and to control and limit the consequences of any failures once it happens. Once a failure happens, network convergence and recovery time becomes very crucial especially when the network failure might have a great financial impact on both the ISPs and their customers. There is always a compromise between the network operational and maintenance (OAM) cost and the complexity of the design of a network. The more complex a network, the greater its OAM cost.

Large networks running real-time applications can be vulnerable to network failures which can be driven by an individual component in the network or as a result of a propagation of certain events. Once a failure happen in a large network, its impact is usually spread very fast and some times might causes a network meltdown. The key to limiting a network failure is to properly design a hierarchal network where failure events can be localized and maintained easily. For example, in networks using link state routing protocols such as a flat large network running OSPF[12] as the IGP, if an OSPF related failure event occurs, all nodes in the flat network would be affected. On the other hand, if the network was segregated into OSPF areas, the effect of the failure event might be localized within its area with minimal impact on nodes in the other areas. A similar analogy can be stated for an ATM network running PNNI. The effect of a network failure might materialize in many ways; some examples would be as follow:

- A noticeable increase in the CPU load of the device experiencing the failure and other devices in the network
- Partial service disruption
- A total network melt down

When designing a network a set of what if conditions must be considered and tested on a similar emulated network prior to deployment. This set of what if conditions must be tailored toward the applications, technology and software features implemented in the network. The set of what if conditions might include but no limited to the following:

- A fiber cut
- A line card or central processor failure
- A sudden bandwidth change over multiple links that might cause generation of control packets
- A node failure
- Simultaneous Multiple node failures
- An addition of a new link or trunk
- The creation of a new hierarchal level, such as the creation of a new OSPF area or a new hierarchal PNNI level.

Network redundancy: This network performance metric is a measure of how redundant the network is. Network redundancy can be accomplished in many ways; we will only consider some of them.

- Re-route around link or card failures, that is alternate path redundancy: when a fiber cut, hardware failure or other type of catastrophic event occurs, the ability of a network to route around such a failure is one of the most important weapons in the fault tolerant network arsenal. Generally speaking, from an OSI perspective the closer you are to layer one, the quicker the restorable time will be. For instance layer 1 APS will generally be much faster than a layer 2 crank back which would generally be much faster than waiting for a layer 4 restorable via TCP. Internal to the DUT, there should be a number of fault tolerant mechanisms to ensure that a fault never

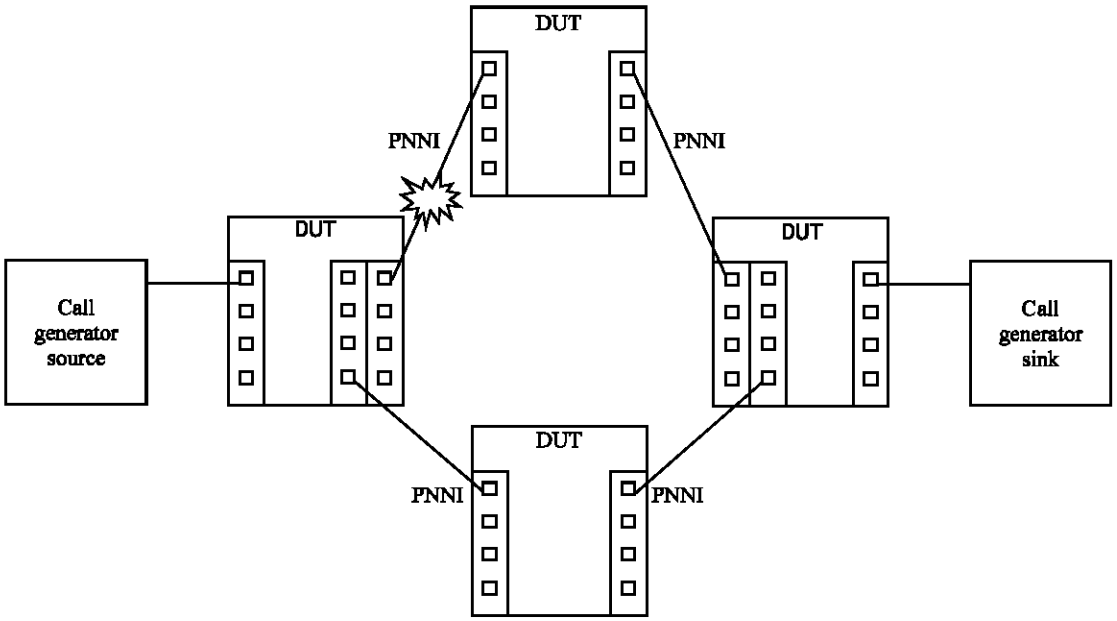

Fig. 14: Alternate path vc restoration time configuration

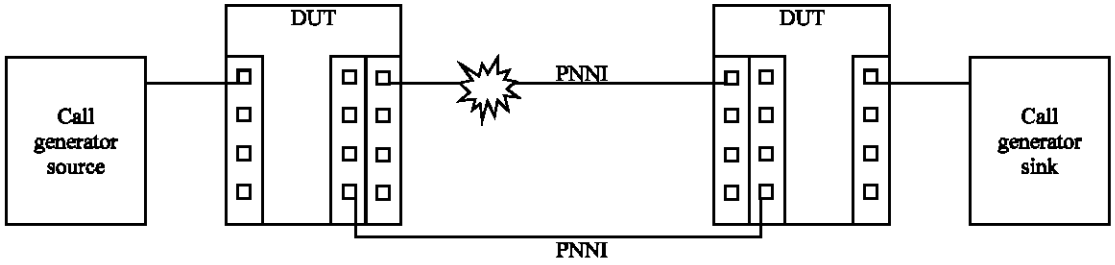

Fig. 15: Alternate path vc restoration time alternative configuration

makes it to the physical (layer 1) or link layer (layer 2) in the first place. The Fig. 14 and 15 show two possible link and card redundancy mechanisms to re-route traffic flowing via VCs over PNNI trunks. Notice that redundant links or PNNI trunks are configured on different cards and not on the same card to account for card failure as well as a fiber cut or link failure.

A measure of the network performance in this case would the time it takes for traffic to re-route over failed paths and the rate at which VCs are being re-routed at.

- Another network redundancy issue is the central processor redundancy. In this case, the use of one or more redundant central processor on a particular switch or router is essential to maintain stability in the network. However, what is more important in this case is the used mechanism for a switch over from an active to a redundant central processor. The switch over must be seamless to other network devices and to applications running on the device. There should be minimal if not no data loss or loss of connectivity due to such a switch over. If the switch over is slow and the network is somewhat large, the switch over might triggers a chain failure events that might cause a network melt down. A central control processor switchover test should be run to determine if there is any service or routing disruption.
- When the network topology is hierarchal, hierarchal redundancy is needed to avoid the creation of any blackout areas. A black out area is a network segment that becomes totally isolated from the rest of the network upon a certain failure. For example, in Fig. 16, if node A.5.1 was down, Peer Group (A,5), PG(A,5), will be isolated from the rest of the network since the node A.5.1 is the only gateway for PG(A,5). However, PG(A,1) has two gateways through node A.1.1 and node A.1.2.
- In certain cases, redundancy can be overdone to a point where it might actually work against the stability of the network. We refer to this as negative

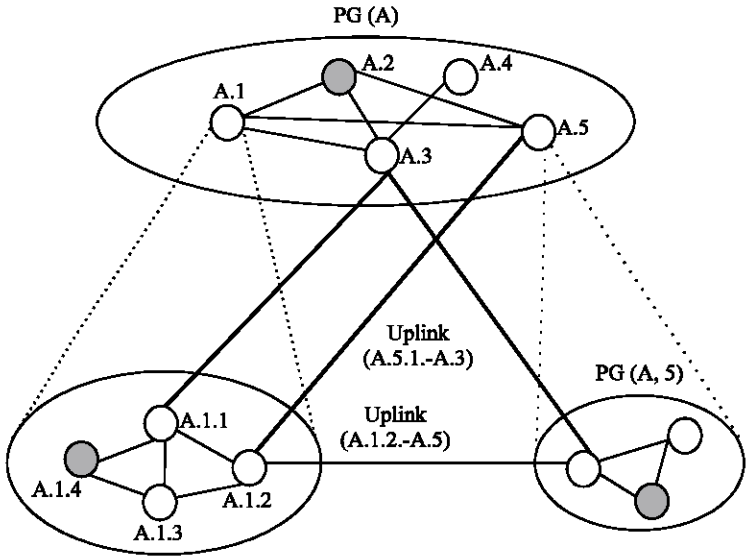

Fig. 16: A two level hierarchal PNNI topology

redundancy. For example, if a network is running a link state protocol as OSPF or PNNI consideration must be taken to limit the number of adjacencies a node might have. This is the case, since such link state protocols are reliable in nature and flood any generated control messages such as LSAs or PTSEs over all links except those received on. The more adjacencies a node has the more CPU cycles would be consumed in processing received control packets. In the case of a network failure event in the network, nodes with high number of adjacencies will become the candidates to fail in the network. Therefore, a network designer must not over do link redundancy and a balance between redundancy, network stability and traffic distribution must be struck.

Number of adjacencies and link state size: To investigate the effect of increased number of adjacencies and the size of a link state on the network stability and performance several tests should be conducted as follow:

- A test should run to measure the time it takes to reboot a node when varying the number of PNNI or OSPF links from 1 to n (where n = the maximum number of configurable PNNI/OSPF adjacencies) on a given switch while using an emulated topology to obtain a large link state database.
- A test should be done by fixing the number of adjacencies on a node to the maximum desired number and increasing the emulated network size by gradually increasing the link state database size and then measuring the reboot time of the node under test.

A test suite should be conducted to see the relation between the node central processor CPU load, the number of adjacencies on the node and the emulated link state data base size. In the case of PNNI, adjacencies might be lost due to excessive delay of PNNI link hello messages that results from excessive load on the processor handling them causing network instability.

Network convergence time: The network convergence time is one of the most critical performance parameter of a network. Upon a network failure event, it is desired that the network go back to a steady operational state. The network convergence time depends on many factors and a network designer must take all of these factors into consideration when building or expanding a network. These factors include the number of nodes in a network, the hierarchy and topology of the network, the maximum number of a adjacencies on a node in the network, how powerful are the used CPUs on the cards, the total number of trunks in a network, the maximum number of hops between any nodes in the network, applications running in the network, maximum number of cards and their types configured on a particular node in the network, the routing protocol or protocols used in the network and several other factors as well. The shorter the convergence time the more stable is the network. A set of what if conditions, as in section 5, should be created and the convergence time should be measured for each one of them for a network operator to have a sense of how stable a network is. These measurements when conducted in the lab, they should be conducted on an emulated network as similar as possible to the real-life network.

Network performance tools: To facilitate testing of a DUT or a network, one must have the needed proper tools with the desired capabilities. In addition to available test equipments, vendors of network devices might have to develop their own tools especially when some of the features offered are proprietary and could only be tested with in house developed tools. Below are the two types of tools a vendor will need to have to perform a comprehensive test suite on the DUT or the network under test.

Emulation tools: Various emulation tools will be required to emulate heavy switch/network load. Some of the emulation needs can be addressed by test equipments. Other areas will be best served by in house tailored tools to better serve any proprietary details specific to the vendor. The following are the types of emulation tools that will likely be required:

- SVC call generator
- Network protocol analyzer
- SPVC creation automation
- PTSE generator for PNNI and LSA generator for OSPF
- Data plane traffic generators/analyzers
- Node spoofing tool. i.e. turning a switch into an entire network with each port on an I/O processor card emulating a full switch to facilitate PTSE flooding scenarios which will help determine how many links can be supported.

Modeling tools

- Memory calculator to calculate the memory consumption of various parameters on the DUT cards to help understand real world capacity which will help Service Providers design large networks.
- CPU calculator to account for steady state and spurious loading on any given switch in a large network. One of the more important things to do with this is to identify heavily loaded switches in the carrier network that would represent the weakest links and to calculate load under various what if scenarios. This will allow a Service Provider to make real cost/risk trade-offs and mitigate any risks that they would consider too great.

The breadth and depth of the various metrics to be explored whether testing, emulating or simulating complex network elements or constellations of network elements are staggering. The intent of this paper was to highlight several key areas based on years of experience in building and deploying multi-service network elements. While it was necessary for the scope of this paper to focus on ATM, the techniques and general methodology are wholly transferable to other protocol suites. The topics covered should serve as a solid foundation to build a performance engineering program in either a carrier or a vendor engineering environment. This focus on performance has proven to be worth its weight in gold for organizations who have embraced it to effectively define the performance envelope for networks and network elements alike.

REFERENCES

1. Anonymous, 1999. AF-TEST-TM-0131.000, ATM Forum Performance Testing Specification. ATM Forum Technical Committee.
2. Anonymous, 1993. ITU-T I.356, B-ISDN ATM Layer Cell Transfer Performance. Telecommunication Standardization Sector of ITU.
3. McDysan, D. and D. Spohn, 1999. ATM Theory and Applications. McGraw-Hill.
4. Handel, R., M. Huber and S. Schroder, 1995. ATM Networks, Addison-Wesley.
5. Halsall, F., 1992. Data Communications, Computer Networks and Protocols, Addison-Wesley.
6. Anonymous, 2002. EANTC, Light Reading ATM/MPLS Multiservice Switch Performance Test Draft Test Plan Version 1.1/sEuropean Advanced Networking Test Center.
7. Anonymous, 1997. ITU G.783 specifications.
8. Borden, M., 1995. Properties of CDV and its accumulation. ATM Forum Technical Committee Contribution.
9. Shuaib, K., S. Nawrot, H. Esayed, M. Lee and T. Saadawi, 1998. Empirical evaluation of ATM CBR traffic transported under diverse operating conditions. Proc. ISCC conference, Athens, Greece.
10. Shuaib, K., T. Saadawil, M. Lee and B. Basch, 2000. Dejittering in the transport of MPEG-2 and MPEG-4 video, Multimedia Systems, 8: 231-239.
11. Anonymous, 1996. ATM User-Network Interface (UNI) Signalling Specification Version 4.0 af-sig-0061.000.
12. Moy, J., 1998. OSPF Version 2 IETF RFC 2328.