Diff-Serv-aware Traffic Engineering Test

Test Results Summary **_Version 1.0_** ***12/18/00***

Introduction

The overall objective of the tests documented here was to examine the functionality of the feature referred to as Diff-Serv-aware Traffic Engineering (DSTE) in a pre-release version of Cisco IOS code. The test was carried out with an eye toward using this feature as part of the implementation of Abilene Premium Service (APS).

Details of the test procedure can be found in the document entitled “DSTE Abilene Lab Test Plan”. The raw data collected during the test execution can be found in various files noted in each section below. Details of the test setup, and the names of test configuration files and other data files associated with the test are listed in the Appendices.

Major Findings

- The basic tunnel functions, including building tunnels, route selection and failure modes, worked as expected.
- The guaranteed bandwidth tunnels with sub-pooling functions also worked as expected, including building tunnels, route selection and preemption.
- While attempting to execute Test Case J (BGP tunnel forwarding test with edge queuing enabled), we were unable to terminate two tunnels on a single router to two different loop-back interfaces. This is being treated as a bug, and is currently under investigation by Cisco.
- While trying to change the queuing configuration on one 7500 POS interfaces the interface shut down. We ultimately ended up having to reboot the router to get it back on line. The 7500 was running the EFT IOS.
- Using QPPB requires configuration on the GigaPop routers to inject BGP communities into the Abilene core. This is a diversion from the “black box” model of APS, where the GigaPops didn’t have any configuration tasks to utilize APS, except for marking packets.
- Overall we found the functionality to be sufficient to recommend proceeding with the program. We assume the IOS image that will be the candidate for a true EFT in a limited Abilene deployment will be regression-tested prior to installation on an Abilene core router.

Zero Case Scenarios

Case A: Baseline test of IOS and hardware configuration

Test Objective:

1. Verify an operational test environment, ensuring that OSPF and BGP are functioning as expected.
2. Confirm that OSPF will re-converge in the event of an interior link failure.

Test Results Summary:

1. Interrogation of the routing tables in all routers indicated that OSPF and BGP were configured correctly and operating as expected.
2. A traceroute from GSR A to an interface on ARy indicated that the preferred path through the core passes through Link L4. After Link L4 was disconnected, interrogation of the routing tables and the same traceroute indicated that the path was now via Links L3 and L5.

Raw Results File:
DSTE-Results-CaseA.pdf

Case B: Baseline test of traffic generation and data collection

Test Objective:

1. Verify correct operation of traffic generators.
2. Verify correct operation of traffic counters.

Test Results Summary:

1. The SmartBits traffic generator was able to pass traffic between ARx and ARy using multiple streams (designated Tunnel 1 – 5). At this point in the test, ARz was not yet setup.
2. The counters work correctly. On both GSR A and GSR C the counters, showing total packets matching the access list, approximately agree with the results from the SmartBits tester.

Raw Results File:
DSTE-Results-CaseB.pdf

Basic TE Tunnel

Case C: Basic TE Tunnel Test

Test Objective:

1. Verify that the basic TE tunnel mechanism works correctly when bandwidth constraints are placed on each link.

2. Verify that the bandwidth pool reservations report the proper amount of bandwidth available.
3. Send traffic across the configured tunnels and verify that the traffic is forwarded properly.
4. Send background traffic (traffic not destined for the tunnel endpoint) across the core and verify that it doesn’t flow through the tunnel.

Test Results Summary:

1. All of the tunnels were built properly, and the third tunnel was routed differently than the first two due to the configured bandwidth constraint, as expected.
2. The CLI showed the proper amount of Global Pool bandwidth remaining on both paths.
3. The SmartBits tester received traffic sent through each tunnel without error or loss. The counters on GSR A for each tunnel agreed (approximately) with the packet count seen by the SmartBits tester.
4. Since the router and the test tool agree on the amount of traffic sent through the tunnel, the assumption is that the background traffic passing through the router is not using the tunnel.

Raw Results File:
DSTE-Results-CaseC.pdf

Case D: Basic TE Tunnel Test and Route Selection

Test Objective:

1. Verify that the path a tunnel takes is affected by per-interface administrative weights.
2. Send traffic over the tunnel and verify that it is forwarded properly.
3. Send background traffic (traffic not destined for the tunnel endpoint) across the core and verify that it doesn’t flow through the tunnel.

Test Results Summary:

1. The path chosen when the tunnel was built was that expected based on administrative weights, and was not the shortest path.
2. The SmartBits tester received traffic sent through the tunnel without error or loss. The counters on GSR A for each tunnel agreed (approximately) with the packet count seen by the SmartBits tester.
3. Since the router and the test tool agree on the amount of traffic sent through the tunnel, the assumption is that the background traffic passing through the router is not using the tunnel.

Raw Results File:
DSTE-Results-CaseD.pdf

Case E: Basic TE Tunnel Failure Mode

Test Objective:

1. Verify the behavior of multiple tunnels configured across two possible paths when one of the paths fails. The test plan provides a matrix of four possible configuration scenarios.
2. Observe the approximate tunnel convergence time in the case of a link failure.

Test Results Summary:

1. The tunnels behaved as expected. In the situations where there was a link failure and bandwidth was available across the alternate path, the tunnel was successfully re-routed. When a path failed and there was no available bandwidth across the alternate path, the tunnel was not built, and was left in a “down” state.
2. We used a fairly crude tool here, but the results are still useful. Using pings from GSR A with a two second timeout, we noted that approximately four ping replies were lost (for a total time of approximately eight seconds) while the tunnel re-converged and inter-AS routing was reestablished. What this actually approximates is the “application recovery time”, a component of which is the time it takes for the tunnel to re-converge.

Raw Results File:

DSTE-Results-CaseE.pdf

Basic GB Tunnel with Sub-pool

CASE F: Basic GB Tunnel with Sub-pooling

Test Objective:

1. Verify that tunnels are created correctly using sub-pools as well as global pools, in the case where the bandwidth constraints are such that the tunnels will all be routed across the shortest path.
2. Send traffic over the tunnels and verify that it is forwarded properly.
3. Send background traffic (traffic not destined for the tunnel endpoint) across the core and verify that it doesn’t flow through the tunnel.
4. Repeat the above steps setting per-interface administrative weights such that the tunnels will prefer the longer path (via Links L3 and L5).

Test Results Summary:

1. All tunnels (both global and sub-pool) were routed correctly over Link L4. The available bandwidth on Link L4 is correct based on the configured tunnels and link bandwidth constraints.
2. The SmartBits tester received traffic sent through each tunnel without error or loss. The counters on GSR A for each tunnel agreed (approximately) with the packet count seen by the SmartBits tester.
3. Since the router and the test tool agree on the amount of traffic sent through the tunnel, the assumption is that the background traffic passing through the router is not using the tunnel.

4. All tunnels (both global and sub-pool) were routed correctly over Link L3, which is the longer path. The available bandwidth on Link L3 is correct based on the configured tunnels and link bandwidth constraints. The SmartBits tester received traffic sent through each tunnel without error or loss. The counters on GSR A for each tunnel agreed (approximately) with the packet count seen by the SmartBits tester.

Raw Results File:
DSTE-Results-CaseF.pdf

Case G: GB tunnel routing test for sub-pool reroute based on bandwidth availability

Test Objective:

1. Verify that a sub-pool tunnel will seek a path other than the shortest path when there is not sufficient reservable bandwidth on the shortest path.
2. Send traffic across the tunnels at varying rates and verify that the traffic is forwarded properly.

Test Results Summary:

1. Four tunnels were built in succession (one global, two sub-pools of that global, then another global) such that the total bandwidth of all four equaled the maximum reservable bandwidth for Link L4 (the shortest path). When the fifth tunnel was built (a sub-pool of the second global tunnel), it took the route over Link L3, since there was no available bandwidth on Link L4.
2. Traffic was successfully passed across all five tunnels with no loss or errors at both 10 Mbps and 19 Mbps per stream (noted in the results as 50% and 95% load). In the case of the 19 Mbps stream, note that this exceeded the bandwidth configured for the tunnel for two cases (Tunnels 2 and 3, configured for 12.4 Mbps each). It seems that in the case where bandwidth is available on the link, and there is no other QoS mechanism configured (CAR, etc.), then traffic will be allowed to exceed the reserved bandwidth for a tunnel.

Raw Results File:
DSTE-Results-CaseG.pdf

Case H: GB Tunnel with Sub-pool with Preemption

Test Objective:

1. Create several global and sub-pool tunnels with different priorities, and verify that the preemption mechanism (based on priority) works as expected. This is the base case, where the last tunnel built exceeds the available bandwidth but has a lower priority than any of the existing tunnels. Note that for this test case, Link L4 is the only available potential path (Links L3 and L5 are disabled).
2. Send traffic across the tunnels at varying rates and verify that the traffic is forwarded properly.

Test Results Summary:

1) The preemption mechanism worked as expected. The first three tunnels (two global and one sub-pool) were setup properly, using priorities 3-5. The fourth tunnel (a sub-pool) requested bandwidth in excess of that available on Link L4 (the only available path). The tunnel was created but not activated, because the priority it was created with (6) was insufficient to preempt any of the existing tunnels.
2) Traffic was successfully passed across all five tunnels with no loss or errors at both 1 Mbps and 5 Mbps per stream (noted in the results as 4% and 20% load).

Raw Results File:
DSTE-Results-CaseH.pdf

Case I: Tunnel Preemption Test part 2

Test Objective:

1. Create several global and sub-pool tunnels with different priorities, and verify that the preemption mechanism (based on priority) works as expected. In this case the last tunnel built (a sub-pool) exceeds the available bandwidth and has a higher priority than the previous sub-pool tunnel built. Note that for this test case, Link L4 is the only available potential path (Links L3 and L5 are disabled).
2. Send traffic across the tunnels at varying rates and verify that the traffic is forwarded properly.

Test Results Summary:

1. The preemption mechanism worked as expected. The first three tunnels (two global and one sub-pool) were setup properly, using priorities 3, 4 and 6 (in order). The fourth tunnel (a sub-pool) requested bandwidth in excess of that available on Link L4 (the only available path). The tunnel was created successfully at priority 5, and subsequently preempted the existing sub-pool tunnel with priority 6.
2. Traffic was successfully passed across all five tunnels with no loss or errors at both 2 Mbps and 4 Mbps per stream (noted in the results as 8% and 16% load).

Raw Results File:
DSTE-Results-CaseI.pdf

BGP routing tests with QOS based routing.

Case J: BGP tunnel forwarding test with edge queuing enabled

Test Objective:

1. Forward EF traffic over tunnels to the correct AS using static routes and a tweaked BGP next hop.
2. Build tunnels anchored to unique loopback interfaces.
3. Send traffic over the tunnels and verify that it is forwarded properly.
4. Check queuing on edge to verify correct operation.

Test Results Summary:
The test required setting tunnels to different loopback interfaces as their destination. We were unable to accomplish this because of a signaling error we received when attempting to build the second tunnel. Cisco is investigating to determine if it is a configuration issue or a bug. The test team voted to move on to other tests using a modified version of the routing procedure that had been proven in previous tests to work. The modified procedure consists of passing BGP routing information via neighbor and network statements as usual and using static routes to force the traffic over the tunnel.

Raw Results File:
None.

Case K: BGP forwarding with edge and core queuing

Test Objective:
1. Create a test bed environment that contains all the components of a comprehensive edge-to-edge service. This was accomplished by combining all of the previously tested components with Modified Deficit Round Robin (MDRR). The tunnel was associated with MDRR priority queues to produce EF behaviors, and CAR was used to provide strict priority edge queuing.
2. Verify that traffic marked EF is sent to the proper destination subnet, and that BE traffic does not use the tunnel.

Test Results Summary:
1. Configuration was successful. The CLI showed that MDRR is associated with the correct COS group.
2. Traffic was forwarded as expected, with the EF traffic flowing across the tunnel and BE traffic flowing extra-tunnel. We verified this by noting the following:
    a. The SmartBits received both the BE and EF traffic correctly, with no loss.
    b. The total amount of traffic sent by the SmartBits tester (1013512 packets) matched the total amount of traffic seen at the egress interface on ARx (POS to GSR A) using the "show policy" command, and at the ingress interface GSR A using the "show interface p5/3 rate" command.
    c. The rate limit counters on the ingress interface to GSR A showed 506756 packets conforming to the description of a BE packet (i.e. marked with DSCP 46). We assume that the rest of the traffic (1013512-506756=506756) was forwarded without using the tunnel.
    d. Note: in hindsight, we should have used the "show interface tunnelx accounting" command to verify the tunnel traffic directly, as we did in Case F.

Raw Results File:
DSTE-Results-CaseK.pdf

Case L: BGP forwarding with edge and core queuing, and QPPB

Test Objective:
Create a test environment using the features configured in the previous test (Case K), and adding the QoS Policy Propagation via BGP (QPPB) mechanism. Then:

1. Verify that QPPB hears BGP communities from an external AS. All EF traffic should arrive at it's appropriate destination. All advertisements and routing tables should be consistent for the configured policy. All traffic should flow over the appropriate interfaces, i.e. EF over tunnels BE as extra-tunnel traffic.
2. Verify that the traffic moves across the appropriate tunnel.
3. Remove MDRR on the core routers and send enough background traffic to flood Link L4. Verify that there is some packet loss due to a lack of protection previously provided by MDRR.

Test Results Summary:

1. The router GSR A, which is responsible for tunnel creation, did "hear the correct community" as demonstrated by the traffic behavior and queue examination. Traffic that was EF eligible that was bound for an EF receiver did get remapped into the correct COS group to take advantage of tunnels with MDRR queuing.
2. The traffic was correctly forwarded. As in Case K, the SmartBits output indicates that the BE and EF traffic arrived at the appropriate destinations.
3. There was some traffic loss noted by the SmartBits tester when MDRR was turned off.

Raw Results File:
DSTE-Results-CaseL.pdf

Abilene Premium Service Building Block Regression Test

Case BB.A: CAR Validation Test

Test Objective:

1. Configure Committed Access Rate (CAR) policing on a core router ingress interface and verify that the rate limiting function works as expected.

Test Results Summary:

1. We sent six streams (for an aggregate stream bandwidth of approximately 20 Mbps) of EF traffic from the SmartBits tester through the interface configured with CAR and found that all the traffic was passed, with no drops. We did not specifically attempt to send enough EF traffic to cause the CAR rate limiting to kick in and begin dropping packets.

Note: We began this test using the current Abilene burst size of 5000 and extended burst size of 5000. At that level there was significant packet drop well before the committed access rate was achieved, using the "packet blaster" stream of traffic from the SmartBits tester. In order to perform this test without interference from premature packet loss, we deemed it necessary to use the formula recommended in a Cisco white paper (see section

BB.A4 of the test plan) for setting the burst parameters. It may be deemed necessary to investigate the issue of optimal CAR settings during another test campaign.

Raw Results File:
DSTE-Results-CaseBB.A.pdf

CASE BB.B: Basic edge queuing strategy test

Test Objective:

1. Configure queuing and CAR on interface pos1A on ARx so that EF traffic (marked with DSCP 46) is given priority over BE traffic. Verify that the average latency for EF traffic is better than (i.e. less than) that for BE traffic.

Test Results Summary:

1. Due to a router misconfiguration, the results for this test were invalid. We noticed the discrepancy soon after the test case was completed, but we ran out of time to repeat the test with the proper configuration.

Note: Class-based Weighted Fair Queuing (CBWFQ) was used for this test, as required by the test plan. During the test, it was learned that the Cisco recommended queuing mechanism for this type of configuration is Low Latency Queuing (LLQ). Subsequent tests that required queuing used LLQ.

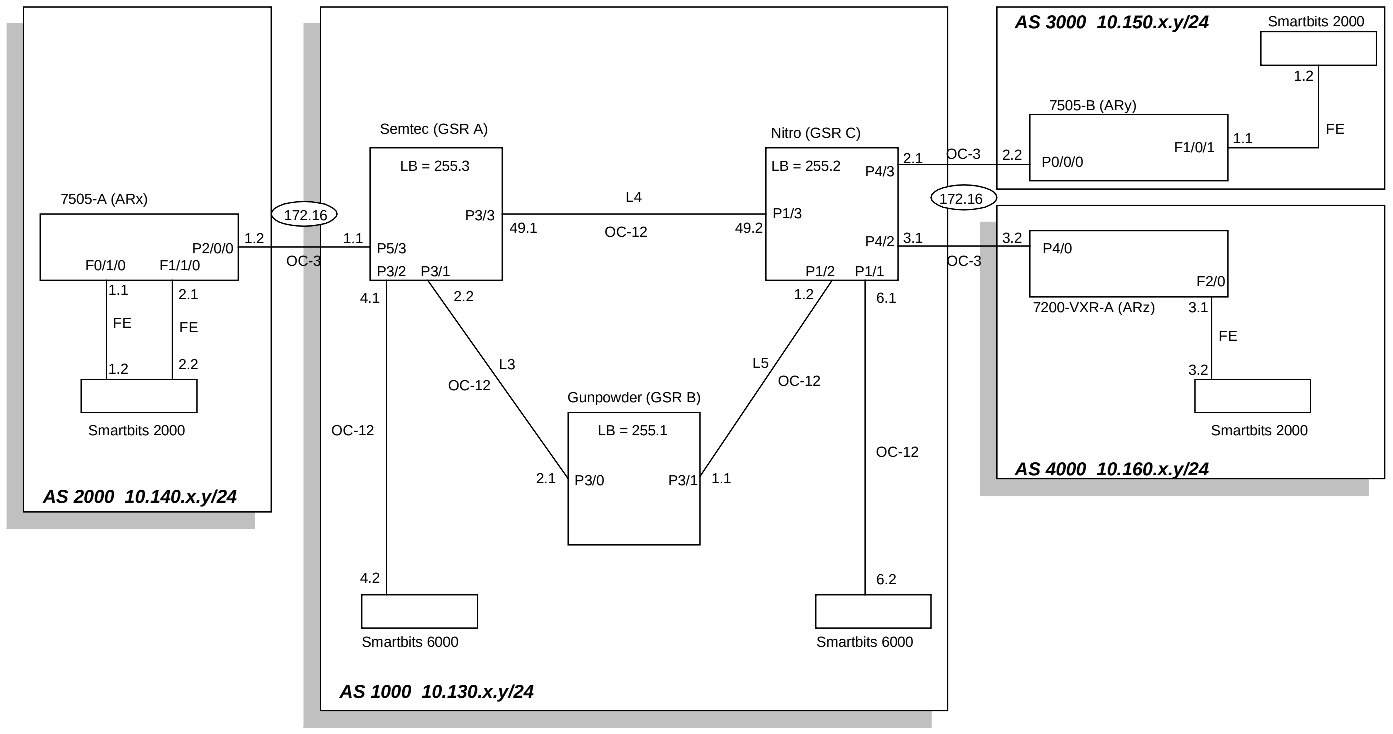
Raw Results File:
DSTE-Results-CaseBB.B.pdf

Overall Conclusion:

In general, the tests were very successful. Overall, Diff-Serv-aware Traffic Engineering appears to be very promising. It clearly provides most of the mechanisms required to create an EF PHB, and ultimately a PDB, when coupled with appropriate queuing strategies. There are a few potential bugs we mentioned in the body of this report but Cisco is examining them and we will work closely with to resolve any outstanding issues.

We can recommend proceeding to a field trial, using a released version of IOS that supports all of the basic Abilene functions as enumerated in the test plan. Once a candidate IOS release is made available, we recommend regression testing it in the lab prior to deployment. During the regression test, we can develop a set of model configurations and a field trial test plan.

Bill Cerveny	John Moore	Paul Schopis
cerveny@advanced.org	jhm@ncstate.net	pschopis@oar.net
Advanced Networks & Services	North Carolina ITEC	Ohio ITEC


7505-A (ARx)
P2/0/0
F0/1/0
F1/1/0
1.1
2.1
FE
FE
1.2
2.2
Smartbits 2000
AS 2000 10.140.x.y/24
1.2
172.16
OC-3
1.1
Semtec (GSR A)
LB = 255.3
P3/3
P5/3
P3/2
P3/1
49.1
L4
OC-12
49.2
Nitro (GSR C)
LB = 255.2
P4/3
P1/3
P4/2
P1/2
P1/1
4.1
2.2
1.2
6.1
L3
OC-12
L5
OC-12
OC-12
OC-12
Gunpowder (GSR B)
LB = 255.1
2.1
P3/0
P3/1
1.1
4.2
6.2
Smartbits 6000
Smartbits 6000
AS 1000 10.130.x.y/24
2.1
OC-3
2.2
172.16
3.1
OC-3
3.2
AS 3000 10.150.x.y/24
Smartbits 2000
1.2
7505-B (ARy)
P0/0/0
F1/0/1
1.1
FE
P4/0
F2/0
7200-VXR-A (ARz)
3.1
FE
3.2
Smartbits 2000
AS 4000 10.160.x.y/24

Appendix B Test Setup Details

Link Specification

L1	OC-3 MM
L3	OC-12 MM
L4	OC-12 MM
L5	OC-12 MM
L6	OC-3 MM
L7	OC-3 MM

GSR A (Semtec) AS 1000

From Test Plan	Actual Interface	IP Address	Speed	Engine
POS 1A	POS 3/3	10.130.49.1	OC-12 MM	2
POS 2A	POS 3/1	10.130.2.2	OC-12 MM	2
POS 3A	POS 5/3	172.16.1.1	OC-3 MM	0
Unlisted SB	POS 3/2	10.130.4.1	OC-12 MM	2
Loopback		10.130.255.3		

GSR B (Gunpowder) AS 1000

From Test Plan	Actual Interface	IP Address	Speed	Engine
POS 1B	POS 3/0	10.130.2.1	OC-12 MM	2
POS 2B	POS 3/1	10.130.1.1	OC-12 MM	2
Loopback		10.130.255.1		

GSR C (Nitro) AS 1000

From Test Plan	Actual Interface	IP Address	Speed	Engine
POS 1C	POS 1/3	10.130.49.2	OC-12 MM	2
POS 2C	POS 1/2	10.130.1.2	OC-12 MM	2
POS 3C	POS 4/2	172.16.3.1	OC-3 MM	0
POS 4C	POS 4/3	172.16.2.1	OC-3 MM	0
Unlisted SB	POS 1/1	10.130.6.1	OC-12 MM	2
Loopback		10.130.255.2		

ARx (7505-A) AS 2000

From Test Plan	Actual Interface	IP Address	Speed
POS 1x	POS 2/0/0	172.16.1.2	OC-3 MM
SB	F 0/1/0	10.140.1.1	100Mbps FDX
Unlisted SB	F 1/1/0	10.140.2.1	100 Mbps FDX
Loopback		10.140.255.1	

ARy (7505-B) AS 3000

From Test Plan	Actual Interface	IP Address	Speed
POS 1y	POS 0/0/0	172.16.2.2	OC-3 MM
SB	F 1/0/1	10.150.1.1	100Mbps FDX
Loopback		10.150.155.1	

ARz (7200) AS 4000

From Test Plan	Actual Interface	IP Address	Speed
POS 1z	POS 4/0	172.16.3.2	OC-3 MM
SB	F 2/0	10.160.3.1	100Mbps FDX

SB (SMB 6000, 128.109.54.98) Gateways = 10.130.x.1

From Test Plan	Actual Interface	IP Address	Type	Speed
	1A	not in use	POS	OC-3/12
	1B	not in use	POS	OC-3/12
	2A	not in use	POS	OC-3/12
	2B	not in use	POS	OC-3/12
Connect to GSR A	3A	10.130.4.2	POS	OC-12
Connect to GSR C	3B	10.130.6.2	POS	OC-12
	4A	not in use	POS	OC-3
	4B	not in use	POS	OC-3
	5A	not in use	GigE	
	5B	not in use	GigE	

SB (SMB 2000, 128.109.54.100) Gatways = 10.130.x.1

From Test Plan	Actual Interface	IP Address	Type
to ARx	Port 1	10.140.1.2	FE
to ARx	Port 4	10.140.2.2	FE
to ARy	Port 2	10.150.1.2	FE
to ARz	Port 3	10.160.3.2	FE

Tunnel Destination Addresses

From Test Plan	Dest. IP Address
Tunnel 1	10.150.1.11
Tunnel 2	10.150.1.12
Tunnel 3	10.150.1.13
Tunnel 4	10.150.1.14
Tunnel 5	10.150.1.15

Appendix C Tester configuration and other files

File Name	Description
DSTE Tunnel Matrix V1_0.pdf	Matrix of SmartBits tester configuration for test cases A-L
DSTE CAR-queuing test Matrix V1_0.pdf	Matrix of SmartBits tester configuration for test cases BB.A-BB.B
GSR Hardware Confiurations.pdf	CLI output of “show version” and “show diagnostics *slot number*” commands for the three GSRs.

Appendix D SmartBits Traffic Generator Testing Methodology

Two Spirent traffic generator chassis' were used in this test: a SmartBits SMB-2000 chassis with four ML7710 10/100BaseT interfaces and a SmartBits SMB-6000 chassis with two POS-6500B OC-3/12 POS interfaces.

The SMB-2000 chassis provided the test traffic for verifying DSTE tunnel functionality. To simplify test setup, five destination IP addresses were defined for each of up to five tunnels. For each individual test, traffic was sourced at the SmartBits 100BaseT interface connected to access router ARx and sinked at the tunnel destination IP address on the SmartBits 100BaseT interface connected to access router ARy.

Generally, all tunnels terminated at the SmartBits interface connected to access router ARy. In test cases K and L, Expedited Forwarding (EF) and Best Effort (BE) traffic was sinked to two SmartBits interfaces connected to routers ARy and ARz to ensure latency and loss findings weren't compromised by unusual performance characteristics. In test case BB.A, traffic was generated in all directions between SmartBits interfaces connected to routers ARx, ARy and ARz. In test case BB.B, traffic was generated as in test case BB.A, but half of the traffic from router ARx and destined for router ARy was sourced with one IP address, which was then marked DSCP 46 by router ARx, and the other half was sourced with a different IP address, which was not marked by router ARx (sent as BE).

The software application SmartFlow controlled tunnel traffic on the SMB-2000 chassis. Smartflow offered the ability to create timed tests and generate results in a format that could be imported into Microsoft Excel. For each test, equally spaced 128-byte IP frames with payloads of "all zeroes" were transmitted for 60 seconds (unless noted otherwise).

The SmartFlow test type most frequently executed, the "frame loss" test, provided information on transmitted and received frames. Usually the most important factors in a test were whether or not packets transmitted into a tunnel actually used the tunnel and were received at the tunnel destination. After the completion of each test, router packet counters for the tunnels were compared with frames transmitted and received by SmartFlow. Where latency measurement was important, a separate test using the SmartFlow "latency" test type was conducted.

In hindsight, the SmartFlow "Jumbo" test type would have probably been a better choice for all cases. In addition to information on frames transmitted and received, the "Jumbo" test type reports simultaneous latency information, which may have been interesting in some tests where the "frame loss" test type didn't capture latency information.

In the raw results document, "percent load" reported by SmartFlow reflected the *total* percent load for the interface and not the percent load for the *individual* traffic flow. A behavior of SmartBits hardware and software, the percent load for the interface defines the sum of equally sized streams or flows on that interface. For example, a 20% load on an interface where there are four streams means that each of the four streams consumed 5% of the load on the interface.

Because SmartFlow creates equally sized flows by default, it was somewhat challenging to simulate flows of differing sizes on one interface; the smallest common denominator of the various flow sizes had to be determined and used as a base to create the desired flow loads. For example, in case test C, to create a load of 700Kbps for one tunnel and 100Kbps loads for two other tunnels, seven identical 100Kbps flows had be created for the first tunnel and a 100Kbps flow was created for each of the two other tunnels.

The SmartBits-6000 chassis generated background traffic across the core network with two POS-6500B OC-3/12 SmartMetrics interfaces. SmartWindow was used to control the background traffic because background traffic could be started and stopped without the timing controls associated with other Netcom applications such as SmartFlow. All frames of generated background traffic were 124-byte IP packets with payloads of “all zeroes.” Background traffic was only generated where it was relevant to the behavior being observed; these tests are noted in the SmartBits tester configuration matrices.