



Inspired Innovation

Impairment White Paper

How to Validate Network Applications Before you Deploy

August 2007

Spirent Communications, Inc.

1325 Borregas Avenue
Sunnyvale, CA 94089 USA

Email: sales-spirent@spirent.com

Web: <http://www.spirent.com>

Americas

T: +1 800.SPIRENT

+818 676.2683

Europe, Middle East, Africa

T: +33 1 6137.2250

Asia Pacific

T: +852 2511.3822

Copyright

© 2007 Spirent Communications, Inc. All Rights Reserved.

All of the company names and/or brand names and/or product names referred to in this document, in particular, the name “Spirent” and its logo device, are either registered trademarks or trademarks of Spirent plc and its subsidiaries, pending registration in accordance with relevant national laws. All other registered trademarks or trademarks are the property of their respective owners.

The information contained in this document is subject to change without notice and does not represent a commitment on the part of Spirent Communications. The information in this document is believed to be accurate and reliable; however, Spirent Communications assumes no responsibility or liability for any errors or inaccuracies that may appear in the document.

How to Validate Network Applications Before you Deploy

Contents

Overview	1
Risks and Rewards.....	2
"How To" Methodology.....	3
Best Practice for Network Testing with WAN Emulation.....	5
Server Migration Case Study.....	7
Delay	7
Jitter	9
Fiber Cut/Network Outage.....	11
Packet Drop	12
Frame Duplication and Reorder.....	14
Combined Impairments	14
Multi-Protocols.....	15
Summary	16

Overview

If your company is like most organizations in today's network-centric world, you need to ensure that your applications run properly over private and public Wide Area Networks (WANs). This is particularly challenging for applications sensitive to network conditions, such as real time transactions, enterprise storage, streaming media, IPTV and VoIP.

The only way to have confidence that a given application, solution or service will operate with acceptable performance when deployed is to test it under expected – and maybe unexpected – production network conditions prior to deployment. Latency, bandwidth limitations, bit errors, dropped packets and packet jitter all impact application performance. Fortunately, it is possible to know how your application functions in the presence of these actual and unfavorable conditions by using an emulated network environment.

This environment can be created in a lab using a network emulator. Network emulators are hardware devices that duplicate network conditions under user control, allowing you to precisely control the addition of errors, delay and other impairments to network traffic. With their ability to vary delay times and emulate Quality of Service (QoS) metrics, network emulators are superior to the alternative of using spools of fiber to characterize long-haul network behavior. With their accuracy and ability to function at full line rate, hardware-based network emulators outperform software simulation tools for validating network applications.

When using a network emulator, however, knowing the appropriate combinations and variations of delays and impairments to emulate is not easy. This white paper is a “how to” guide for getting started. The goal is to equip network managers with the knowledge they need to run their critical networked applications in an “emulated” environment in order to establish, maintain and optimize the performance and availability of these solutions.

Risks and Rewards

The risks of network conditions causing serious problems in networked application performance are very real. The following are actual examples:

- A major bank running a storage extension solution using Fibre Channel over SDH experiences Fibre Channel buffer credit starvation. This limits system throughput and greatly increases the time it takes to perform a system backup. Bit errors, packet drops and other types of impairments increase retransmissions required, which also decreases throughput performance.
- A large insurance company finds an auto-negotiation problem between two different vendors' director class switches while setting up a new converged network. The problem is caused by excessive jitter from one of the devices and leads to the product's redesign by the vendor. The problem is fixed but it delays deployment of the new network by three months.
- A global information service company suffers an unexpected outage of its main system. When their back-up system takes over operations, the company experiences significant performance issues caused by the distance between their backup system and central data operation. Their service is not interrupted, but the experience identifies critical flaws with their current disaster recovery solution.

“How To” Methodology

With today’s network emulation tools, it is possible to validate network applications in a lab environment by replicating the actual real world network. This network replication consists of three items:

- I. An emulated Wide Area Network
- II. Actual applications (or application traffic generator), and
- III. Actual network endpoint elements (servers, switches, routers, storage arrays, mobile devices, etc.)

To emulate the network, it is important to have as much information about the final production environment as possible. The closer the emulated network represents the production network, the better. Good starting points are: knowing the distance of the network (and delays associated with this distance), and understanding the minimum guaranteed Quality of Service (QoS) values of the network Service Level Agreement (SLA). For example, this includes knowing the guaranteed bandwidth, maximum delay and bit error rate (BER) of the production network. These values should represent the base line test.

Determining the expected bandwidth usage from your application, the number of users, and the amount of other traffic on the network are also important. Network load significantly affects application performance. This load factor will include not only how much the network is being utilized, but also how many clients are using your particular application.

The baseline test should include running your application at its expected bandwidth with the expected number of clients. The emulated network should also be loaded to reflect a production network scenario although this may not always be possible. In such cases, there are ways to minimize the risk:

- Test with a smaller available bandwidth so the ratio between bandwidth utilization to available bandwidth is equal to the expected ratio with a realistically loaded system.
- Characterize performance under higher impairment conditions to verify that performance exceeds expectations under less than guaranteed service levels, thereby giving more confidence the application will still have acceptable performance if bandwidth utilization is heavier.

From this baseline, a risk assessment is conducted to characterize performance of the service or solution under worsening conditions (e.g., if performance drops significantly just beyond the minimum guaranteed service level then this could be deemed high risk). More importantly, this approach also determines the minimum required service level to achieve sufficient application performance, reducing the possibility of over-engineering a solution.

A robust characterization of application performance will include various “what if?” analyses covering key network impairment parameters. Corner cases should be explored, and “break points” where application performance becomes unacceptably poor should be identified in a proper validation test process.

Interoperability is an important criterion in any network emulation test. In many cases, inconsistencies in auto-negotiation, jitter tolerance and queue prioritization between different servers and switches will wreak havoc on network applications. A correct test methodology is to use actual network devices with a WAN emulator.

Best Practice for Network Testing with WAN Emulation

1. Gain an understanding of the expected network environment to increase the relevancy of the testing. This includes:
 - Network QoS parameters
 - Bandwidth usage from your application or solution, and
 - Realistic loading on the system under test
2. Perform a baseline test under optimal conditions – e.g., minimum expected delay and BER, and largest available bandwidth.
3. Characterize performance beyond the optimal guaranteed network conditions – this provides information for assessing risk.
4. Employ actual network endpoint devices.

So how do you start the process of validating your network application performance prior to deployment? At Spirent, we recommend a lab test running your actual application using a hardware-based network emulator to create typical delays and impairments. The network emulator should be capable of running at full line rate at the bandwidth of your network without introducing unwanted impairments. It should also be precisely and dynamically controllable for delays and impairments to be easily increased and decreased in small increments during testing.

We recommend emulating impairments at layers one, two and three of the OSI stack, for the following types of impairments:

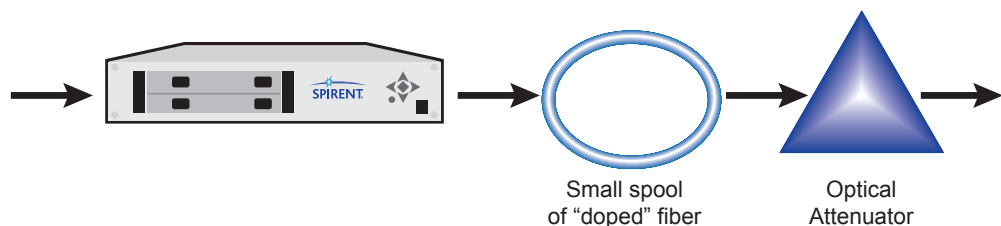
- Delay
- Bit Errors
- Packet Drop
- Packet Jitter
- Fiber Cut
- Packet Reorder
- Frame duplication
- Bandwidth Clamping
- Congestion Control

"With 20ms of latency our applications were straining ... at 80ms, some applications stopped working entirely"

While there is a role for emulating higher layer (layers four through seven) impairments, it is of a secondary concern. The reason for this is higher layer errors *result* from lower layer impairments. This is because higher layer data (such as transport, session, presentation and application information) is carried by the lower layers. Therefore, impairing a Layer Two Ethernet frame will impact the TCP/IP signal, and, more importantly, the end-user application.

For example, consider what happens when there is excessive delay or packet drop. When delay occurs, the source does not receive an acknowledgement in a timely manner, which causes the TCP window size to shrink. In the case of errors, the TCP/IP protocol, which uses checksums to detect errors and validate data integrity, will drop packets if the checksums do not match. This also shrinks the TCP window size, in effect turning your network “fire hose” into a “straw.” If the TCP window size shrinks to zero, the application is halted.

Also, before moving forward, it is worth mentioning that there is another very different type of distance emulation beyond the focus of this paper. This emulation impairs the optical characteristics of light signals traveling through fiber by creating polarization mode dispersion (PMD) and attenuation to mimic the inherent effects of light pulses propagating through optical media. Of course, these optical impairments can be emulated and tested separately or in conjunction with the impairments described here. To conduct them at the same time, we recommend placing a network emulator in serial with a 10-15km spool of doped fiber (to create dispersion) and a low cost adjustable attenuator as shown in the figure below. For more sophisticated PMD emulation that emulates specific types of fiber, a specialized PMD emulator will be needed.



Network emulation with optical impairments

Server Migration Case Study

A leading information technology services provider began a major project to move its client's mainframe systems to a central service center 1,000 miles away. But before they could perform the migration it was critical to test the impact of such a distance on mainframe applications. Using network emulators from Spirent Communications, the team introduced delay into the live production environment between the users and the mainframe, exactly emulating the setup they wanted to implement with the proposed network. Transmission latency times of up to 80ms were emulated. Other impairments included bandwidth throttling down to 40Mbps, reordering and corrupting Ethernet frames, and introducing packet jitter. All of these tests were designed to match impairments that could occur on the deployed network.

The initial test showed that the client's critical business applications – as well as ODBC (Open Database Connectivity) and JDBC (Java Database Connectivity) applications – did not work with the delay that the emulated distance introduced. With 20ms of latency, applications were straining. Tasks that normally took 60 minutes to run took 100 minutes. At 80ms, some apps stopped working entirely. Fortunately, the team detected the failure in real time and dynamically eliminated the delay to avoid end user problems on the live network. The team then focused on tuning and tweaking these applications to operate with the several millisecond delay associated with the 1,000 mile distance. Once optimized, they retested the applications using the emulator and validated performance was acceptable to end users.

Delay

Transmission delay on a network due to distance and latency through network elements can be estimated using a few simple assumptions. You may recall from high school physics that light travels at 300,000 km/sec in a vacuum. In the real world, light moves through optical fiber at about two-thirds of this amount, or approximately 200km each millisecond. This assumption is appropriate for synchronous long-haul optical networks, such as those running SONET/SDH or DWDM.

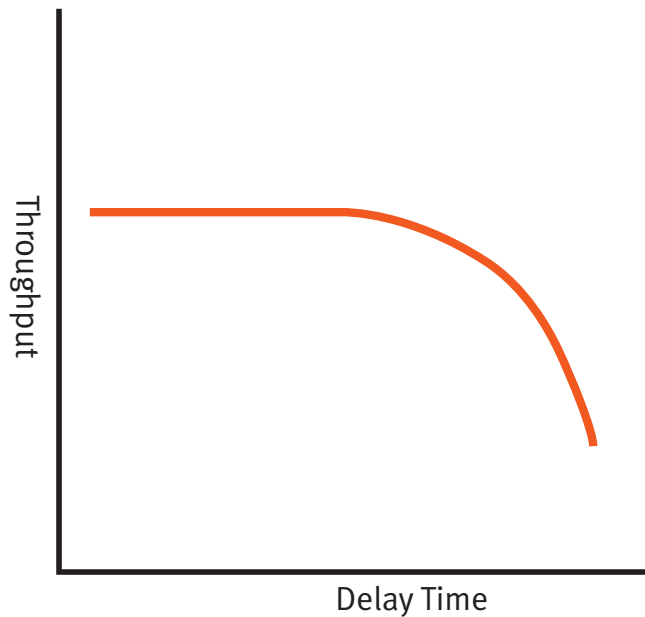
Next, you need to add the latency through switches, routers and other “end point” network elements outside of the core optical network ring. Among other items, this includes latency caused by:

- The clock recovery process
- FEC processes
- Encoding and decoding activities (such as for MPEG and secure transmissions)
- QoS prioritization, and
- Jitter buffer designed to absorb incoming packet jitter

This amount of delay can vary widely. In our experience, we have seen latencies range from under one millisecond through each switch or router to much higher amounts. For example, latency incurred by the FEC process alone can range up to 10ms and jitter buffer in some configurations runs as high as 100ms.

Adding these two delay amounts together – distance delay plus router/switch delay – gives an approximation for overall network delay. However, an even better method would be to measure actual throughput times if your network already exists. In pre-deployed “emulated” environments, this is not possible.

You will find that increasing the delay time reduces the throughput, as depicted in the figure below (throughput vs. delay time graph). At some level of delay, throughput will essentially drop to zero. The shape and slope of the line will vary depending on the actual application. Influencing parameters include packet size, packet density and protocol (Ethernet or Fibre Channel, for example).

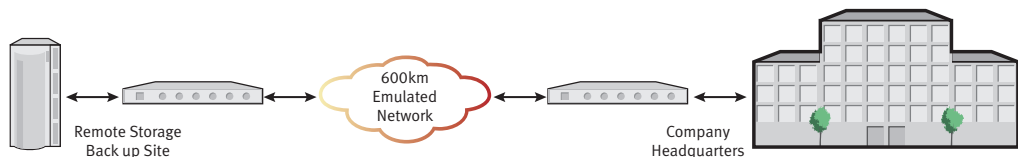


Throughput vs. Delay Time

The following example of an actual customer situation demonstrates how network delay times can be established for pre-deployment testing using the Spirent Communications Network Emulator:

A large European insurance company was conducting lab testing prior to setting up a remote storage site. The distance between primary and remote sites was estimated at 600km. Based on this, the company estimated data transmission delay in the fiber should be approximately 4ms (3ms based on speed of light in optics plus 1ms for repeaters). An additional 8ms was added to conservatively account for latency encountered in the switches and routers present in the network due to clock recovery and jitter buffer. (This data was calculated from the specs and configurations for each switch and router.) Therefore the total latency set in the network emulator was 12ms in each direction.

Next, an interesting thing happened. This company checked with its carrier, France Telecom, on the amount of latency “guaranteed” between the two central offices 600km apart. The service guarantee was only 22ms! Therefore, the network engineers increased the delay setting on the network emulator from 12ms up to 30ms, to cover 22ms from the carrier plus 8ms from their own switches and routers. In their testing, they gradually upped the delay amount to 40ms to safely validate that the backup storage application would perform acceptably under the service level guarantee.



Jitter

Network jitter, defined as variations in the time between packets, should not be confused with “optical” or “clock” jitter, which refers to deviations between successive high-frequency pulses of an optical signal. Network jitter is a regular occurrence caused by network congestion, timing drift, queuing, loading variability or routing changes. Jitter can have a deleterious effect on network applications, particularly multimedia Triple Play applications like IPTV, VoIP and streaming media. Jitter also impacts IP network throughput as described in the “land speed record” example.

To handle this inherent network jitter, the receiving elements are usually designed with a certain amount of “buffer” to absorb the incoming jitter, as described above as one of the “causes” of network latency. We have seen network receivers running

with jitter buffer “budgets” of up to 100ms to compensate for IP network jitter. Jitter absorption is a tricky process and needs to be tested thoroughly by your network receiving devices to ensure data is not modified unexpectedly during this process.

On long distance networks, jitter is also a major factor in limiting network throughput. Consider the University of Tokyo case, which recently set the Internet2 Land Speed Record¹ for the highest “bandwidth” times “distance” IP network transmission. In setting this record, the Tokyo team, led by Dr. Kei Hiraki, successfully transmitted 1485 gigabytes at an average rate of 7.99 Gbps over a distance of 30,000 kilometers using IPv4. (They also set the IPv6 record over the same distance.)

“What was most useful about the Spirent emulator was that it inserted a perfectly jitterless delay. This allowed us to determine whether a reduction in bandwidth was caused by round-trip time or by some other factors such as jitter somewhere in the network. As a result, we concluded that jitter on the actual network is a major factor in the reduction in efficiency in ultra long distance Internet communication. The precise and accurate delay inserted by the Spirent emulator was instrumental in identifying jitter as the cause.”

Dr. Kei Hiraki, University of Tokyo

In preliminary testing using a 10 gigabit Ethernet WAN PHY Network Emulator from Spirent Communications, the effects of jitter on network performance were judged to be significant even when compared to the roughly 500ms round-trip delay time.

Jitter impairment testing is a necessary ingredient in any network application validation plan because jitter affects data integrity, delay and throughput. Jitter testing is very straightforward with a network emulator, and is done by injecting controlled packet jitter into the line (often combined with other impairments such as bit errors). In the Spirent emulator, we create

Key Features to look for in an Impairment Emulator

- **Hardware-based for Precision and Accuracy**
 - Allows for impairments to be isolated for troubleshooting
 - Increases repeatability
- **True full line rate capable**
 - Data rates that match your current and future requirements
- **Programmable Architecture to protect your investment**
- **Easy to use – GUI, Scripting API and/or front panel control**
- **Integrated multi-protocol solutions**
 - SONET/SDH, Ethernet, Fibre Channel, OTN
- **Controlled Layer 1, 2 and 3+ impairments**
- **Dynamic control of impairments**
 - Essential for characterizing performance boundaries
- **Real-time statistics and alarm monitoring**

1. For more information see: <http://lsr.internet2.edu/>.

jitter by changing the delay between each Ethernet frame on a frame by frame basis. The net result is the gap between packets shrinks to any selected value between the original amount and the minimum 80 nanoseconds required by the Ethernet standard, or stretches to the maximum positive inter-packet gap increase specified by the user.

A question often raised is, “how much jitter do I emulate when doing jitter impairment testing?” This is best answered by checking with your network provider on the jitter level guaranteed for the service you are provided. This should also be stated in your Service Level Agreement (SLA). According to the TIA and ITU-T Test Profiles, a “well managed” network will have peak to peak jitter of less than 40ms. A “best effort” network will have jitter of less than 150ms, while an “unmanaged” network can have jitter of up to 500ms².

Fiber Cut/Network Outage

How do you simulate a total loss of signal (LOS) on your network such as a fiber cut or network outage? What is the affect on your application when a momentary signal loss occurs? How long can LOS exist before your application cannot recover?

These questions can and should be answered in the lab before deployment. This test is configured by changing the laser output off and then back on after a short amount of time. At a minimum, the fiber cut time period should be set to 50ms. This is because on a SONET/SDH ring, when a fiber is cut, the network must restore through the “protect” ring path within 50ms. The outage duration may be changed to longer periods depending on the application’s needs.

When you consider how much harm even a short network outage can have on your data, the importance of testing LOS becomes clear. As an example, let’s look at how much data is on a high speed network link at any given time. Since the speed of light in fiber is five nanoseconds per meter, at 10Gbps every meter of fiber holds 50 data bits. (Each bit is approximately two cm long). This means that it takes only 20km to hold an entire megabit of data! That is a boatload of data to have in flight for such a relatively short distance.

Viewed another way, consider the following: If we assume New York and San Francisco are 4000km apart, the amount of data that is "in flight" on a 10Gbps fiber optic data link between the two cities is at least 200 megabits (with a transmission time of 20ms). That’s enough data for 18,000 pages of text, approximately volumes A through M of the Encyclopedia Britannica (which is 26,000 pages long for the 2004 edition)!³

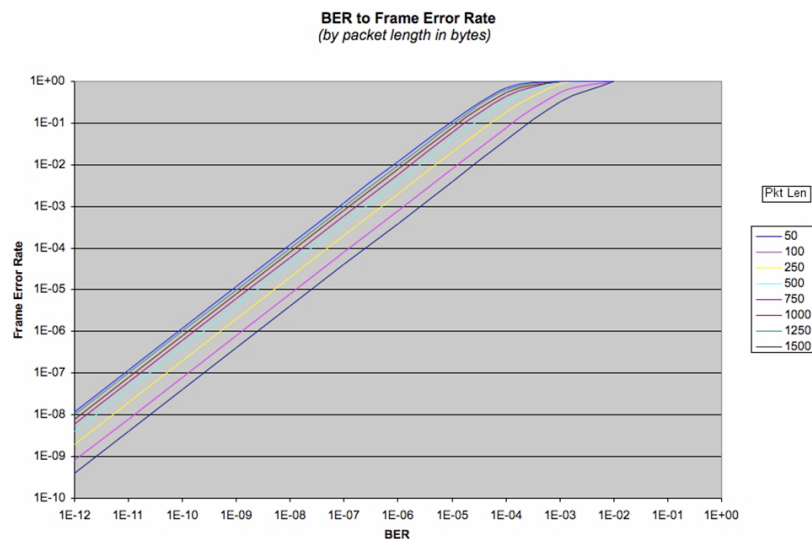
2. TIA 921 and ITU-T G.NIMM Test profiles based on QoS Classes, 2005.
3. As computed by Charles Webb, CTO Anue Systems Inc., 2005.

Service interruptions due to network delays and impairments are a recipe for disaster. For this reason, many companies are working to better understand the impact of delays and impairments on their application traffic throughputs. Service providers, in turn, are implementing more robust protection and restoration mechanisms. Bell Canada is a case in point. Bell has been able to represent extreme cases on their network with Spirent Network Emulators. Using emulators that handle data rates from 100Mbps to 10Gbps on the same platform, Bell emulates a range of delays and errors that represent their nationwide network. In addition, Bell Canada can identify and understand network weaknesses before customers experience issues.

Packet Drop

Packets are usually dropped due to two reasons: bit errors and congestion on the network. This packet loss, in turn, degrades throughput performance. Though some impairments described above, such as jitter and delay, will result in packets being dropped, successful pre-deployment testing also includes specifically dropping packets on both random and deterministic bases to assess the impact on network applications.

Emulating packet drops should start with injecting bit errors in the 10 bit domain into the network. Though the rate of bit error insertion can be set to match your specific SLA – stated in terms of packet or frame error rate (FER), bit error rate (BER) or error-free seconds (EFS) – a high quality link generally has a Bit Error Rate of 1×10^{-9} or lower. Bit errors can be distributed uniformly at first and then, to more accurately reflect real network behavior, should be distributed in a statistically “bursty” manner as with a Gaussian (i.e., normal) or Poisson distribution function. Assuming an average Ethernet frame size of 1500 bytes, the 1×10^{-9} BER translates to having one errored frame in approximately every 100K frames.



Frame Error Rate vs Bit Error Rate

Erroring a stream of bits in 8b/10b domain will create running disparity and code word errors, and this will create packet errors. On most network emulators, the length of errors is programmable from one bit to a large number like 64,000 bits. By varying the length of the error stream, one can stress physical layer components like the clock recovery circuits and the Ethernet framing logic.

Next, CRC corruptions that can result from data corruption should be created. Again, a setup of one in 100K Ethernet frames is a good test setting to emulate an extremely high quality network. Finally, straight packet loss should be emulated by dropping one in 100K Ethernet frames.

The amount of packet loss can then be increased to stress your network application. How high to go? It depends on your application and your network. Real time voice and video applications (such as IPTV, Video on Demand and VoIP) are more sensitive to packet loss than most data applications. Storage and financial transaction apps are more sensitive than e-mails.

In terms of real world networks, even “well managed” networks can experience random packet loss of as high as one in every 2000 packets⁴. “Ordinary” Web traffic is far worse. Measurements of *average* packet loss on the web range from 1.28%⁵ to 3%⁶. *Maximum* packet loss measurements and measurements outside of North America are even higher.

As with the BER settings, these packet corruption and loss impairments will more accurately mirror reality if injected according to a random or normal distribution function instead of a uniform or periodic distribution. Most decent network emulators allow such distribution settings on their impairments.

To test system robustness, data transfers should be initiated *while* these bit errors, CRC corruptions and packet drops functions are being applied. Impairments should be run for at least 30 minutes (or the duration of the transfer, whichever is longer). At the completion of the transfer, the data integrity should be checked at the receiving location to ensure no errors were introduced during the transmission.

If your network application is typical, you will see significant degradation of throughput as packets are dropped. The level of throughput decline will depend on your particular application and higher layer protocols. In one study of the file transfer protocol, packet loss of only 3% caused throughput to drop by 50%⁷.

4. TIA 921 and ITU-T G.NIMM Test Profiles based on QoS Classes, 2005.

5. Xaffire (formerly Matrix NetSystems), measured in July 2003.

6. Internet Traffic Report (internettrafficreport.com) North America, April 2005.

7. NASA study on the network impact of packet loss, 1998.

Frame Duplication and Reorder

Infrequently, a frame may be duplicated in the network and reach the destination twice. Hardware or software bugs or protection switches could cause this. For this test, duplication of one in every one million frames is reasonable.

Likewise, packet reorder on well managed networks is a rare event, on the order of one in a million. This occurs when packets arrive in different order than they were sent due to bugs, protection switches, changes to routing or switching tables, jitter or diverse network paths. If you are using an FEC scheme, it should correct out of order packets, and indeed this FEC scheme should be tested through network emulation. If and when packets do arrive out of sequence, the amount of reordering that occurs is likely to be small. In our experience, any given packet that is reordered will slip no more than 8 packets out of place.

Combined Impairments

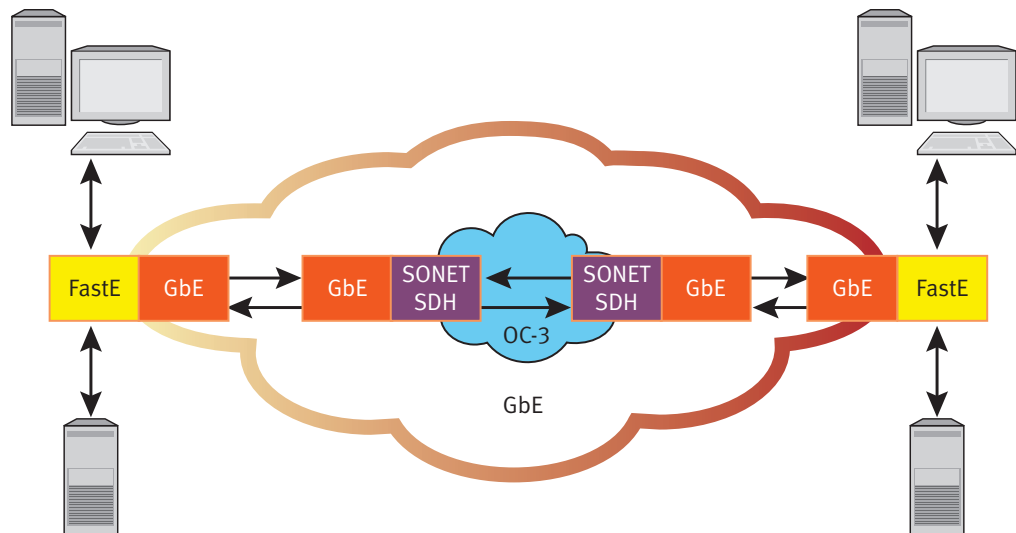
Let's assume you have diligently run each of the key impairments described above – delay, jitter, bit errors, packet loss, bandwidth clamping (with and without congestion control), and frame duplication/reorder – one after the other in your emulated network setup. What's next? An obvious answer is to run multiple impairment profiles at once. By combining impairments, you will further shake out errors and characterize the performance of your network applications in unfriendly network conditions.

A common combined network impairment test includes simultaneously injecting delay, jitter, packet loss, CRC corruption and frame duplication.

Multi-Protocols

Another type of “combination” testing involves emulating different protocols at different times during the course of validation. This represents the real world situation where application data is almost always mapped onto and carried by various lower layer protocols. Multi-protocol emulation is a valuable capability provided by some network emulators.

A TV broadcasting company performed multi-protocol emulation testing to fully test its streaming video application before a major international sporting event. The company planned to receive live video feeds from the event in China to North America for real time editing and processing. They were concerned about the effects of the 300ms delay time on the video processing application. Multiple feeds were Ethernet based but GFP frame mapped onto a SONET OC-3 (155 Mbps) link. The company’s network engineers emulated 300ms of delay time first at the physical, SONET layer, and then at the Ethernet layer using the same Spirent Network Emulator. The first test emulated the TDM circuit to stress the framing mechanisms, while the second test focused on emulating the end user’s application experience.



SONET/SDH and Ethernet Network Emulation using one emulator

Summary

At the end of the day, validating network applications is all about performance. Your goal in a validation procedure is to ascertain that network performance is acceptable from an end-user perspective. Network errors, congestion, delays and other impairments impact application performance. This impact can be significant, driving the need for thorough pre-deployment validation testing of all networked applications.

Application validation testing should be conducted under “real world” conditions. *If it is to be deployed on the WAN, then it should be tested as if on the WAN!* Preferably such testing should be done in a lab environment. There are different ways of doing this. You can build out a miniature network in your lab, use a third-party interoperability facility, or “borrow” the live production network. In many cases, the best option is to use a network emulator. Using a network emulator is easier, less expensive, more controllable and more repeatable than testing on a live or mock network.

With a network emulator and the right validation plan (derived from the information described in this “how to” white paper), you can:

- Characterize performance of new applications, even under unfavorable network conditions, prior to deployment
- Save time and money by increasing the relevance, effectiveness and efficiency of your testing
- Reduce risk while building confidence your solutions will perform as expected
- Assist with problem replication and troubleshooting
- Maintain customer satisfaction and loyalty



Inspired Innovation