Black Book

ixia

Edition 10

Application Delivery

http://www.ixiacom.com/blackbook

June 2014

Your feedback is welcome

Our goal in the preparation of this Black Book was to create high-value, high-quality content. Your feedback is an important ingredient that will help guide our future books.

If you have any comments regarding how we could improve the quality of this book, or suggestions for topics to be included in future Black Books, please contact us at ProductMgmtBooklets@ixiacom.com.

Your feedback is greatly appreciated!

Contents

How to Read this Book

The book is structured as several standalone sections that discuss test methodologies by type. Every section starts by introducing the reader to relevant information from a technology and testing perspective.

Each test case has the following organization structure:

Overview	Provides background information specific to the test case.
Objective	Describes the goal of the test.
Setup	An illustration of the test configuration highlighting the test ports, simulated elements and other details.
Step-by-Step Instructions	Detailed configuration procedures using Ixia test equipment and applications.
Test Variables	A summary of the key test parameters that affect the test's performance and scale. These can be modified to construct other tests.
Results Analysis	Provides the background useful for test result analysis, explaining the metrics and providing examples of expected results.
Troubleshooting and Diagnostics	Provides guidance on how to troubleshoot common issues.
Conclusions	Summarizes the result of the test.

Typographic Conventions

In this document, the following conventions are used to indicate items that are selected or typed by you:

- **Bold** items are those that you select or click on. It is also used to indicate text found on the current GUI screen.
- *Italicized* items are those that you type.

Dear Reader

Ixia's Black Books include a number of IP and wireless test methodologies that will help you become familiar with new technologies and the key testing issues associated with them.

The Black Books can be considered primers on technology and testing. They include test methodologies that can be used to verify device and system functionality and performance. The methodologies are universally applicable to any test equipment. Step by step instructions using Ixia's test platform and applications are used to demonstrate the test methodology.

This tenth edition of the black books includes twenty two volumes covering some key technologies and test methodologies:

Volume 1 – Higher Speed Ethernet

Volume 2 – QoS Validation

Volume 3 – Advanced MPLS

Volume 4 – LTE Evolved Packet Core

Volume 5 – Application Delivery

Volume 6 – Voice over IP

Volume 7 – Converged Data Center

Volume 8 – Test Automation

Volume 9 – Converged Network Adapters

Volume 10 – Carrier Ethernet

Volume 11 – Ethernet Synchronization

Volume 12 – IPv6 Transition Technologies

Volume 13 – Video over IP

Volume 14 – Network Security

Volume 15 – MPLS-TP

Volume 16 – Ultra Low Latency (ULL) Testing

Volume 17 – Impairments

Volume 18 – LTE Access

Volume 19 – 802.11ac Wi-Fi Benchmarking

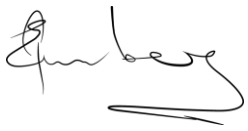
Volume 20 – SDN/OpenFlow

Volume 21 – Network Convergence Testing

Volume 22 – Testing Contact Centers

A soft copy of each of the chapters of the books and the associated test configurations are available on Ixia's Black Book website at http://www.ixiacom.com/blackbook. Registration is required to access this section of the Web site.

At Ixia, we know that the networking industry is constantly moving; we aim to be your technology partner through these ebbs and flows. We hope this Black Book series provides valuable insight into the evolution of our industry as it applies to test and measurement. Keep testing hard.



Errol Ginsberg, Acting CEO

Application Delivery

Test Methodologies

The focus of these test cases is to provide hands-on guidance on how to set up and execute a series of related tests that assess the impact on performance of a unified security device when enabling application and content aware features.

Application Delivery Testing Overview

Today's IP networks have evolved beyond providing basic local and global connectivity using traditional routing and packet-forwarding capabilities. Converged enterprise and service provider networks support a complex application delivery infrastructure that must recognize, prioritize, and manage multiplay traffic with differentiated classes of service. The emergence of integrated service routers (ISRs), application-aware firewalls, server load balancers, and deep packet inspection (DPI) devices is enabling businesses to deliver superior application performance and security while improving the quality of experience (QoE) for its users.

Equipment manufacturers need a comprehensive test solution for validating the capabilities, performance and scalability of their next-generation hardware platforms. The foundation begins with the ability to generate stateful application traffic such voice, video, peer-to-peer (P2P), and data services in order to measure the key performance indicators for each application. Meeting this challenge requires a comprehensive test methodology that addresses the complexity of performance and scale testing requirements.

Application Layer Forwarding

Inspection of the application data within a packet makes available the information necessary to determine the true usage of the traffic: interactive content, video, web page contents, file sharing, etc. It also makes it possible to detect viruses, spam, and proprietary information within data packets. For example, Windows Messenger uses HTTP, with a special setting in the User-Agent field of a message. In order to apply the appropriate QoS policy for instant messaging, the HTTP message must be parsed for this value.

Traditional stateful packet inspection looks at the IP and TCP/UDP headers to decide where and how packets to forward the packets.

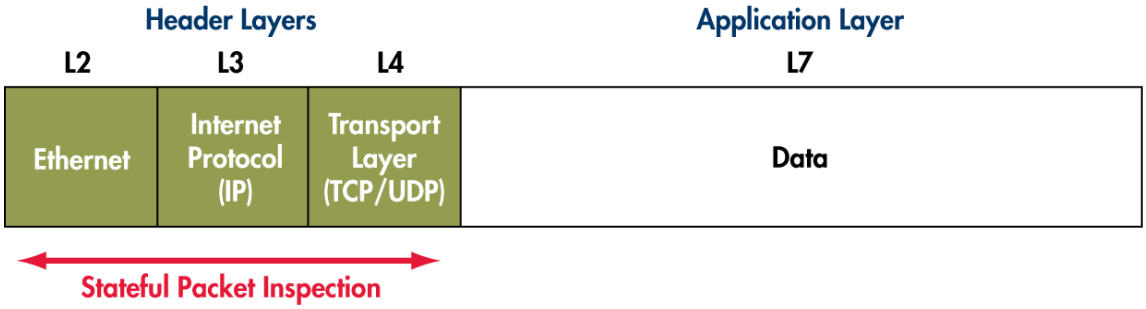


Figure 1. Traditional Packet Inspection

The essential information found there includes the source and destination IP address, TCP/UDP port number and type of service (TOS). The TCP/UDP port numbers have well-known associations; for example TCP/21 is associated with FTP, TCP/80 with HTTP, TCP/25 with SMTP and TCP/110 with POP3. This 5-tuple of information from layers 3 and 4 is the classic

means by which firewalls, routers and other switching devices decide on whether to and where to forward packets and with what priority.

This information is not sufficient to satisfy the requirements for multiplay services in a mixed customer environment. Additional elements of each packet must be inspected.

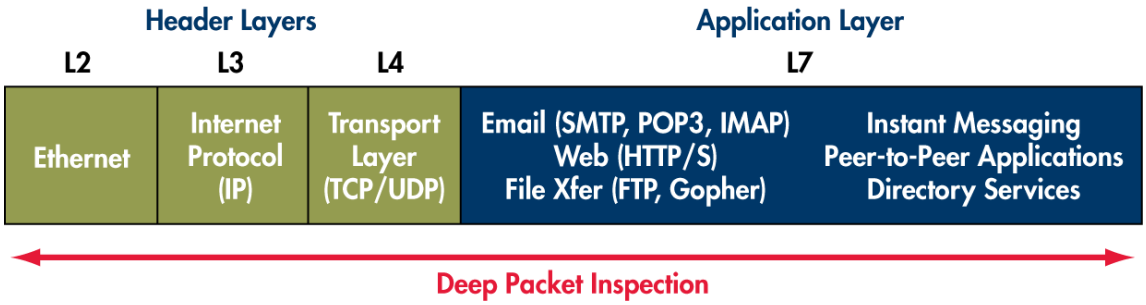


Figure 2. Deep Packet Inspection

The application layer (layer 7) of the packet holds information specific to a protocol. All bits and bytes are inspected by deep packet inspection engines, allowing network devices to finely classify traffic based on type and source. For example not only can you identify the traffic as email using SMTP, you can now identify the source application as Microsoft Outlook™ by examining the application signature. The information can be used to provide:

- Subscriber and service based QoS policing
- Peer-to-peer bandwidth management
- Denial of service (DoS) and intrusion prevention
- Email virus and content filtering
- Web content filtering, phishing

Security Threats

Losses due to security breaches that result in theft, downtime and brand damage now stretch into the tens of millions of dollars per year for large enterprises, according to Infonetics Research. Attacks and failures are seen at every level – from online applications, to networks, to mobile and core infrastructures.

Conventional security software and appliances, such as anti-virus protection and firewalls, have increasingly reduced the number of attacks, but the total losses continue to grow. The 2007 CSI Computer Crime and Security Survey reported that in 2006 the average loss per survey respondent more than doubled when compared to the year before.

Security issues have pushed defenses into network devices and have spawned a number of auxiliary security enforcement devices. These functions include:

- Intrusion prevention systems (IPSs)

- Unified threat management systems
- Anti-virus, anti-spam, and phishing filters

Increasingly, application-aware devices are performing security functions – largely because the information they need is now available through deep packet inspection.

Measuring Application Performance

The requirements for testing application-aware devices are complex. The challenge lies in creating complete and true stateful application traffic flows to exercise the deep packet inspection capabilities of the device.

A new generation of high-scale, high-performance devices can handle millions of sessions, and process packets in the 100's of Gigabits per second. These platforms have to be pushed to its limits and beyond to ensure that it will function at optimum levels and properly apply policies to manage traffic. This type of testing involves the use of a wide range of multiplay traffic:

- Data, including HTTP, P2P, FTP, SMTP, POP3
- Video, including IGMP, MLD, RTSP/RTP
- Voice, including SIP, MGCP

Determining the ability of a device to process content intelligently requires determining key performance indicators for each service or application. The quality expectation varies between services, hence a comprehensive set of metrics is required to tailor to get visibility into how a device is performing. These include:

- HTTP/web response time for loading web pages and content
- VoIP call setup time and voice quality
- Consistent and reliable video delivery and quality
- Peer-to-peer (P2P) throughput
- DNS queries speed and response time
- Email transfer performance and latency

Negative tests must also be applied to ensure that attack traffic is correctly classified and that it does not affect normal traffic performance. Of particular importance is the testing of devices and networks under the influence of distributed denial of services (DDoS) such as SYN floods.

Scalability testing is of particular importance for capacity planning. NEMs must publish limits that their customer can use to scale up and manage future growth.

This test plan is focused on providing expert guidelines of how to measure the performance of today's application-aware platforms with a true application traffic mix.

Focused is placed on providing in depth technical background for different scenarios that are relevant to test, and a thorough and complete methodology to execute the test.

Getting Started Guide

For application performance tests, Ixia's Layer 7 test solution, IxLoad™, is used in this methodology. The following section is designed for new users who wish to become familiar with the user interface and workflow to create and run tests in IxLoad.

The user interface is presented below. The **Test Configuration** pane is used to navigate the various configuration windows used to create the test elements, including network, traffic, user behavior and other advanced capabilities.

The **Statistics** pane is a real-time statistics viewer during an active test. All of the key performance indicators are present in this view.

The **Analyzer** pane is a real-time packet capture tool and a decode viewer. It offers a simple, powerful way to capture traffic on the test ports for debugging purposes. Packet analysis is done to show conversations between peers and client/server flows.

User Interface Settings

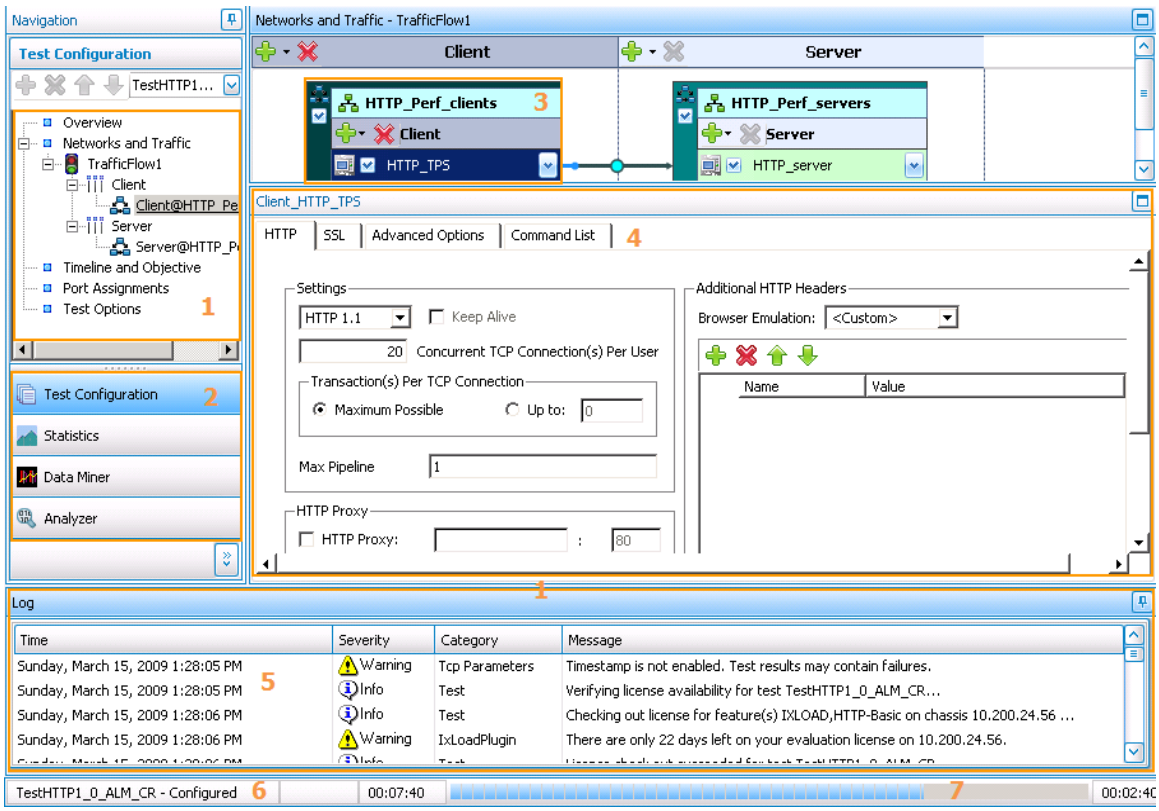


Figure 3. Main IxLoad GUI

The numbers in the following discussion correspond to the numbers in figure above.

1. The left navigation panel is used to select all configuration windows. It contains the network and traffic configurations, setting test duration and objective, and assigning test ports.

2. This panel switches views between test configuration, looking at real-time statistics, or accessing the **Analyzer** view for analyzing captured packets.

3. The network and protocol configuration object is called a **NetTraffic**. The network IP addresses, protocol configuration, and request pages are configured by selecting the network or the activity object and configuring the details in the window below it.

4. Detailed configuration for network and protocol configuration is done here. Network settings, protocol configuration, page sizes and user behavior (i.e. what pages to request) are configured here.

5. The log window provides real-time test progress and indicates warnings and test errors. Keep this window active to become familiar with IxLoad's workflow and test progress.

6. The test status is indicated here, such as **Unconfigured** or **Configured**. **Configured** refers to an active test configuration present on the test ports.

7. Test progress is indicated here for a running test, with total test time and remaining duration. The test objective and duration is configurable from the **Timeline and Objective** view that is accessed from the tree panel from (**1**).

Before getting started, refer to the following figure to understand IxLoad's workflow.

User Workflow for Configuring and Running a Series of Tests

These are the steps to create and run a series of tests in IxLoad.

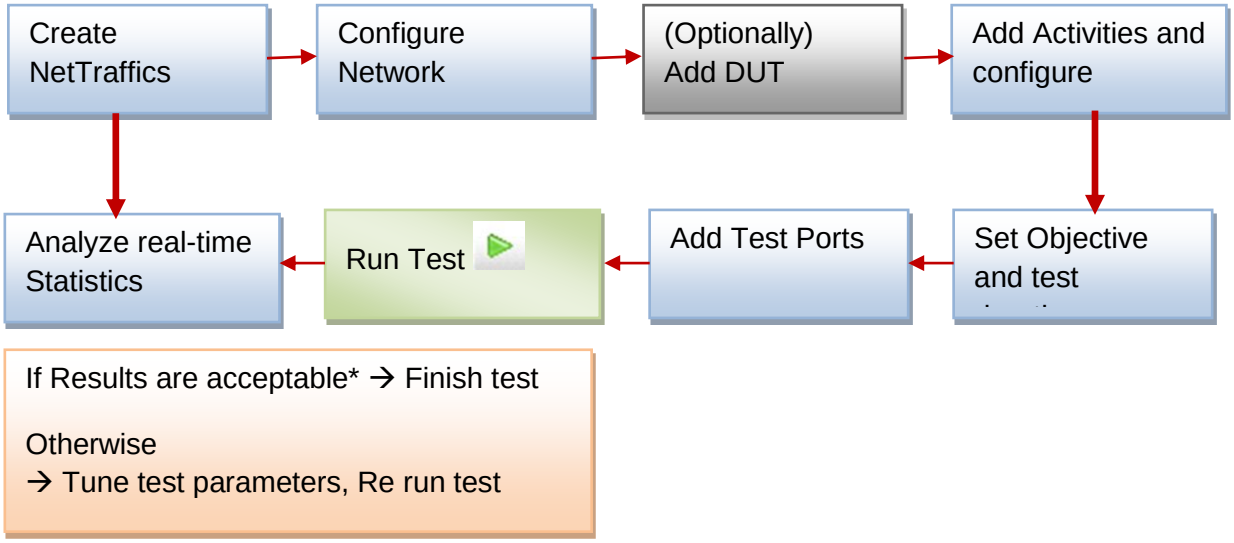


Figure 4. User workflow for configuring and running a series of tests in IxLoad

* Acceptable test results are based on the target desired versus what was actually attained. Additionally, for each type of test, key performance metrics should be examined to determine if the results obtained can be considered acceptable.

It is important to establish a baseline performance. A baseline test is one that only uses the test ports; the test profile is configured to be very similar to the actual desired profile to determine the test tool limit. The baseline performance can be used to scale up and build the test profile to appropriately measure the DUT performance.

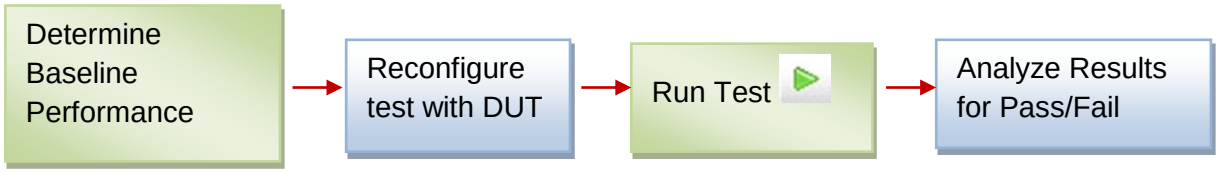


Figure 5. Establishing a baseline

Maximum Performance Testing

Application-aware devices' perform needs to be validated under different workloads - which include traffic profiles that simulate for different periods of time that represent peak and nominal performance capacities. The peak performance is an important metric which indicates the best performance of a device in an optimal environment.

Increasingly, the networks that deploy content-aware devices have become complex – making intelligent decisions for content delivery based on application information. The complexity can also be considered multi-dimensional in that it attempts to deliver good performance for a variety of non-optimal traffic profiles that are present in production networks.

There are several key performance indicators that are generally used to determine the maximum performance of a device under test (DUT), which are most often measured under optimal conditions. The key performance metrics are listed below.

Table 1. Key performance metrics

Metric	Description
Connection	A single TCP connection between two hosts, using connection establishment (3-way handshake).
Transaction	A single application request and its associated response. Transactions use an established TCP connection to send and receive messages between two hosts.
Concurrent connections	Multiple TCP connections established between two or more hosts.
Connections per second	The per-second rate at which new TCP connections are initiated.
Transactions per second	The per-second rate at which application transactions are initiated and serviced.
Throughput	The rate at which data is sent and received, measured in bits per second. When measuring performance of an application-aware device, goodput is used.
Protocol latency	The time elapsed between a sending a protocol request and receiving the reply. Refer to TTFB and TTLB for more information.
TTFB	Time to first byte – The time elapsed before the client receives the first byte of the HTTP response.
TTLB	Time to last byte – The time elapsed before the client receives the last byte of the HTTP response.

Other performance metrics may be of interest to the tester to characterize the firewall/SLB performance, and tests can be designed to meet these objectives.

Test Case: Maximum Connections per Second

Overview

Determine the maximum rate of TCP connections that a device can service per second. Connection per second can be determined in multiple ways:

1. A TCP connection establishment (SYN, SYN-ACK, ACK), followed by a complete layer 7 transaction (Request, Response), and TCP teardown (FIN, ACK).
2. A TCP connection establishment (SYN, SYN-ACK, ACK), followed by a partial or incomplete layer 7 transaction (Request), and TCP teardown (FIN, ACK).
3. A TCP connection establishment (SYN, SYN-ACK, ACK), followed by a partial or incomplete layer 7 transaction (Request), and forced TCP teardown (RST).

The most desirable approach is the first option, in which a complete and successful transaction at layer 7 happens. However, it's also possible that the device can handle new TCP sessions but not all layer 7 transactions. In this case, the second approach provides another meaningful performance metric that focuses on only the layer 4 performance of the device. The third approach can also be used to further stress the device by forcing connection teardowns.

Objective

Performance metrics required: Maximum connections per second. This metric has real-world significance in that it provides a raw performance metric of how well the DUT is able to accept and service new connections**.**

Setup

The setup requires at least one server and one client port. The HTTP client traffic will pass through the DUT to reach the HTTP server. The HTTP client and server must be connected to the DUT using a switched network.

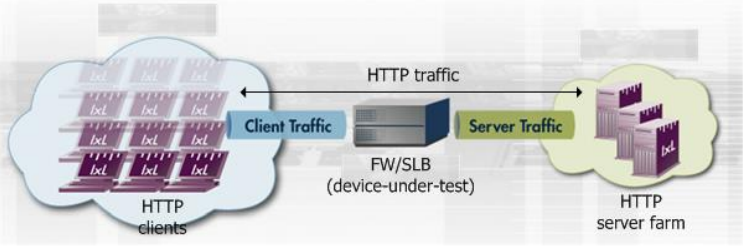


Figure 6. Connections per second test setup

Test Variables

Test Tool Variables

The following test configuration parameters provide the flexibility to create the traffic profile that a device would experience in a production network.

Table 2. HTTP configuration parameters

Parameters	Description
HTTP clients	100 IP addresses or more, use sequential or “use all” IP addresses
HTTP client parameters	HTTP/1.0 without keep-alive 20 TCP connections per user 1 Transaction per TCP connection
TCP parameters	TCP RX and TX buffer at 4096 bytes
HTTP client command list	1 GET command – payload of 1-128 byte
HTTP servers	1 per Ixia test port, or more
HTTP server parameters	Random response delay – 0 – 20 ms Response timeout – 300 ms

DUT Test Variables

There are several scenarios that may be used to test a DUT. The following table outlines some of the capabilities of the DUT that can be switched on to run in a certain mode.

Table 3. Examples of DUT scenarios

Device(s)	Variation	Description
Server load balancer	Activate packet filtering rules	• Configure SLB engine for “stickiness” • Change the algorithm for load balancing • Use Bridged or Routed Mode for servers
Firewall	Activate access control rules	• Configure Network Address Translation or Disable for Routed Mode
Firewall security device	Enable deep content inspection (DPI) rules	• Advanced application-aware inspection engines enabled • IDS or threat prevention mechanisms enabled

Step-by-Step Instructions

Configure the test to run a baseline test, which is an Ixia port-to-port test, in order to verify the test tool’s performance. Once you have obtained the baseline performance, setup the DUT and the test tool as per the **Setup** section above. Refer to the **Test and DUT Variables** section for recommended configurations. Note that the network configurations must change between running the port-to-port and DUT test. Physical cabling will change to connect the test ports to the DUT. A layer 2 switch that has a high-performance backplane is highly recommended in a switched environment.

Reference baseline performance: Fill in the table below with the baseline performance to use as a reference of the test tool's performance, based on the quantity and type of hardware available.

Table 4. Reference baseline performance form

Performance indicator	Value per port pair	Load module type
Connections/sec		

1. Launch IxLoad. In the main window, you will be presented with the Scenario Editor window. All test configurations will be done here.

   To get familiar with the IxLoad GUI, see the Getting Started Guide section.

2. Add the client **NetTraffic** object. Configure the **Client** network with total IP count, gateway and VLAN, if used.

   Add the server **NetTraffic**. Also configure the total number of servers that will be used. For performance testing, use *1* server IP per test port.

   For a step-by-step workflow, see Appendix A.

Figure 7. IxLoad Scenario Editor view with Client and Server side NetTraffics and Activities

3. The TCP parameters that is used for a specific test type is important in optimizing the test tool. Refer to the **Test Variables** section to set the correct TCP parameters.

   There are several other parameters that can be changed. Leave them at their default values unless you need to change them as per testing requirements.

   For a step-by-step workflow, see Appendix B.

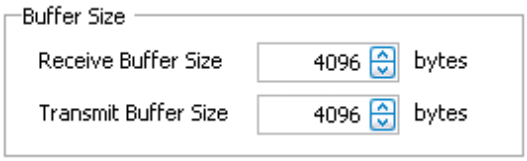


Figure 8. TCP Buffer Settings Dialogue

4. Configure the **HTTP server**. Add the **HTTP Server Activity** to the server **NetTraffic**. The defaults are sufficient for this testing.

   For a step-by-step workflow, see Appendix C.

5. Configure the **HTTP client**. Add the **HTTP Client Activity** to the client **NetTraffic**. Refer to **Test Variables** section to configure the HTTP parameters.

   You can use advanced network mapping capabilities to use sequence IPs or use all IPs configured.

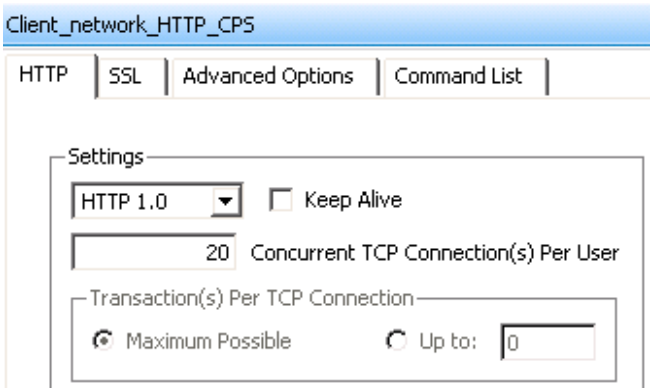


Figure 9. HTTP Client Protocol Settings Dialogue

   For a step-by-step workflow, see Appendix D.

6. Having setup client and server networks and the traffic profile, the test Objective can now be configured.

   Go to the **Timeline and Objective** view. The test **Objective** can be applied on a per-activity or per-protocol basis. The iterative objectives will be set here and will be used between test runs to find the maximum TPS for the device.

   The following should be configured:

   - **Test Objective**. Begin by attempting to send a large number of connections per second through the DUT. If the published or expected value for **MAX_RATE** is known, this value is a good starting point, and will become the targeted value for the test (**TARGET_MAX_RATE**).

Network Traffic Mapping	Objective Type	Objective Value
TrafficFlow1		
Client_network@HTTP_Pe...	Connections/Second	5100
HTTP_CPS	Connections/Second	5100
Server_network@HTTP_P...	N/A	N/A
HTTP_server	N/A	N/A

Figure 10. Test Objective Settings Dialogue

   For a step-by-step workflow, see Appendix E.

7. Once the Test Objective is set, the “Port CPU” on the bottom indicates the total number of ports that is required. See the Appendix below on adding the ports to the test.

For a step-by-step workflow, see Appendix F.

Run the test for a few minutes to reach a steady-state. Steady state is referred as the **Sustain** duration in the test. Continue to monitor the DUT with respect to the target rate and any failure/error counters. See the **Results Analysis** section for important statistics and diagnostics information.

In most cases interpretation of the statistics is non-trivial, including what they mean under different circumstance. The **Results Analysis** section that follows provides a diagnostics-based approach to highlight some common scenarios, the statistics being reported, and how to interpret them.

8. Iterate through the test setting TARGET_MAX_RATE to the steady value attained during the previous run. To determine when the DUT has reached its MAX_RATE, see the Results Analysis section on interpreting results before making a decision.

   The test tool can be started and stopped in the middle of a test cycle, or wait for it to be gracefully stopped using the test controls shown here. 

   For a step-by-step workflow, see Appendix F.

Results Analysis

The maximum connections/sec performance requires an iterative method in which the test is run multiple times, changing a number of test input parameters. The DUT configuration must also be configured so that it performs optimally, based on the **Test Variables** section.

The following the key performance statistics that must be monitored. The importance of these statistics is that it helps identify if the device has reached its saturation point, and identify issues. Also interpreting the results in the correct manner will ensure that transient network, device or test tool behavior do not create a false negative condition.

Table 5. Key performance indicators that require monitoring

Metric	Key Performance Indicators	Statistics View
Performance metrics	Connections/sec Total connections, Number of Simulated Users, Throughput	HTTP Client – Objectives HTTP Client – Throughput
Application level transactions Application level failure monitoring	Requests Sent, Successful, Failed Request Aborted, Timeouts, Session Timeouts Connect time, 4xx, 5xx errors	HTTP Client – Transactions HTTP Client – HTTP Failures HTTP Client – Latencies
TCP Connection Information TCP Failure monitoring	SYNs sent, SYN/SYN-ACKs Received RESET Sent, RESET Received, Retries, Timeouts	HTTP Client – TCP Connections HTTP Client – TCP Failures
Other Indicators	Per URL statistics, Response Codes	HTTP Client – Per URL HTTP Client – xx Codes

Real-Time Statistics

The graph below provides a view of the real-time statistics for the test. Real-time statistics provide instant access to key statistics that should be examined for failures at the TCP and HTTP protocol level.

The statistics below are real-time test-level performance observations. The **Connection Rate** statistic indicates that the DUT is able to sustain approximately 4500 connections per second.

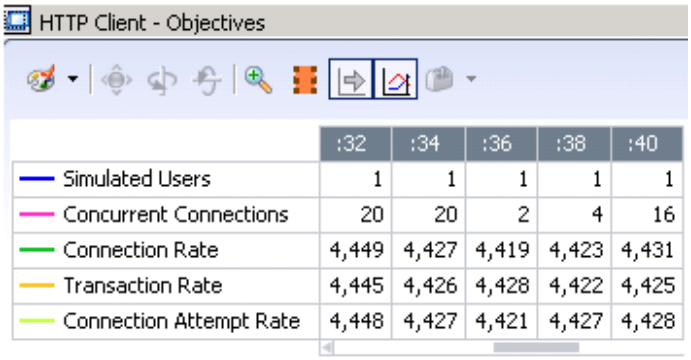


Figure 11. HTTP Client Statistics View – showing statistics for the duration of the test

The TCP connections view can quickly identify connectivity issues or indicate if the device is unable to keep up with the targeted test objective.

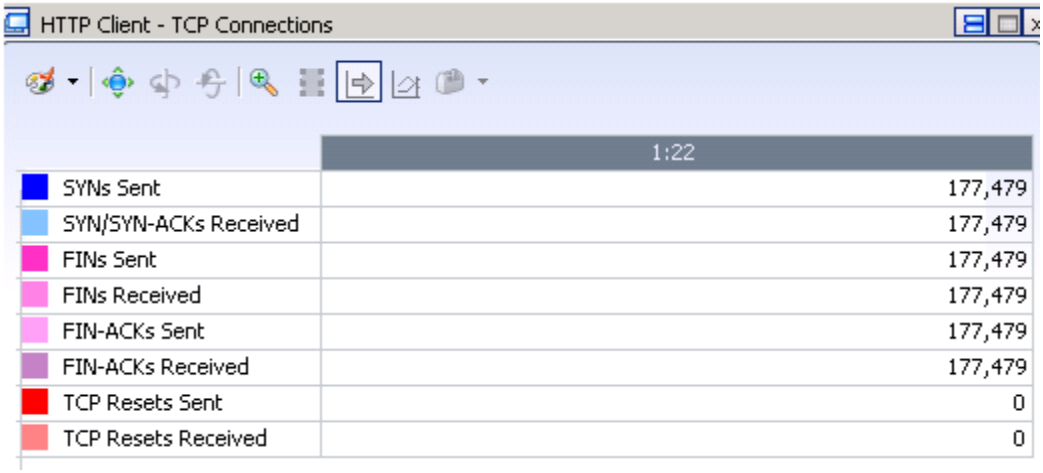


Figure 12. TCP Statistics View – showing the type of TCP packet exchanges

Troubleshooting and Diagnostics

Table 6. Troubleshooting and diagnostics

Issue	Diagnosis, Suggestions
Addition of more test ports does not increase performance	The DUT may have reached saturation. Check for TCP resets received by the client or server. In the case where the DUT is terminating connections from the client side, also check the device statistics for CPU utilization.
A large number of TCP resets received on the client and/or server side, during the beginning on the test – during ramp up. When in steady-state (Sustain), there are no or very minimal TCP timeouts and retries seen on the client and/or server side.	This indicates that the DUT or the test tool is “ramping up” and that there are many TCP sessions being opened and the DUT may not be “ready” to handle them. If the device uses multiple processing cores, then this behavior is possible when the additional load “turns on” the processing cores, but not before that.
A large number of TCP resets are received on the client and/or server side, throughout the test.	If there are continuous failures observed during steady-state, it’s possible that the device is reaching saturation. Reduce the objective until the failures are acceptable.
A small number of TCP failures are observed (timeout/retry), is this acceptable?	In general, small levels of TCP failures should be an acceptable behavior when a device is running at its maximum level. The transient nature of network, device and TCP behavior cannot guarantee zero failures. However, using the no failures is subjective, and may be required.

Test Case: Maximum Concurrent Connections

Overview

Determine the maximum number of active TCP sessions the DUT can sustain.

An active TCP session can be measured as follows:

- Create a TCP connection establishment (SYN, SYN-ACK, ACK), followed by several layer 7 transactions (Request, Response), and TCP session teardown.

In general, the duration of an established TCP connection is a configurable parameter, based on the scenario.

In general, the maximum number on concurrent connections that a DUT can handle is a function of DUT's memory; the higher the memory allocated to service connections, the larger the value.

Objective

Performance metrics required: Maximum concurrent connections.

This metric has real-world significance in that it provides a raw performance metric of DUT scalability and support for a large number of sustained TCP connections. For example, web services do millions of transactions per day, and the total concurrent connections a web server or server load balancer can handle is critical to ensure a surge in transactions or a long lived connection is maintained successfully.

Setup

The setup requires at least one server and one client port. The HTTP client traffic will pass through the DUT to reach the HTTP server. The HTTP client and server must be connected to the DUT using a switched network.

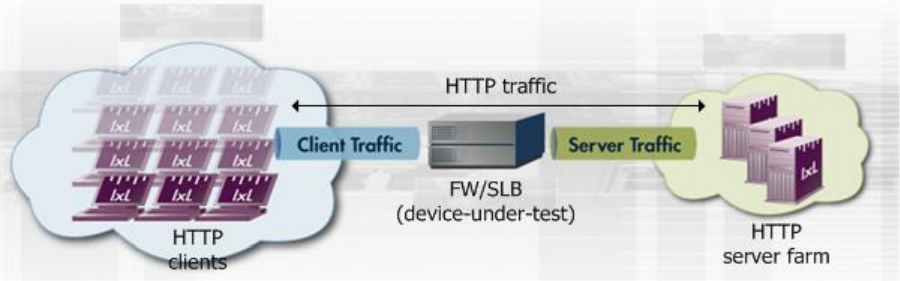


Figure 13. Concurrent connections test setup

Test Variables

Test Tool Variables

The following test configuration parameters provide the flexibility to create the traffic profile that a device would experience in a production network.

Table 7. Test configuration parameters

Parameter	Description
HTTP clients	100 IP addresses or more, use sequential or “use all” IP addresses
HTTP client parameters	HTTP1.0 without keep-alive 20 TCP connections per user or more 1 Transaction per TCP connection
HTTP client command list	1 GET command – payload of 1 byte
TCP Parameters	TCP RX and TX buffer at 1024 bytes
HTTP servers	1 per Ixia test port, or more
HTTP server parameters	Random response delay – 0 – 20 ms Response timeout – 300 ms

DUT Variables

There are several DUT scenarios. The following table outlines some of the capabilities of the DUT that can be switched on to run in a certain mode.

Table 8. Sample DUT scenarios

Device(s)	Variation	Description
Server load balancer	Activate packet filtering rules	• Configure SLB engine for “stickiness” • Change the algorithm for load balancing • Use Bridged or Routed Mode for servers
Firewall	Activate access control rules	• Configure Network Address Translation or Disable for Routed Mode
Firewall security device	Enable deep content inspection (DPI) rules	• Advanced application-aware inspection engines enabled • IDS or threat prevention mechanisms enabled

Step-by-Step Instructions

Configure the test to run a baseline test, which is an Ixia port-to-port test, in order to verify the test tool's performance. Once you have obtained the baseline performance, setup the DUT and the test tool as per the **Setup** section above. Refer to the **Test and DUT Variables** section for recommended configurations. Note that the network configurations must change between running the port-to-port and DUT test. Physical cabling will change to connect the test ports to the DUT. A layer 2 switch that has a high-performance backplane is highly recommended.

Reference baseline performance: Fill in the table below with the baseline performance to use as a reference of the test tool's performance, based on the quantity and type of hardware available.

Table 9. Reference baseline performance form

Performance indicator	Value per port pair	Load module type
Concurrent Connections		

1. Launch IxLoad. In the main window, you will be presented with the **Scenario Editor** window. All the test configurations will be made here.

   To get familiar with the IxLoad GUI, see the Getting Started Guide section.

2. Add the client **NetTraffic** object. Configure the **Client** network with total IP count, gateway and VLAN, if used.

   Add the server **NetTraffic**. Also configure the total number of servers that will be used. For performance testing, use *1* server IP per test port.

   For a step-by-step workflow, see Appendix A.

Figure 14. IxLoad Scenario Editor view with Client and Server side NetTraffics and Activities

3. The TCP parameters that are used for a specific test type are important when optimizing the test tool. Refer to the **Test Variables** section to set the correct TCP parameters.

There are several other parameters that can be changed. Leave them at their defaults values unless you need to change them for testing requirements.

For a step-by-step workflow, see Appendix B.

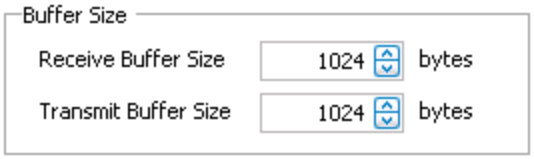


Figure 15. TCP Buffer Settings Dialogue

4. Configure the **HTTP server**. Add the **HTTP Server Activity** to the server **NetTraffic**. The defaults are sufficient for this testing.

   For a step-by-step workflow, see Appendix C.

5. Configure the **HTTP client**. Add the **HTTP Client Activity** to the client **NetTraffic**. Refer to **Test Variables** section to configure the HTTP parameters.

   You can use advanced network mapping capabilities to use sequence IPs or use all IPs configured.

   For a step-by-step workflow, see Appendix D.

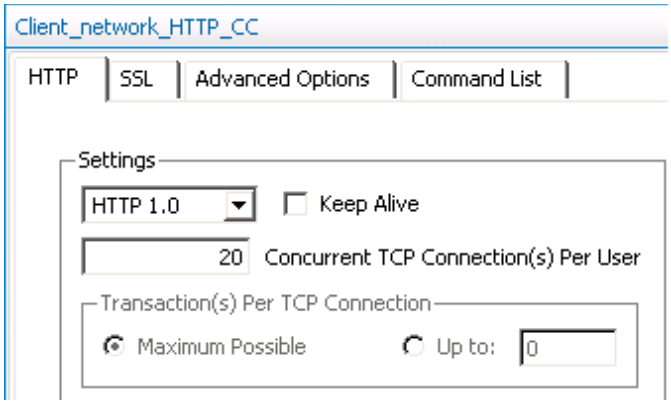


Figure 16. HTTP Client Protocol Settings Dialogue

6. Having setup the client and server networks and the traffic profile, the test objective can now be configured.

   Go to the **Timeline and Objectives** view. The test **Objective** can be applied on a per-activity or per-protocol basis. The iterative objectives will be set here and will be used between test runs to find the maximum concurrent connections for the device.

   The following should be configured:

   - **Test Objective**. Begin by attempting to send a large number of connections per second through the DUT. If the published or expected value for **MAX_CONN** is known, this is a good starting point, and will become the targeted value for the test (**TARGET_MAX_CONN**).

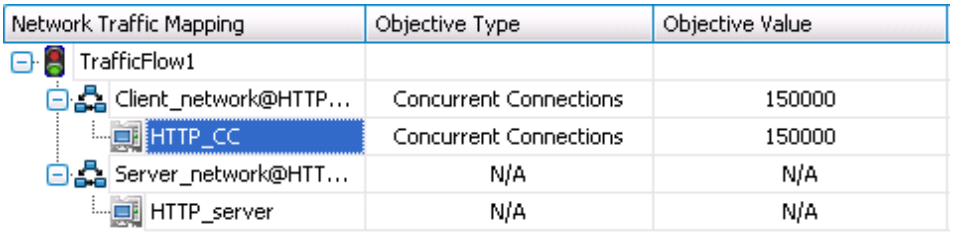


Figure 17. Test Objective Settings Dialogue

For a step-by-step workflow, see Appendix E.

7. Once the Test Objective is set, the **Port CPU** on the bottom indicates the total number of ports that is required. See the Appendix below on adding the ports to the test.

   For a step-by-step workflow, see Appendix F.

   Run the test for few minutes for the performance to attempt to reach a steady-state. Steady state is referred as “Sustain” duration in the test. Continue to monitor the DUT for the target rate and any failure/error counters. See the Results Analysis section for important statistics and diagnostics information.

   In most cases interpretation of the statistics is non-trivial, including what they mean under different circumstance. The **Results Analysis** section that follows provides a diagnostics-based approach to highlight some common scenarios, the statistics being reported, and how to interpret them.

8. Iterate through the test setting **TARGET_MAX_CONN** to the steady value attained during the previous run. To determine when the DUT has reached its **MAX_CONN**, see the **Results Analysis** section on interpreting results before making a decision.

   The test tool can be started and stopped in the middle of a test cycle, or wait for it to be gracefully stopped using the test controls shown here. 

   For a step-by-step workflow, see Appendix F.

Results Analysis

The maximum active/concurrent TCP connections requires an iterative method in which the test is run multiple times, changing a number of test input parameters. The DUT configuration must also be configured so that it performs optimally, based on the **Test Variables** section.

The following lists the key performance statistics that must be monitored. These statistics help determine if the device has reached its saturation point and to identify issues. Interpreting the results in the correct manner will also ensure that transient network, device or test tool behavior does not create a false negative condition.

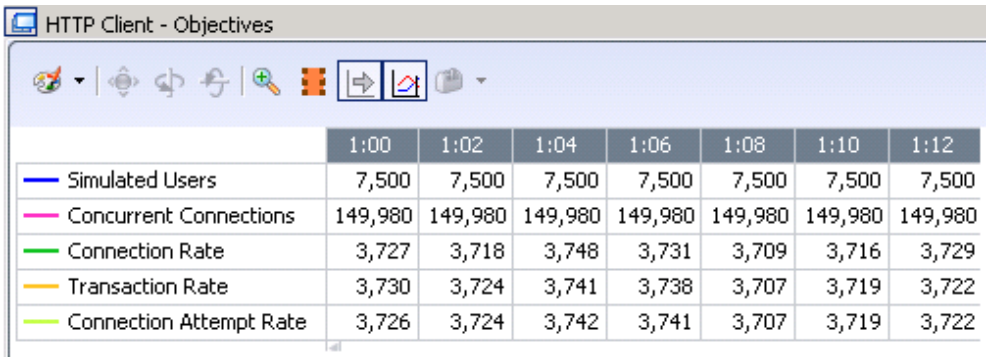
Table 10. Key performance statistics to monitor

Metric	Key Performance Indicators	Statistics View
Performance metrics	Connections/sec Total connections, Number of Simulated Users, Throughput	HTTP Client – Objectives HTTP Client – Throughput
Application level transactions Application level failure monitoring	Requests Sent, Successful, Failed Request Aborted, Timeouts, Session Timeouts Connect time, 4xx, 5xx errors	HTTP Client – Transactions HTTP Client – HTTP Failures HTTP Client – Latencies
TCP Connection Information TCP Failure monitoring	SYNs sent, SYN/SYN-ACKs Received RESET Sent, RESET Received, Retries, Timeouts	HTTP Client – TCP Connections HTTP Client – TCP Failures
Other Indicators	Per URL statistics, Response Codes	HTTP Client – Per URL HTTP Client – xx Codes

Real-Time Statistics

The graph below provides a view of the real-time statistics for the test. Real-time statistics provide instant access to key statistics that should be examined for failures at the TCP and HTTP protocol level.

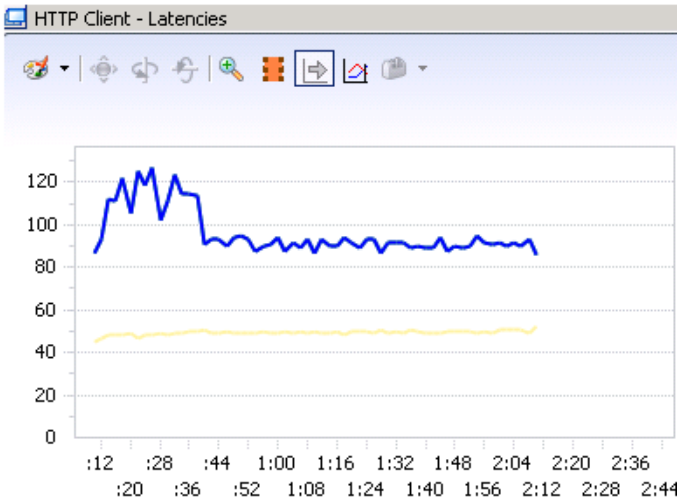
The statistics below are real-time observations of the achieved performance. The **Concurrent Connection** statistic shows that the DUT is able to sustain approximately 150,000 connections.



	1:00	1:02	1:04	1:06	1:08	1:10	1:12
Simulated Users	7,500	7,500	7,500	7,500	7,500	7,500	7,500
Concurrent Connections	149,980	149,980	149,980	149,980	149,980	149,980	149,980
Connection Rate	3,727	3,718	3,748	3,731	3,709	3,716	3,729
Transaction Rate	3,730	3,724	3,741	3,738	3,707	3,719	3,722
Connection Attempt Rate	3,726	3,724	3,742	3,741	3,707	3,719	3,722

Figure 18. HTTP Client Statistics View – Showing concurrent connections results

To identify if a device has reached its active connections limit, refer to the **Latencies** view. In this graph, the **TTFB** was higher during the "ramp-up" period to indicate the device/server accepting a burst of connection requests. An increasing **Connect Time** and/or **TTFB** is an indication that the device is slowing down.



	:12	:14	:16	:18	:20	:22
Connect Time (µs)	47	48	48	48	49	47
Time To First Byte (TTFB) (µs)	93	112	111	122	106	125
Time To Last Byte (TTLB) (µs)	93	112	111	122	106	125

Figure 19. HTTP Client Latency Statistics View

Troubleshooting and Diagnostics

Table 11. Troubleshooting and diagnostics

Issue	Diagnosis, Suggestions
The CC never reaches steady state; it's always moving up and down and the variation is reasonably high.	If device is not reaching steady-state, check the Connect time, and TCP failures. If the Connect time keeps increasing, then it's an indication that the device may not be able to service or maintain any connections. Check the Simulated Users count – if it is very high then it's an indication that the test tool is attempting to reach the target and its system resources are depleting; add more test ports or check the configuration to determine any possible issue.
What are the optimal statistics that can be used to certify that the optimal concurrent connections (CC) metric has been reached?	The relatively quick way to know if the device is at maximum capacity is to incrementally add new test ports, and see if the overall CC increases. If TCP failures occur and new connections are failing, then it's an indication that the limit has been reached. Other indications include a high latency, including Connect Time, TTFB and TTLB. Concurrent connections metric is a function of system memory; look at the memory usage on the device, should indicate it's near its high water mark.

Test Case: Maximum Transactions per Second

Overview

This test will determine the maximum transaction rate that a device can support. A transaction refers to a request sent and its corresponding response.

For example, when a web browser connects to a web site, first it establishes an initial TCP connection with a three-way TCP handshake. The page requested may contain several objects, such as web pages, images, style guides for the browser to use, flash or embedded objects, and active scripts. Following the initial TCP connection, multiple objects are downloaded to the browser in sequence or in parallel, using a HTTP feature called pipelining.

For example Browsing to http://www.ixiacom.com shows that a total of 87 “Requests” were made.

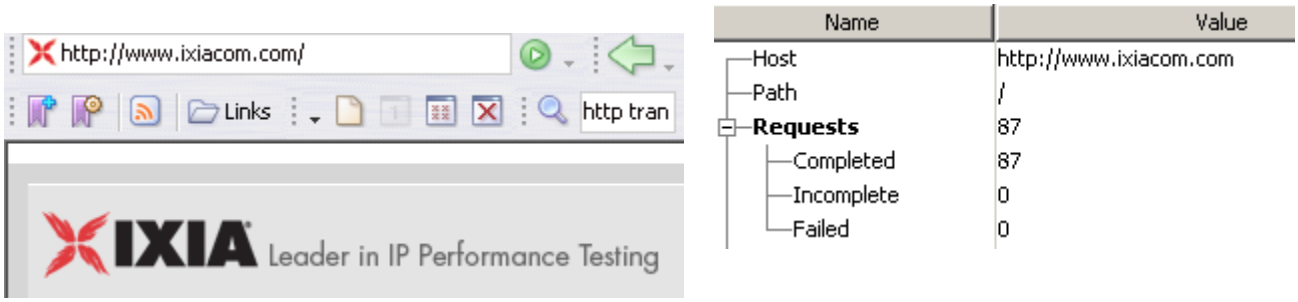


Figure 20. One URL request results in 87 individual requests for all the objects on the web page

The timeline below shows the order of the individual transactions, the type, size and time it look to download each object.

#	Resource	...	Mime Type	Header	Body	Time	Host
1	<default>	...	text/html	570 B	34.06 KB	250ms	www.ixiacom.com
2	style.css	...	text/css	229 B	19.50 KB	281ms	www.ixiacom.com
3	print.css	...	text/css	229 B	10.69 KB	172ms	www.ixiacom.com
4	superfish.css	...	text/css	228 B	5.96 KB	172ms	www.ixiacom.com
5	bg_header.gif	...	image/gif	227 B	450 B	172ms	www.ixiacom.com
6	logo.gif	...	image/gif	228 B	2.35 KB	204ms	www.ixiacom.com
7	btn_ok.gif	...	image/gif	227 B	304 B	203ms	www.ixiacom.com
8	home_big.jpg	...	image/jpeg	231 B	25.34 KB	203ms	www.ixiacom.com
9	t_press.gif	...	image/gif	227 B	290 B	312ms	www.ixiacom.com
10	bg_content.gif	...	image/gif	227 B	269 B	297ms	www.ixiacom.com

Figure 21. A break down on each request with corresponding object type and size

A single TCP connection usually supports multiple transactions. This is possible when using HTTP 1.0 with keep-alive and HTTP 1.1 by default. The number of transactions per TCP connection is a configurable option for most operating systems and browsers.

For example, in the case where the **Transactions per TCP Connection** is set to *3*, the following illustration shows multiple TCP connections and their associated transactions for each TCP connection.

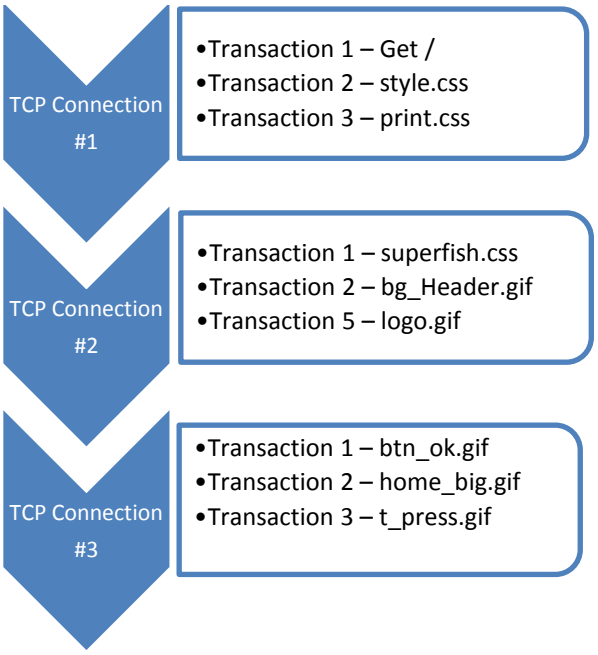


Figure 22. Illustration of the transaction distribution over the TPC connections

The test must be setup in a way to minimize the number of TCP connections, so as to allow the largest number of application level transactions.

Objective

Performance metrics required: Maximum transactions per second

Determining the transactions per second has real world significance in measuring the speed and response of downloading pages and objects. It can be used to determine the quality of the user's experience when browsing web sites and interacting with various web applications, including social web sites, photo sites and multimedia/video content.

Setup

The setup requires at least one server and one client port. The HTTP client traffic will pass through the DUT to reach the HTTP server. The HTTP client and server must be connected to the DUT using a switched network.

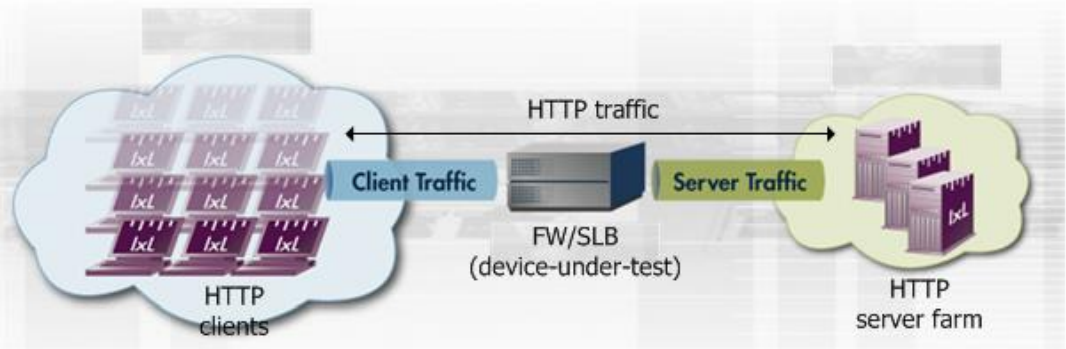


Figure 23. Transactions per second test setup

Test Variables

Test Tool Variables

The following test configuration parameters provide the flexibility to create a traffic profile that a device would experience in a production network.

Table 12. Test configuration parameters

Parameter	Description
HTTP clients	100 IP addresses or more, use sequential or “use all” IP addresses
HTTP client parameters	HTTP 1.1 20 TCP connections per user Maximum transactions per TCP connection
HTTP client pages to request	1 GET command – payload of 1 byte
TCP Parameters	TCP RX and TX buffer at 4096 bytes
HTTP servers	1 per Ixia test port, or more
HTTP server parameters	Random response delay – 0 – 20 ms Response timeout – 300 ms

DUT Variables

There are several DUT scenarios. The following table outlines some of the capabilities of the DUT that can be switched on to run in a certain mode.

Table 13. Sample DUT scenarios

Device(s)	Variation	Description
Server load balancer	Activate packet filtering rules	• Configure SLB engine for “stickiness” • Change the algorithm for load balancing • Use Bridged or Routed Mode for servers
Firewall	Activate access control rules	• Configure Network Address Translation or Disable for Routed Mode
Firewall security device	Enable deep content inspection (DPI) rules	• Advanced application-aware inspection engines enabled • IDS or threat prevention mechanisms enabled

Step-by-Step Instructions

Configure the test to run a baseline test, which is an Ixia port-to-port test, in order to verify the test tool’s performance. Once you have obtained the baseline performance, setup the DUT and the test tool as per the **Setup** section above. Refer to the **Test Tool and DUT Variables** section for recommended configurations. Note that the network configurations must change between running the port-to-port and DUT test. Physical cabling will change to connect the test ports to the DUT. A layer 2 switch that has a high-performance backplane is highly recommended.

Reference baseline performance: Fill in the table below with the baseline performance to use as a reference of the test tool’s performance, based on the quantity and type of hardware available.

Table 14. Reference baseline performance form

Performance indicator	Value per port pair	Load module type
Transactions/sec		

1. Launch IxLoad. In the main window, you will be presented with the **Scenario Editor** window. All test configurations will be performed here.

To become familiar with the IxLoad GUI, see the Getting Started Guide section.

2. Add the client **NetTraffic** Object. Configure the Client network with total IP count, gateway and VLAN, if used.

   Add the server NetTraffic. Also configure the total number of servers that will be used. For performance testing, use 1 server IP per test port.

For a step-by-step workflow, see Appendix A.

Figure 24. IxLoad Scenario Editor view with client and server side NetTraffics and Activities

3. The TCP parameters that are used for a specific test type are important for test tool optimization. Refer to the **Test Variables** section to set the correct TCP parameters.

   There are several other parameters that can be changed. Leave them set to their default values unless you need to change them for testing requirements.

   For a step-by-step workflow, see Appendix B.

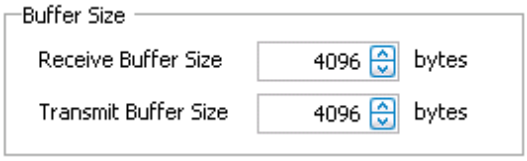


Figure 25. TCP Buffer Settings Dialogue

4. Configure the **HTTP server**. Add the **HTTP Server Activity** to the server **NetTraffic**. The defaults are sufficient for this testing.

   For a step-by-step workflow, see Appendix C.

5. Configure the **HTTP client**. Add the **HTTP Client Activity** to the client **NetTraffic**. Refer to **Test Variables** section to configure the HTTP parameters.

   You can use advanced network mapping capabilities to use sequence IPs or use all IPs configured.

   For a step-by-step workflow, see Appendix D.

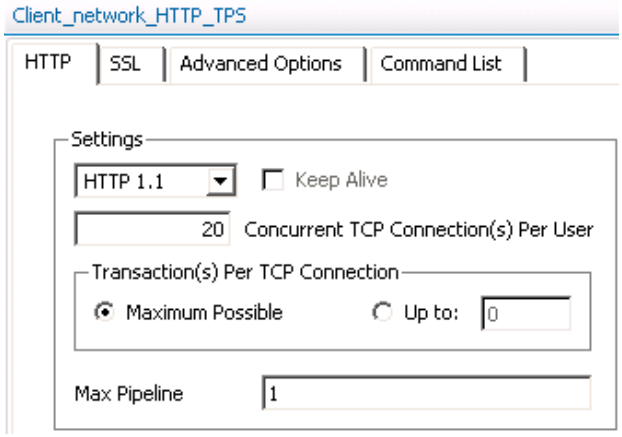


Figure 1 HTTP Client Protocol Settings Dialogue

6. Having setup the client and server networks and the traffic profile, the test objective can now be configured.

   Go to the **Timeline and Objective** view. The test **Objective** can be applied on a per-activity or per-protocol basis. The iterative objectives will be set here and will be used between test runs to find the maximum TPS for the device.

   The following should be configured:

   - **Test Objective**. Begin by attempting to send a large number of connections per second through the DUT. If the published or expected value for **MAX_TPS** is known, this value is a good starting point, and will become the targeted value for the test (**TARGET_MAX_TPS**).

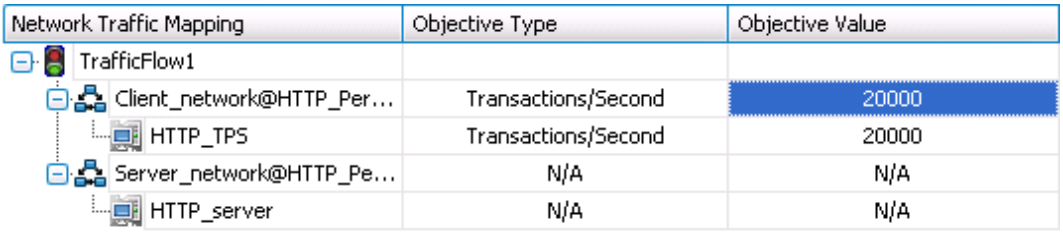


Figure 26. Test Objective Settings Dialogue

   For a step-by-step workflow, see Appendix E.

7. Once the **Test Objective** is set, the **Port CPU** on the bottom indicates the total number of ports that are required. See the Appendix F below on adding the ports to the test.

   For a step-by-step workflow, see Appendix F.

   Run the test for few minutes to allow the DUT to reach a steady state. Steady state is referred to as the **Sustain** duration in the test. Continue to monitor the DUT for the target rate and any failure/error counters. See the **Results Analysis** section for important statistics and diagnostics information.

   In most cases interpretation of the statistics is non-trivial, including what they mean under different circumstance. The **Results Analysis** section that follows provides a diagnostics-based approach to highlight some common scenarios, the statistics being reported, and how to interpret them.

8. Iterate through the test setting **TARGET_MAX_TPS** to the steady value attained during the previous run. To determine when the DUT has reached its **MAX_TPS**, see the **Results Analysis** section on interpreting results before making a decision.

   The test tool can be started and stopped in the middle of a test cycle, or wait for it to be gracefully stopped using the test controls shown here. 

   For a step-by-step workflow, see Appendix F.

Results Analysis

The maximum transactions per second performance requires an iterative method in which the test is run multiple times, changing a number of test input parameters. The DUT configuration must also be configured so that it performs optimally, based on the **Test Var**iables section.

The following the key performance statistics that must be monitored. The importance of these statistics is that it helps identify if the device has reached its saturation point, and identify issues. Also interpreting the results in the right manner will ensure that transient network, device or test tool behavior do not create a false negative condition.

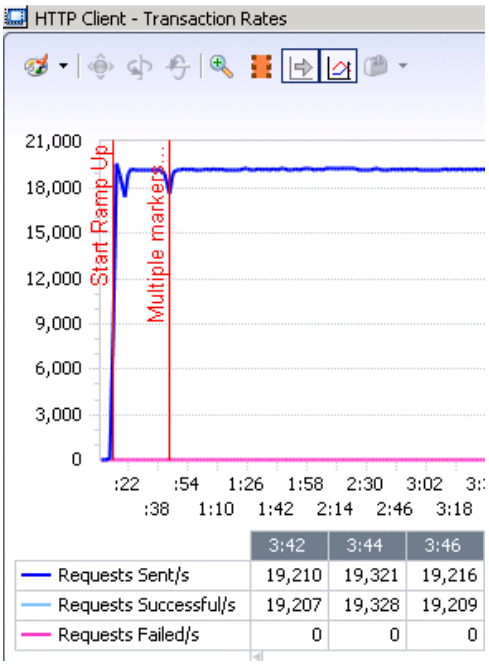
Table 15. Key performance indicators that should be monitored

Metric	Key Performance Indicators	Statistics View
Performance metrics	Connections/sec Total connections, Number of Simulated Users, Throughput	HTTP Client – Objectives HTTP Client – Throughput
Application level transactions Application level failure monitoring	Requests Sent, Successful, Failed Request Aborted, Timeouts, Session Timeouts Connect time, 4xx, 5xx errors	HTTP Client – Transactions HTTP Client – HTTP Failures HTTP Client – Latencies
TCP Connection Information TCP Failure monitoring	SYNs sent, SYN/SYN-ACKs Received RESET Sent, RESET Received, Retries, Timeouts	HTTP Client – TCP Connections HTTP Client – TCP Failures

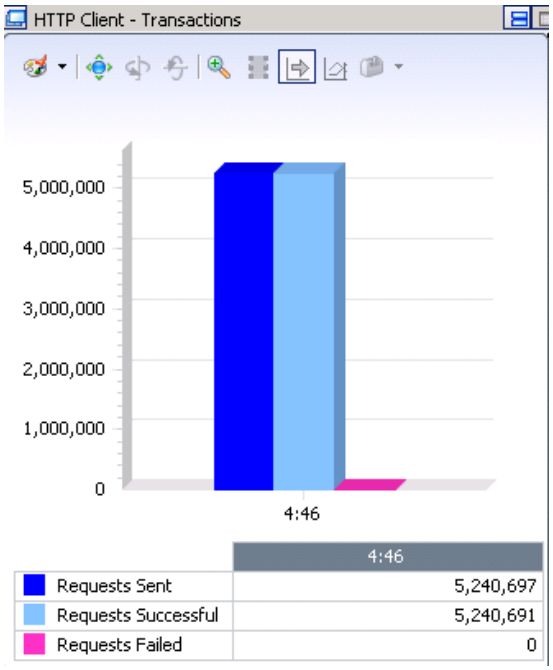
Real-Time Statistics

The graph below provides a view of the real-time statistics for the test. Real-time statistics provide instant access to key statistics that should be examined for failures at the TCP and HTTP protocol level.

The following graph indicates the target of 19000 TPS was reached. The total **Transactions Sent and Successful** indicates no failures at the application level.



HTTP Transaction Rate Statistics View



Cumulative HTTP Transactions Statistics View

Test Case: Maximum Throughput

Overview

This test will establish maximum throughput performance of a device.

It is important to note that the interpretation of application-layer throughput differs from the general understanding of how throughput is measured. See the illustration below for the two ways in which throughput can be computed and recorded.

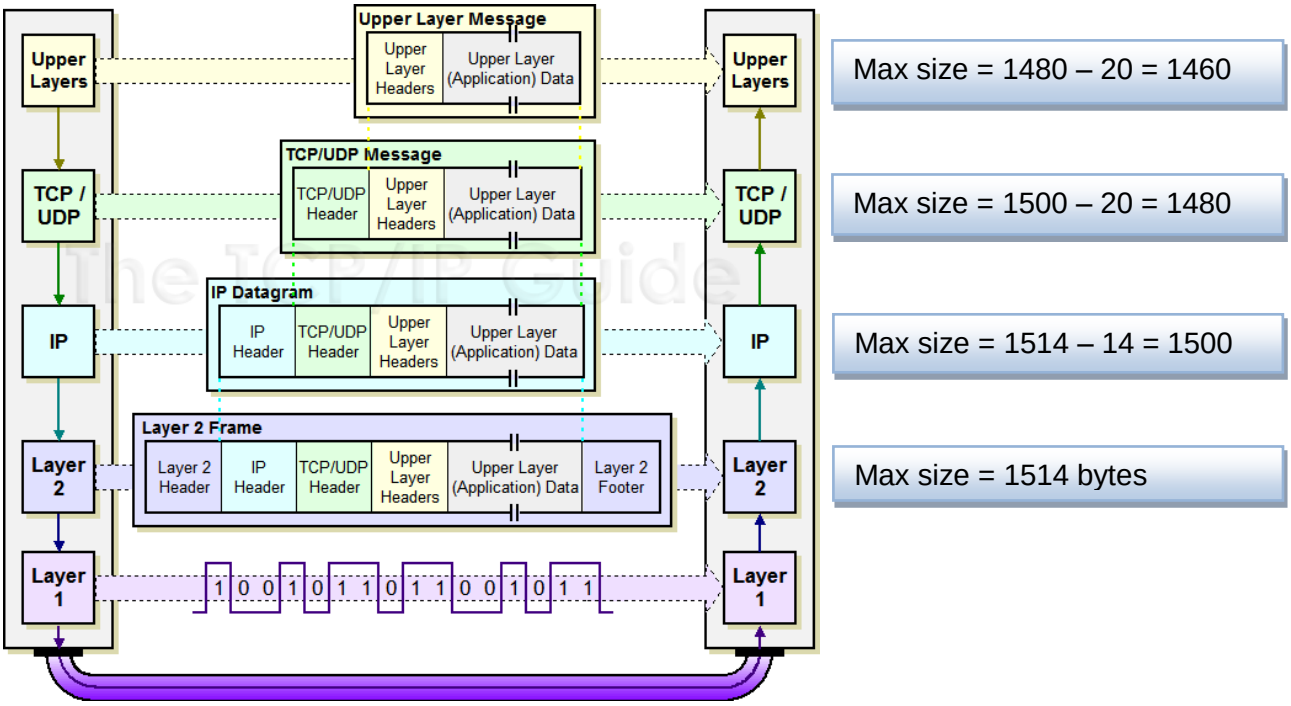


Figure 27. Layer 2 Frame, TCP/IP packet size guide

Throughput is computed for the entire layer 2 frame, or the total bits per second on the wire.

Goodput is also referred to as the application-layer throughput. It is generally used to provide a meaningful performance characterization of a device without factoring the overhead of TCP and lower protocols. Retransmitted packets are not factored into the goodput metric.

Objective

Performance metrics required: Maximum Throughput

Setup

The setup requires at least one server and one client port. The HTTP client traffic will pass through the DUT to reach the HTTP server. The HTTP client and server must be connected to the DUT using a switched network.

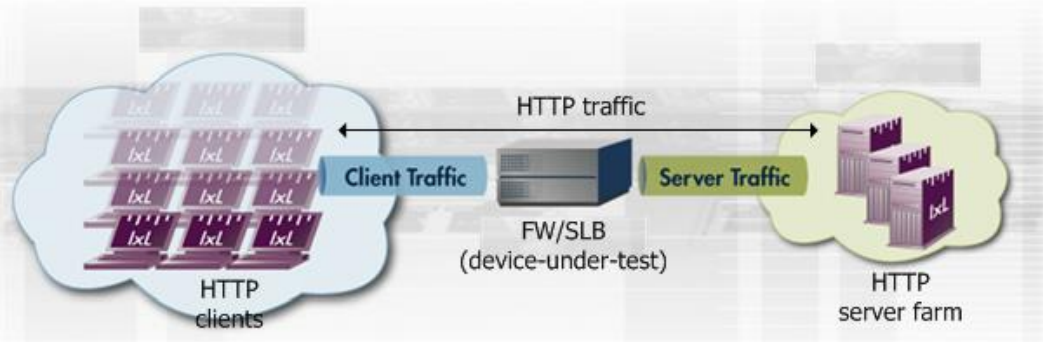


Figure 28. Throughput test setup

Test Variables

Test Tool Variables

The following test configuration parameters provide the flexibility to create the traffic profile that a device would experience in a production network.

Table 16. Test configuration parameters

Parameter	Description
HTTP clients	100 IP addresses or more, use sequential or “use all” IP addresses
HTTP client parameters	HTTP 1.1 20 TCP connections per user Maximum transactions per TCP connection
HTTP client pages to request	1 GET command – payload of 1MB, 512kB, 1024 bytes, 512 bytes
TCP Parameters	Client TCP - RX 32768 bytes, TX 4096 bytes Server TCP – RX 4096 bytes, TX 32768 bytes
MSS	1460, 500, 256, 128 bytes
HTTP servers	1 per Ixia test port, or more
HTTP server parameters	Random response delay – 0 – 20 ms Response timeout – 300 ms

DUT Variables

There are several DUT scenarios. The following table outlines some of the capabilities of the DUT that can be switched on to run in a certain mode.

Table 17. Sample DUT scenarios

Device(s)	Variation	Description
Server load balancer	Activate packet filtering rules	• Configure SLB engine for “stickiness” • Change the algorithm for load balancing • Use Bridged or Routed Mode for servers
Firewall	Activate access control rules	• Configure Network Address Translation or Disable for Routed Mode
Firewall security device	Enable deep content inspection (DPI) rules	• Advanced application-aware inspection engines enabled • IDS or threat prevention mechanisms enabled

Step-by-Step Instructions

Configure the test to run a baseline test, which is an Ixia port-to-port test, in order to verify the test tool’s performance. Once you have obtained the baseline performance, setup the DUT and the test tool as per the **Setup** section below. Refer to the **Test and DUT Variables** section for recommended configurations. Note that the network configurations must change between running the port-to-port and DUT test. Physical cabling will change to connect the test ports to the DUT. A layer 2 switch that has a high-performance backplane is highly recommended.

Reference baseline performance: Fill in the table below with the baseline performance to use as a reference of the test tool’s performance, based on the quantity and type of hardware available.

Table 18. Reference baseline performance form

Performance indicator	Value per port pair	Load module type
Throughput		

1. Launch IxLoad. In the main window, you will be presented with the **Scenario Editor** window. All the test configurations will be made here.

   To get familiar with the IxLoad GUI, see the Getting Started Guide section.

2. Add the client **NetTraffic** object. Configure the **Client** network with total IP count, gateway and VLAN, if used.

Add the server **NetTraffic**. Also configure the total number of servers that will be used. For performance testing, use *1* server IP per test port.

For a step-by-step workflow, see Appendix A.

Figure 29. IxLoad Scenario Editor view with Client and Server side NetTraffics and Activities

3. The TCP parameters that is used for a specific test type is important for optimizing the test tool. Refer to the **Test Variables** section to set the correct TCP parameters.

   There are several other parameters that can be changed. Leave them at their default values unless you need to change them for testing requirements.

   For a step-by-step workflow, see Appendix B.

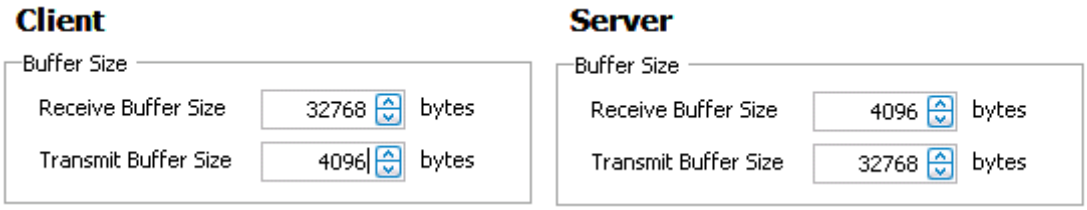


Figure 30. TCP Buffer Settings for Client and Server traffic

4. Configure the **HTTP server**. Add the **HTTP Server Activity** to the server **NetTraffic**. The defaults are sufficient for this testing.

   For a step-by-step workflow, see Appendix C.

5. Configure the **HTTP client**. Add the **HTTP Client Activity** to the client **NetTraffic**. Refer to **Test Variables** section to configure the HTTP parameters.

   You can use advanced network mapping capabilities to use sequence IPs or configure all IP addresses.

   For a step-by-step workflow, see Appendix D.

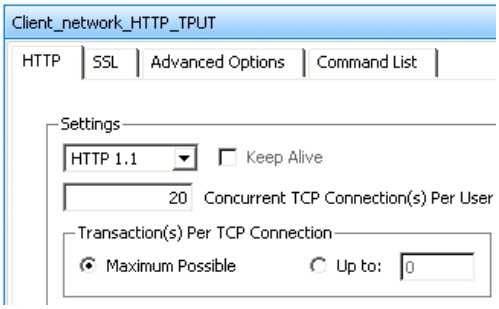


Figure 31. HTTP Client Protocol Settings Dialogue

6. Having setup client and server networks, and the traffic profile, the test objective can now be configured.

   Go to the **Timeline and Objective** view. The test **Objective** can be applied on a per-activity or per-protocol basis. The iterative objectives will be set here and will be used between test runs to find the maximum throughput for the device.

   The following should be configured:

   - **Test Objective**. Begin by attempting to send a large number of connections per second through the DUT. If the published or expected value for **MAX_TPUT** is known, this value is a good starting point, and will become the targeted value for the test (**TARGET_MAX_TPUT**).

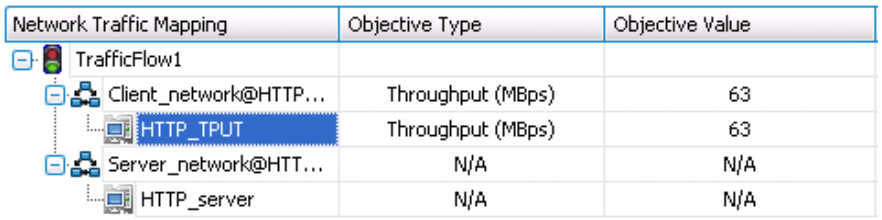


Figure 32. Test Objective Settings Dialogue

   For a step-by-step workflow, see Appendix E.

7. Once the **Test Objective** is set, the **Port CPU** on the bottom indicates the total number of ports that are required. See the Appendix F below on adding the ports to the test.

   For a step-by-step workflow, see Appendix F.

   Run the test for few minutes for the performance to reach a steady state. Steady state is referred as the **Sustain** duration in the test. Continue to monitor the DUT for the target rate and any failure/error counters. See the **Results Analysis** section for important statistics and diagnostics information.

   In most cases interpretation of the statistics is non-trivial, including what they mean under different circumstance. The **Results Analysis** section that follows provides a diagnostics-based approach to highlight some common scenarios, the statistics being reported, and how to interpret them.

8. Iterate through the test setting **TARGET_MAX_TPUT** to the steady value attained during the previous run. To determine when the DUT has reached its **MAX_TPUT**, see the **Results Analysis** section on interpreting results before making a decision.

   The test tool can be started, stopped in the middle of a test cycle, or wait for it to be gracefully stopped using the test controls shown here. 

   For a step-by-step workflow, see Appendix F.

Results Analysis

Finding the maximum throughput performance requires an iterative method in which the test is run multiple times, changing a number of the test input parameters. The DUT configuration must also be configured so that it performs optimally, based on the **Test Variables** section.

The following are the key performance statistics that must be monitored. The importance of these statistics is that they help identify if the device has reached its saturation point and to identify issues. Also interpreting the results in the correct manner will ensure that transient network, device or test tool behavior do not create a false negative condition.

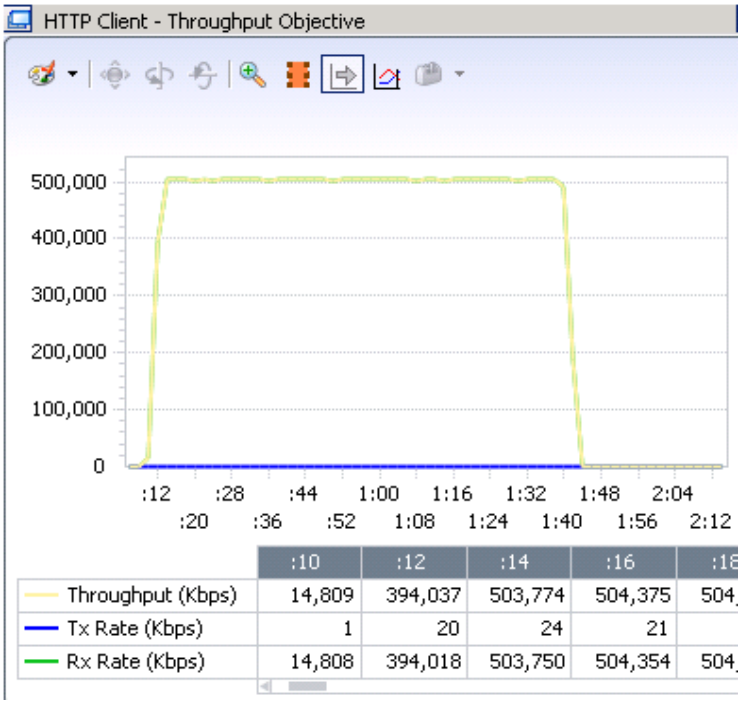
Table 19. Key performance statistics that should be monitored

Metric	Key Performance Indicators	Statistics View
Performance metrics	Connections/sec Total connections, Number of Simulated Users, Throughput	HTTP Client – Objectives HTTP Client – Throughput
Application level transactions Application level failure monitoring	Requests Sent, Successful, Failed Request Aborted, Timeouts, Session Timeouts Connect time, 4xx, 5xx errors	HTTP Client – Transactions HTTP Client – HTTP Failures HTTP Client – Latencies
TCP Connection Information TCP Failure monitoring	SYNs sent, SYN/SYN-ACKs Received RESET Sent, RESET Received, Retries, Timeouts	HTTP Client – TCP Connections HTTP Client – TCP Failures

Real-Time Statistics

The graph below provides a view of the real-time statistics for the test. Real-time statistics provide instant access to key statistics that should be examined for failures at the TCP and HTTP protocol level.

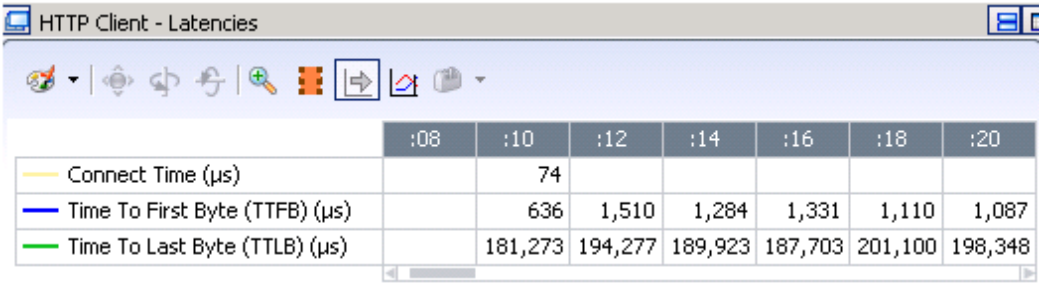
The following graph indicates the target throughput of 500Mbps was reached. The throughput is the **Goodput,** as described earlier. The Tx and Rx rates add to the overall throughput.



	:10	:12	:14	:16	:18
Throughput (Kbps)	14,809	394,037	503,774	504,375	504
Tx Rate (Kbps)	1	20	24	21	
Rx Rate (Kbps)	14,808	394,018	503,750	504,354	504

Figure 33. HTTP Client Throughput Statistics View

The latency view is useful for the throughput test to indicate that the DUT is able to process the packets in a timely manner. The graph below shows the system is performing well – the **TTLB** is high as expected because the requested page is 1Mbyte and the **TTLB** also indicates an average of 200ms sustained latency in delivering the requests.



	:08	:10	:12	:14	:16	:18	:20
Connect Time (µs)		74					
Time To First Byte (TTFB) (µs)		636	1,510	1,284	1,331	1,110	1,087
Time To Last Byte (TTLB) (µs)		181,273	194,277	189,923	187,703	201,100	198,348

Figure 34. HTTP Client Latency Statistics View

Troubleshooting and Diagnostics

Table 20. Troubleshooting and diagnostics

Issue	Diagnosis, Suggestions
The target throughput is not reached. Throughput goes up and down.	If the device is not reaching steady-state, check the TCP failures. High TCP timeout and RST packets can indicate that the device is unable to handle load. Add more test ports. If the result is the same, then the device cannot process more packets.
The Simulated User count is increasing, but the test tool is unable to reach target throughput.	Check the Simulated Users count – if it is very high then it's an indication that the test tool is attempting to reach the target and its system resources are depleted; add more test tools or check the configuration to determine any possible issue. Check for TCP failures to indicate any network issues. Check device statistics for ingress and egress packet drops.

Test Case: Application Forwarding Performance under DoS Attacks

Overview

This test determines the degree of degradation that the denial of service (DoS) attacks have on a DUT's application forwarding performance.

Firewalls and DPI systems support advanced capabilities to protect it and the active user sessions from attacks. The security features of these devices add to the processing overhead of such devices and often come at the expense of impeding the overall performance of the system.

There are several approaches for testing the resiliency of a device under attack. This test focuses on determining the performance impact when the DUT it is subjected to a network-based attack, such as a SYN Flood.

Objective

Determine the impact of network-based attacks on the performance of an application-aware device while processing and forwarding legitimate traffic.

Setup

The setup requires at least one server and one client port. In this test the HTTP client traffic will pass through the DUT to reach the HTTP server. Then dynamic DoS (DDoS) and malicious traffic will be introduced, with the appropriate inspection engines enabled on the DUT. To test realistic network conditions, several other legitimate protocols can be added.

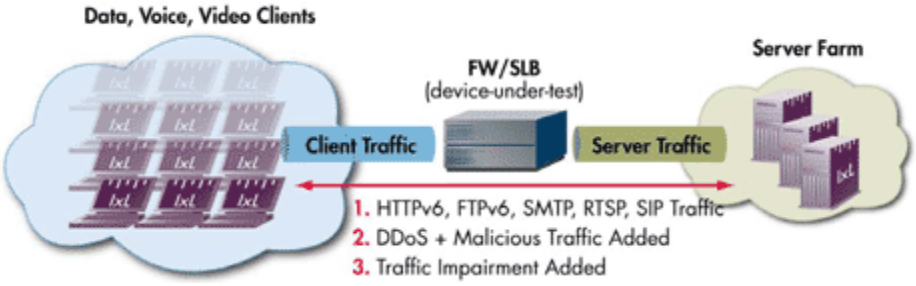


Figure 35. HTTP and DoS Attack Test Topology

Test Variables

Test Tool Variables

The following test configuration parameters provide the flexibility to create the traffic profile that a device would experience in a production network.

Table 21. Test tool variables

Parameter	Description
Client Network	100 IP addresses or more, use sequential or “use all” IP addresses
HTTP client parameters	HTTP/1.1 without keep-alive 3 TCP connections per user 1 Transaction per TCP connection
TCP parameters	TCP RX and TX buffer at 4096 bytes
HTTP client command list	1 GET command – payload of 128k-1024k byte
HTTP servers	1 per Ixia test port, or more
HTTP server parameters	Random response delay – 0 – 20 ms Response timeout – 300 ms
DoS attacks	ARP flood attack, evasive UDP attack, land attack, ping of death attack, ping sweep attack, reset flood attack, smurf attack, SYN flood attack, TCP scan attack, tear-drop attack, UDP flood attack, UDP scan attack, unreachable host attack and Xmas tree attack
Other protocols	FTP, SMTP, RTSP, SIP or combination

DUT Test Variables

There are several DUT scenarios. The following table outlines some of the capabilities of the DUT that can be switched on to run in a certain mode.

Table 22. Sample DUT scenarios

Device(s)	Variation	Description
Server load balancer	Activate packet filtering rules	• Configure SLB engine for “stickiness” • Change the algorithm for load balancing • Use Bridged or Routed Mode for servers
Firewall	Activate access control rules	• Configure Network Address Translation or Disable for Routed Mode
Firewall security device	Enable deep content inspection (DPI) rules	• Advanced application-aware inspection engines enabled • IDS or threat prevention mechanisms enabled

Step-by-Step Instructions

Configure the test to run a baseline test, which is an Ixia port-to-port test, in order to verify the test tool's performance. Once you have obtained the baseline performance, setup the DUT and the test tool as per the **Setup** section below. Refer to the **Test and DUT Variables** section for recommended configurations. Note that the network configurations must change between running the port-to-port and DUT test. Physical cabling will change to connect the test ports to the DUT. A layer 2 switch that has a high-performance backplane is highly recommended.

Reference baseline performance: Fill in the table below with the baseline performance to use as a reference of the test tool's performance, based on the quantity and type of hardware available.

Table 23. Reference baseline performance form

Performance indicator	Value per port pair	Load module type
Throughput		
Connections/sec		
Transactions/sec		

Once you have the baseline enable the security features of the DUT:

- Enable application-aware inspection engines for virus, spam and phishing attacks, which may be global parameters and/or access-lists
- Enable application-gateway or proxy services for specific protocols used in the test - e.g., SIP NAT traversal (STUN)

1. Launch IxLoad. In the main window, you will be presented with the **Scenario Editor** window. All test configurations will be performed here. To become familiar with the IxLoad GUI, see the Getting Started Guide section.
2. Add the client **NetTraffic** object. Configure the client network with total IP count, gateway and VLAN, if used.

Add the server **NetTraffic** and configure the total number of servers that will be used.

For a step-by-step workflow, see Appendix A.

Figure 36. IxLoad Scenario Editor view with client and server side NetTraffics and Activities

3. The TCP parameters that are used for a specific test type are important for optimizing the test tool. Refer to the Test Variables section to set the correct TCP parameters.

   There are several other parameters that can be changed. Leave them at their default values unless you need to change them for testing requirements.

   For a step-by-step workflow, see Appendix B.

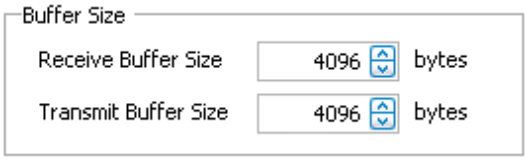


Figure 37. TCP Buffer Settings Dialogue

4. Add the **HTTP Server Activity** to the server **NetTraffic**. Configure the **HTTP Server Activity**; the default values should be sufficient for this test.

   For a step-by-step workflow, see Appendix C.

5. Add the **HTTP Client Activity** to the client **NettTaffic**. Configure the HTTP client with the parameters defined in the **Test Variables** section above.

   You can use advanced network mapping capabilities to use sequence IPs or use all IPs configured.

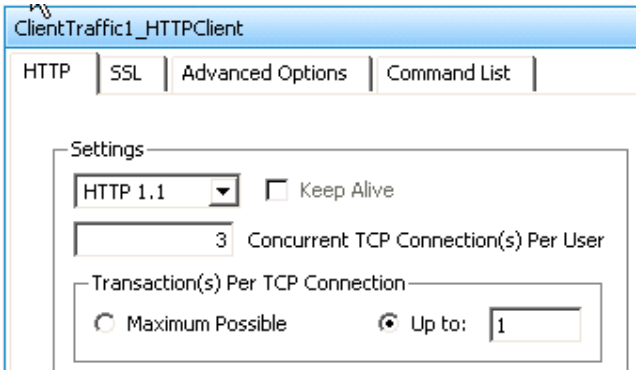


Figure 2 HTTP Client Protocol Settings Dialogue

For a step-by-step workflow, see Appendix D.

6. On the client traffic profile used to stress test the DUT, add a **DoSAttacks Activity**. Configure the **DoSAttack** client with the relevant DDoS attack signatures. You can optionally add other protocols to create a more stressful environment.

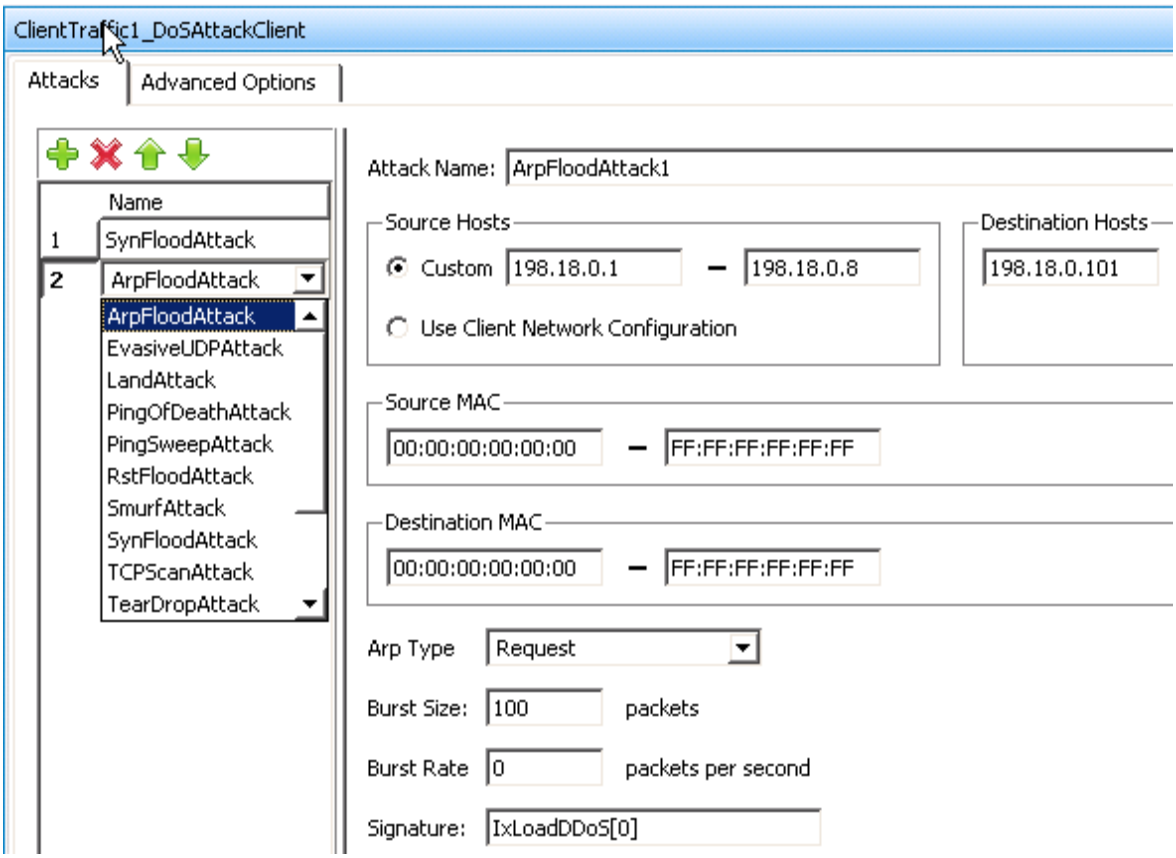


Figure 38. DoS Attack Client Settings Dialogue

If you want the attacks to originate from the same IP addresses as the legitimate HTTP traffic enable the **Use Client Network Configuration** option. Alternatively, you can originate the attack from a different set of IP addresses.

There are several layer 7 DoS attacks to consider; you can add multiple attacks by clicking the button. Use discretion in assembling the attacks to be initiated against the servers or DUT, and configure the **Destination Hosts** appropriately.

7. On the server side profile add a **PacketMonitorServer** activity to monitor any attacks that were not discarded by a DUT, that is attacks that make it through the DUT.

   To configure the **PacketMonitorServer** activity, simply select the corresponding **DDoS Client** activity. If the **Automatic** configuration mode is selected, the filters will be automatically imported from DoS attack configuration from the client network. Alternatively you can use the **Manual** configuration mode to specify custom signatures.

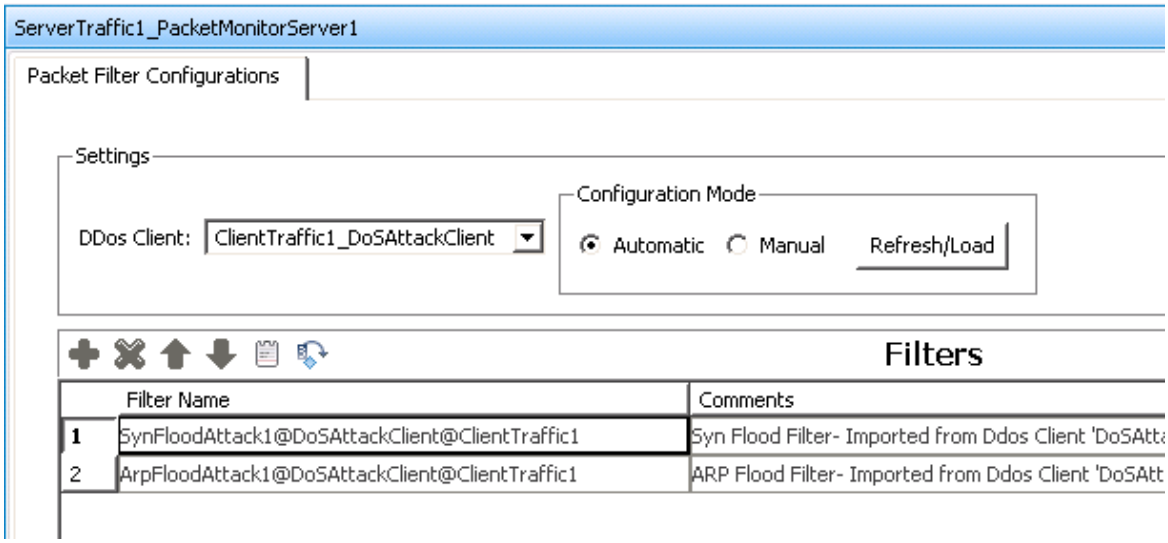


Figure 39. Packet Monitor Sever Settings Dialogue

8. Having setup client and server networks and the traffic profile, the test objective can now be configured.

   Go to the **Timeline and Objective** view. The test **Objective** can be applied on a per-activity or per-protocol basis. The iterative objectives will be set here and will be used between test runs to find the maximum TPS for the device.

   Begin with setting the Throughput objective or one of the other metrics at the maximum achieved with no DoS attacks – this performance metric is the Reference baseline performance that was determined first.

   For a step-by-step workflow, see Appendix E.

9. Once the **Test Objective** is set, the **Port CPU** on the bottom indicates the total number of ports that are required.

   For the **DoSAttack** activity, set the objective to *1* simulated user and run the test. Use an iterative process to increase the simulated users to find the point at which the throughput or desired objective starts to degrade.

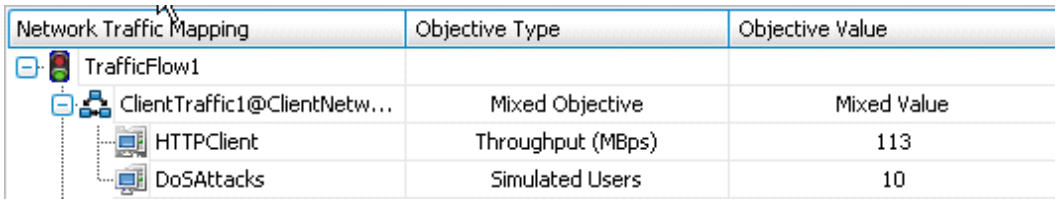


Figure 40. Test Objective Settings Dialogue

For a step-by-step workflow, see Appendix F.

10. Iterate through the test, setting different values for the **Simulated Users** objective for the DoS attack, which will gradually increase the intensity of the DoS attack directed at the DUT. Record the application throughput CPS, TPS and throughout metrics for the test. Monitor the DUT for the target rate and any failure/error counters. Stop the iterative process when the DUT application forwarding performance drops below an acceptable level.

Results Analysis

To analyze the impact of DoS attacks on the DUT application forwarding performance you need to compare the results from the performance baseline test case and results of DoS attack test case. Also it is critical to analyze how various types of DoS attacks impact the performance.

The following the key performance statistics that must be monitored. These statistics will help you identify if the device has reached its' saturation point, and identify issues.

Table 24. Key performance statistics to monitor

Metric	Key Performance Indicators	Statistics View
Performance metrics	Connections/sec Total connections, Number of Simulated Users, Throughput	HTTP Client – Objectives HTTP Client – Throughput
Application level transactions Application level failure monitoring	Requests Sent, Successful, Failed Request Aborted, Timeouts, Session Timeouts Connect time	HTTP Client – Transactions HTTP Client – HTTP Failures HTTP Client – Latencies
TCP Connection Information TCP Failure monitoring	SYNs sent, SYN/SYN-ACKs Received RESET Sent, RESET Received, Retries, Timeouts	HTTP Client – TCP Connections HTTP Client – TCP Failures
DoS Attacks	Successful, Failed Packets Bytes Sent	DDoS Client – Successful Packets DDoS Client – Failedl Packets DDoS Client – Bytes Sent
Packet Monitor	Packets Received, Filtered and Allowed	Packet Monitor Server – Packet Statistics Total

Real-Time Statistics

The graph below provides a view of the real-time statistics for the test. Real-time statistics provide instant access to key statistics that should be examined for failures at the TCP and HTTP protocol level.

In the following graph you can see the throughput value was 410Mbps before the DoS attacks began and how the throughput performance drops as the DoS attack intensity increases.

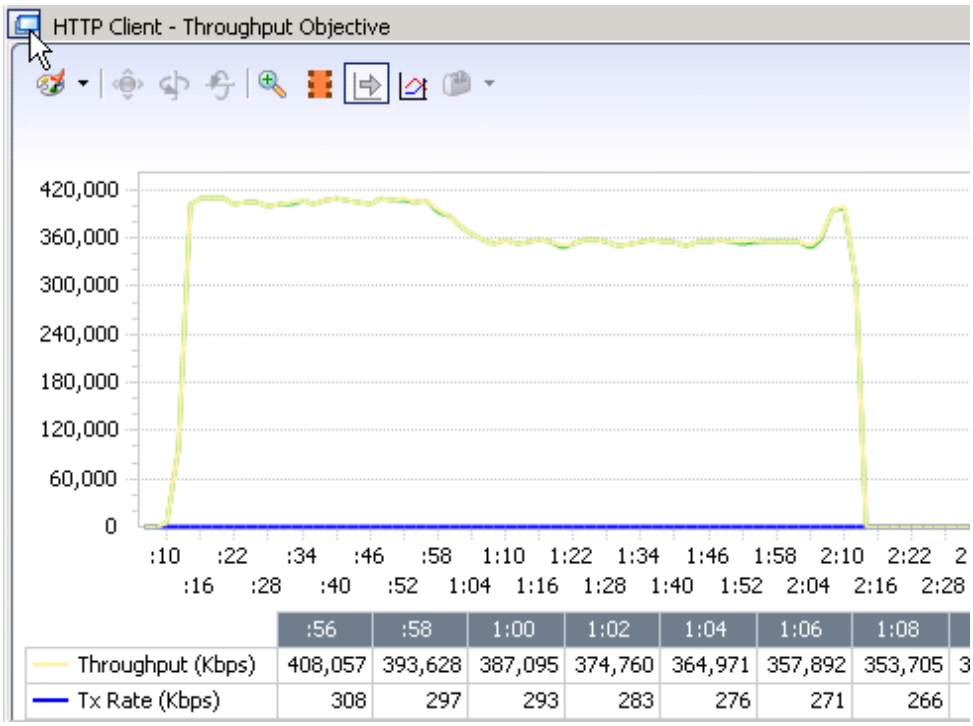
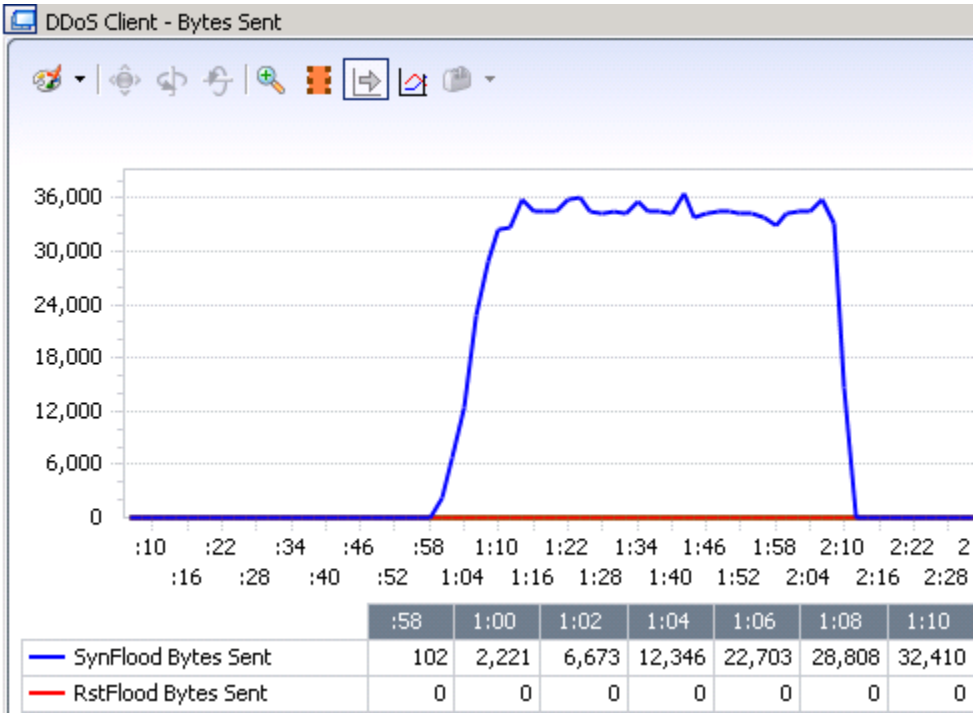


Figure 41. HTTP Throughput Statistics view – notice at 0.58 sec is the DoS attacks begin and the throughput starts to drop

This graph below shows the corresponding DoS attack rate. You can see at the start of the test that there were no DoS attacks generated and during that period the throughput graph showed a steady 410Mbps. When the DoS attacks started around 58 second the throughput degradation began.



	:58	1:00	1:02	1:04	1:06	1:08	1:10
SynFlood Bytes Sent	102	2,221	6,673	12,346	22,703	28,808	32,410
RstFlood Bytes Sent	0	0	0	0	0	0	0

Figure 42. DDoS Client Bytes Sent Statistics View – notice the time the SynFlood Attack begin and the corresponding effect on the throughput graph above

Other metric of interest are TCP and transactions failures. In some test runs, however, you may not see any TCP failures or transaction failures during a DoS attack. When you compare the total number of TCP connections serviced or total throughput during the DoS attack as in the case above you may notice degradation compared to the baseline test case values.

A jump in latency is also observed during DoS attacks, as shown below. At the same time at the DoS attacks start you see an increase the in HTTP time to last byte (**TTLB**) latency values. This indicates the device's inability to sufficiently transfer data at the inflection point when the attacks began.

	:52	:54	:56	:58	1:00	1:02	1:04	1:06	1:08
Connect Time (µs)	81	80	82	82	82	83	83	82	84
Time To First Byte (TTFB) (µs)	184	183	182	184	185	184	184	184	186
Time To Last Byte (TTLB) (µs)	30,425	30,069	31,202	31,644	32,787	33,649	34,280	34,821	34,288

Figure 43. HTTP Client Latency Statistics View – shows the latency gets higher at the same time as the DoS attacks begin

Utilizing the **Packet Monitor** statistics, the distribution of legitimate HTTP traffic and filtered traffic can be determined. In this case the filter was set to catch the SynFlood attacks.

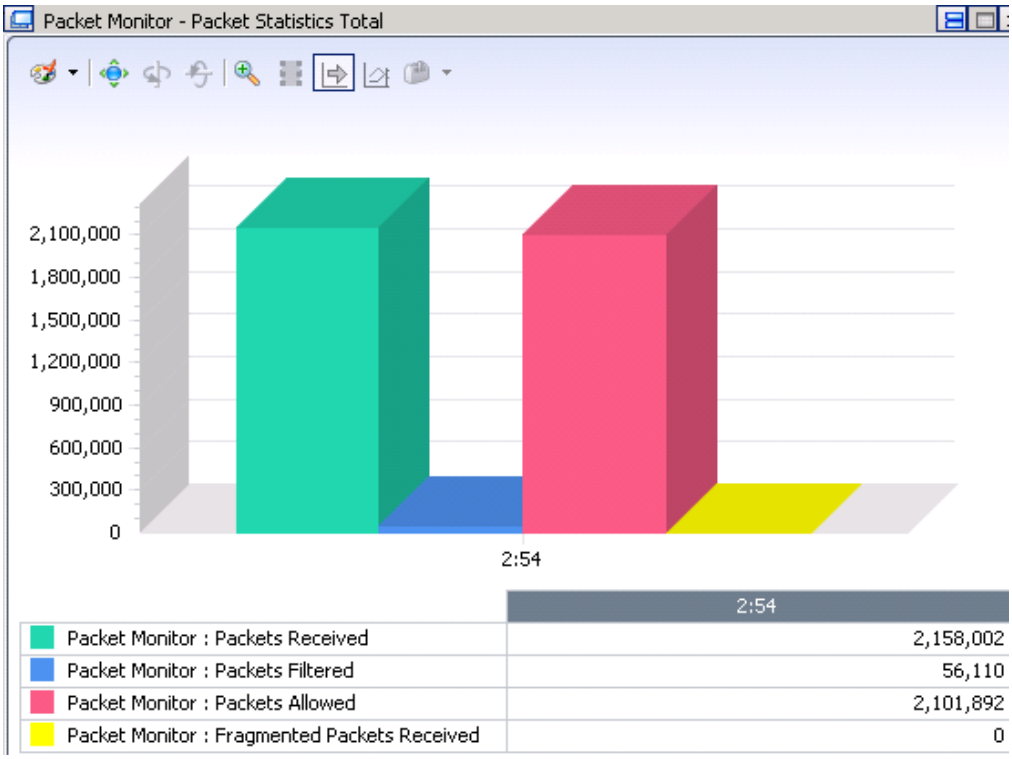


Figure 44. Packet Monitor Statistics View

Troubleshooting and Diagnostics

Table 25. Troubleshooting and diagnostics

Issue	Diagnosis, Suggestions
A large number of TCP resets are received on client and/or server side throughout the test.	If there are continuous failures observed during steady-state operation, it's possible that the device is reaching saturation due to DoS attacks. This is very common during SYN flood attacks.
The throughput goes up and down.	If the device does not reaching steady-state, check the TCP failures. High TCP timeout and RST packets can indicate issues with the device being unable to handle load due to the DoS attacks.
The Simulated User count is increasing. The test tool is unable to reach target Throughput	If the Simulated User count is increasing and the throughput is not met, it indicates the test tool is actively seeking to reach the target. Check for TCP failures to indicate the effects of the DoS attack.

Impact of Inspection Rules and Filters on Application Performance

Overview

Firewalls are predominantly used for protection of and access to networks and resources from trusted and untrusted sources. In the past firewalls supported a few key security functions such as IP routing, NAT/PAT, packet filtering based on IP type, TCP/UDP port, and blocking of traffic based on L2/L3 packet header information.

The rest of security and protection in a data center was done at the host level, using software and hardware solutions that provided anti-virus, anti-spam, web, VPN remote access and various application-aware filtering for more robust protection of server and host systems.

The picture below is a typical deployment of point solutions for protecting a network.

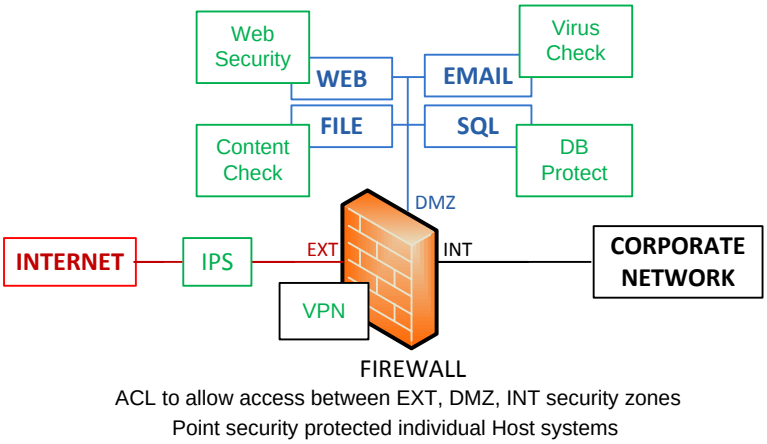


Figure 45. Enterprise network protection using point solutions

The increase of network-based attacks accelerated the need to establish a multi-vector threat protection networks on different network segments. The unification of various point solutions have provided much more knowledge and intelligence to today's modern firewalls. The intelligence comes from performing deep packet inspection on hundreds of applications in real-time to make access/deny decisions. Some of the key features that have come together into a single platform include application-aware and content-specific firewall inspection, malware and virus protection for mail and several other protocols that are used to carry out attacks, sophisticated web filtering to have visibility in the nature of web transactions, VPN integration and protection from outside clients, and high-risk threat mitigation using intrusion prevention techniques to protect internal resources from having to protect themselves.

Integrated Security Devices

The move towards an integrated security gateway solution is driven by the explosion of applications and services delivered over HTTP and other application protocols. Exploits are carried out using these standard protocols; as a result there are more ways to exploit application

and host vulnerabilities. Traditional firewalls that implement allow/deny rules based on IP+TCP port number are inadequate for identifying any such threats, since all its decisions are simply based on layer 4 information and not on the contents of the layer 7 payload.

An integrated security gateway/firewall that supports application and/or content-aware inspection fully understands application protocols. This includes requests, responses, status codes, timers, while maintaining the session context and state of all TCP and related secondary connections. Integrated gateways/firewalls make decisions based on the nature of content.

A firewall application proxy sits between clients and servers, inspecting and dropping all suspicious packets. It alleviates backend servers from ever interacting with untrusted clients and handling malicious packets or attacks. The application proxy enforces protocol conformance and reduces malicious content from sneaking past firewalls that are embedded deep into the application payload.

With application awareness that is integrated into the firewall's routing decisions, firewalls can provide a new and powerful way to protect networks at the edge.

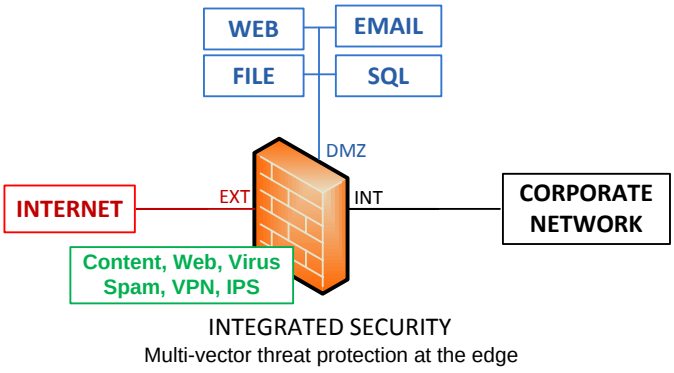


Figure 46. Integrated security at the network edge

The unification of different security functions into a single platform also simplifies management, reduces cost, improves overall security posture and reduces potential security related incidents.

Performance and Feature Characterization

Firewalls usually perform exceptionally well with minimal protection features enabled. As different application-aware rules/filters are enabled, application flows must be analyzed deeper into the payload before they are allowed or denied. More packet inspection requires more firewall CPU processing, memory usage and complex session state maintenance, all of which impacts firewall performance. It is important to understand how each feature impacts performance. It helps to understand where performance is sacrificed in return for a higher degree of protection. It is critical for capacity planning and provides an empirical way to choose a firewall and its feature set based on the importance of features and the tradeoff in performance.

Let's look at the example below.

Table 26. Performance versus functionality

Feature	Performance for Vendor A	Performance for Vendor B	Feature importance
1000 ACL for servers	500Mbps	300Mbps	50%
10 Attack protection	700Mbps	600Mbps	50%
ACL + Attack	400Mbps	500Mbps	100%

Vendor A performs better with access control lists (ACL) or attack protection alone. With both features enabled however, Vendor B is better. All else being equal, Vendor B has a better feature to performance ratio. This type of analysis is important in understanding how well a device will perform in a network with a combination of features enabled.

Testing performance of application-aware firewalls

The focus of this test case is to determine the performance curve of a firewall or unified security device that supports complex application-aware features. The performance metric of interest is steady-state throughput with one or more features enabled. A typical traffic load profile of web, enterprise and Internet traffic will be created to exercise these capabilities and determine this performance metric.

Performance Metric

It is important to note that the interpretation of application-layer throughput differs from the general understanding of how throughput is measured. See the illustration below for the 2 ways in which Throughput can be computed and recorded.

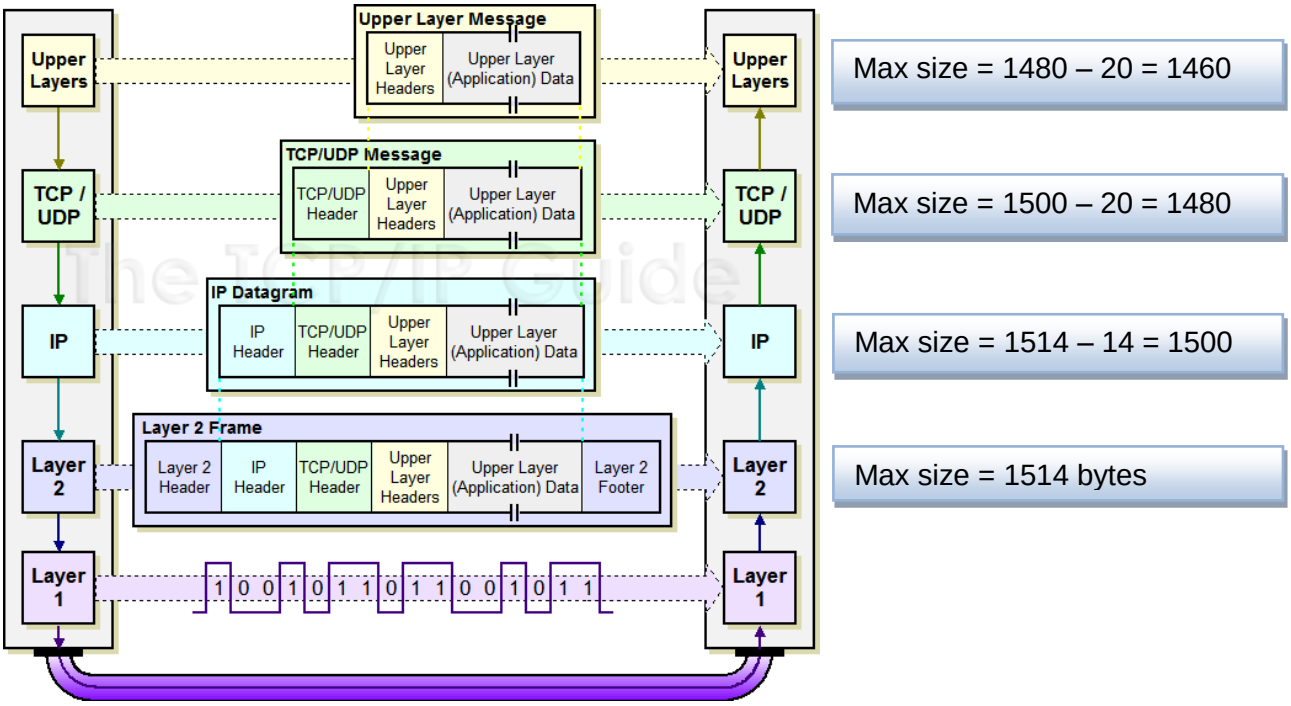


Figure 47. Layer 2 frame, TCP/IP packet size guide

Throughput is computed for the entire layer 2 frame, thereby calculating the total bits per second on the wire.

Goodput is defined as the application-layer throughput. It is generally used to provide a meaningful performance characterization of a device without factoring in the overhead of TCP and lower protocols. Retransmitted packets are not factored into the goodput metric.

Test Case: Application Filtering with Access Control Lists

Overview

Basic application filtering capabilities are used to ensure that all traffic flows are correctly passed, adhering to basic firewall/security rules with minimal application-layer inspection. Basic application filtering ensures that various enterprise protocols work correctly through a firewall and run at the optimally.

ACLs are the primary technique for setting up and managing application traffic. ACLs are allow or deny rules based on layer 3 and layer 4 header information. A typical access list may look like this:

```
Allow/deny <source IP> <source tcp/udp port> <dest IP> <dest tcp/udp port>
```

Figure 48. Layer 2 frame, TCP/IP packet size guide

Objective

The purpose of this test case is to configure the firewall with a default security mode that consists of various access control lists that use IP address, TCP and UDP port information to filter traffic. The firewall allows or denies traffic while the test measures the baseline performance.

Performance metrics required: **Maximum Throughput**.

Setup

The firewall should be configured with basic packet inspection that allows traffic from an internal network to reach untrusted or external networks. The primary way to setup access rules is to configure several ACL rules based on the protected subnets, the type of traffic to allow, and action to take for unwanted traffic.

Here is an example firewall configuration:

```
Allow TCP/80 traffic from INT network to EXT network.
Allow ICMP traffic from INT network to EXT network.
Block all ICMP traffic from EXT network to INT network.
```

Figure 49. General firewall configuration using ACLs

The test tool requires at least one server and one client port, as shown below.

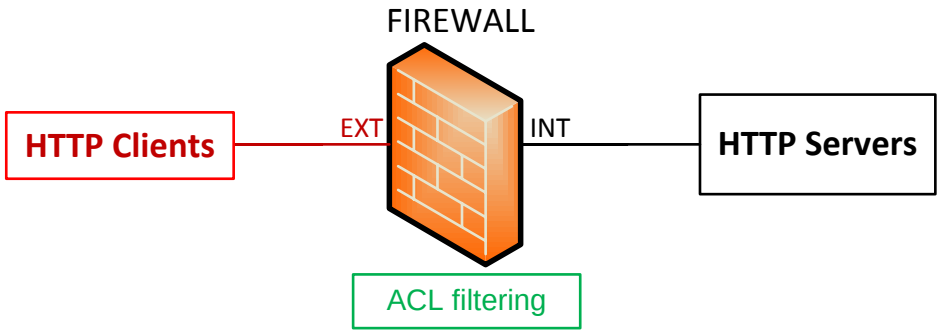


Figure 50. Test Case Setup

Test Variables

Test Tool Configuration

To exercise firewall ACL processing performance, 1000+ source IP addresses are configured on the test tool, and for each new TCP connection, a new IP address and random source ephemeral TCP port is used.

The test tool configuration profile is shown below. Configure one NetTraffic pair, with HTTP client and server protocol activities as specified below.

Table 27. Test tool configuration for test case

Parameters	Client	Server
Network configuration	1,000-100,000 IP addresses “Use all” IP addresses option	1 IP address per test port
TCP parameters	TCP RX 32768, TCP TX 1024	TCP RX 1024, TCP TX 32768
HTTP parameters	HTTP/1.1 with keep-alive 10 TCP connections per user Max Transactions	Default settings
HTTP command list	1 GET command – payload of 500-1024000 bytes	N/A
Test Objective	Throughput	N/A

DUT Test Variables

The firewall should be configured with basic protection and access control lists that will allow test traffic through from trusted to untrusted zones.

Table 28. Firewall configuration for Test case

Feature	Configuration	Description
Basic firewall	Configure security interfaces or zones	Security zones/interfaces are used to create trusted and untrusted networks
Basic network protection	ARP, Fragmentation, ICMP, SYN protection	Enable basic network protection features that would be used in a production network
Route or NAT	Enable NAT or no-NAT zone configurations	Configure Network Address Translation or Disable for Routed Mode where source IP is not changed
100+ ACL	Enable source IP or source + destination IP	ACLs will be used to allow several HTTP traffic based on filters used and deny all others

Step-by-Step Instructions

Configure the test to run a baseline test, which is an Ixia port-to-port test, in order to measure the test tool's baseline performance. Configure IxLoad as per the instructions below. Once you have obtained the baseline performance, setup the DUT and the test tool as per the Setup section above. Refer to the Test Variables section for recommended configurations. Note that the network configurations must change between running the port-to-port and DUT test. Physical cabling will change to connect the test ports to the DUT. A layer 2 switch with a high-performance backplane is highly recommended in a switched environment.

Reference baseline performance: fill in the table below with the baseline performance to use as a reference of the test tool's performance, based on the quantity and type of hardware available.

Table 29. Reference baseline performance form

Test Case	Performance indicator	Value per port pair	Load module type
1. Basic ACL/FW	Throughput		

Note: For each test case there can be several runs, based on the use of individual firewall features. This reference performance matrix should be used to determine the test tool performance when connected port to port.

These Step-by-Step Instructions should be followed for each test case that is run. The test cases require individual test runs, and each test case has a set of test tool and DUT.

1. Configure the DUT as specified in Test Variables, based on the test case being executed.

2. Launch IxLoad. In the main window, you will be presented with the **Scenario Editor** window. All the test configurations will be made here.

   To become familiar with the IxLoad GUI, see the Getting Started Guide.

3. Add the Client **NetTraffic** object. Configure the Client network with total IP count, gateway and VLAN if used.

   Add the Server NetTraffic object. Configure the total number of servers IP addresses that will be used. For performance testing, use one server IP per test port. For a step-by-step workflow, see Appendix A: Configuring IP and Network Settings.

4. The TCP parameters that are used for a specific test type are important optimizing optimizing the test tool. Refer to the Test Variables section to set the correct TCP parameters for the **Client** and **Server** network.

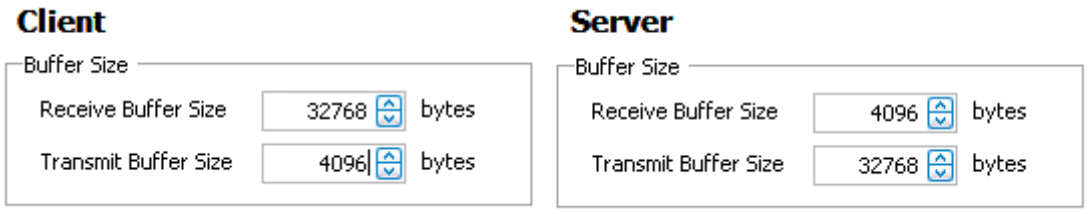


Figure 51. TCP buffer settings for Client and Server traffic

There are several other parameters that can be changed. Leave them to defaults unless you need to change them as per testing requirements.

For a step-by-step workflow, see Appendix B: Configuring TCP Parameters.

5. On the Server **NetTraffic** object, add one or more Server protocol activities as specified in the test case. The defaults are sufficient for this test, unless specifically noted.

   For a step-by-step workflow, see Appendix C: Configuring HTTP Servers.

6. On the Client **NetTraffic** object, add one or more Client protocol activities as specified in the test case. Configure each client Activity as per the test case profile.

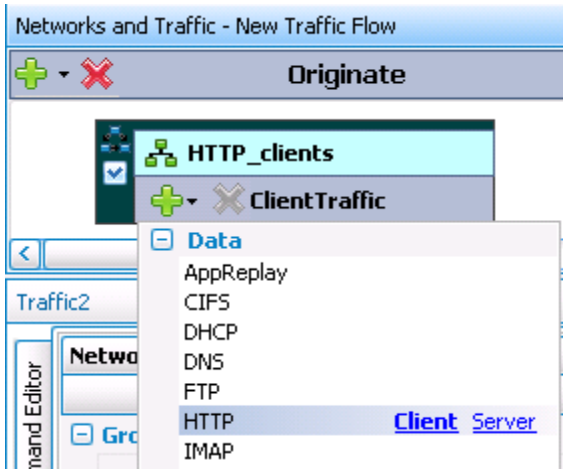


Figure 52. Adding a new Client Activity for a NetTraffic Object

7. Select the Activity name to access **User Source IP Mapping** and use **Cycle Users Through All Source IPs** option.

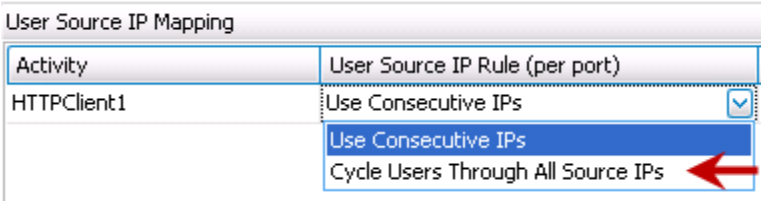


Figure 53. User Source IP Mapping dialog

8. Having setup the Client and Server networks, and the traffic profile, the Test Objective should now be configured. Go to the **Objective and Timeline** view. The test Objective can be applied on a per-Activity or per-protocol basis. This is where the iterative Objectives will be set between test runs to find the maximum throughput for the device.

   The following should be configured:

   - Test Objective. Begin by attempting to send a large number of connections per second through the DUT. If the published or expected value for **MAX_TPUT** is known, this value is a good starting point, and will become the targeted value for the test (**TARGET_MAX_TPUT**).

     For a step-by-step workflow, see Appendix E: Setting the Test Load Profile and Objective.

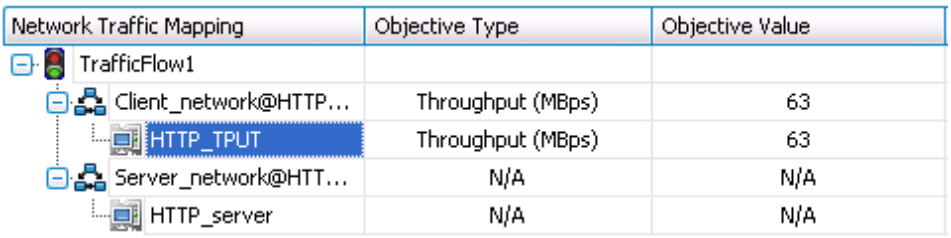


Figure 54. Test Objective Settings Dialogue

Once the Test Objective is set, the **Port CPU** on the bottom indicates the total number of ports that are required. See the Appendix F: Adding Test Ports and Running Tests on adding the ports to the test. For a step-by-step workflow, see Appendix F: Adding Test Ports and Running Tests.

NOTE: Tests with multiple Activities. For test cases with multiple application protocol Activities, the Test Objective is set per Activity. Hence, **TARGET_MAX_TPUT** should be equally distributed across the protocol Activities or can be set to multiple Objectives with the total not exceeding the **TARGET_MAX_TPUT** for all Activities combined.

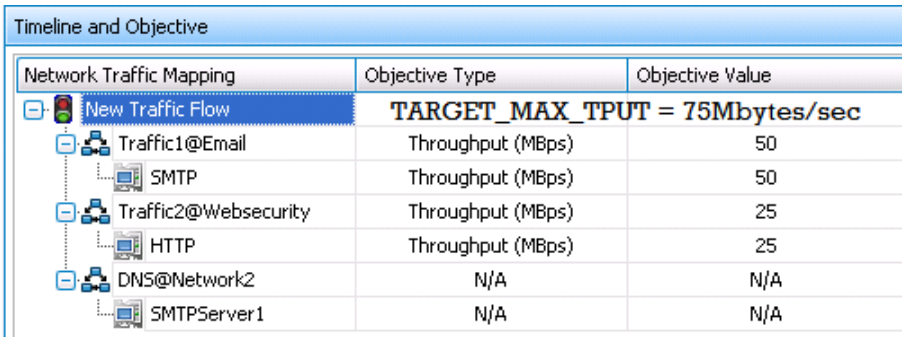


Figure 55. Multiple test objectives

9. Run the test for few minutes to allow the performance to reach a steady-state. Steady state is referred as Sustain duration in the test. Continue to monitor the DUT for the target rate and any failure/error counters. See the Common Results Analysis section for important statistics and diagnostics information.

   It is not trivial to interpret all statistics in all cases. The section that follows provides a diagnostics-based approach to highlight some common scenarios, the statistics being reported, and how to interpret them.

10. Iterate through the test setting **TARGET_MAX_TPUT** to the steady value attained during the previous run. To determine when the DUT has reached its **MAX_TPUT**, see the section on interpreting results before making a decision.

    The test tool can be started, stopped in the middle of a test cycle, or wait for it to be gracefully stopped using the test controls shown here. 

    For a step-by-step workflow, see Appendix F: Adding Test Ports and Running Tests.

Results

The steady-state throughput performance metric can be obtained using the **Throughput Objective** views in the **Statistics** viewer. Please refer to the Common Results Analysis section following test case 4 for detailed analysis and troubleshooting.

Test Case: Content Inspection

Overview

Content inspection, also known as application-aware proxy, performs general application recognition and support. It is used to inspect various applications that run over TCP, UDP transport in order to verify that they are behaving normally.

A firewall must support protocol decode and maintain complete TCP and application state per connection in order for a number of Internet and enterprise applications to work properly through the firewall. Protocols include SIP/RTP, RTSP/RTP, various IM protocols, FTP, RPC, DCE, and SMTP,. Managing TCP, TCP+UDP and dynamically handling secondary TCP/UDP connections is necessary to intelligently support and secure sessions in real-time. Application proxies may also inspect layer 7 protocol exchanges for conformance, protect against malformed packets/header attacks, and protect against unsolicited connections.

When application inspection is used, the process of matching of initial connection packets and forwarding subsequent packets will be handled differently by the firewall. The connection will be routed through a different firewall path in order for the inspection to take place and for the full session state to be maintained.

Objective

The purpose of this test case is to measure the performance of the firewall with various enterprise applications in addition to basic ACL processing.

Performance metrics required: Maximum Throughput.

Setup

The firewall should be configured with the existing ACL from the first test case. In additional the application inspection proxy should be enabled.

Below is an example of the firewall configuration.

```
Allow TCP/80 traffic from INT network to EXT network.
Allow ICMP traffic from INT network to EXT network.
Block all ICMP traffic from EXT network to INT network.
Enable application proxy for HTTP, RTSP, DNS, FTP, SMTP.
```

Figure 56. Generic firewall configuration with application proxy enabled

The test tool setup at least one server and one client port.

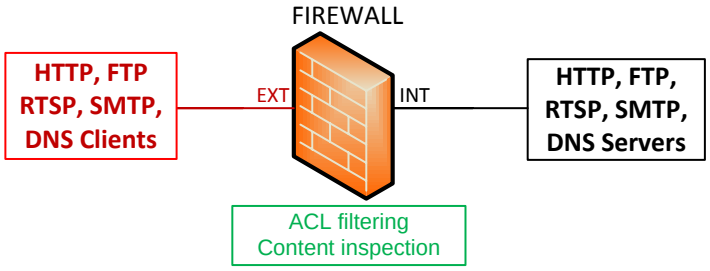


Figure 57. Test case 2 setup

Test Variables

Test Tool Configuration

This section specifies the test tool configuration profile. Configure one **NetTraffic** pair per application protocol Activity as specified below.

HTTP client and server configuration

Table 30. HTTP configuration for test Case 2

Parameters	Client	Server
Network configuration	1,000-100,000 IP addresses “Use all” IP addresses option	1 IP address per test port
TCP parameters	TCP RX 32768, TCP TX 1024	TCP RX 1024, TCP TX 32768
HTTP parameters	HTTP/1.1 with keep-alive 10 TCP connections per user Max Transactions	Default settings
HTTP command list	1 GET command – payload of 500-1024000 bytes	N/A
Test Objective	Throughput	N/A

Table 31. FTP configuration for Test Case 2

FTP client and server configuration

Parameters	Client	Server
Network configuration	1,000 IP addresses “Use all” IP addresses option	1 IP address per test port
TCP parameters	TCP RX 32768, TCP TX 1024	TCP RX 1024, TCP TX 32768
FTP parameters	Set FTP Mode to Passive	N/A
FTP command list	1 GET command – payload of 500-1024000 bytes	N/A
Test Objective	Throughput	N/A

DNS client and server configuration

Table 32. DNS configuration for test case 2

Parameters	Client	Server
Network configuration	1,000 IP addresses “Use all” IP addresses option	1 IP address per test port
TCP parameters	N/A	N/A
DNS parameters	Change retries to Zero	N/A
DNS command list	Configure A, MX, NS record with Recursion set to Off	N/A
Test Objective	Queries per Second	N/A

RTSP client and server configuration

Table 33. RTSP configuration for test case 2

Parameters	Client	Server
Network configuration	1,000 IP addresses “Use all” IP addresses option	1 IP address per test port
TCP parameters	TCP RX 32768, TCP TX 1024	TCP RX 1024, TCP TX 32768
RTSP parameters	Set RTP Transport to UDP	Add 4Mbps per stream content /videoaudio.mp4 | MPEG2 | 300 second
RTSP command list	{PlayMedia} / videoaudio.mp4	N/A
Test Objective	Simulated Users *Assuming constant-bit-rate, the number of SU will be the desired throughput divided bitrate per stream*	N/A

SMTP client and server configuration

Table 34. SMTP configuration for test case 2

Parameters	Client	Server
Network configuration	1,000 IP addresses “Use all” IP addresses option	1 IP address per test port
TCP parameters	TCP RX 32768, TCP TX 1024	TCP RX 1024, TCP TX 32768
SMTP parameters	Default	Session Limit per port 50,000
SMTP command list	{Send} | RandomLarge | 100	N/A
Test Objective	Throughput	N/A

DUT Test Variables

This firewall should be configured with basic protection, access control lists and application-aware proxy/gateway (ALG) for the specific protocols that will be tested, as outlined in the Test Tool Configuration section.

Table 35. Firewall configuration for Test Case 2

Feature	Configuration	Description
Basic firewall	Configure security interfaces or zones	• Security zones/interfaces are used to create trusted and untrusted networks
Basic network protection	ARP, Fragmentation, ICMP, SYN protection	• Enable basic network protection features that would be used in a production network
Route or NAT	Enable NAT or no-NAT zone configurations	• Configure Network Address Translation or Disable for Routed Mode where source IP is not changed
100+ ACL	Enable source IP or source + destination IP	• ACLs will be used to allow several HTTP traffic based on filters used and deny all others
Enable ALG	*SMTP, RTSP, FTP, HTTP, DNS (and any others)*	• *Configure application-aware proxy (ALG) for specific protocols that will be tested*

Step-by-Step Instructions

Configure the test to run a baseline test, which is an Ixia port-to-port test, in order to measure the test tool's baseline performance. Configure IxLoad as per the instructions below. Once you have obtained the baseline performance, setup the DUT and the test tool as per the Setup section above. Refer to the Test Variables section for recommended configurations. Note that the network configurations must change between running the port-to-port and DUT test. Physical cabling will change to connect the test ports to the DUT. A layer 2 switch with a high-performance backplane is highly recommended in a switched environment.

Reference baseline performance: fill in the table below with the baseline performance to use as a reference of the test tool's performance, based on the quantity and type of hardware available.

Table 36. Reference baseline performance form

Test Case	Performance indicator	Value per port pair	Load module type
2. Basic ACL/FW + Application Proxy	Throughput		

Note: For each test case there can be several runs, based on the use of individual firewall features. This reference performance matrix should be used to determine the test tool performance when connected port to port.

These Step-by-Step Instructions should be followed for each test case that is run. The test cases require individual test runs, and each test case has a set of test tool and DUT.

1. Configure the DUT as specified in the DUT Test Variables, based on the test case being executed. Launch IxLoad. In the main window, you will be presented with the **Scenario Editor** window. All the test configurations will be made here.

   To become familiar with the IxLoad GUI, see the Getting Started Guide.

3. Add the Client **NetTraffic** object. Configure the Client network with total IP count, gateway and VLAN if used.

   Add the Server NetTraffic object. Configure the total number of servers IP addresses that will be used. For performance testing, use one server IP per test port. For a step-by-step workflow, see Appendix A: Configuring IP and Network Settings.

4. The TCP parameters that are used for a specific test type are important optimizing optimizing the test tool. Refer to the Test Variables section to set the correct TCP parameters for the **Client** and **Server** network.

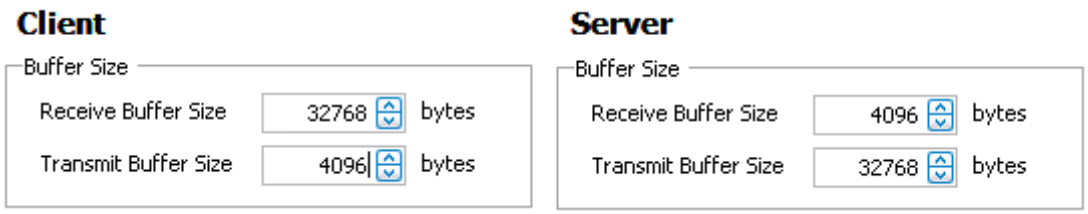


Figure 58. TCP Buffer Settings for Client and Server traffic

There are several other parameters that can be changed. Leave them to defaults unless you need to change them as per testing requirements.

For a step-by-step workflow, see Appendix B: Configuring TCP parameters.

5. On the Server **NetTraffic** object, add one or more Server protocol activities as specified in the test case. The defaults are sufficient for this testing unless specifically noted.

   For a step-by-step workflow, see Appendix C: Configuring HTTP Servers.

6. On the Client **NetTraffic** object, add one or more Client protocol activities as specified in the test case. Configure each client Activity as per the test case profile.

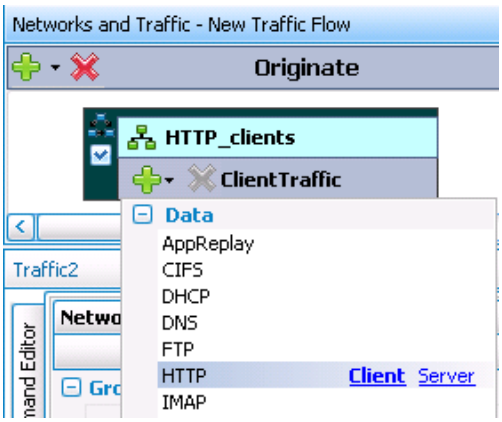


Figure 59. TCP Adding a new Client Activity for a NetTraffic object

7. Select the Activity name to access **User Source IP Mapping** and use **Cycle Users Through All Source IPs** option.

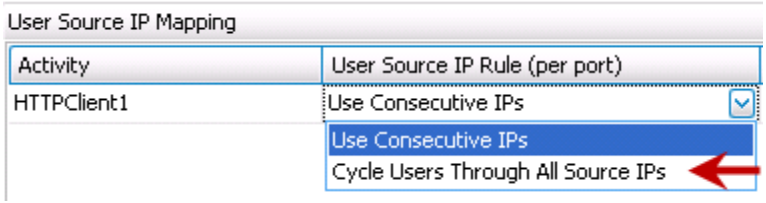


Figure 60. User Source IP Mapping dialog

8. Having setup the Client and Server networks, and the traffic profile, the Test Objective should now be configured. Go to the **Objective and Timeline** view. The test Objective can be applied on a per-Activity or per-protocol basis. This is where the iterative Objectives will be set between test runs to find the maximum throughput for the device.

   The following should be configured:

   - Test Objective. Begin by attempting to send a large number of connections per second through the DUT. If the published or expected value for **MAX_TPUT** is known, this value is a good starting point, and will become the targeted value for the test (**TARGET_MAX_TPUT**).

     For a step-by-step workflow, see Appendix E: Setting the Test Load Profile and Objective.

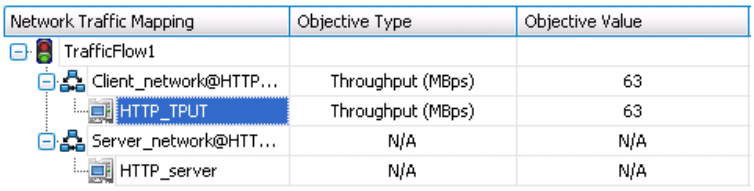


Figure 61. Test Objective Settings Dialogue

Once the Test Objective is set, the **Port CPU** on the bottom indicates the total number of ports that are required. See the Appendix F: Adding Test Ports and Running Tests on adding the ports to the test. For a step-by-step workflow, see Appendix F: Adding Test Ports and Running Tests.

Note: Tests with multiple Activities. For test cases with multiple application protocol Activities, the Test Objective is set per Activity. Hence, **TARGET_MAX_TPUT** should be equally distributed across the protocol Activities or can be set to multiple Objectives with the total not exceeding the **TARGET_MAX_TPUT** for all Activities combined.

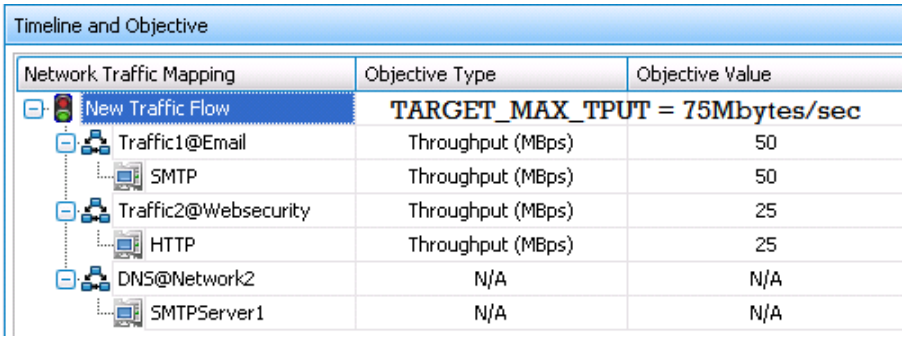


Figure 62. Multiple Test Objectives

9. Run the test for few minutes to allow the performance to reach a steady-state. Steady state is referred as Sustain duration in the test. Continue to monitor the DUT for the target rate and any failure/error counters. See the Common Results Analysis section for important statistics and diagnostics information.

   It is not trivial to interpret all statistics in all cases. The Common Results Analysis section that follows provides a diagnostics-based approach to highlight some common scenarios, the statistics being reported, and how to interpret them.

   Iterate through the test setting **TARGET_MAX_TPUT** to the steady value attained during the previous run. To determine when the DUT has reached its **MAX_TPUT**, see the Common Results Analysis section on interpreting results before making a decision.

   The test tool can be started, stopped in the middle of a test cycle, or wait for it to be gracefully stopped using the test controls shown here. 

   For a step-by-step workflow, see Appendix F: Adding Test Ports and Running Tests.

Results

The steady-state throughput performance metric can be obtained using the **Throughput Objective** views in the **Statistics** viewer. Please refer to the Common Results Analysis section following test case 4 for detailed analysis and troubleshooting.

Test Case: Web Security Filtering

Overview

Websites represent potential risk. They may contain malicious content that is disguised within websites or within scripts. Malicious content includes virus, adware, tracking cookies, Trojans, and spyware. The most common practice for preventing enterprise users from malicious web content is to block site URLs, IP addresses, and known spammer e-mails, servers and domains. Web security is not only used to protect users, but it is increasingly used to manage Internet usage, by specifying which sites are accessible, the type of content that can be accessed, such as streaming video, Internet messaging, webmail, social networks, and blocking access to sites that don't conform to company policies.

Deep packet inspection (DPI) may be used to reduce this vulnerability. The amount of DPI that is used is a function of the complexity and nature of the filters that are enabled. DPI filters force all HTTP traffic to be routed through network processors; this can have a dramatic impact on performance.

Objective

The purpose of this test case is to measure throughput performance when one or more web security inspection features are enabled.

The list of web security features will vary by product. Measure performance for individual features alone where possible, then enable multiple features before measuring the performance curve.

Performance metrics required: Maximum Throughput.

Setup

The firewall should be configured with the basic web filtering/security feature enabled and tested. Additional web security features should be turned on and tested until all web security protection mechanisms are enabled and tested.

Here is an example firewall configuration:

Allow TCP/80 traffic from INT network to EXT network.
Allow ICMP traffic from INT network to EXT network.
Block all ICMP traffic from EXT network to INT network.
Enable one after another
- *Web security feature 1*
- *Web security feature*
- *…*
- *Web security all features*

Figure 63. Firewall Configuration with Content Inspection

The test tool setup at least one server and one client port.

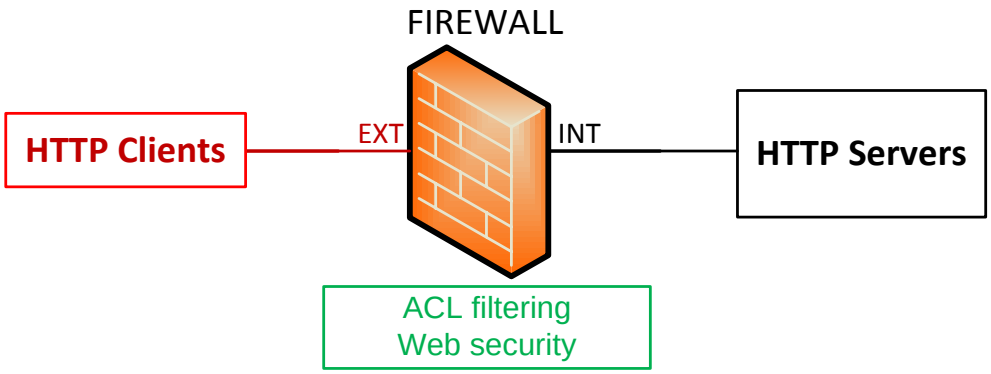


Figure 64. Test Case 3 Setup

Test Variables

Test Tool Configuration

The test tool configuration profile is shown below. Configure one NetTraffic pair, with HTTP client and server protocol activities as specified below.

HTTP client and server configuration

Table 37. HTTP configuration for test case 3

Parameters	Client	Server
Network configuration	- 1,000 IP addresses - "Use all" IP addresses option - Configure source IP from known spam list, use: http://www.spamhaus.org	- 1 IP address per test port - Configure server IP from known spam list, use: http://www.spamhaus.org
TCP parameters	TCP RX 32768, TCP TX 1024	TCP RX 1024, TCP TX 32768
HTTP parameters	- HTTP/1.1 with keep-alive - 10 TCP connections per user - Max Transactions	Create and upload Real files - file1.mp3 – 1MB - file2.zip – 2MB - blockedsite.html – 10kB (anything) - iecar.com – download from http://www.eicar.org/anti_virus_test_file.htm
HTTP command list	- GET file1.mp3 - GET file2.zip - GET blockedsite.html - GET iecar.com	N/A
Test Objective	Throughput	N/A

DUT Test Variables

There are several scenarios that may be used to test a DUT. The following table outlines some of the capabilities of the DUT that can be selected to run in a certain mode.

Table 38. Firewall configuration for Test Case 3

Feature	Configuration	Description
Basic firewall	Configure security interfaces or zones	• Security zones/interfaces are used to create trusted and untrusted networks
Basic network protection	ARP, Fragmentation, ICMP, SYN protection	• Enable basic network protection features that would be used in a production network
Route or NAT	Enable NAT or no-NAT zone configurations	• Configure Network Address Translation or Disable for Routed Mode where source IP is not changed
Enable ALG	SMTP, RTSP, FTP, HTTP, DNS (and any others)	• Configure ALG (fix up) for specific protocols that will be tested
Enable Web security features	*Enable Malware* *Enable Spyware* *Enable bad IP list* *Enable restricted site* *Enable format scanning* *Enable other Web security features*	• *Configure full Web security protection features* • *Test performance one feature at a time*

Step-by-Step Instructions

Configure the test to run a baseline test, which is an Ixia port-to-port test, in order to measure the test tool's baseline performance. Configure IxLoad as per the instructions below. Once you have obtained the baseline performance, setup the DUT and the test tool as per the Setup section above. Refer to the Test Variables section for recommended configurations. Note that the network configurations must change between running the port-to-port and DUT test. Physical cabling will change to connect the test ports to the DUT. A layer 2 switch with a high-performance backplane is highly recommended in a switched environment.

Reference baseline performance: fill in the table below with the baseline performance to use as a reference of the test tool's performance, based on the quantity and type of hardware available.

Table 39. Reference baseline performance form

Test Case	Performance indicator	Value per port pair	Load module type
3. Basic ACL/FW + Web Filtering	Throughput		

Note: For each test case there can be several runs, based on the use of individual firewall features. This reference performance matrix should be used to determine the test tool performance when connected port to port.

These Step-by-Step Instructions should be followed for each test case that is run. The test cases require individual test runs, and each test case has a set of test tool and DUT.

1. Configure the DUT as specified in the DUT Test Variables based on the test case being executed.

2. Launch IxLoad. In the main window, you will be presented with the **Scenario Editor** window. All the test configurations will be made here.

   To become familiar with the IxLoad GUI, see the Getting Started Guide.

3. Add the Client **NetTraffic** object. Configure the Client network with total IP count, gateway and VLAN if used.

   Add the Server NetTraffic object. Configure the total number of servers IP addresses that will be used. For performance testing, use one server IP per test port. For a step-by-step workflow, see Appendix A: Configuring IP and Network Settings.

4. The TCP parameters that are used for a specific test type are important for optimizing the test tool. Refer to the Test Variables section to set the correct TCP parameters for the **Client** and **Server** network.

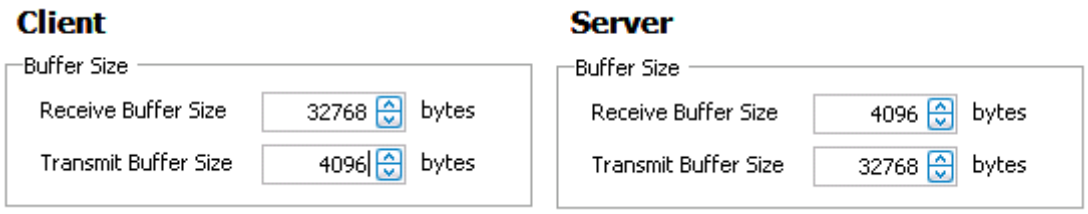


Figure 65. TCP Buffer Settings for Client and Server traffic

There are several other parameters that can be changed. Leave them to defaults unless you need to change them as per testing requirements.

For a step-by-step workflow, see Appendix B: Configuring TCP parameters.

On the Server NetTraffic object, add one or more **Server protocol** activities as specified in the test case. The defaults are sufficient for this testing unless specifically noted to change.

For a step-by-step workflow, see Appendix C: Configuring HTTP Servers.

5. On the Client **NetTraffic** object, add one or more Client protocol activities as specified in the test case. Configure each client Activity as per the test case profile.

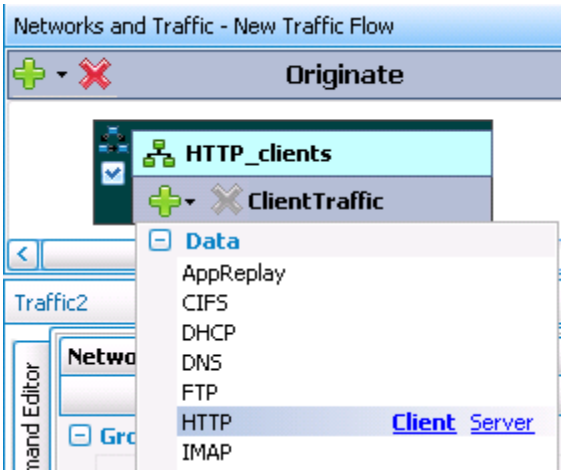


Figure 66. Adding a new Client Activity for a NetTraffic Object

6. Select the Activity name to access **User Source IP Mapping** and use **Cycle Users Through All Source IPs** option.

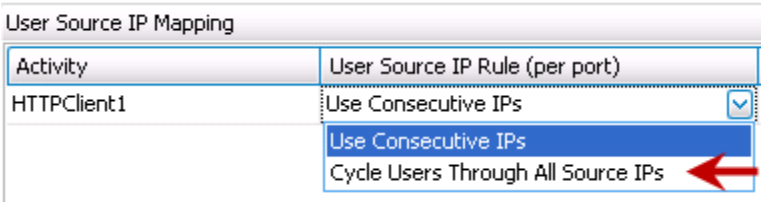


Figure 67. User Source IP Mapping dialog

7. Having setup the Client and Server networks, and the traffic profile, the Test Objective should now be configured. Go to the **Objective and Timeline** view. The test Objective can be applied on a per-Activity or per-protocol basis. This is where the iterative Objectives will be set between test runs to find the maximum throughput for the device.

   The following should be configured:

   - Test Objective. Begin by attempting to send a large number of connections per second through the DUT. If the published or expected value for **MAX_TPUT** is known, this value is a good starting point, and will become the targeted value for the test (**TARGET_MAX_TPUT**).

     For a step-by-step workflow, see Appendix E: Setting the Test Load Profile and Objective.

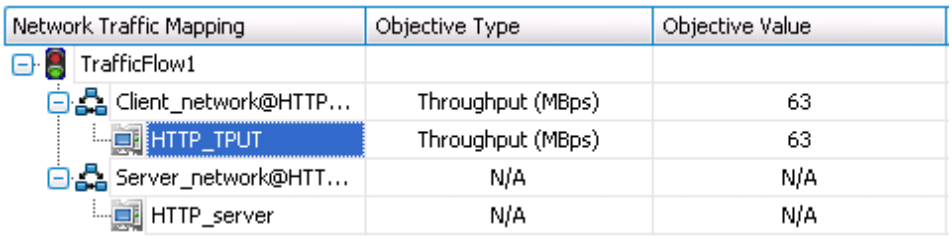


Figure 68. Test Objective Settings Dialogue

Once the Test Objective is set, the **Port CPU** on the bottom indicates the total number of ports that are required. See the Appendix F: Adding Test Ports and Running Tests on adding the ports to the test. For a step-by-step workflow, see Appendix F: Adding Test Ports and Running Tests.

Note: Tests with multiple Activities. For test cases with multiple application protocol Activities, the Test Objective is set per Activity. Hence, **TARGET_MAX_TPUT** should be equally distributed across the protocol Activities or can be set to multiple Objectives with the total not exceeding the **TARGET_MAX_TPUT** for all Activities combined.

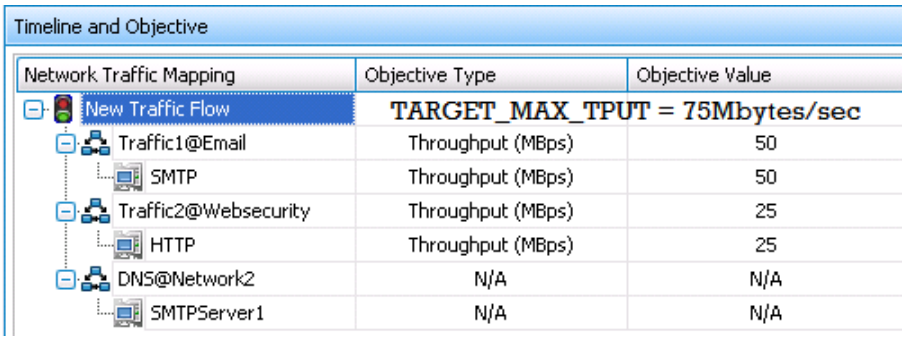


Figure 69. Multiple Test Objectives

8. Run the test for few minutes to allow the performance to reach a steady-state. Steady state is referred as Sustain duration in the test. Continue to monitor the DUT for the target rate and any failure/error counters. See the Common Results Analysis section for important statistics and diagnostics information.

   It is not trivial to interpret all statistics in all cases. The Common Results Analysis section that follows provides a diagnostics-based approach to highlight some common scenarios, the statistics being reported, and how to interpret them.

9. Iterate through the test setting **TARGET_MAX_TPUT** to the steady value attained during the previous run. To determine when the DUT has reached its **MAX_TPUT**, see the Common Results Analysis section on interpreting results before making a decision.

   The test tool can be started, stopped in the middle of a test cycle, or wait for it to be gracefully stopped using the test controls shown here. 

For a step-by-step workflow, see Appendix F: Adding Test Ports and Running Tests.

Results

The steady-state throughput performance metric can be obtained using the **Throughput Objective** views in the **Statistics** viewer. Please refer to the Common Results Analysis section following test case 4 for detailed analysis and troubleshooting.

Test Case: Anti-Virus and Anti-Spam Filtering

Overview

E-mail is a backbone of enterprise and Internet communication. Enterprises want to ensure a 24/7 operation. Malicious content, attacks, unsolicited mail and spam has only increased, requiring people to spend exorbitant amount of time sifting through unwanted and dangerous content daily. An effort to provide multiple levels of protection to e-mail systems is a well-conceived strategy.

Perimeter firewalls and security devices incorporate protection at the edge of the network to protect from virus, phishing, and spam content before it gets to mail servers, where a secondary scan is used to further mitigate risks and reduce spam content.

Objective

A firewall that supports inbound SMTP message and content processing requires different configuration options for scanning for virus, content, spam, images, files, etc. Each of these options require DPI that may impact performance. Real-time e-mail delivery into in boxes is not a user expectation. However the amount of e-mail that a firewall can adequately process should be measured before it becomes a performance bottleneck should be known.

Performance metrics required: Maximum Throughput.

Setup

The firewall should be configured with the basic security feature enabled; e-mail security features should then be enabled one at a time, followed by a combination of features as required.

Here is an example firewall configuration:

Allow TCP/80 traffic from INT network to EXT network.
Allow ICMP traffic from INT network to EXT network.
Block all ICMP traffic from EXT network to INT network.
Enable AV, AS and Malware protection for SMTP, IMAP, POP
Enable AV, AS and Malware protection for HTTP (web email)

Figure 70. Firewall configuration with anti-virus, spam and malware protection

The test tool setup at least one server and one client port. The clients and servers must be connected to the DUT on a low latency switched network.

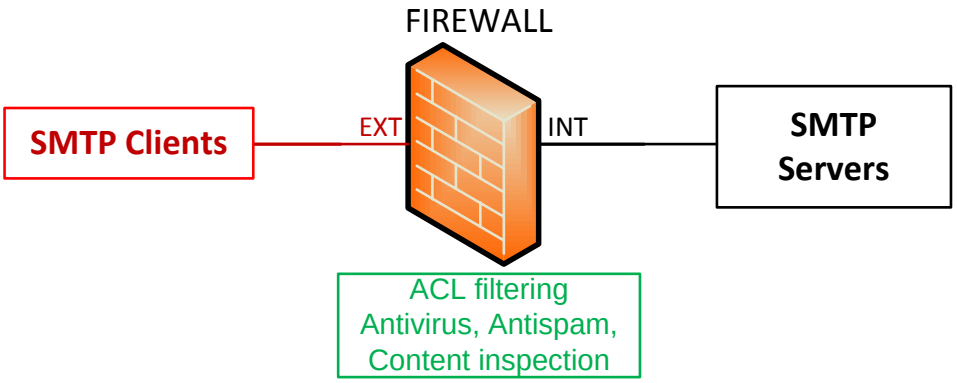


Figure 71. Test Case 4 Setup

Test Variables

Websites today deliver a rich experience for users using a number of advanced web client-side scripts, programming languages and other technologies that bring together a number of features. In addition to offering a highly interactive content.

Test Tool Configuration

The test tool configuration profile is shown below. Configure one NetTraffic pair, with HTTP client and server protocol activities as specified below.

HTTP client and server configuration

Table 40. HTTP configuration for test case 4

Parameters	Client	Server
Network configuration	1,000-100,000 IP addresses “Use all” IP addresses option	1 IP address per test port
TCP parameters	TCP RX 32768, TCP TX 1024	TCP RX 1024, TCP TX 32768
HTTP parameters	HTTP/1.1 with keep-alive 10 TCP connections per user Max Transactions	Default settings
HTTP command list	1 GET command – payload of 500-1024000 bytes	N/A
Test Objective	Throughput	N/A

SMTP client and server configuration

Table 41. SMTP Configuration for Test Case 4

Parameters	Client	Server
Network configuration	1,000 IP addresses "Use all" IP addresses option	1 IP address per test port
TCP parameters	TCP RX 32768, TCP TX 1024	TCP RX 1024, TCP TX 32768
SMTP parameters	Default	Session Limit per port 50,000
SMTP command list	{Send} | RandomLarge | 100	N/A
Test Objective	Throughput	N/A

DUT Test Variables

There are several scenarios that may be used to test a DUT. The following table outlines some of the capabilities of the DUT that can be selected to run in a certain mode.

Table 42. Firewall configuration for Test Case 4

Feature	Configuration	Description
Basic firewall	Configure security interfaces or zones	• Security zones/interfaces are used to create trusted and untrusted networks
Basic network protection	ARP, Fragmentation, ICMP, SYN protection	• Enable basic network protection features that would be used in a production network
Route or NAT	Enable NAT or no-NAT zone configurations	• Configure Network Address Translation or Disable for Routed Mode where source IP is not changed
AV, Spam, Malware, Spyware	*Enable Anti-virus* *Eable Spam checking* *Enable Spyware, Malware, Trojans* *Enable other content inspection*	• *Enable complete virus, spam, malware, email based attacks, and other content specific rules for SMTP inbound* • *Enable Quarantine or delivery mechanism for suspect content* • *Enable one feature at a time*

Step-by-Step Instructions

Configure the test to run a baseline test, which is an Ixia port-to-port test, in order to measure the test tool's baseline performance. Configure IxLoad as per the instructions below. Once you have obtained the baseline performance, setup the DUT and the test tool as per the Setup section above. Refer to the Test Variables section for recommended configurations. Note that the network configurations must change between running the port-to-port and DUT test. Physical cabling will change to connect the test ports to the DUT. A layer 2 switch with a high-performance backplane is highly recommended in a switched environment.

Reference baseline performance: fill in the table below with the baseline performance to use as a reference of the test tool's performance, based on the quantity and type of hardware available.

Table 43. Reference baseline performance form

Test Case	Performance indicator	Value per port pair	Load module type
4. Basic ACL/FW + Anti-virus, Spam	Throughput		

Note: For each test case there can be several runs, based on the use of individual firewall features. This reference performance matrix should be used to determine the test tool performance when connected port to port.

These Step-by-Step Instructions should be followed for each test case that is run. The test cases require individual test runs, and each test case has a set of test tool and DUT.

1. Configure the DUT as specified in the Test Variables based on the test case being executed.
2. Launch IxLoad. In the main window, you will be presented with the **Scenario Editor** window. All the test configurations will be made here.

   To become familiar with the IxLoad GUI, see the Getting Started Guide.
3. Add the Client **NetTraffic** object. Configure the Client network with total IP count, gateway and VLAN if used.

Add the Server NetTraffic object. Configure the total number of servers IP addresses that will be used. For performance testing, use one server IP per test port. For a step-by-step workflow, see Appendix A: Configuring IP and Network settings.

4. The TCP parameters that are used for a specific test type are important optimizing optimizing the test tool. Refer to the Test Variables section to set the correct TCP parameters for the **Client** and **Server** network.

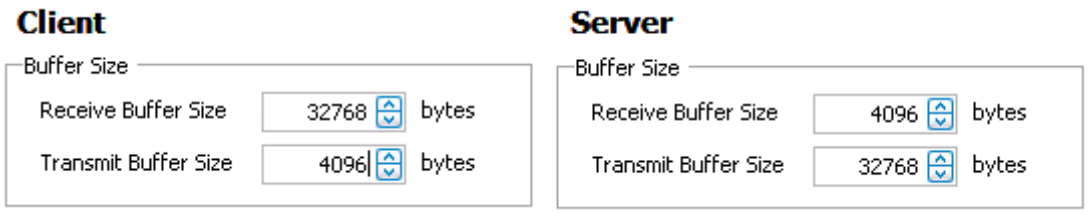


Figure 72. TCP Buffer Settings for Client and Server Traffic

There are several other parameters that can be changed. Leave them to defaults unless you need to change them as per testing requirements.

For a step-by-step workflow, see Appendix A: Configuring IP and Network Settings.

5. .On the Server **NetTraffic** object, add one or more Server protocol activities as specified in the test case. The defaults are sufficient for this test, unless specifically noted.

   For a step-by-step workflow, see Appendix C: Configuring HTTP Servers.

6. On the Client **NetTraffic** object, add one or more Client protocol activities as specified in the test case. Configure each client Activity as per the test case profile.

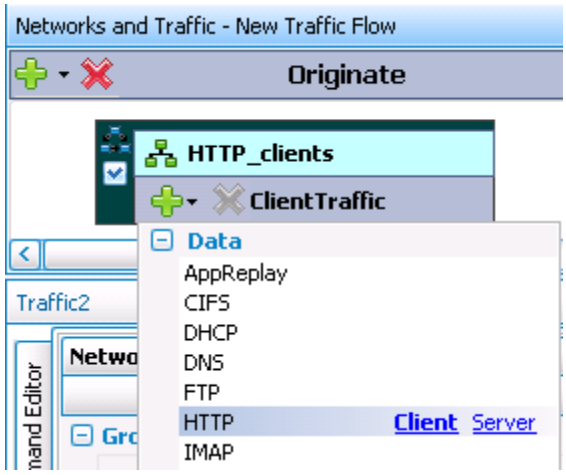


Figure 73. Add a New Client Acitivity for a NetTraffic Object

7. Select the Activity name to access **User Source IP Mapping** and use **Cycle Users Through All Source IPs** option.

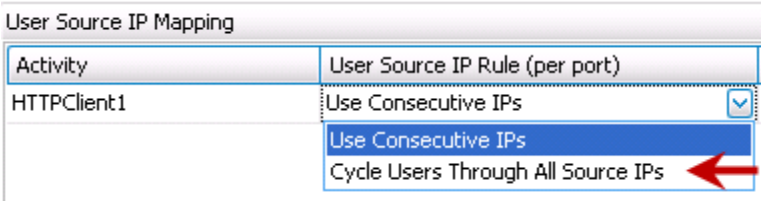


Figure 74. User Source IP Mapping dialog

8. Having setup the Client and Server networks, and the traffic profile, the Test Objective should now be configured. Go to the **Objective and Timeline** view. The test Objective can be applied on a per-Activity or per-protocol basis. This is where the iterative Objectives will be set between test runs to find the maximum throughput for the device.

   The following should be configured:

   - Test Objective. Begin by attempting to send a large number of connections per second through the DUT. If the published or expected value for **MAX_TPUT** is known, this value is a good starting point, and will become the targeted value for the test (**TARGET_MAX_TPUT**).

For a step-by-step workflow, see Appendix E: Setting the Test Load Profile and Objective.

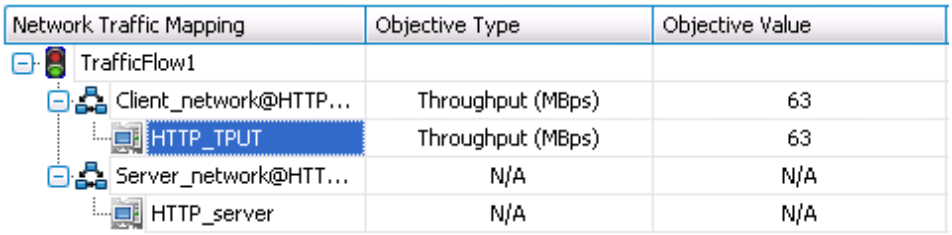


Figure 75. Test Objective Settings Dialogue

Once the Test Objective is set, the **Port CPU** on the bottom indicates the total number of ports that are required. See the Appendix F: Adding Test Ports and Running Tests on adding the ports to the test. For a step-by-step workflow, see Appendix F: Adding Test Ports and Running Tests.

Note: Tests with multiple Activities. For test cases with multiple application protocol Activities, the Test Objective is set per Activity. Hence, **TARGET_MAX_TPUT** should be equally distributed across the protocol Activities or can be set to multiple Objectives with the total not exceeding the **TARGET_MAX_TPUT** for all Activities combined.

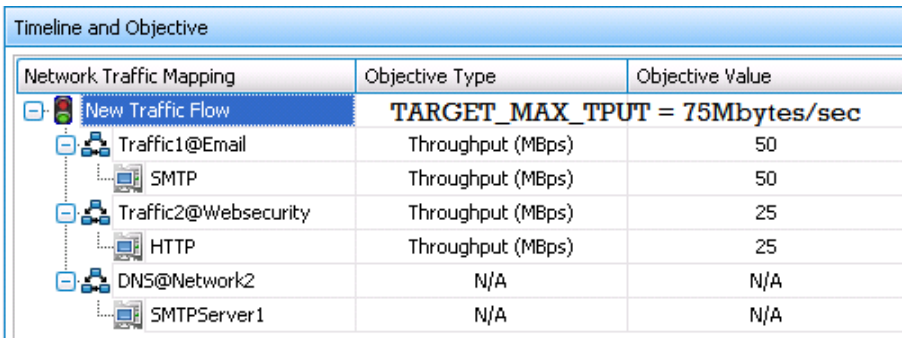


Figure 76. Multiple Test Objectives

9. Run the test for few minutes to allow the performance to reach a steady-state. Steady state is referred as Sustain duration in the test. Continue to monitor the DUT for the target rate and any failure/error counters. See the Common Results Analysis section for important statistics and diagnostics information.

   It is not trivial to interpret all statistics in all cases. The Common Results Analysis section that follows provides a diagnostics-based approach to highlight some common scenarios, the statistics being reported, and how to interpret them.

10. Iterate through the test setting **TARGET_MAX_TPUT** to the steady value attained during the previous run. To determine when the DUT has reached its **MAX_TPUT**, see the Common Results Analysis section on interpreting results before making a decision.

The test tool can be started, stopped in the middle of a test cycle, or wait for it to be gracefully stopped using the test controls shown here. 

For a step-by-step workflow, see Appendix F: Adding Test Ports and Running Tests.

Results

The steady-state throughput performance metric can be obtained using the **Throughput Objective** views in the **Statistics** viewer. Please refer to the Common Results Analysis section following test case 4 for detailed analysis and troubleshooting.

Common Results Analysis

Finding the maximum throughput requires an iterative method of running and re-running tests by changing a number of test input parameters. The DUT configuration must also be set so that it performs optimally, based on the test variation section.

Performance metrics

The following table lists the key performance statistics that must be monitored. These statistics help identify whether the device has reached its saturation point, as well as problem issues. Interpreting results in the correct manner will ensure that transient network, device or test tool behavior does not create a false negative condition.

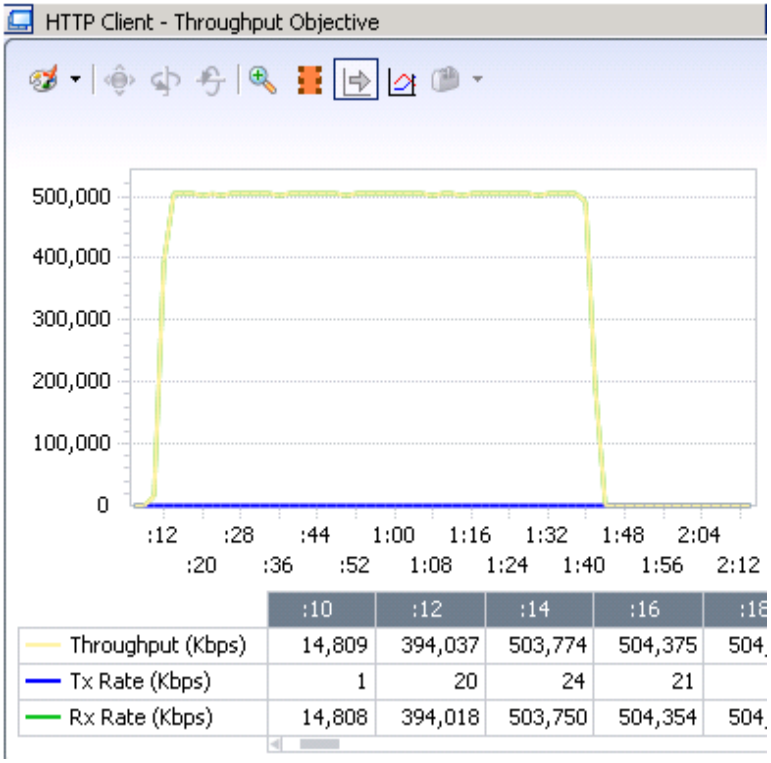
Table 44. Key Performance metrics

Metric	Key Performance Indicators	Statistics View
Performance metrics	Connections/sec Total connections, Number of Simulated Users, Throughput	HTTP Client – Objectives HTTP Client – Throughput
Application level transactions Application level failure monitoring	Requests Sent, Successful, Failed Request Aborted, Timeouts, Session Timeouts Connect time, 4xx, 5xx errors	HTTP Client – Transactions HTTP Client – HTTP Failures HTTP Client – Latencies
TCP Connection Information TCP Failure monitoring	SYNs sent, SYN/SYN-ACKs Received RESET Sent, RESET Received, Retries, Timeouts	HTTP Client – TCP Connections HTTP Client – TCP Failures

Real-Time Statistics

The graph below provides a view of the real-time statistics for the test. Real-time statistics provide instant access any failures at the TCP and HTTP protocol level.

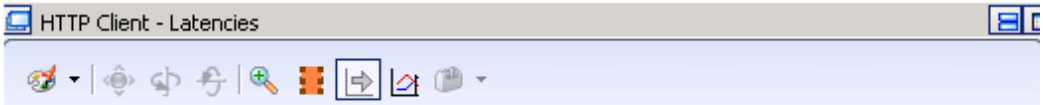
The following graph indicates that the target throughput of 500Mbps was reached. The throughput is the goodput, as described earlier. The Tx and Rx rates add to the overall throughput.



	:10	:12	:14	:16	:18
Throughput (Kbps)	14,809	394,037	503,774	504,375	504
Tx Rate (Kbps)	1	20	24	21	
Rx Rate (Kbps)	14,808	394,018	503,750	504,354	504

Figure 77. HTTP Client Throughput Statistics View

The latency view is useful for throughput testing to indicate if the DUT is able to process the packets in a timely manner.



	:08	:10	:12	:14	:16	:18	:20
Connect Time (µs)		74					
Time To First Byte (TTFB) (µs)		636	1,510	1,284	1,331	1,110	1,087
Time To Last Byte (TTLB) (µs)		181,273	194,277	189,923	187,703	201,100	198,348

Figure 78. HTTP Client Latency Statistics View

Test tool reference performance

Fill in the table below with the baseline performance of the test tool to use as a reference, based on the quantity and type of hardware available.

Table 45. Reference test tool performance matrix

Test Case	Performance indicator	Value per port pair	Load module type
1. Basic ACL/FW	Throughput		
2. Basic ACL/FW + Application Proxy			
3. Basic ACL/FW + Web Filtering			
4. Basic ACL/FW + Anti-virus, Spam			

Content inspection features and performance matrix

Fill in the table below with the steady state throughput performance that was obtained with the specific application aware filters and rules enabled.

Table 46. Complete performance matrix with different content inspection features enabled

Product	Basic Firewall	Content inspection	Web security	Anti-virus, Anti-spam

Troubleshooting and Diagnostics

Table 47. Troubleshooting and diagnostics

Issue	Diagnosis, Suggestions
Addition of more test ports does not increase performance	The DUT may have reached saturation. Check for TCP resets received by the client or server. In the case where the DUT is terminating connections from the client side, also check the device statistics for CPU utilization.
A large number of TCP resets received on the client and/or server side, during the beginning on the test – during ramp up. When in steady-state (Sustain), there are no or very minimal TCP timeouts and retries seen on the client and/or server side.	This indicates that the DUT or the test tool is “ramping up” and that there are many TCP sessions being opened and the DUT may not be “ready” to handle them. If the device uses multiple processing cores, then this behavior is possible when the additional load “turns on” the processing cores, but not before that.
A large number of TCP resets are received on the client and/or server side, throughout the test.	If there are continuous failures observed during steady-state, it’s possible that the device is reaching saturation. Reduce the objective until the failures are acceptable.
A small number of TCP failures are observed (timeout/retry), is this acceptable?	In general, small levels of TCP failures should be an acceptable behavior when a device is running at its maximum level. The transient nature of network, device and TCP behavior cannot guarantee zero failures. However, using the no failures is subjective, and may be required.
The target throughput not reached, throughput goes up and down	If device is not reaching steady-state, check the following: the TCP failures. High TCP timeout and RST packets can indicate issues with device unable to handle load Add more test ports; if the result is the same, then device cannot process more packets
The Simulated User is increasing, the test tool is unable to reach target Throughput	Check the Simulated Users count – if it is very high then it’s an indication that the test tool is attempting to reach the target and its system resources are depleting; add more test tools or check the configuration to determine any possible issue Check for TCP failures to indicate any network issues. Check device statistics for ingress and egress packet drops.

Impact of Traffic Management on Application Performance

Traffic Management

Traffic management is an encompassing term that groups a series of mechanisms used to positively impact network traffic flow. The mechanisms include admission control, traffic policing and traffic shaping. The most common traffic management goals for converged IP networks are to ensure quality of service (QoS) and service level agreements (SLA). These concepts are discussed below.

Traffic Shaping

Traffic shaping involves either delaying or dropping packets in favor of others when necessary. The most common network conditions that require traffic shaping are congestion and contention. The shaping actions are based on a variety of factors, but they are usually related to a traffic contract or SLA that must be guaranteed even under congestion conditions.

Traffic shaping is often used in conjunction with other mechanisms that perform functions that are necessary for traffic shaping to take place. One such function identifies and logically separates traffic into classes that have similar characteristics, so that shaping can occur. It's impossible to prioritize traffic classes with respect to each other if the traffic classes can't be identified. There are multiple ways to do this, depending on the environment. Differentiated services and VLAN tagging (IEEE 802.1q) are two such mechanisms.

The need for modern, flexible techniques that can be flexibly applied has resulted in the development of deep packet inspection (DPI). DPI inspects the contents of each packet in order to identify traffic type. But it goes further than that; DPI also maintains the state of each data flow and monitors the behavior of the flow to help identify it. It is even used for QoS enforcement and shaping in some systems, since it is able to affect traffic classes in a more stateful and intelligent fashion. DPI techniques are intelligent enough to drop or delay specific packets with respect to others, since it understands the impact of the delay or drop on the application won't be as destructive. The argument is that the random discard and blindly drop of packets in a flow would actually increase congestion because of retransmissions, etc.

The most common type of shaping involves grouping flows into classes that have the same traffic characteristics and then applying a common policy for the group. Once a type of flow is identified, it can then be considered and treated in the same fashion as all of the other flows having the same or similar characteristics, but differently from flows belonging to a separate group. Each traffic class has its own first-in first-out (FIFO) queue, which is normally regulated by the token bucket algorithm or a more modern enhancement. The algorithm is a control mechanism that allows traffic to flow at a certain average rate, but also allows for burstiness.

Traffic policing

Traffic policing is a traffic management function that ensures that the traffic at an network ingress point conforms to a traffic contract. Contracts can be viewed in various ways. A contract may be simply expressed in terms of bandwidth per user or group of users, or it may be more

granular, where certain applications are prioritized over others. DPI-based techniques make even more detailed contracts possible, enabling limiting based on the available bandwidth for very specific applications.

Packets are dropped by the policing function when the contract is exceeded. Traffic shaping, on the other hand, ensures that traffic conforms to a contract. Properly shaped traffic should never violate the policing function checks if both functions are based on the same traffic contract.

Admission control

Admission control determines whether to allow or deny access of a traffic flow into the network based on network resource availability. Based on its knowledge of current traffic types and bandwidth usage and of the QoS requirements for a new traffic flow, admission control can make an intelligent decision to allow the flow or not. The most common IP-based protocol used for this function is the RSVP protocol (Resource ReSerVation Protocol), described in the IETF RFC2205. It reserves resources across a network in order to comply with the flow's requested QoS. A number of extensions have been added to RSVP since its inception in order to accommodate newer demands, most notably RFC3209, which introduces traffic engineering – called RSVP-TE.

QoS and QoE

Different traffic types have different network requirements. This has become more evident and challenging as multi-service delivery on converged networks become more common. Network must handle traffic in such a way as to deliver different QoS expectations for each traffic type. QoS expectations are based on each traffic type's characteristics.

QoS requirements for most data protocols are usually low. Data traffic is not particularly latency sensitive, and also withstands packet loss easily. TCP handles retransmissions of lost packets such that data always arrives intact. Jitter is not a factor.

Real-time traffic, on the other hand, must be dealt with much more carefully. It has strict requirements for low jitter and latency. Voice traffic that exhibits high end-to-end latency is perceived as inacceptable. VoIP can sustain some packet loss, however, without having a disastrous effect on the quality. This is as opposed to video traffic, which is very sensitive to packet loss.

Table 48. Service traffic types and their QoS requirements

Services	QoS Expectations	Performance Attributes
High speed Internet	Low – best effort	• Variable bandwidth consumption • Latency and loss tolerant
Enterprise/ business services	High – critical data	• High bandwidth consumption • Highly sensitivity to latency • High security
Peer-to-peer	Low – best effort	• Very high bandwidth consumption • Latency and loss tolerant
Voice	High – low latency and jitter	• Low bandwidth – 21-320 Kbps per call • One-way latency < 150ms • One-way jitter < 30 ms
IPTV	High – low jitter and extremely low packet loss	• Very high bandwidth consumption • Very sensitive to packet loss
Mobility services	Moderate – low packet loss	• Moderate bandwidth consumption • Highly sensitive to latency
Gaming and interactive services	High – low packet loss	• Variable bandwidth consumption • One-way latency < 150ms • One-way jitter < 30 ms

Referring to Table 48, it is evident that certain types of traffic have stricter requirements than others; if those requirements are not met, users' QoE will be significantly degraded. Voice, video, and gaming all require special treatment because of their similar characteristics and sensitivities. Most data services, such as HTTP, FTP, e-mail and peer-to-peer, (P2P) are far less sensitive to the same errors. Delays when using these protocols doesn't normally negatively impact users' perception because latency and jitter arne't a factor and most use TCP as a reliable transport layer, which can retransmit lost data, and that the real-time requirements of latency and jitter aren't really a factor. In fact, for P2P traffic, which is very bandwidth hungry, it may be more desirable to limit total bandwidth than delay.

Test Case: Impact of Increased Best-Efforts Traffic on Real-Time Traffic QoS

Overview

This is an iterative test case that will verify the ability of a DUT's traffic management mechanism to sustain acceptable QoE levels for real-time traffic.

Objective

The objective of this test is to verify the performance of a DUT's traffic management functionality to maintain real-time traffic QoS under increasing load of best-effort traffic. A properly implemented network would leave real-time traffic's QoS relatively unaffected, while the QoS for best-effort traffic would decrease. This would demonstrate that real-time traffic received a higher priority than best-effort traffic.

Setup

This test uses multiple ports with varying types of traffic. Each traffic type (VoIP, video and data) will use its own NetTraffic pair, for a total of three NetTraffic pairs. For real-time traffic, VoIP (SIP) and video on demand (VOD) will be used, while HTTP, FTP, POP3 and SMTP will be used for best-effort data traffic.

Step-by-Step Instructions

In order to assess the effect of best-effort traffic on real-time traffic, it is first necessary to establish a baseline measurement. A baseline test involves the measurement of QoS when the DUT is subject exclusively to real-time traffic. This baseline will be used as the standard to judge the DUT's ability to handling increasing amounts of best-effort traffic.

In the following steps, we'll configure the final test. In order to perform the baseline test, merely disable the client and server NetTraffics for the best-effort , set a moderate traffic rate for the remaining traffic activities and run the test, noting the QoS values.

1. Launch IxLoad. In the main window, the **Scenario Editor** will be used for all test configuration.

   To become familiar with the IxLoad GUI, see the Getting Started Guide.

2. Add the client and server **NetTraffic** objects for the VoIP traffic. Configure the client and server networks with total IP count, and gateway and VLAN addresses, if used. The number of IP addresses used must be equal or greater than the number of user agents to be simulated.

3. Add and configure the **SIP client** and **SIP server** activities to the appropriate client and server **NetTraffics**. Refer to the **Test Variables** section above to configure the SIP parameters. See **Error! Reference source not found.** for configuration details.

Figure 79. IxLoad Scenario Editor with client and server side NetTraffics and VoIP activities

4. Add the client and server **NetTraffic** objects for the video traffic. Configure the client and server networks with total IP count, and gateway and VLAN addresses, if used.

5. Add and configure the **IPTV/Video client** and **IPTV/Video server** activities to the appropriate client and server **NetTraffics**. Refer to the **Test Variables** section above to configure the IPTV/Video parameters. See Table 50 for configuration details.

Figure 80. IxLoad Scenario Editor with client and server side NetTraffics and Video activities

6. Add the client and server **NetTraffic** objects for the best-effort traffic. Configure the client and server networks with total IP count, and gateway and VLAN addresses, if used.

7. Add and configure the **HTTP client** and **HTTP server** activities to the appropriate client and server **NetTraffics**. Refer to the **Test Variables** section above to configure the HTTP parameters. See Table 51 for configuration details.

8. Add and configure the **POP3 client** and **POP3 server** activities to the appropriate client and server **NetTraffics**. Refer to the **Test Variables** section above to configure the POP3 parameters. See Table 52 for configuration details.

9. Add and configure the **SMTP client** and **SMTP server** activities to the appropriate client and server **NetTraffics**. Refer to the **Test Variables** section above to configure the SMTP parameters. See Table 53 for configuration details.

10. Add and configure the **FTP client** and **FTP server** activities to the appropriate client and server **NetTraffics**. Refer to the **Test Variables** section above to configure the FTP parameters. See Table 54 for configuration details.

Figure 81. IxLoad Scenario Editor with client and server side NetTraffics and best effort data activities

The **Networks** and **Traffic** configuration should appear as in **Error! Reference source not ound.**

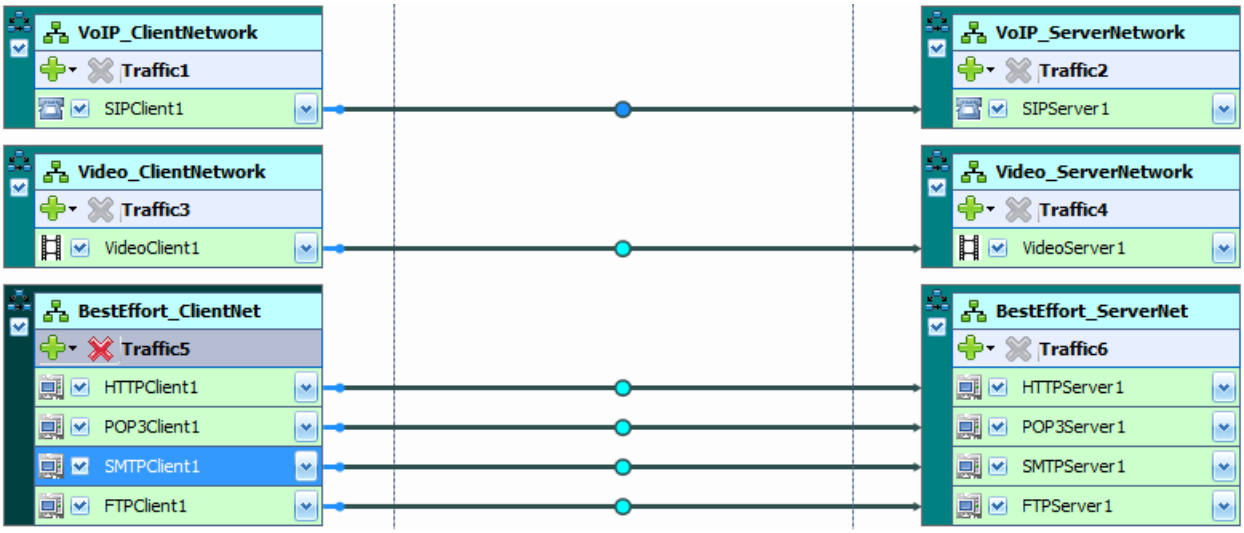


Figure 82. Complete Network and Traffic configuration

11. Having setup the client and server networks and the traffic profile, the test objective can now be configured.

    Open the **Timeline and Objective** view. Test **Objectives** will be applied on a per-activity basis. The iterative objectives will be set here and will be used between test runs to achieve the desired results.

    The following should be configured:

    - **Test Objective for SIP Client**. Set a moderate number of users such that when added to the video objective, it does not saturate the network to the maximum capacity.
    - **Test Objective for IPTV/Video Client**. Set a moderate number of users such that when added to the SIP objective, it does not saturate the network to the maximum capacity.

- **Test Objectives for best-effort clients** (HTTP, FTP, POP3 and SMTP). Set a series of moderate numbers of users such that when added together with the real-time activity objectives above, the total reaches close to the maximum throughput allowed by the network under test. This set of objectives will be iterated multiple times for the test.

Network Traffic Mapping	Objective Type	Objective Value	Timeline	Iteration Time	Total Time
New Traffic Flow					
Traffic1@VoIP_ClientNetwork	User Agents	800	Timeline1	000:00:56	000:00:56
SIPClient1	User Agents	800	Timeline1	000:00:56	000:00:56
Traffic3@Video_ClientNetwork	Simulated Users	500	Timeline2	000:01:00	000:01:00
VideoClient1	Simulated Users	500	Timeline2	000:01:00	000:01:00
Traffic5@BestEffort_ClientNet	Simulated Users	Mixed Value	Timeline3	000:01:05	000:01:05
HTTPClient1	Simulated Users	5000	Timeline3	000:01:05	000:01:05
POP3Client1	Simulated Users	400	Timeline3	000:01:05	000:01:05
SMTPClient1	Simulated Users	400	Timeline3	000:01:05	000:01:05
FTPClient1	Simulated Users	100	Timeline3	000:01:05	000:01:05
Traffic2@VoIP_ServerNetwork	N/A	N/A	<Match Longest>	000:01:05	000:01:05
SIPServer1	N/A	N/A	<Match Longest>	000:01:05	000:01:05
Traffic4@Video_ServerNetwork	N/A	N/A	<Match Longest>	000:01:05	000:01:05
VideoServer1	N/A	N/A	<Match Longest>	000:01:05	000:01:05
Traffic6@BestEffort_ServerNet	N/A	N/A	<Match Longest>	000:01:05	000:01:05
HTTPServer1	N/A	N/A	<Match Longest>	000:01:05	000:01:05
POP3Server1	N/A	N/A	<Match Longest>	000:01:05	000:01:05
SMTPServer1	N/A	N/A	<Match Longest>	000:01:05	000:01:05
FTPServer1	N/A	N/A	<Match Longest>	000:01:05	000:01:05

Figure 83. Figure 3 Test Objective Settings Dialog

Select the HTTP objective line, and using the **Timeline** pane below the **Objectives** pane set the Time to First Iteration to a value that allows the real-time traffic to reach sustain time before the best-effort traffic starts. The same parameter will be replicated to the POP3, SMTP and ftp timelines automatically.

For each of the activities, select moderate ramp-up values that will allow the activities to reach sustain time in a reasonable amount of time, without overload the network under test.

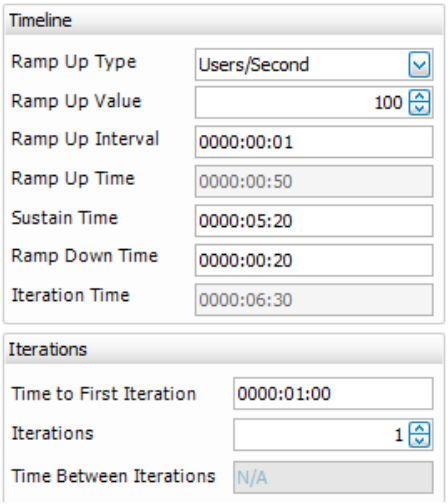


Figure 84. Timeline Settings Dialog for the http activity

12. After the **Test Objective** is set, the **approximate amount of ports required for the test** value on the bottom of the window indicates the total number of ports that are required for each load module type. Add each of the **NetTraffics** to an individual port. A minimum of six ports will be required to run the test.

13. Run the test.

    As the best-effort traffic ramps up, keep an eye on the QoS statistics for the real-time traffic.

    Interpretation of result statistics can sometimes be difficult, deducing what they mean under different circumstance. The **Results Analysis** section below provides a diagnostics-based approach to highlight some common scenarios, the statistics being reported, and how to interpret them.

14. Iterate through the test by incrementally increasing the objectives on the best effort traffic activities.

    The test tool can be started and stopped in the middle of a test cycle, or you can wait for it to be gracefully stopped using the test controls shown here. 

    To change the objectives while the test is running, make the **Run** window (**Error! Reference ource not found.**) visible by hovering over or clicking on the **Run** tab on the top left of the screen while in the **Statistics** view. The **Run** screen will appear, with the configured objectives shown. You can modify the objectives by either moving the corresponding slider or by entering a new value into the appropriate row of the **Objective Value** column.

Network Traffic Mapping	Objective Type	Objective Value	Run-Time Objective Value	Timeline	Total Time	Iterations	State
New Traffic Flow							
Traffic1@VoIP_ClientNetwork	User Agents	100		Timeline1	0000:06:42		Preparing Configuration
SIPClient1	User Agents	100		Timeline1	0000:06:42	0	Preparing Configuration
Traffic3@Video_ClientNetwork	Simulated Users	100		Timeline2	0000:06:44		Preparing Configuration
VideoClient1	Simulated Users	100		Timeline2	0000:06:44	0	Preparing Configuration
Traffic5@BestEffort_ClientNet	Simulated Users	Mixed Value		Timeline3	0000:06:30		Preparing Configuration
HTTPClient1	Simulated Users	5000		Timeline3	0000:06:30	0	Preparing Configuration
POP3Client1	Simulated Users	400		Timeline3	0000:06:30	0	Preparing Configuration
SMTPClient1	Simulated Users	400		Timeline3	0000:06:30	0	Preparing Configuration
FTPClient1	Simulated Users	100		Timeline3	0000:06:30	0	Preparing Configuration
Traffic2@VoIP_ServerNetwork	N/A	N/A		<Match Longest>	0000:06:44		Preparing Configuration
SIPServer1	N/A	N/A		<Match Longest>	0000:06:44	0	Preparing Configuration
Traffic4@Video_ServerNetwork	N/A	N/A		<Match Longest>	0000:06:44		Preparing Configuration
VideoServer1	N/A	N/A		<Match Longest>	0000:06:44	0	Preparing Configuration
Traffic6@BestEffort_ServerNet	N/A	N/A		<Match Longest>	0000:06:44		Preparing Configuration
HTTPServer1	N/A	N/A		<Match Longest>	0000:06:44	0	Preparing Configuration
POP3Server1	N/A	N/A		<Match Longest>	0000:06:44	0	Preparing Configuration
SMTPServer1	N/A	N/A		<Match Longest>	0000:06:44	0	Preparing Configuration
FTPServer1	N/A	N/A		<Match Longest>	0000:06:44	0	Preparing Configuration

Figure 85. Run screen showing the objectives during the test run

Test Variables

Test tool variables

Use the following test tool configuration profiles for each protocol activity pair.

SIP client and server configuration

Table 49. SIP Configuration for test case

Parameters	Client	Server
Network configuration	1,000 IP addresses “Use all” IP addresses option	1,000 IP address per test port
TCP parameters	Default	Default
SIP parameters	Default	Default
SIP scenarios	{Originate Call} | {Voice Session} | {End Call}	{Receive Call using 180} | {Voice Session}
SIP Perform MOS	Enabled, for all streams	N/A
Test Objective	User Agents	N/A

All default SIP parameters should be used except where noted above. For the **Voice Session** command of the scenario (both client and server), select *speech.wav* as the **Audio Pool File**, and set it to **Repeat Continuous** for *30s*.

IPTV/Video client and server configuration

Table 50. IPTV/Video configuration for Test case

Parameters	Client	Server
Network configuration	1,000 IP addresses “Use all” IP addresses option	1 IP address per test port
TCP parameters	Default	Default
IPTV/Video parameters	Default	Default
Mode	Video Client	Video
Quality Metrics	Enable	N/A
Test Objective	Simultaneous Users	N/A

When configuring of the **IPTV/Video server** activity, select the **Video Config** tab, and edit stream0 such that it uses a real payload, and select the Cloud-vs11-with audio(2Mbps).ts video file from the list of samples provided with the IxLoad installation.

HTTP client and server configuration

Table 51. HTTP configuration for Test case

Parameters	Client	Server
Network configuration	5,000 IP addresses "Use all" IP addresses option	1 IP address per test port
TCP parameters	TCP RX 32768, TCP TX 1024	TCP RX 1024, TCP TX 32768
HTTP parameters	HTTP/1.1 with keep-alive 3 TCP connections per user Max Transactions	Default settings
HTTP command list	1 GET command – payload of 256K bytes	N/A
Test Objective	Simultaneous Users	N/A

POP3 client and server configuration

Table 52. POP3 Configuration for Test case

Parameters	Client	Server
Network configuration	Same as HTTP client above	Same as HTTP server above
TCP parameters	Same as HTTP client above	Same as HTTP client above
POP3 parameters	Default	Default
Server Mail Messages	N/A	HTMLRandom: Random 1K to 32K HTML body
Test Objective	Simultaneous Users	N/A

SMTP client and server configuration

Table 53. SMTP Configuration for Test case

Parameters	Client	Server
Network configuration	Same as HTTP client above	Same as HTTP server above
TCP parameters	Same as HTTP client above	Same as HTTP client above
SMTP parameters	Default	Default
SMTP Mail Messages	RandomSmall: small random body random attachments	N/A
Test Objective	Simultaneous Users	N/A

FTP client and server configuration

Table 54. FTP Configuration for Test case

Parameters	Client	Server
Network configuration	Same as HTTP client above	Same as HTTP server above
TCP parameters	Same as HTTP client above	Same as HTTP client above
FTP parameters	Set FTP Mode to Passive	N/A
FTP command list	1 GET command – payload of 512000 bytes	N/A
Test Objective	Simultaneous Users	N/A

Note: if VLAN priority or DiffServ TOS is used by the network under test, you will need to set these values for each of the activities above, individually as necessary.

DUT test variables

The device or network under test should be configured to perform traffic shaping for at least two classes of traffic: best effort and real-time. Real-time traffic should be prioritized higher than best effort traffic.

Results Analysis

Since the focus of this test is QoS and QoE, only related statistics and results categories will be highlighted. The QoS key performance indicators (KPIs) that should be monitored on the real-time traffic are the video and voice call MOS scores. If they degrade significantly while the best effort traffic is increased, then the traffic management scheme is not working properly. It's the most obvious indicator: if the quality of the call degrades, there's a problem.

Less obvious indicators relate to other QoE variables, such as setup and teardown times, and retransmission of signaling requests. Significantly longer call setup times negatively impacts users' experience. Additionally, frequent SIP or RTSP retransmissions can be a symptom of signaling bandwidth challenges, which contribute to long setup times, or worse yet, service unavailability.

Table 55. Key performance indicators that require monitoring

Metric	Key Performance Indicators	Statistics View
VoIP and video QoS	Voice MOS Video MOS Jitter	SIP Client - RTP Video Client – MOS Scores SIP Client – RTP Jitter Video Client – Jitter Distribution
VoIP and video setup and teardown times	RTSP setup latency RTSP teardown latency	Video conditional view
Other Indicators	INVITE messages re-transmitted Non-INVITE messages re-transmitted	SIP Client – Sent Messages

The following figures show some of these statistics for real-time traffic QoS. Note that the **Video Client RTSP** latency statistics are obtained using conditional statistics.

SIP Client - RTP

Drill Down

Stat Name	2:44	2:46	2:48	2:50	2:52
Calls With Interrupted Path ...	0	0	0	0	0
Calls Without Path Confirma...	0	0	0	0	0
MOS Average Instant	4.410	4.410	4.410	4.410	4.410
MOS Best Instant	4.410	4.410	4.410	4.410	4.410
MOS Worst Instant	4.410	4.410	4.410	4.410	4.410
MOS Best	4.410	4.410	4.410	4.410	4.410
MOS Worst	4.410	4.410	4.410	4.410	4.410
MOS Average Per Call	4.410	4.410	4.410	4.410	4.410
MOS Best Per Call	4.410	4.410	4.410	4.410	4.410
MOS Worst Per Call	4.410	4.410	4.410	4.410	4.410
Bytes Sent	96,066,453	97,786,472	99,505,631	101,226,453	102,938,734
Packets Sent	558,526	568,526	578,521	588,526	598,481
Tx Packets Dropped	0	0	0	0	0
Bytes Received	96,037,041	97,757,060	99,476,219	101,197,041	102,908,986
Packets Received	558,355	568,355	578,350	588,355	598,308
Bad Packets Received	0	0	0	0	0
Misordered Packets Received	0	0	0	0	0
Packets Dropped By Jitter B...	0	0	0	0	0
Duplicate Packets Received	0	0	0	0	0
Lost Packets	0	0	0	0	0

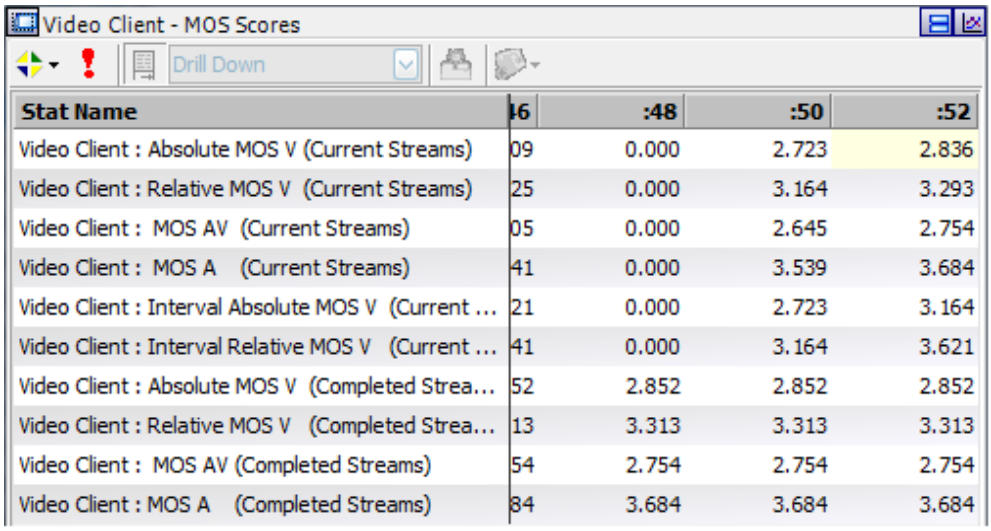
Figure 86. VoIP QoS statistics for test case

SIP Client - RTP Jitter

Drill Down

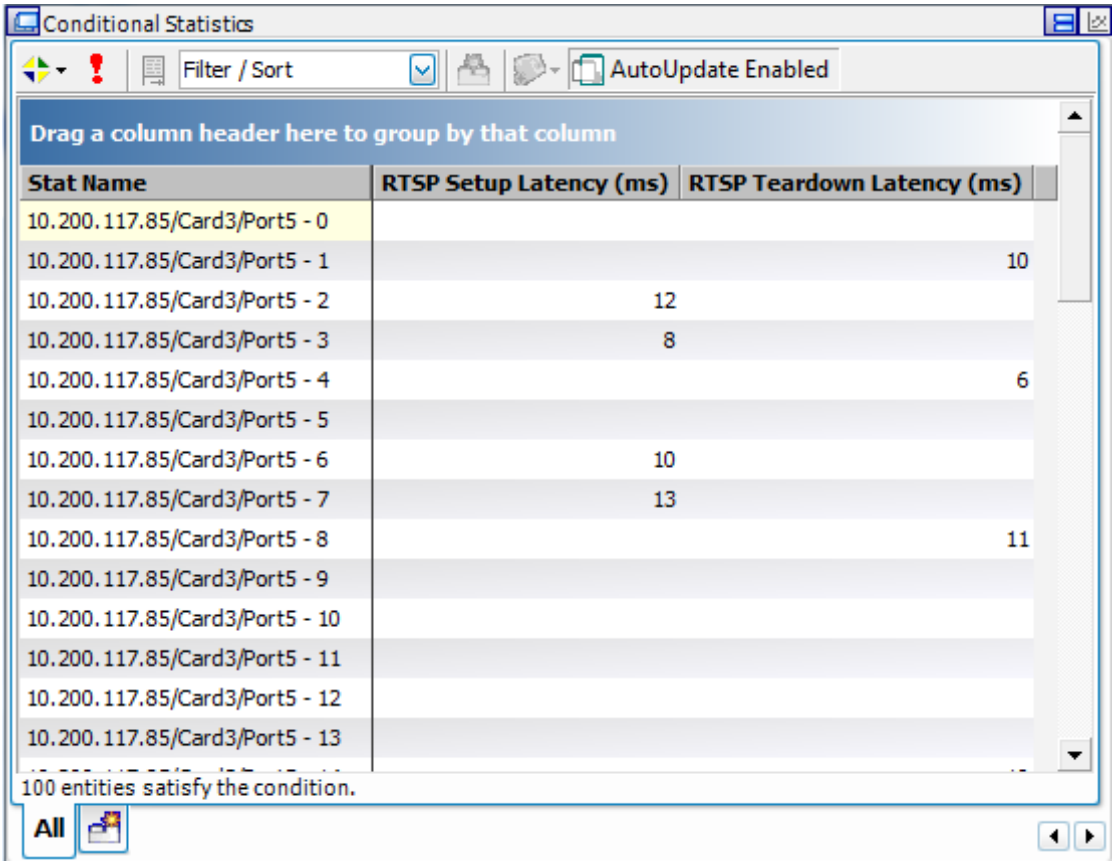
Stat Name	2:52	2:54	2:56	2:58
Jitter Min (ns)	62	21	20	20
Jitter Max (ns)	51,840	51,840	51,840	51,840
Packets With Jitter Up To 1ms	597,908	599,379	599,408	600,188
Packets With Jitter Up To 3ms	0	0	0	0
Packets With Jitter Up To 5ms	0	0	0	0
Packets With Jitter Up To 10ms	0	0	0	0
Packets With Jitter Up To 20ms	0	0	0	0
Packets With Jitter Up To 40ms	0	0	0	0
Packets With Jitter More Th...	0	0	0	0

Figure 87. VoIP jitter statistics for test case



Stat Name	46	:48	:50	:52
Video Client : Absolute MOS V (Current Streams)	09	0.000	2.723	2.836
Video Client : Relative MOS V (Current Streams)	25	0.000	3.164	3.293
Video Client : MOS AV (Current Streams)	05	0.000	2.645	2.754
Video Client : MOS A (Current Streams)	41	0.000	3.539	3.684
Video Client : Interval Absolute MOS V (Current ...	21	0.000	2.723	3.164
Video Client : Interval Relative MOS V (Current ...	41	0.000	3.164	3.621
Video Client : Absolute MOS V (Completed Strea...	52	2.852	2.852	2.852
Video Client : Relative MOS V (Completed Strea...	13	3.313	3.313	3.313
Video Client : MOS AV (Completed Streams)	54	2.754	2.754	2.754
Video Client : MOS A (Completed Streams)	84	3.684	3.684	3.684

Figure 88. Video MOS sores statistics for test case



Stat Name	RTSP Setup Latency (ms)	RTSP Teardown Latency (ms)
10.200.117.85/Card3/Port5 - 0		
10.200.117.85/Card3/Port5 - 1		10
10.200.117.85/Card3/Port5 - 2	12	
10.200.117.85/Card3/Port5 - 3	8	
10.200.117.85/Card3/Port5 - 4		6
10.200.117.85/Card3/Port5 - 5		
10.200.117.85/Card3/Port5 - 6	10	
10.200.117.85/Card3/Port5 - 7	13	
10.200.117.85/Card3/Port5 - 8		11
10.200.117.85/Card3/Port5 - 9		
10.200.117.85/Card3/Port5 - 10		
10.200.117.85/Card3/Port5 - 11		
10.200.117.85/Card3/Port5 - 12		
10.200.117.85/Card3/Port5 - 13		

Figure 89. Video RTSP latency statistics for test case

In order to view conditional statistics for video **RTSP Setup Latency** and **RTSP Teardown Latency**, select the **Conditional Statistics** view (**Error! Reference source not found.**) from he statistics views in the left pane. From the **Conditional Statistics** view, select the **Filter/Sort** drop-down list, followed by **Presentation**. Expand the **Video Client – Conditional** group, scroll down, select the appropriate statistics and add them to the right pane (**Error! Reference ource not found.**).

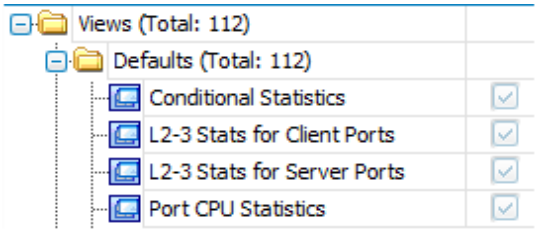


Figure 90. Conditional Statistics view selection

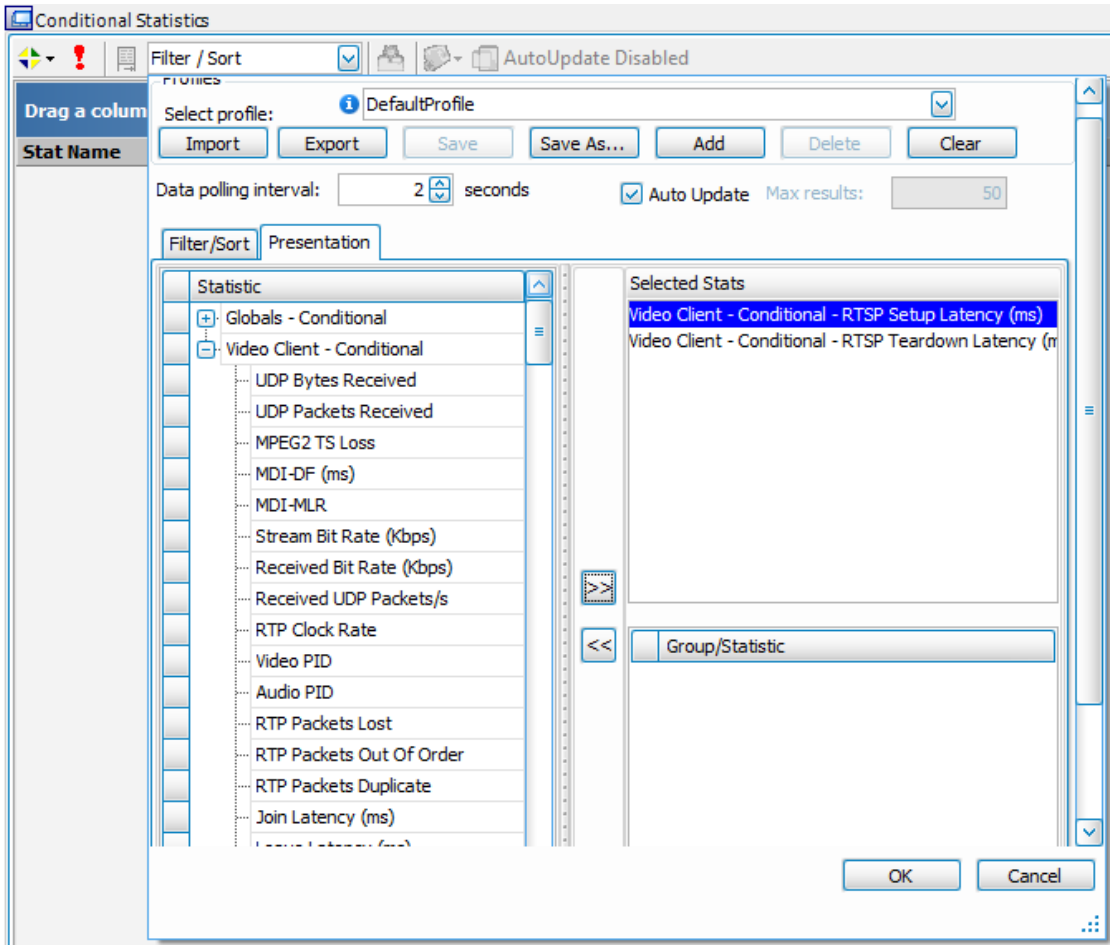


Figure 91. The Filter/Sort drop down window in the Conditional Statistics view

Test Case: Impact of Varying Real-Time Traffic on Best-Efforts Traffic QoS

Overview

This is an iterative test case that will verify the ability a DUT's traffic management mechanism to downgrade QoE levels for best effort traffic when real-time traffic increases.

Objective

The objective of this test is to verify the performance of a DUT's traffic management functionality to maintain real-time traffic QoS under the influence of increasing real-time traffic and a constant load of the best effort traffic. A properly implemented network would keep the real-time traffic QoS high, sacrificing the best-effort traffic's QoS. This would demonstrate that real-time traffic received a higher priority than best-effort traffic.

Setup

In order to assess the effect of best-effort traffic on real-time traffic, it is first necessary to establish a baseline measurement. A baseline test involves the measurement of QoS when the DUT is subject exclusively to real-time traffic. This baseline will be used as the standard to judge the DUT's ability to handling increasing amounts of best-effort traffic.

Step-by-Step Instructions

1. Follow steps 1-10 for the previous test case to configure the **NetTraffics** and **Activities**. See Step 1through Step 10 for configuration details.
2. Having setup the client and server networks and the traffic profile, the test objective can now be configured.

   Open the **Timeline and Objective** view. Test **Objectives** will be applied on a per-activity basis. The iterative objectives will be set here and will be used between test runs to achieve the desired results.

   The following should be configured:

   - **Test Objectives for best-effort clients** (HTTP, FTP, POP3 and SMTP). Set a series of medium scale number of users such that when added together, the total throughput does not reach the maximum throughput allowed by the DUT.
   - **Test Objective for SIP Client**. Set a moderate number of users such that when added to the video objective and the best-effort traffic, the total does not saturate the network to the maximum capacity. This objective will be iterated multiple times for the test.

- **Test Objective for IPTV/Video Client**. Set a moderate number of users such that when added to the SIP objective and the best-effort traffic, it does not saturate the network to the maximum capacity. This objective will be iterated multiple times for the test.

Select the **SIP Objective** line, open the **Timeline** pane below the **Objectives** pane, and set the **Time to First Iteration** to a value that allows the best-effort traffic to reach sustain time before the real-time traffic starts. Set the same value for the **Video Objective**.

For each of the activities, select moderate ramp-up values that allow the activities to reach sustain time in a reasonable amount of time, without overloading the network under test.

3. Once the **Test Objective** is set, the **approximate amount of ports required for the test** value on the bottom indicates the total number of ports that are required for each type of load module. Add each of the **NetTraffics** to an individual port. A minimum of six ports will be required to run the test.

4. Run the test.

   As the real-time traffic ramps up, keep an eye on the QoS statistics for the real-time traffic.

   Interpretation of result statistics can sometimes be difficult, deducing what they mean under different circumstance. The **Results Analysis** section below provides a diagnostics-based approach to highlight some common scenarios, the statistics being reported, and how to interpret them.

5. Iterate through the test by incrementally increasing the objectives on the real-time traffic activities.

   The test tool can be started and stopped in the middle of a test cycle, or you can wait for it to be gracefully stopped using the test controls shown here. 

   To change the objectives while the test is running, make the **Run** window (**Error! Reference ource not found.**) visible by hovering over or clicking on the **Run** tab on the top left of the screen while in the **Statistics** view. The **Run** screen will appear, with the configured objectives shown. You can modify the objectives by either moving the corresponding slider or by entering a new value into the appropriate row of the **Objective Value** column.

Network Traffic Mapping	Objective Type	Objective Value	Run-Time Objective Value	Timeline	Total Time	Iterations	State
New Traffic Flow							
Traffic1@VoIP_ClientNetwork	User Agents	100		Timeline1	0000:06:42		Preparing Configuration
SIPClient1	User Agents	100		Timeline1	0000:06:42	0	Preparing Configuration
Traffic3@Video_ClientNetwork	Simulated Users	100		Timeline2	0000:06:44		Preparing Configuration
VideoClient1	Simulated Users	100		Timeline2	0000:06:44	0	Preparing Configuration
Traffic5@BestEffort_ClientNet	Simulated Users	Mixed Value		Timeline3	0000:06:30		Preparing Configuration
HTTPClient1	Simulated Users	5000		Timeline3	0000:06:30	0	Preparing Configuration
POP3Client1	Simulated Users	400		Timeline3	0000:06:30	0	Preparing Configuration
SMTPClient1	Simulated Users	400		Timeline3	0000:06:30	0	Preparing Configuration
FTPClient1	Simulated Users	100		Timeline3	0000:06:30	0	Preparing Configuration
Traffic2@VoIP_ServerNetwork	N/A	N/A		<Match Longest>	0000:06:44		Preparing Configuration
SIPServer1	N/A	N/A		<Match Longest>	0000:06:44	0	Preparing Configuration
Traffic4@Video_ServerNetwork	N/A	N/A		<Match Longest>	0000:06:44		Preparing Configuration
VideoServer1	N/A	N/A		<Match Longest>	0000:06:44	0	Preparing Configuration
Traffic6@BestEffort_ServerNet	N/A	N/A		<Match Longest>	0000:06:44		Preparing Configuration
HTTPServer1	N/A	N/A		<Match Longest>	0000:06:44	0	Preparing Configuration
POP3Server1	N/A	N/A		<Match Longest>	0000:06:44	0	Preparing Configuration
SMTPServer1	N/A	N/A		<Match Longest>	0000:06:44	0	Preparing Configuration
FTPServer1	N/A	N/A		<Match Longest>	0000:06:44	0	Preparing Configuration

Figure 92. Run screen showing the objectives during the test run

Test Variables

Test tool variables

The test tool variables are identical to the variables used in test 1. Please refer to Table 49 through Table 54 for details.

DUT test variables

The DUT, or the network under test if applicable, should be configured to perform traffic shaping for at least two classes of traffic: best effort and real-time. Real-time traffic should be prioritized higher than best effort traffic.

Results Analysis

For test 2, the objective was to monitor the impact of increasing real-time traffic on the QoS of the best-effort traffic, while ensuring that the real-time traffic exhibits acceptable QoS KPIs. For this test, the best-effort KPIs should actually decrease as the real-time traffic is increased.

Table 56. Key performance indicators that require monitoring

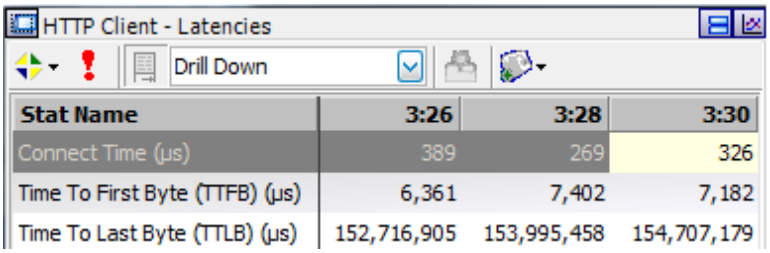
Metric	Key Performance Indicators	Statistics View
VoIP and video QoS	Voice MOS Video MOS Jitter	SIP Client - RTP Video Client – MOS Scores SIP Client – RTP Jitter Video Client – Jitter Distribution
VoIP and video setup and teardown times	RTSP setup latency RTSP teardown latency	Video conditional view
Best-effort data latencies	HTTP connect time HTTP Time to first byte HTTP time to last byte	HTTP Client - Latencies
Best-effort data TCP failures	TCP resets sent TCP retries	HTTP Client – TCP Failures FTP Client – TCP Failures
Best-effort data throughput	SMTP throughput POP3 throughput	SMTP Client – Throughput objective POP3 Client – Throughput objective
Other Indicators	INVITE messages re-transmitted Non-INVITE messages re-transmitted	SIP Client – Sent Messages

Table 56 shows a series of KPIs for both the real-time traffic and best-effort traffic that require monitoring. Specifically, real-time traffic indicators that will degrade are:

- **Latencies**. The time to establish a connection should become slower, and the time required to download a web page should increase as well. Good indicators of these conditions are the connect time, time to first byte, and time to last byte.

- **TCP failures**. As best-effort traffic is de-prioritized with increasing real-time traffic, the probability of getting TCP failures will increase. This will be indicated by the TCP failure KPIs listed in Table 56.
- **Data throughput**. As traffic increases, the throughput for the best-effort data should decline as the total bandwidth allocated for best-effort traffic decreases. All of the activities used for best-effort traffic in this test have statistics views for the throughput.

Some of the real-time statistic views for the KPIs mentioned above are shown in figures below.



Stat Name	3:26	3:28	3:30
Connect Time (µs)	389	269	326
Time To First Byte (TTFB) (µs)	6,361	7,402	7,182
Time To Last Byte (TTLB) (µs)	152,716,905	153,995,458	154,707,179

Figure 93. HTTP client latencies statistics view

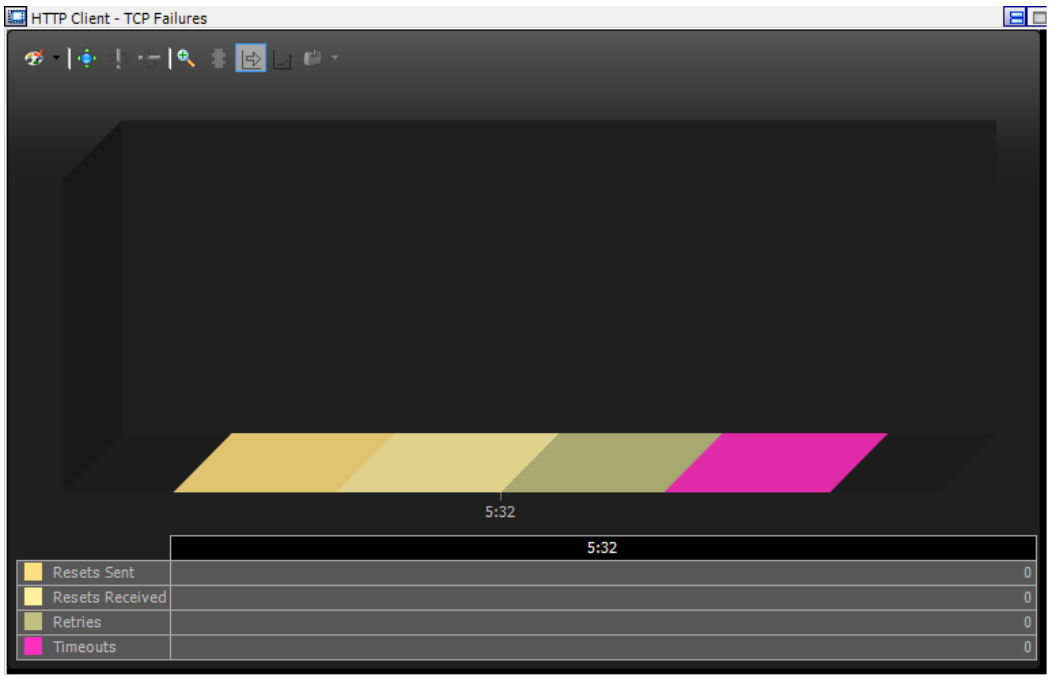
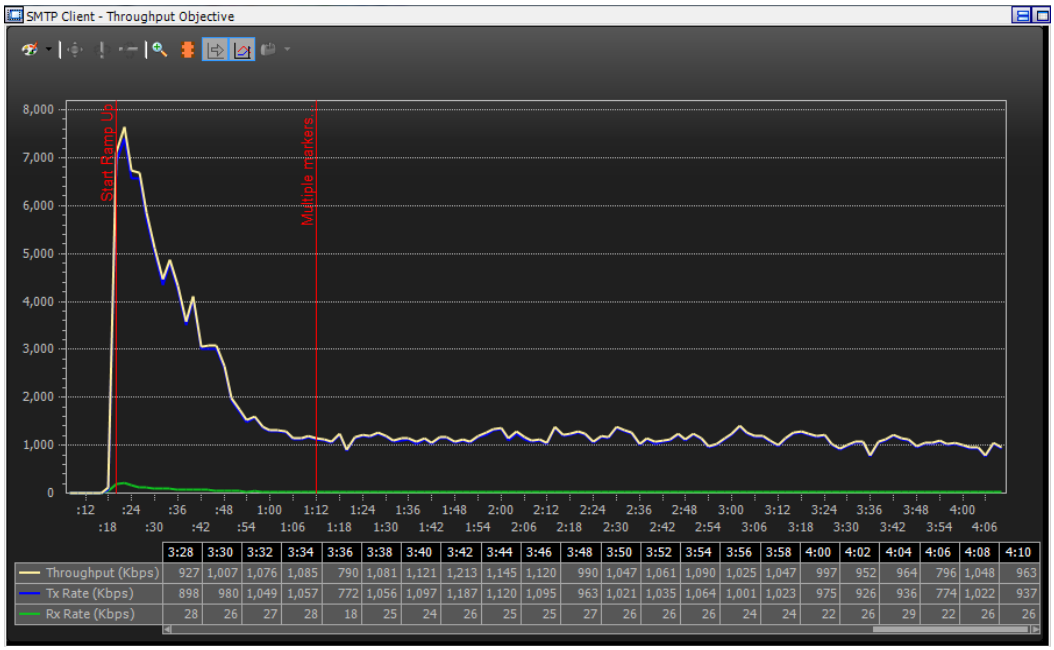


Figure 94. HTTP TCP errors statistics view



	3:28	3:30	3:32	3:34	3:36	3:38	3:40	3:42	3:44	3:46	3:48	3:50	3:52	3:54	3:56	3:58	4:00	4:02	4:04	4:06	4:08	4:10
Throughput (Kbps)	927	1,007	1,076	1,085	790	1,081	1,121	1,213	1,145	1,120	990	1,047	1,061	1,090	1,025	1,047	997	952	964	796	1,048	963
Tx Rate (Kbps)	898	980	1,049	1,057	772	1,056	1,097	1,187	1,120	1,095	963	1,021	1,035	1,064	1,001	1,023	975	926	936	774	1,022	937
Rx Rate (Kbps)	28	26	27	28	18	25	24	26	25	25	27	26	26	26	24	24	22	26	29	22	26	26

Figure 95. SMTP client throughput objective statistics view

Test Case: Maximum Capacity and Performance of DPI Devices

Overview

Deep packet inspection (DPI) is a technique used to uncover the true nature of the traffic traversing IP networks. It works by inspecting each packet layer, including layer 7 protocol contents. Traditional packet inspection mechanisms only analyze layer 2 through 4 contents, including the source address, destination address, source port, destination port and the protocol type.

DPI analyzes the application layer packets, identifying the application source and contents.

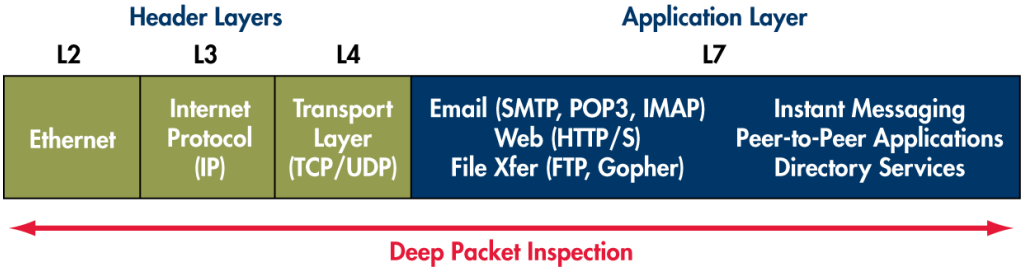


Figure 96. Deep Packet Inspection

The application-awareness offered by DPI was initially used by service providers to identify and manage P2P traffic and security threats. Providers are now looking towards DPI to manage their subscriber community and deliver application-level quality of service (QoS) appropriate for each type of service traffic – voice, video, P2P, gaming, email, etc.

In order to identify and classify application types, DPI devices need to inspect every bit of traffic traversing its path and match patterns against signature or heuristic libraries containing patterns that correspond with standard Internet protocols or application behavior of certain protocol flows.

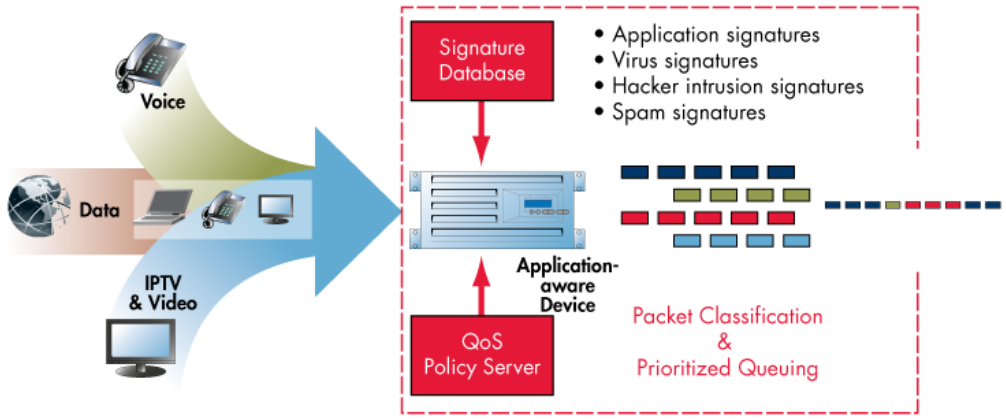


Figure 97. DPI Architecture

Once a traffic flow has been inspected and classified, the QoS policies that are relevant to that application type or subscriber need to be applied to ensure that the traffic is placed in the appropriate priority queue. These activities make DPI a resource intensive task, especially when the device is processing tens of gigabits per second of traffic from millions of subscribers.

DPI Usage

The benefits of DPI technology far outweigh the complexity. DPI gives service providers the application awareness with per-subscriber granularity that is needed to unlock the revenue potential of their converged IP networks. With greater awareness and intelligence, operators can provide better value to subscribers, content owners and advertisers. Service providers and mobile operators are looking at or already deploying these advanced DPI devices to deliver the advanced functionalities and features discussed in Table 57.

Table 57. DPI Features and Deployment Use Cases

Features	Description
Stateful inspection and monitoring	Inspects every IP flow, identifying the application contained within the flow and the subscriber that generated it. This provides service providers with a complete picture of network utilization on a per-application and per-subscriber basis.
Security	Identifies denial of service/distributed denial of service (DoS/DDoS), zero day attacks and other traffic anomalies that may impact network performance and integrity.
Traffic management and policy enforcements	Prioritizes, expedites and shapes application traffic flows to control congestion and ensure fair use of the network resources. Service provider can define dynamic quality of service (QoS) policies to be applied to application traffic in order to manage the way in which bandwidth is apportioned to applications and subscribers.
Tiered service management	Mobile and broadband service providers use DPI as a means to implement tiered service plans, offering differentiated services. This enables operators to tailor service offerings to individual subscribers and increase their average revenue per user (ARPU). A policy is created per user or group of users. The DPI system enforces that policy, allowing users access to different services and applications with varying quality of service.
Dynamic subscriber management and billing	Some DPI infrastructures have centralized quota management and accounting capabilities. This allows operators to monitor real-time subscriber usage and to implement consumption-based billing models or quota thresholds.

Features	Description
Lawful intercept, copyright enforcement and ad Insertion	Service providers are sometimes required by various government agencies to allow lawful intercept. Copyright owners, such record labels and movie studios are also turning towards DPI technology to enforce copyright protection. Operators are also looking at ways to monetize subscriber-specific information that they gained while monitoring service and usage patterns that enable external marketing companies to perform targeted advertising.

DPI Deployment

DPI devices are typically deployed towards the edge of the service provider network, as shown in the following figure, making it a critical point in the end-to-end service delivery.

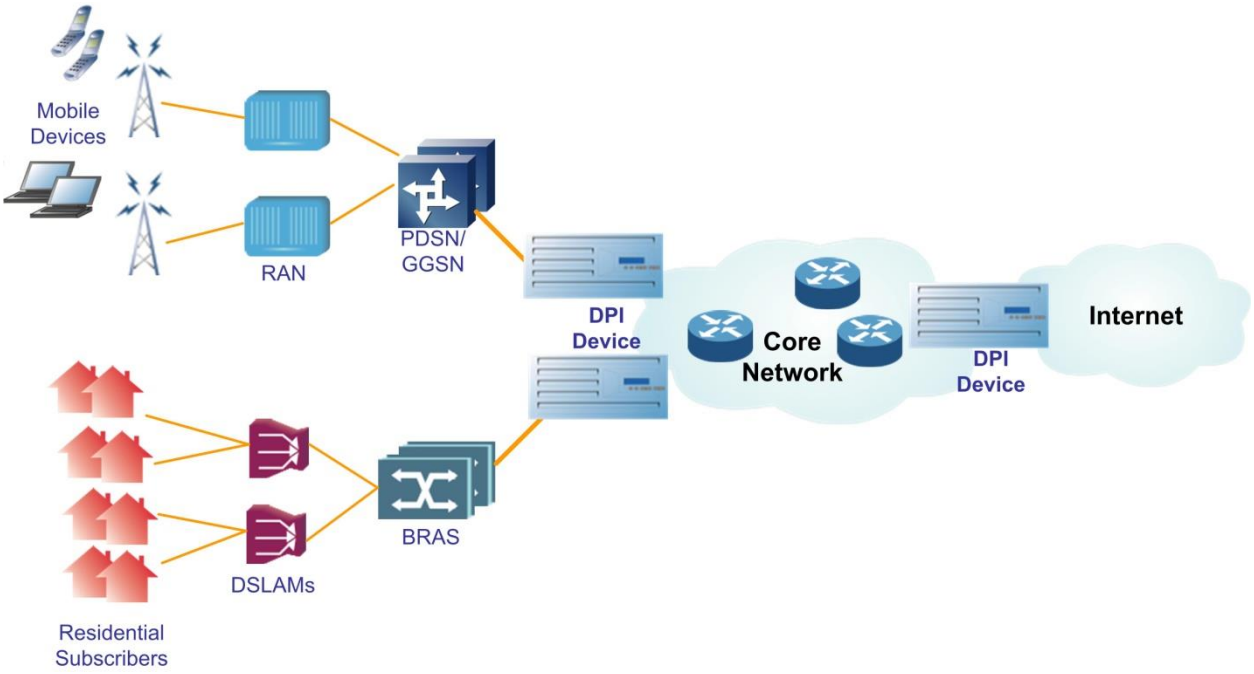


Figure 98. Typical DPI Deployment

All the traffic that enters the core network will traverse the DPI devices, where every flow is inspected, classified and forwarded with the appropriate policy. A poor performing DPI device could be a serious bottleneck, affecting service quality and, in turn, subscriber satisfaction.

DPI devices can play a crucial role in the management and the long-term profitability of next generation multiplay networks. When service providers and mobile operators are evaluating these devices, they need to determine the scalability of the DPI device so that they can accurately model the network capacity accounting for expected subscriber growth.

Key DPI Scalability Metrics

The scalability of DPI devices needs to be measured in a multi-dimensional fashion, where the following metrics are simultaneously maximized to determine the performance under realistic conditions.

- Maximum number of subscribers that can be supported
- Maximum concurrent connections, or flows, that can be inspected and classified
- Additional connections per second that can be serviced when the device is already servicing a large number of flows
- Maximum throughput that can be sustained under peak subscriber load

These are some of the system performance quoted by some DPI vendors:

Table 58. Typical performance of DPI platforms

Metric	Vendor A	Vendor B	Vendor C
Throughput	240G	20G	80G
Subscribers	5M	500K	5 M
Concurrent Connections/Flows	200M	N/A	40 M
New Connections per second	6 M	N/A	1 M

Most DPI devices are deployed in clusters, so that capacity can be incrementally increased. The performance figures in Table 2 are for fully populated system. Performance and capacity can be measured in modular fashion with a smaller test bed, however.

Another key point to note is that with any application-aware device, performance is adversely impacted if advanced functions are enabled. Therefore it is critical to understand and measure the performance under real-world conditions with features that are used in your network.

Test Case: Measuring Max DPI Capacity and Performance with HTTP

Objective

Determine the scalability and capacity limits of a DPI device under realistic load conditions.

Each performance metric can be measured in isolation using the methodologies defined in the **Application Delivery** Black Book chapter; for example, maximum connections per second, maximum concurrent connections and maximum throughput. These test cases provide a good baseline of the maximum capacity under ideal conditions.

The test methodology discussed here maximizes multiple performance metrics simultaneously in a multi-dimensional fashion that mimics realistic DPI deployment conditions. The results of the test reveal the true capacity in terms of number of subscribers, concurrent flows/services and maximum bandwidth.

Setup

The setup requires multiple server and client/subscriber ports to emulate end-to-end service delivery. In this test HTTP application traffic will be used as the benchmark because web traffic is the most prevalent traffic type carried. To test more realistic network conditions, several other protocols can be added.

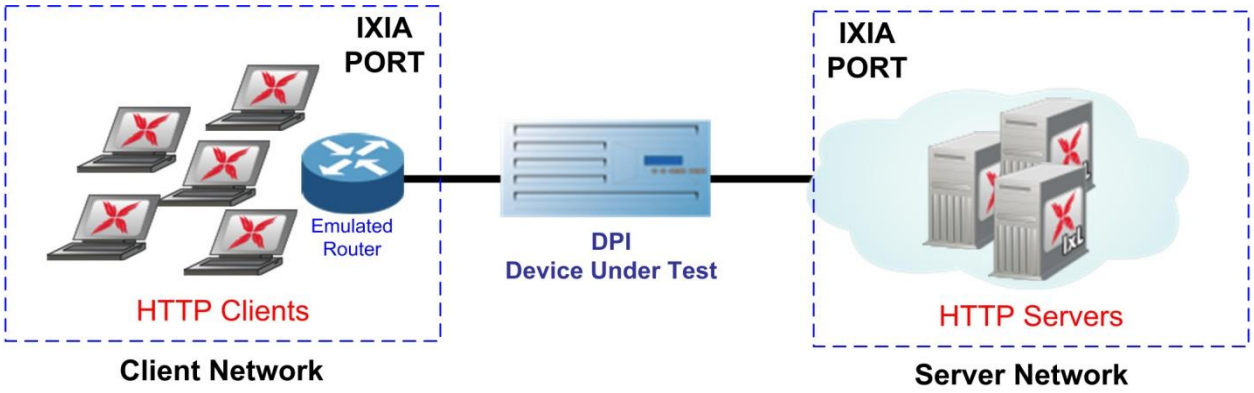


Figure 99. Figure 4 DPI Capacity Test Topology

Test Variables

Test Tool Variables

The following test configuration parameters provide the flexibility to create the HTTP traffic profile that is representative of a high load condition that is sustained during a peak-hours usage.

Table 59. Test tool variables

Parameter	Description
Client network	• 1.2 million IP addresses, or as many IP addresses as required to emulate a subscriber community. • Vary the MSS parameter to smaller values to emulate a mobile subscriber
HTTP client parameters	• HTTP/1.1 without keep-alive • 8 TCP connections per user representing 8 parallel connections per user • 1 transaction per TCP connection
TCP parameters	• TCP Rx and Tx buffer of 4096 bytes
HTTP client command list	• 1-3 GET commands – payload of 8k-256K byte page size
HTTP servers	• 1 per Ixia test port, or more
HTTP server parameters	• Random response delay: 0 – 20 ms • Response timeout: 300 ms
Other protocols	• Use IxLoad's application replay feature to emulate protocol flows that are unique to a provider network or subscriber community, FTP, SMTP, RTSP, SIP or a combination

DUT Test Variables

There are several DUT scenarios. The following list outlines some of the capabilities of the DUT that can be switched on.

- Enable stateful inspection engines
- Increase the number of traffic management policies
- Enable blocking and traffic shaping mechanisms
- Enable subscriber monitoring and management features
- Enable URL and other content filters including copyright enforcement

Step-by-Step Instructions

Configure a baseline test, which is an Ixia port-to-port test, in order to verify the test tool's performance. Once you have obtained the baseline performance, setup the DUT and the test tool as per the **Setup** section above. Refer to the **Test and DUT Variables** sections above for recommended configurations. When configuring the DUT test, a layer 2 switch with a high-performance backplane is highly recommended.

Fill in the table below with the baseline performance to use as a reference of the test tool's performance, based on the quantity and type of hardware available.

Table 60. Reference baseline performance form

Performance indicator	Value per port pair	Load module type
Throughput		
Connections/sec		
Concurrent Connections		

1. Launch IxLoad. In the main window, all of the test configurations will be made using the **Scenario Editor** window.

   To get familiar with the IxLoad GUI, see the Getting Started Guide section.

2. Add the client **NetTraffic** object. Configure the **Client** network with the total of number of IP addresses.

   Since DPI devices don't directly interface with the subscriber network you will need to simulate an **Emulated Router** to protect the DPI device from any ARP storm.

   a. Add an **IP Stack** with an **Emulated Router.** The emulated router gateway should connect to DUT interface.

      To add the **Emulated Router** right-click on the **MAC/VLAN Stack** and follow the cascaded menu as show in the following figure.

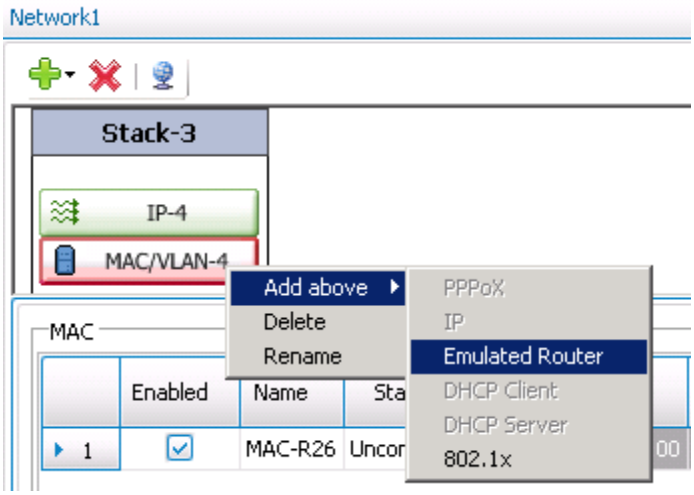
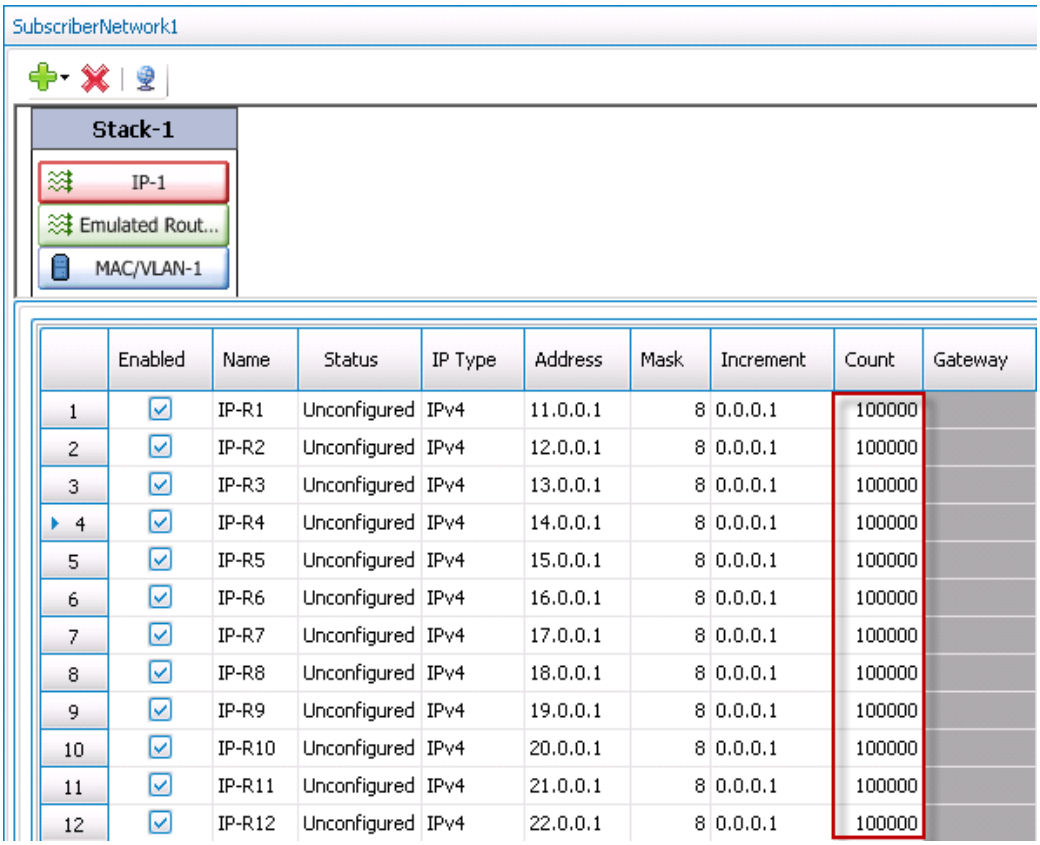


Figure 100. Creating an IP Network Stack with Emulated Routers

b. On the **IP stack**, configure the desired number of static IP addresses. Typically Ixia load modules support 128K IP addresses per port. If you are using an Acceleron-NP load module, you can create 12 network ranges with100K IP addresses for each port, which will deliver 1.2 million IP addresses. Also ensure you have 12 emulated routers created, one for each IP range.



	Enabled	Name	Status	IP Type	Address	Mask	Increment	Count	Gateway
1	☑	IP-R1	Unconfigured	IPv4	11.0.0.1	8	0.0.0.1	100000	
2	☑	IP-R2	Unconfigured	IPv4	12.0.0.1	8	0.0.0.1	100000	
3	☑	IP-R3	Unconfigured	IPv4	13.0.0.1	8	0.0.0.1	100000	
4	☑	IP-R4	Unconfigured	IPv4	14.0.0.1	8	0.0.0.1	100000	
5	☑	IP-R5	Unconfigured	IPv4	15.0.0.1	8	0.0.0.1	100000	
6	☑	IP-R6	Unconfigured	IPv4	16.0.0.1	8	0.0.0.1	100000	
7	☑	IP-R7	Unconfigured	IPv4	17.0.0.1	8	0.0.0.1	100000	
8	☑	IP-R8	Unconfigured	IPv4	18.0.0.1	8	0.0.0.1	100000	
9	☑	IP-R9	Unconfigured	IPv4	19.0.0.1	8	0.0.0.1	100000	
10	☑	IP-R10	Unconfigured	IPv4	20.0.0.1	8	0.0.0.1	100000	
11	☑	IP-R11	Unconfigured	IPv4	21.0.0.1	8	0.0.0.1	100000	
12	☑	IP-R12	Unconfigured	IPv4	22.0.0.1	8	0.0.0.1	100000	

Figure 101. IP Network Ranges with 1.2 million IP addresses representing unique subscriber

3. Add the server **NetTraffic**. Also configure the total number of servers that will be used. Ensure you have at least *1* server IP per test port.

   For a step-by-step workflow, see Appendix A.

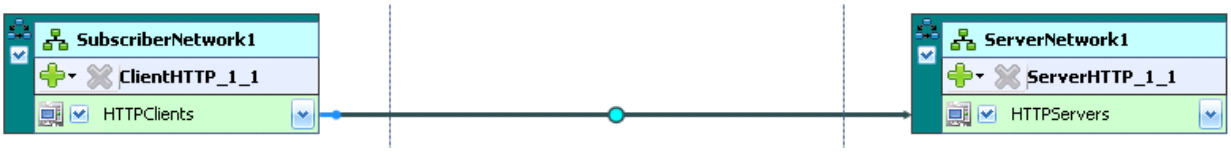


Figure 102. IxLoad Scenario Editor view with Client and Server side NetTraffics and Activities

4. TCP parameters are important when optimizing the test tool. Refer to the **Test Variables** defined in Table 3, to set the correct TCP parameters.

   There are several other parameters that can be changed. Leave them at their defaults values unless you need to modify them for testing requirements.

   For a step-by-step workflow, see Appendix B.

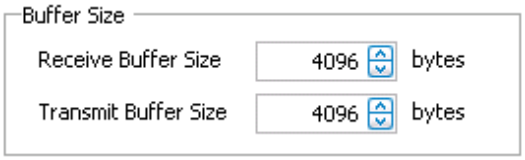


Figure 103. TCP Buffer Settings Dialog

5. Configure the **HTTP server**. Add the **HTTP Server Activity** to the server **NetTraffic**. The defaults are sufficient for this testing.

   For a step-by-step workflow, see Appendix C.

6. Configure the **HTTP client**. Add the **HTTP Client Activity** to the client **NetTraffic**.

   For a step-by-step workflow, see Appendix D.

   a. Refer to **Test Variables** section above to configure the HTTP parameters.

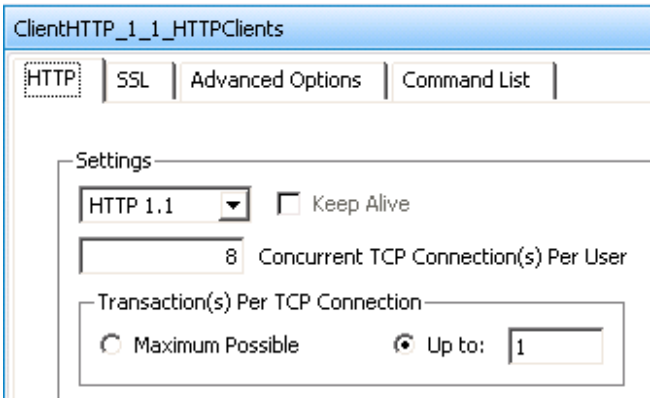


Figure 104. HTTP client protocol settings dialog

   b. Configure the transaction size for your connection. A single transaction size can be used for all the subscriber connection or it may be varied. Keep in mind that smaller page sizes will maximize the connections per second although throughput will decrease. It will be necessary to experiment with a few different transaction combinations to characterize both the connections per second (CPS) and throughput performance.

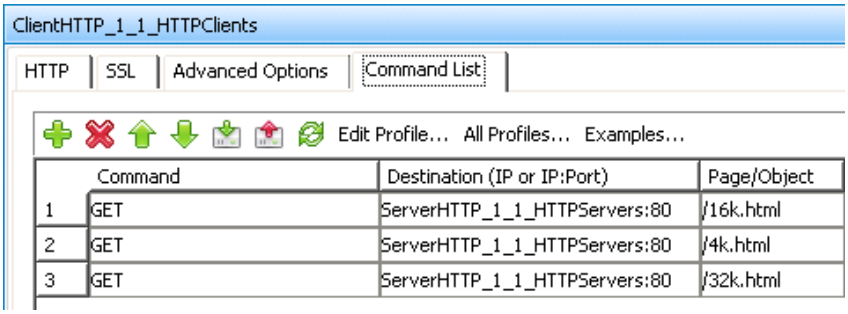


Figure 105. HTTP Command List

7. Having setup the client and server networks and the traffic profile, the test objective can now be configured.

   Open the **Timeline and Objectives** view. Select the test **Objective Type** to be **Concurrent Connections** and **Objective Value** to be *9,600,000*. The concurrent connection objective is chosen in order to activate all the subscribers and all connections per user - 1.2 million subscribers multiplied by 8 connections per subscriber equals 9.6 million concurrent connections.

   The following should be configured:

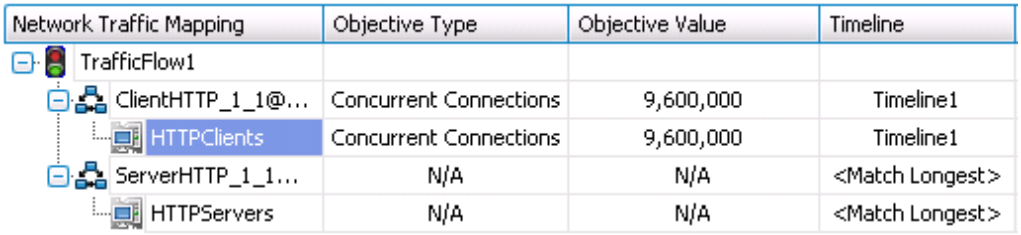


Figure 106. Test Objective Settings

   For a step-by-step workflow, see Appendix E.

8. Run the test for few minutes to allow the performance to reach a steady-state. Steady state is referred as **Sustain** duration in the test. Continue to monitor the DUT for the target rate and any failure/error counters. See the **Results Analysis** section below for important statistics and diagnostics information.

   Interpretation of result statistics can sometimes be difficult, deducing what they mean under different circumstance. The **Results Analysis** section below provides a diagnostics-based approach to highlight some common scenarios, the statistics being reported, and how to interpret them.

9. Iterate through the test varying the number of subscribers, the numbers of connections per subscriber and size of the transactions to determine the steady state capacity of the DUT.

   a. The test tool can be started and stopped in the middle of a test cycle, or wait for it to be gracefully stopped using the test controls shown here. 

   For a step-by-step workflow, see Appendix F.

Results Analysis

To analyze forwarding performance and determine the maximum capacity it is necessary to examine multiple test iterations, comparing the results for all test parameter variations, including:

- Number of IP addresses or subscribers
- Number of connection per subscribers
- Number of concurrent flows/connections
- Transaction size

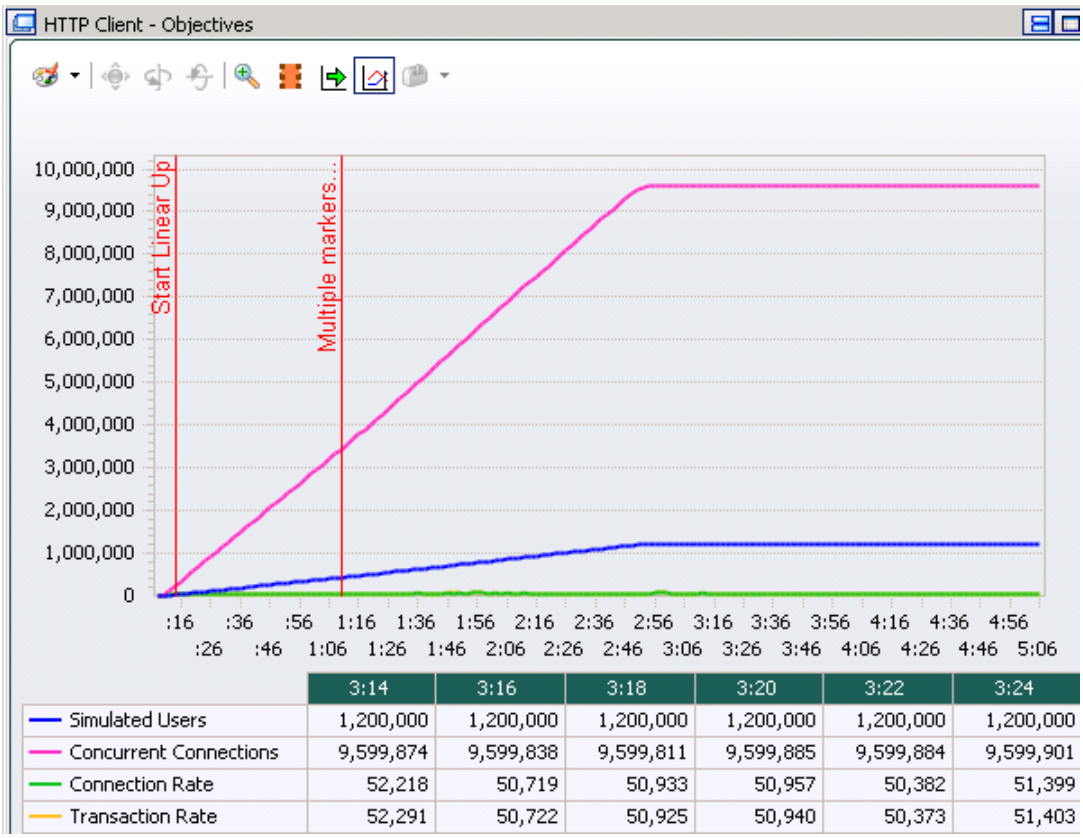
Table 61 lists the key performance statistics that must be monitored. These statistics help determine if the device has reached its saturation point and to troubleshoot performance issues. Interpreting the results in the correct manner will also ensure that transient network, device or test tool behavior does not create a false positive condition.

Table 61. Key HTTP performance statistics

Metric	Key Performance Indicators	Statistics View
Performance metrics	Connections/sec, total connections, number of simulated users, throughput	HTTP Client – Objectives HTTP Client – Throughput
Application level transactions Application level failure monitoring	Requests sent, successful, failed request aborted, timeouts, 4xx, 5xx errors, session timeouts, connect time, time to first/last byte	HTTP Client – Transactions HTTP Client – HTTP Failures HTTP Client – Latencies
TCP connection information TCP failure monitoring	SYNs sent, SYN/SYN-ACKs received, RESET sent, RESET received, retries, timeouts	HTTP Client – TCP Connections HTTP Client – TCP Failures

Real-Time Statistics

The figure below**Error! Reference source not found.** provides a real-time view of the easured performance for a test run. The **Concurrent Connection** statistic shows that the DUT is able to sustain approximately 9.6 million concurrent flows/connections. 1.2 million **Simulated Users** are also achieved, which is equivalent to the number of subscribers that were emulated, ensuring all the IP addresses configured in the test were utilized.



	3:14	3:16	3:18	3:20	3:22	3:24
Simulated Users	1,200,000	1,200,000	1,200,000	1,200,000	1,200,000	1,200,000
Concurrent Connections	9,599,874	9,599,838	9,599,811	9,599,885	9,599,884	9,599,901
Connection Rate	52,218	50,719	50,933	50,957	50,382	51,399
Transaction Rate	52,291	50,722	50,925	50,940	50,373	51,403

Figure 107. HTTP Client Objective Statistics View

The figure below **Error! Reference source not found.** is a real-time view of the **HTTP hroughput,** which is equivalent to the DUT's goodput.

HTTP Client - Throughput Objective

	3:14	3:16	3:18	3:20	3:22	3:24
Throughput (Kbps)	8,251,162	8,004,224	8,037,351	8,040,488	7,952,019	8,110,678
Tx Rate (Kbps)	19,111	18,562	18,639	18,649	18,439	18,811
Rx Rate (Kbps)	8,232,051	7,985,663	8,018,712	8,021,839	7,933,553	8,091,580

Figure 108. HTTP Client Throughput (goodput)

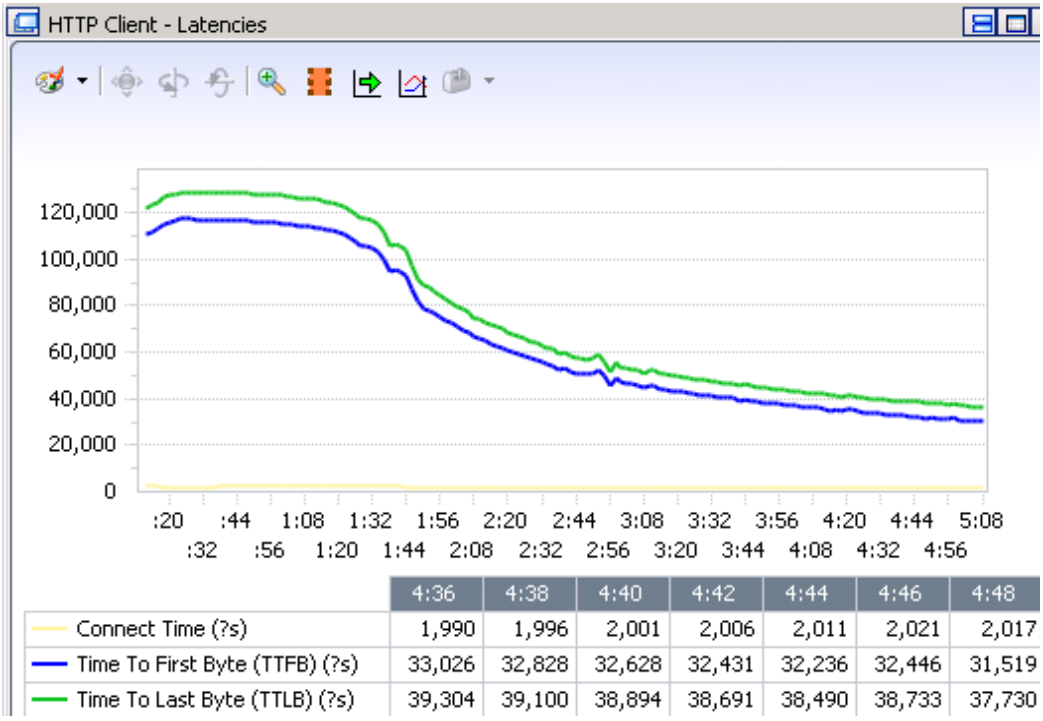
The throughput can be viewed by reviewing the **L2-3 Stats.** These statistics indicate the device's sustained forwarding performance. This view is very useful, as most DPI devices report their line-rate throughput instead of goodput.

	3:14	3:16	3:18
Bits Received Rate (Kb/s) [L2-3 Stats for Client Ports]	8,618,011	8,637,386	8,585,439
Bits Sent Rate (Kb/s) [L2-3 Stats for Client Ports]	294,606	293,813	293,304

Figure 109. Layer 2-3 statistics showing line-rate throughput

The connections per second and throughput metrics are directly influenced by the transaction/page size used in the test. If the goal of the test is to maximize the CPS in conjunction with concurrent flows then the page size will have to be small. If throughput has to be maximized with concurrent flows, the page size must be large. Packet size tends to be smaller in mobile networks, which can be modeled with smaller transaction size or by altering the MSS value in the network layer parameters.

To identify if a device has reached its active connections limit, refer to the **Latencies** view, as shown in the following figure**Error! Reference source not found.**. In this graph, the **TTFB** was igher during the ramp-up period, indicating that the device/server accepting a burst of connection requests. A continuous increase is also visible in the **Connect Time,** which is an indication that the device is slowing down.



	4:36	4:38	4:40	4:42	4:44	4:46	4:48
Connect Time (?s)	1,990	1,996	2,001	2,006	2,011	2,021	2,017
Time To First Byte (TTFB) (?s)	33,026	32,828	32,628	32,431	32,236	32,446	31,519
Time To Last Byte (TTLB) (?s)	39,304	39,100	38,894	38,691	38,490	38,733	37,730

Figure 110. HTTP Client Latency statistics

Troubleshooting and Diagnostics

Table 62. Troubleshooting and diagnostics

Issue	Diagnosis and Suggestions
The concurrent connections value never reaches steady state. It's always moving up and down and the variation is reasonably high.	Check the connect time and TCP failures. If the connect time keeps increasing, then it's an indication that the device may not be able to service or maintain any connections.
What are the best statistics that can be used to certify that the optimal capacity has been reached?	The relatively quick way to know if the device is at maximum capacity is to incrementally add new test ports, and see if the overall concurrent connections, connections per second and throughput metrics increase. If TCP failures occur and new connections are failing, then it's an indication that the limit has been reached. Other indications include a high latency, including connect Time, TTFB and TTLB. Concurrent flows and number of simultaneous subscribers that can be supported is a function of system memory. Refer to the memory usage on the device, which should indicate that it's near its high water mark.

Test Case: Maximum DPI Capacity and Performance with Multiplay Traffic

Objective

Determine the scalability and capacity limits of a DPI device with a realistic blend of application traffic. The previous test methodology Test Case: Measuring Max DPI Capacity and Performance with HTTP provided a good baseline performance metric under high load condition. In this test methodology a blended traffic mix will be used in order to assess the performance when inspection and traffic management features of the DPI device are fully exercised.

This test methodology will also maximize multiple performance metrics, including the number of subscribers, parallel application flows and throughput to mimic realistic DPI deployment conditions. The results will provide an understanding of the true capacity that can be supported when crucial functionality is enabled.

Setup

This test requires multiple server and client/subscriber ports to emulate end-to-end service delivery. In this test a multiplay application traffic mix will be used to exercise the inspection engine and traffic management capabilities of the DUT under high load condition.

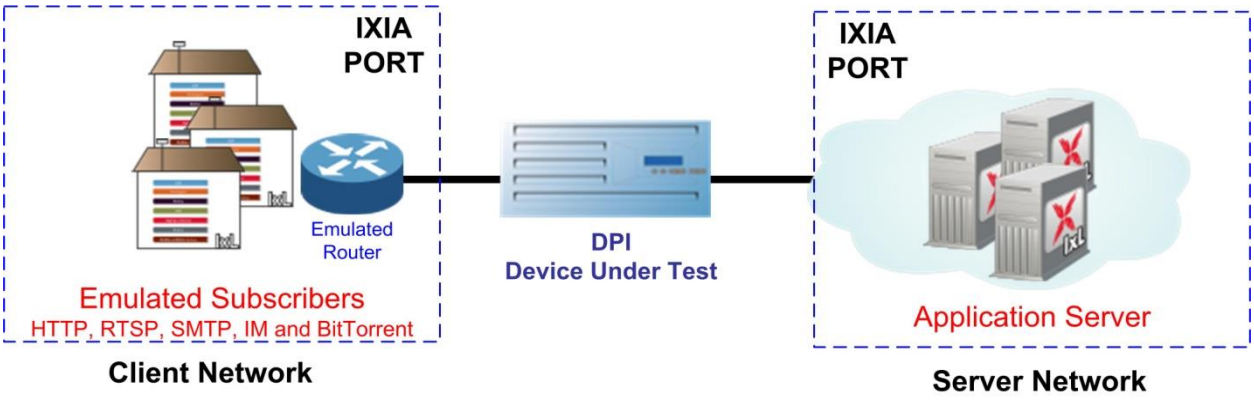


Figure 111. DPI Capacity Test Topology

Test Variables

Test Tool Variables

The following test configuration parameters are used to create a traffic profile that represents a subscriber community under a sustained peak-hour load.

Table 63. Test Tool Variables

Parameter	Description
Client network	120K IP addresses, or as many IP addresses as are required to emulate a subscriber community
TCP parameters	TCP Rx and Tx buffer at 4096 bytes
Application replay initiator parameters	Five custom flows representing five different application types. For this test we will use Facebook (HTTP), Streaming Media (RTSP), Yahoo Messenger, Email (SMTP) and BitTorrent. Any application mix that is representative of your subscriber community can be created. Five concurrent flows
Application replay responder parameters	One IP range per Ixia test port, or more

DUT Test Variables

There are several DUT scenarios. The following list outlines some of the DUT's capabilities that can be switched on.

- Enable stateful inspection engines
- Increase the number of traffic management policies
- Enable blocking and traffic shaping mechanisms
- Enable subscriber monitoring and management features
- Enable URL and other content filters including copyright enforcement

Step-by-Step Instructions

Configure a baseline test, which is an Ixia port-to-port test, in order to verify the test tool's performance. Once you have obtained the baseline performance, setup the DUT and the test tool as per the Setup section above. Refer to the Test and DUT Variables sections above for recommended configurations. When configuring the DUT test, a layer 2 switch with a high-performance backplane is highly recommended.

Fill in the table below with the baseline performance to use as a reference of the test tool's performance, based on the quantity and type of hardware available.

Table 64. Reference baseline performance form

Performance indicator	Value per port pair	Load module type
AppReplay Throughput		
AppReplay Connections/sec		
AppReplay Concurrent Connection		

1. Launch IxLoad. In the main window, all of the test configurations will be made using the **Scenario Editor** window.

   To get familiar with the IxLoad GUI, see the Getting Started Guide section.

2. Add the client **NetTraffic** object. Configure the **Client** network with the total of number of IP addresses.

   Since DPI devices don't directly interface with the subscriber network you will need to simulate an **Emulated Router** to protect the DPI device from any ARP storm.

   a. Add an **IP Stack** with an **Emulated Router.** The emulated router gateway should connect to DUT interface.

To add the **Emulated Router** right-click on the **MAC/VLAN Stack** and follow the cascaded menu as show in the following figure.

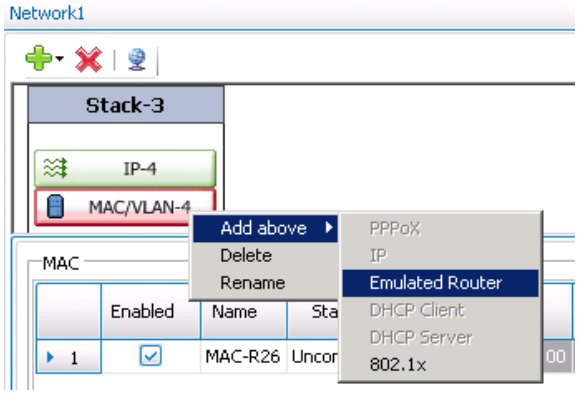
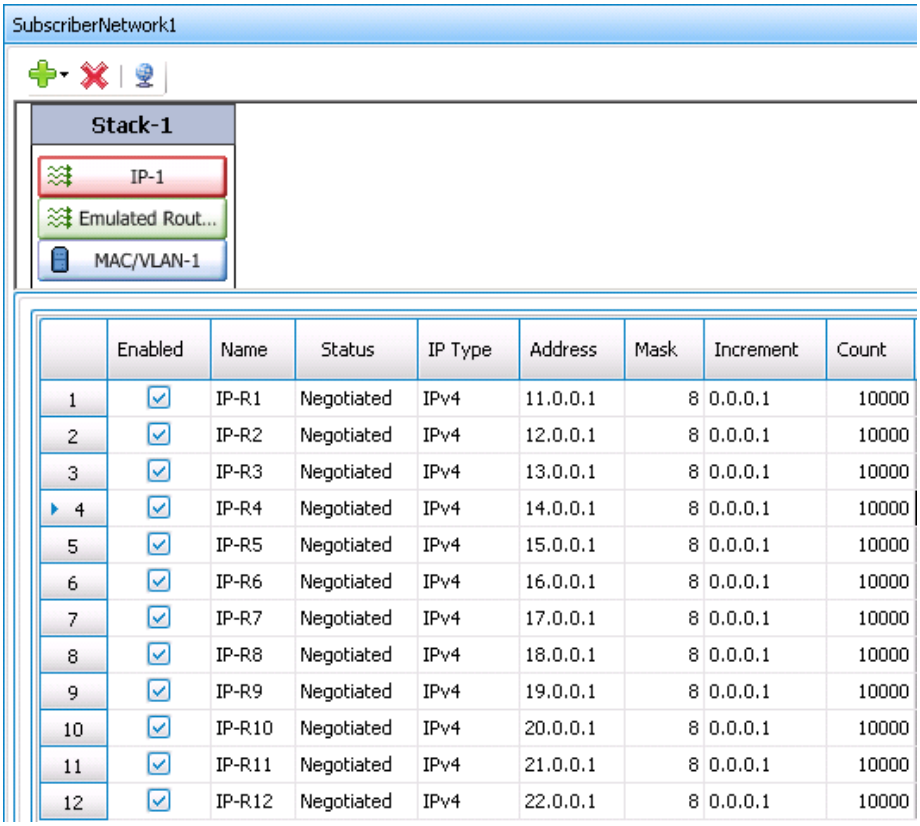


Figure 112. Creating an IP Network Stack with Emulated Routers

b. On the **IP stack**, configure the desired number of static IP addresses. Typically Ixia load modules support 128K IP addresses per port. If you are using an Acceleron-NP load module, you can create 12 network ranges with 10K IP addresses for each port, which will emulate 120K IP addresses for the test. Also ensure you have 12 emulated routers created, one for each IP range.



	Enabled	Name	Status	IP Type	Address	Mask	Increment	Count
1	☑	IP-R1	Negotiated	IPv4	11.0.0.1	8	0.0.0.1	10000
2	☑	IP-R2	Negotiated	IPv4	12.0.0.1	8	0.0.0.1	10000
3	☑	IP-R3	Negotiated	IPv4	13.0.0.1	8	0.0.0.1	10000
4	☑	IP-R4	Negotiated	IPv4	14.0.0.1	8	0.0.0.1	10000
5	☑	IP-R5	Negotiated	IPv4	15.0.0.1	8	0.0.0.1	10000
6	☑	IP-R6	Negotiated	IPv4	16.0.0.1	8	0.0.0.1	10000
7	☑	IP-R7	Negotiated	IPv4	17.0.0.1	8	0.0.0.1	10000
8	☑	IP-R8	Negotiated	IPv4	18.0.0.1	8	0.0.0.1	10000
9	☑	IP-R9	Negotiated	IPv4	19.0.0.1	8	0.0.0.1	10000
10	☑	IP-R10	Negotiated	IPv4	20.0.0.1	8	0.0.0.1	10000
11	☑	IP-R11	Negotiated	IPv4	21.0.0.1	8	0.0.0.1	10000
12	☑	IP-R12	Negotiated	IPv4	22.0.0.1	8	0.0.0.1	10000

Figure 113. IP network ranges with 240K IPs representing unique subscriber

3. Add the server **NetTraffic**. Also configure the total number of servers that will be used. Ensure you have at least *1* server IP per test port.

   For a step-by-step workflow, see Appendix A.

4. TCP parameters are important when optimizing the test tool. Refer to the **Test Variables** section in Table 1 to set the correct TCP parameters.

   There are several other parameters that can be changed. Leave them at their defaults values unless you need to modify them for testing requirements.

   For a step-by-step workflow, see Appendix B.

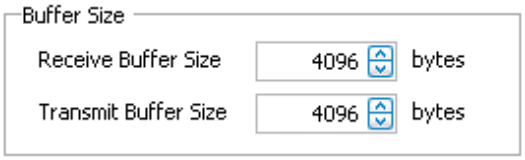


Figure 114. TCP Buffer Settings

5. Configure the **AppReplay Initiator** and **AppReplay Responder** activities. Add the **AppReplay Peer Activity** to both the client and server NetTraffic.

Figure 115. IxLoad scenario editor view with Client/Server NetTraffics and AppReplay Peer Activities

   a. Refer to the **Test Variables** section above to configure the **AppReplay Initiator** parameters.

      The **AppReplay Activity** can be used to replay application flows using captured packets. The application flows are then replayed statefully. This accommodates devices that require stateful message handshakes in order to operate properly. AppReplay leverages IxLoad's powerful infrastructure to provide massive scalability and performance.

   b. Add a **Custom Flow** for each application type.

      i. Add a **Custom Flow – TCP** command for each **Application Flow** type.

      ii. Specify the **Destination** as the **AppReplay Peer** Activity configured on the server side **NetTraffic.**

      iii. Give the **Flow Identifier** a meaningful name that you can reference later, during result analysis

      iv. Specify the **Capture File** location.

v. If there are multiple flows within the capture file you can use the **Follow First SYN** or the **User defined Filter** options to isolate the flow that you want to replay.

vi. Follow the 5 steps above to create **Custom Flows** that represent Streaming Media, Yahoo Messenger, Email and BitTorrent.

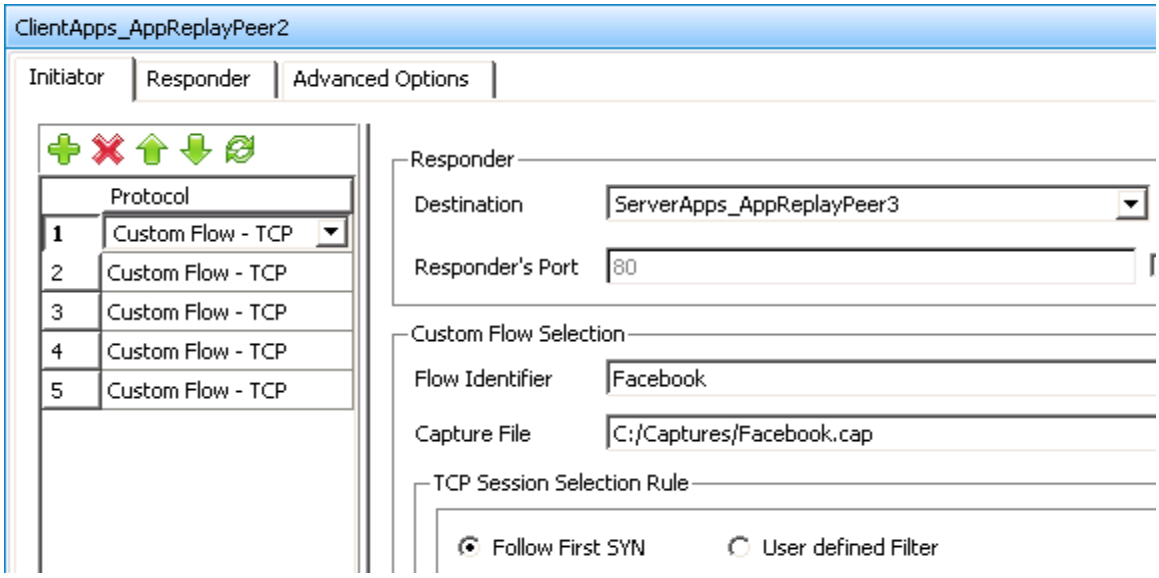


Figure 116. AppReplay peer activity command settings

c. In the **Advanced Options** tab define the **Maximum Concurrent Flow(s) per user** as *5*, so that all 5 flows will be replayed in parallel, for each subscriber. Also enable **Verify Length** as the **Payload Verification Option** for maximum performance.

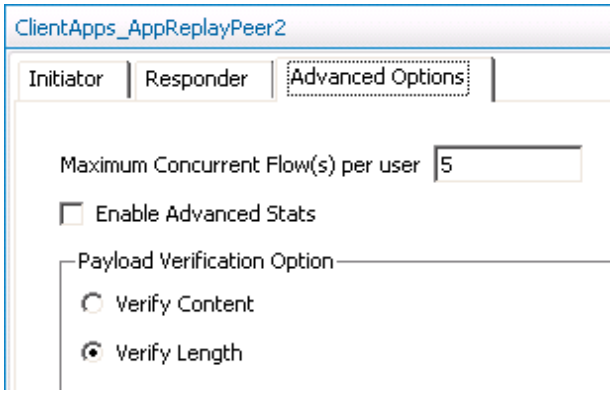


Figure 117. AppReplay peer activity advanced options

d. The responder AppReplay peer Activity on the server side will be automatically configured based on the client side initiator settings. Only the **Payload Verification Option** needs to be set as **Verify Length** for maximum performance.

6. Having setup the client and server networks and the traffic profile, the test objective can now be configured.

Open the **Timeline and Objectives** view. Select the test **Objective Type** to be **Initiator Peer Count** and **Objective Value** to be *120,000.* The **Initiator Peer Count** objective is chosen in order to activate all the subscribers and all 5 custom flows per user – 120K subscribers multiplied by 5 connections per subscriber equals 600K concurrent connections/flows.

The following should be configured:

Network Traffic Mapping	Objective Type	Objective Value	Timeline
TrafficFlow1			
ClientApps@SubscriberNetw...	Initiator Peer Count	120,000	Timeline1
AppReplayPeer2	Initiator Peer Count	120,000	Timeline1
ServerApps@ServerNetwork1	N/A	N/A	<Match Longest>
HTTPServer	N/A	N/A	<Match Longest>

Figure 118. Test objective settings

For a step-by-step workflow, see Appendix E.

7. Run the test for few minutes to allow the performance to reach a steady-state. Steady state is referred as **Sustain** duration in the test. Continue to monitor the DUT for the target rate and any failure/error counters. See the **Results Analysis** section below for important statistics and diagnostics information.

   Interpretation of result statistics can sometimes be difficult, deducing what they mean under different circumstance. The **Results Analysis** section below provides a diagnostics-based approach to highlight some common scenarios, the statistics being reported, and how to interpret them.

8. Iterate through the test varying the number of subscribers, the numbers of connections per subscriber and size of the transaction to determine the steady state capacity of the DUT.

   a. The test tool can be started and stopped in the middle of a test cycle, or wait for it to be gracefully stopped using the test controls shown here. 

For a step-by-step workflow, see Appendix F.

Results Analysis

To analyze forwarding performance and determine the maximum capacity it is necessary to examine multiple test iterations, comparing the results for all test parameter variations, including:

- Number of IP addresses or subscribers
- Number of concurrent flows/connections per subscriber
- Different application flows and size/duration of the flows

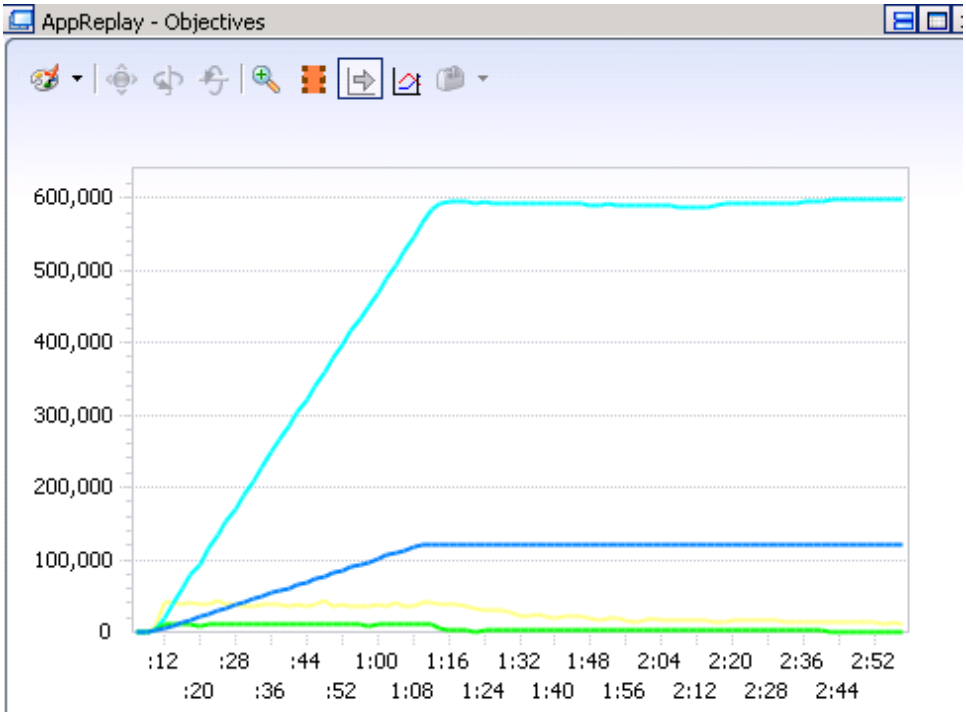
Table 65 lists the key performance statistics that must be monitored. These statistics help determine if the device has reached its saturation point and to identify any issues. Interpreting the results in the correct manner will also ensure that transient network, device or test tool behavior does not create a false positive condition.

Table 65. Key AppReplay performance statistics to monitor

Metric	Key Performance Indicators	Statistics View
Performance metrics	Connections/sec, total connections, number of simulated users, throughput	AppReplay – Objectives AppReplay – Initiator Throughput
Application level transactions Application level failure monitoring	Flows Initiated, succeeded, failed, error. Per-flow statistics if advanced stats option is enabled	AppReplay – Flows AppReplay – Per Flow Stats Initiator HTTP Client – Latencies
TCP connection information TCP failure monitoring	SYNs sent, SYN/SYN-ACKs received, RESET sent, RESET received, retries, timeouts. Connect time latency.	AppReplay – TCP Connections AppReplay – TCP Failures AppReplay – TCO Connections Latency Bin

Real-Time Statistics

Error! Reference source not found. provides a real-time view of the measured performance or a test run. The **Concurrent Connection** statistic shows that the DUT is able to sustain approximately 600K concurrent flows/connections. 120K **Initiator Peers** were emulated, which is equivalent to subscribers, ensuring all the IP configured addresses were utilized.



	2:48	2:50	2:52	2:54	2:56
AppReplay Peer : Initiator Peer Count	120,000	120,000	120,000	120,000	120,000
AppReplay Peer : Connection Rate	710	743	820	804	786
AppReplay Peer : Concurrent Connections	596,753	596,638	596,754	597,023	597,221
AppReplay Peer : Transactions Rate	13,665	14,439	12,963	11,744	12,406

Figure 119. AppReplay objective statistics

The following figure is the real-time view of the **AppReplay Throughput,** which is equivalent to the good put of the DUT.

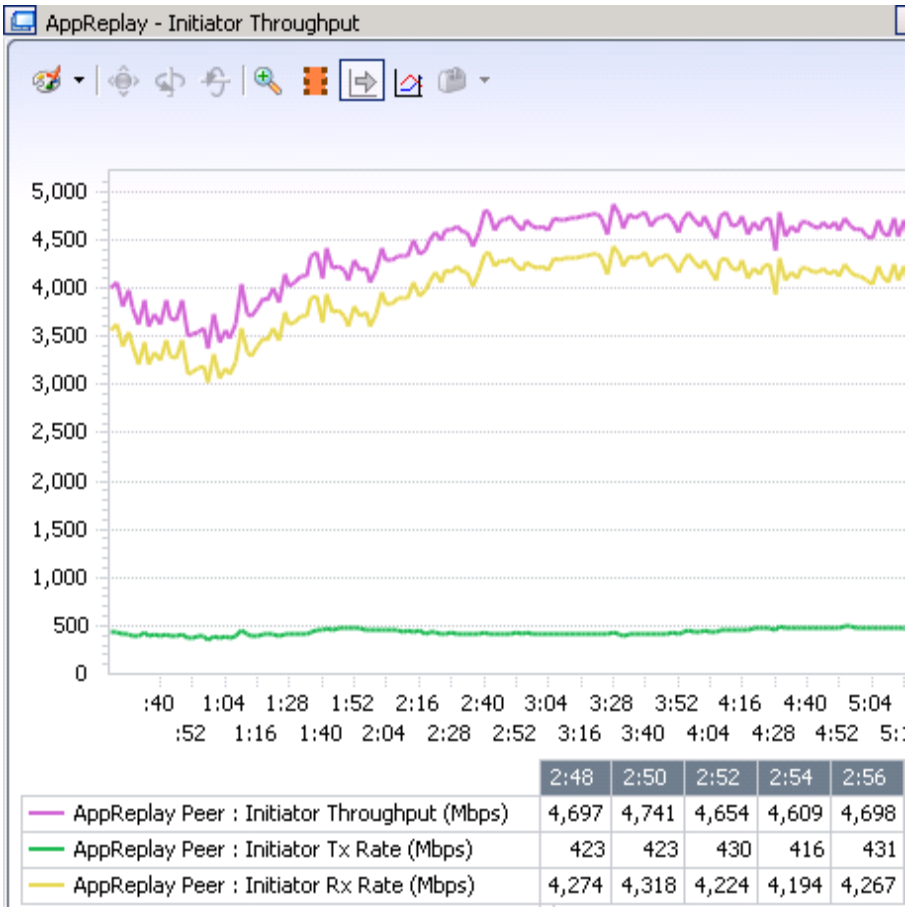


Figure 120. AppRepaly Throughput (Goodput) Statistics View

The throughput can be viewed by reviewing the **L2-3 Stats**, this indicates the device's forwarding performance that was sustained.

	2:48	2:50	2:52
10.200.134.136/Card04/Port13 : Bits Sent Rate (Kb/s) [L2-3 Stats for Client Ports]	691,588	689,680	697,109
10.200.134.136/Card04/Port13 : Bits Received Rate (Kb/s) [L2-3 Stats for Client Ports]	4,628,863	4,647,679	4,594,074

Figure 121. Layer 2-3 Statistics

AppReplay per-flow statistics provide an insight with per-application granularity. To view the per-flow statistics, the **Enable Advanced Stats** option should be enabled in the **AppReplay Advanced Options** dialogue before the test starts.

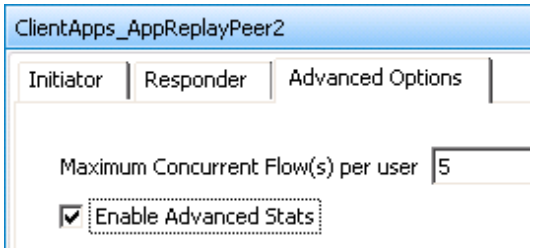
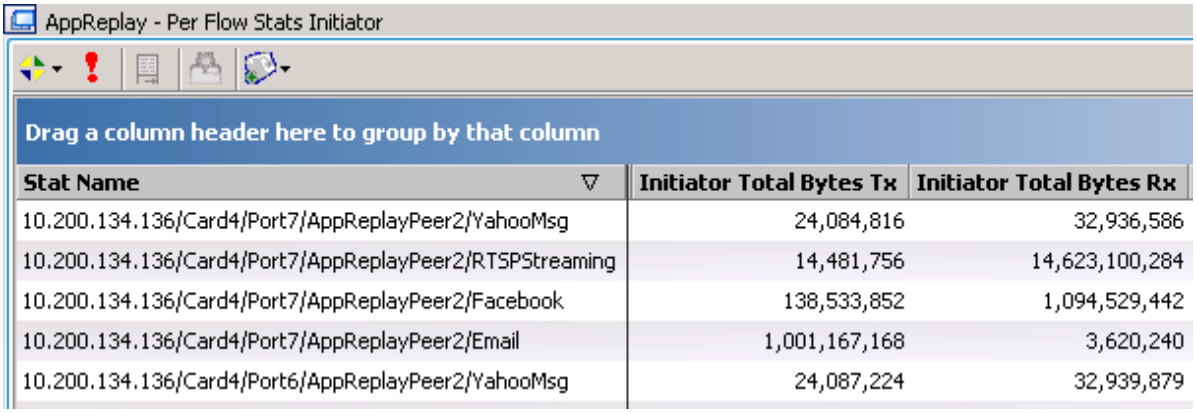


Figure 122. AppRepaly Peer Activity Advanced Options

The AppReplay per flow statistics view can be used to analyze the number of flows initiated and succeeded, number of bytes transmitted and received, errors, and latency measurements on per flow basis.



Stat Name	Initiator Total Bytes Tx	Initiator Total Bytes Rx
10.200.134.136/Card4/Port7/AppReplayPeer2/YahooMsg	24,084,816	32,936,586
10.200.134.136/Card4/Port7/AppReplayPeer2/RTSPStreaming	14,481,756	14,623,100,284
10.200.134.136/Card4/Port7/AppReplayPeer2/Facebook	138,533,852	1,094,529,442
10.200.134.136/Card4/Port7/AppReplayPeer2/Email	1,001,167,168	3,620,240
10.200.134.136/Card4/Port6/AppReplayPeer2/YahooMsg	24,087,224	32,939,879

Figure 123. AppReplay per flow statistics view

Troubleshooting and Diagnostics

Table 66. Troubleshooting and diagnostics

Issue	Diagnosis, Suggestions
What are the optimal statistics that can be used to certify that the optimal capacity has been reached?	The relatively quick way to know if the device is at maximum capacity is to incrementally add new test ports, and see if the overall concurrent connections, connections per second and throughput metrics increase. If TCP failures occur and new connections are failing, then it's an indication that the limit has been reached. Other indications include a high latency, including connect Time, TTFB and TTLB. Concurrent flows and number of simultaneous subscribers that can be supported is a function of system memory. Refer to the memory usage on the device, which should indicate that it's near its high water mark.

Test Case: Validate DPI Application Signature Database Accuracy with AppLibrary

Overview

The explosion of web oriented applications, connected smart devices and associated multimedia applications possess a tremendous burned on application delivery networks in terms of traffic diversity and complexity. It is both the sheer number of different applications and the pace at which new applications are released (or existing applications updated with potentially different behavioral characteristics) that contribute to this reality.

To ensure successful operation in these tough conditions, there is little or no room for error left for DPI devices. At the very core of every successful DPI device lays a powerful and mostly up to date application signature database. Such databases contain traffic patterns that help in identifying as many applications as possible and even specific actions contained within a particular application message exchange. A good example of such actions is the different activities that can be performed using Facebook: login/logout, upload photo, update status or play different games such as Farmville. Application control mechanisms embedded in DPI devices leverages the application signature database to perform granular decisions on whether to block even specific actions within certain application flows.

Objective

Determine the detection accuracy of a DPI`s Application Signature Database under real life traffic using the most common applications and protocols.

The detection accuracy is measured by configuring the DUT to block certain application flows (like peer to peer sharing), but also specific actions within an application flow (block Farmville but allow other type of Facebook actions).

The validity of such test is certified only by emulating real life traffic conditions. This is accomplished by using a mix of real application flows and by controlling the objective distribution load over all application flows.

Setup

This setup requires one server and one client/subscriber test port to emulate the entire application mix. However more ports can be used to scale the performance of the test tool.

Figure 124. DPI Accuracy Test Topology

Test Variables

Test Tool Variables

The following test configuration parameters are used to create a traffic profile that represents various business and consumer subscribers using the most common applications under a sustained peak-hour load.

Table 67. Test Tool Variables

Parameter	Description
Client network	4K IP addresses
TCP parameters	TCP Rx and Tx buffers at 4096 bytes
Application Mix parameters	• 16K Simulated Users mimic different user profiles running a broad range of applications like business, social media, Peer to Peer sharing, email, video streaming and others. • 14 different real application flows representing different activities or actions will be emulated: various Facebook flows (with different actions like login, news feed page, update status, Farmville game, and so on), Outlook, Citrix, Netflix, iTunes, YouTube, BitTorrent, Gnutella, and so on. • To better emulate real world traffic scenarios, even application flows generated by mobile users can be used. • Each application flow will have associated a deterministic percentage of the global objective value. Other application mixes that are representative for your subscriber community can be created as well.

DUT Test Variables

There are several DUT scenarios. The following list outlines some of the DUT's capabilities that can be switched on.

- Enable stateful inspection engines
- Increase the number of traffic management policies
- Enable blocking and traffic shaping mechanisms
    - Block major application: *BitTorrent*
    - Block specific actions within an application: *Farmville* to be blocked while other types of *Facebook* activities will be allowed
- Enable subscriber monitoring and management features

Step-by-Step Instructions

Configure a baseline test, which is an Ixia port-to-port test, in order to verify the test tool's performance. Once you have obtained the baseline performance, setup the DUT and the test tool as per the **Setup** section above. Refer to the Test Tool and DUT Variables sections above for recommended configurations. When configuring the DUT test, a layer 2 switch with a high-performance backplane is highly recommended.

Fill in the table below with the baseline performance to use as a reference of the test tool's performance, based on the quantity and type of hardware available.

Table 68. Reference Test Tool baseline performance

Performance indicator	Value per port pair	Load module type
Application Mix Throughput		
Application Mix Connections/sec		
Application Mix Transactions/sec		
Application Mix Flows/sec		
Application Mix Concurrent Connection		

1. Start IxLoad. In the main window, all of the test configurations are made using the Scenario Editor window.

To get familiar with the IxLoad GUI, see the *Getting Started Guide* section.

2. Click the **Add AppMix** button to add **AppMix** NetTraffic objects. This action creates both originate and terminate NetTraffics.

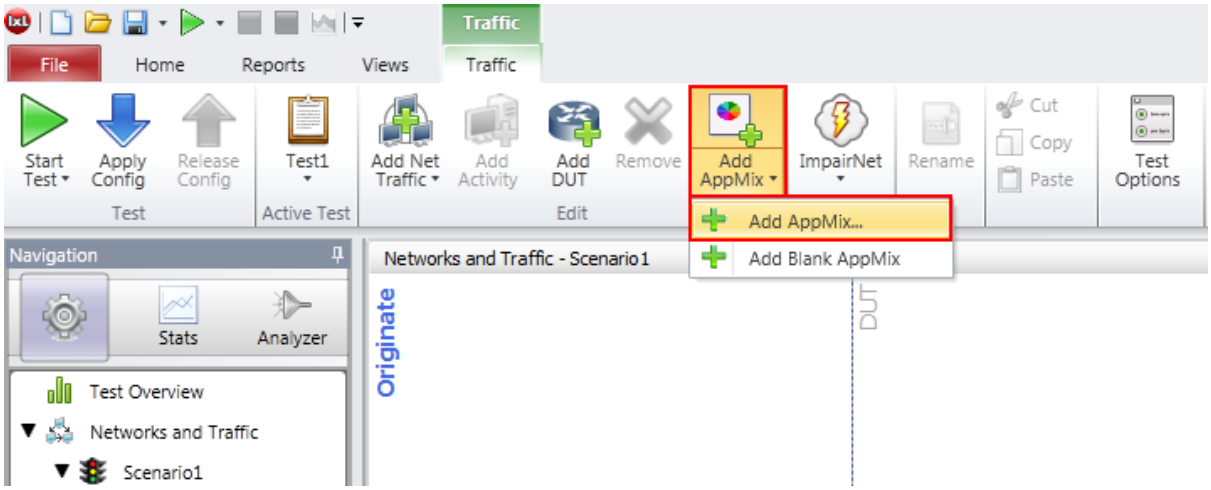


Figure 125. Create an AppMix

3. Configure the **Originate** network with the total number of IP addresses required.

The DPI devices do not directly interface with the subscriber network, so you must simulate an **Emulated Router** to protect the DPI device from any ARP storm.

a. To add the **Emulated Router,** right-click the **MAC/VLAN** Stack and follow the context sensitive menu as shown in the below Figure.

The emulated router gateway must connect to the DUT interface.

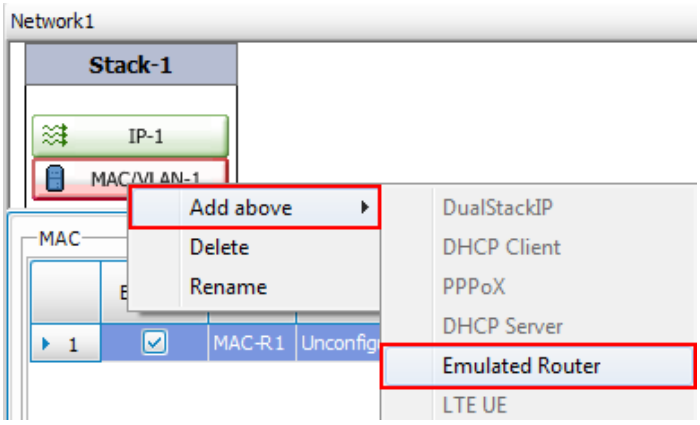


Figure 126. Creating an IP Stack with Emulated Router

b. On the IP stack, configure the desired number of static IP addresses. For this test, configure 4K IP addresses for the **Originate NetTraffic**.

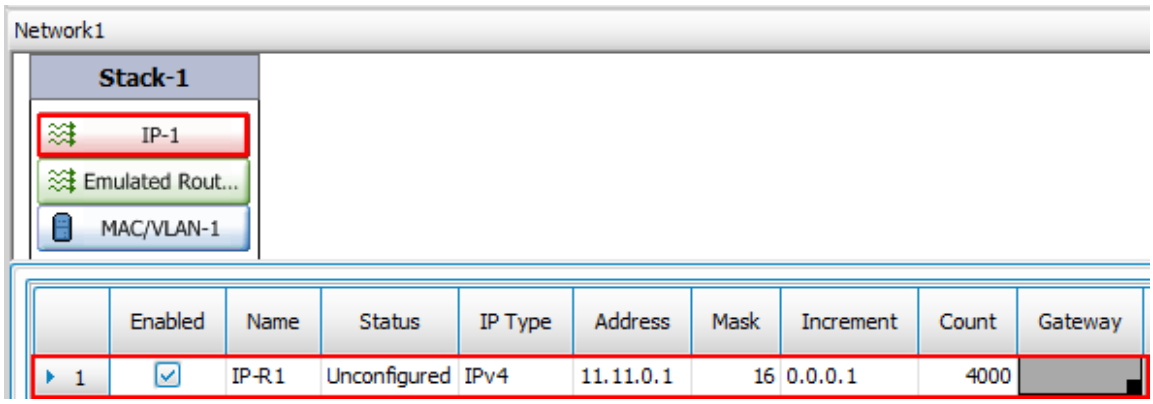


Figure 127. Client Network IP Range

According to the test hardware that is being used, the test can be further scaled. For example, if you are using an Xcellon Ultra-NP load module, you can create 12 network ranges with 10K IP addresses for each port, which emulates 120K IP addresses for the test. Also ensure that you have 12 emulated routers created, one for each IP range.

4. Configure the **Terminate** network with the total number of IP addresses required. For the purpose of this test, configure 100 IP addresses for the **Terminate NetTraffic**.

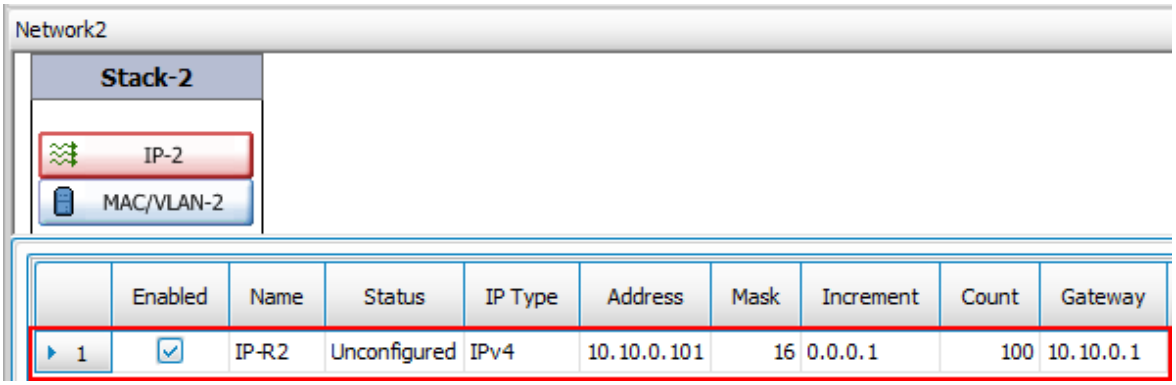


Figure 128. Server Network IP Range

For a step-by-step workflow, see Appendix A: Configuring IP and Network settings.

5. TCP parameters are important when optimizing the test tool. Refer to the Test Tool Variables section in Table 1 to set the correct TCP parameters.

6. There are several other parameters that can be changed. Leave them at their defaults values unless you need to modify them for testing requirements.

   For a step-by-step workflow, see Appendix B.

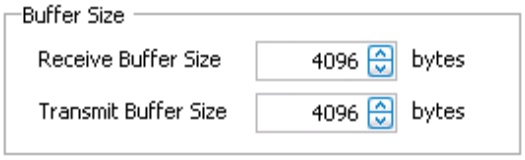


Figure 129. TCP Buffer Settings

7. Configure the **AppMix** activity with the application flows to be emulated.

The **AppMix** activity is the container of all application flows. **AppMix** levarages Ixia`s powerful **AppLibrary** framework to define scalable application mixes with deterministic distributions for multi-application testing.

a. To start adding application flows to the application mix, first select the originate **AppMix** activity, and click **Add AppFlow.**

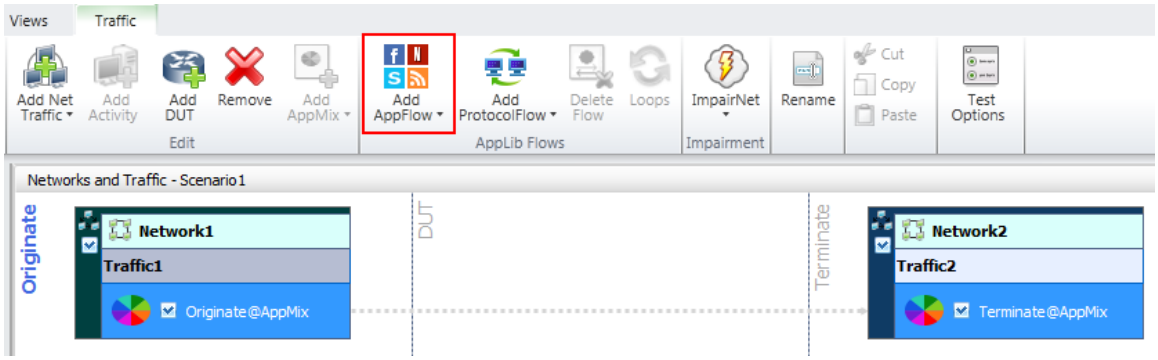


Figure 130. Accessing the Application Flow Library

A new dialog window displays all application flow category folders. All available application flows are grouped within category folders according to their practical usage purpose. You can navigate to the relevant category folder for the desired application flow.

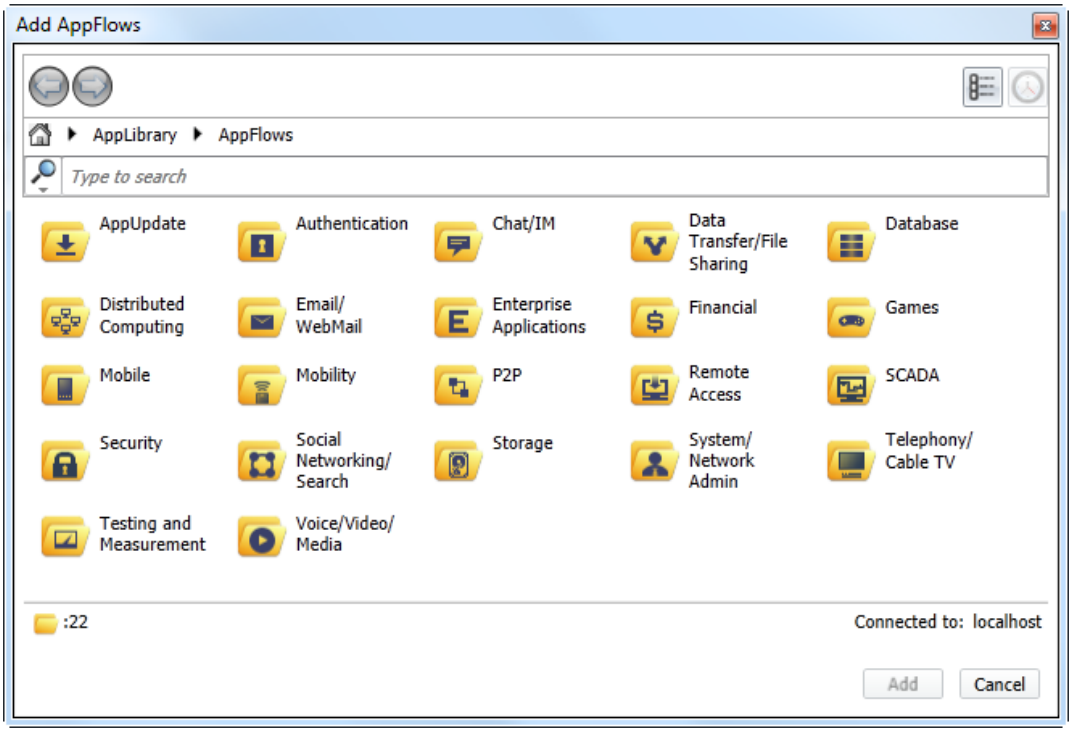


Figure 131. AppLibrary Category Folders

b. The application flows to be used in this test are grouped in the following categories:

- Social Networking: *Facebook*
- Games: *Farmville*
- Voice/Video/Media: *Netflix Service Provider, YouTube Service Provider, BBC iPlayer, iTunes Mobile App Store*
- Peer to Peer: *BitTorrent Service Provider, Gnutella*
- Email/WebMail: *Yahoo Mail, Exchange-Outlook Send Mail*
- Mobile generated: *Google Play Search, View, and Download*
- Test and Measurement: *HTTP Service Provider*
- Remote Access: *Citrix*
- Database: *Oracle Database*

Enabled	AppFlow	Flow Size (bytes)	Value	Percentage
	Type to filter by name		16000	100 %
☑	Facebook1	394024	1440	9 %
☑	Farmville1	2624239	960	6 %
☑	BBC iPlayer1	417645	640	4 %
☑	Netflix Service Provider1	19001779	2880	18 %
☑	iTunes Mobile App Store1	5967977	1440	9 %
☑	BitTorrent Service Provider1	105110	1440	9 %
☑	Gnutella1	432980	1280	8 %
☑	Yahoo Mail1	93552	320	2 %
☑	Exchange-Outlook Send Mail	12914	160	1 %
☑	HTTP Service Provider1	100520	1920	12 %
☑	Google Play Search, View, and Download1	292970	960	6 %
☑	YouTube Service Provider1	2059712	2240	14 %
☑	Citrix1	573995	160	1 %
☑	Oracle Database1	15091	160	1 %

Figure 132. Application Flow List to be Emulated

You can add other application flows as well in the mix according to test requirements.

8. Having setup the client and server networks and the traffic profile, now the test objective and traffic distribution can be configured.

From the main configuration window located just below the scenario editor, configure the **Objective Type** as *Simulate Users* and **Objective Value** to *16K*. Next, you can configure individual flow percentages to ensure that the test load distribution (simulated users in this case) adheres to the real traffic profile as seen in production networks.

For this test, it emulates a traffic profile where 18% of the users use *Netflix*, 14% of the users use *YouTube*, 12% of the users use *HTTP Browsing* and so on. The entire percentage distribution will be configuring as per below figure:

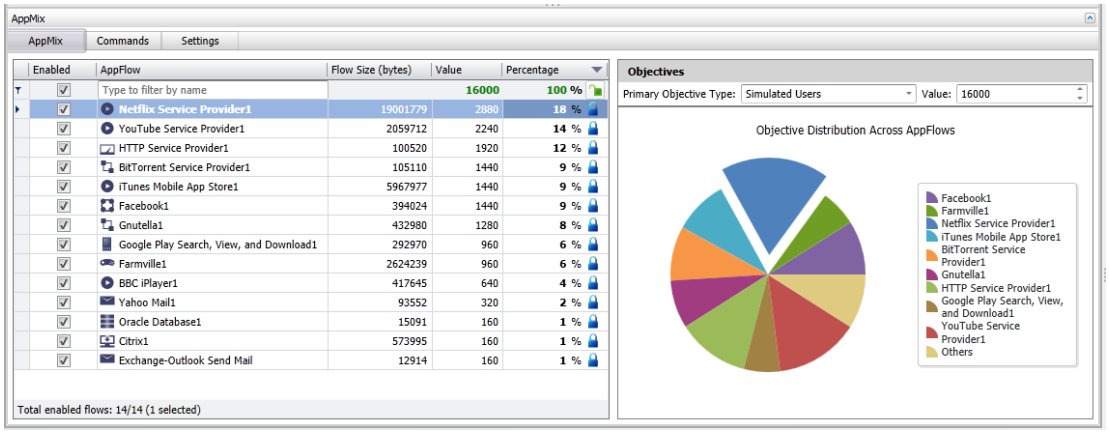


Figure 133. AppMix Objective and Distribution Load Percentages

9. Run the test for a few minutes to allow the performance to reach a steady-state. Steady state is referred as **Sustain** duration in the test. Continue to monitor the DUT for the application types it detected (and, where applicable, blocked) and any failure/error counters.

Interpretation of result statistics can sometimes be difficult, deducing what they mean under different circumstances. The Results Analysis section below provides a diagnostics-based approach to highlight some common scenarios, the statistics being reported, and how to interpret them.

10. Iterate through the test by varying the number of subscribers, the number of application flows within the mix and the diversity of the applications to determine the DUT`s capability of properly identifying the applications under different traffic conditions.

    a. You can start and stop the test tool in the middle of a test cycle, or wait for it to be gracefully stopped using the test controls shown here:

For a step-by-step workflow, see Appendix F.

Results Analysis

To analyze DUT`s capability of properly identifying the applications, it is necessary to examine multiple test iterations, comparing the results for all test parameter variations, including:

- Number of subscribers.
- Number of application flows within an AppMix container.
- Diversity of the application flows.

All application types are expected to be properly identified by the DUT`s DPI engine (leveraging the Application Signature Database) and proper actions must be performed according to the configured traffic management policies (refer to the Test Variable section for DUT configuration details).

First the DUT`s reporting needs to be checked to ensure that all application types used in the test were correctly identified. Additionally, the DUT must accurately apply the configured traffic management policies and report the amount and type of traffic subject to such management policies.

In parallel, the below table lists the key performance statistics that must be monitored on the Test Tool. These statistics help to determine, if the device has reached its saturation point and to identify any issues. Interpreting the results in the correct manner also ensures that transient network, device or test tool behavior does not create a false positive condition:

Table 69. Key AppMix performance statistics to monitor

Metric	Key Performance Indicators	Statistics View
Performance metrics - global	Throughput, connection/transaction/flow rates, total connections and flows, number of simulated users	AppMix
Application level performance Application level failure monitoring	Throughput, connection/transaction/flow rates, number of simulated users, connect time and flow lifetime flows initiated/successful/failed, transactions initiated/successful/failed connections initiated/successful/failed	AppMix Per Flow Initiator, AppMix Per Flow Responder
TCP connection information TCP failure monitoring	SYNs sent, SYN/SYN-ACKs received, RESET sent, RESET received, retries, timeouts.	AppMix Per Flow Initiator TCP Details, AppMix Per Listening Port TCP Details

Real-Time Statistics

The statistic view shown in the figure below (**AppMix Per Flow Initiator**) provides valuable information, extremely useful for monitoring individual application flow performance. It offers insights into different KPIs (key performance indicators) with a per-application flow granularity.

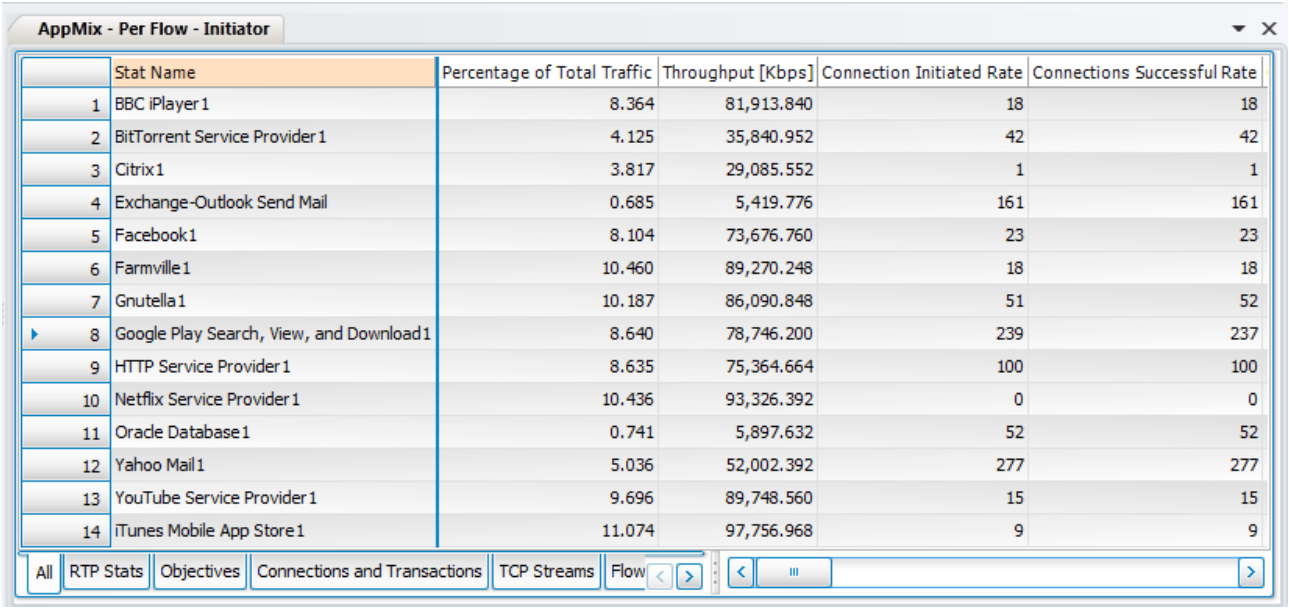
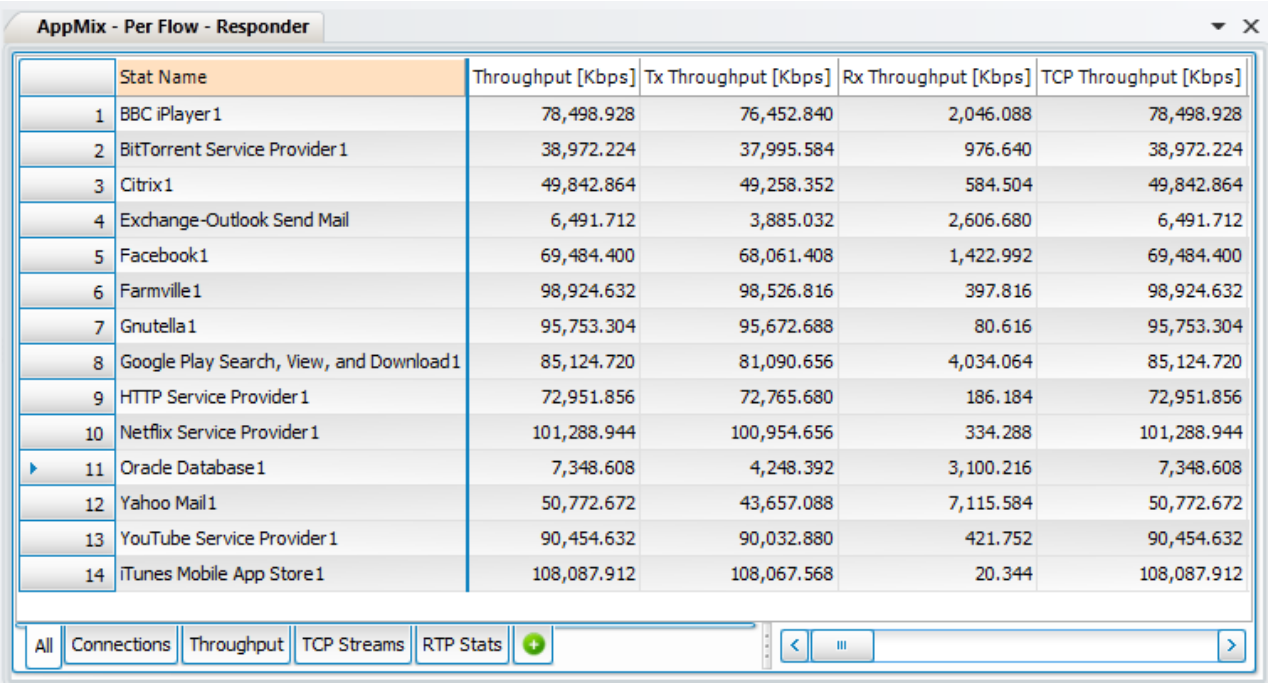


Figure 134. AppMix – Per Flow – Initiator View

Another static view that also provides an insight with per-application granularity is the **AppMix Per Flow Responder:**



	Stat Name	Throughput [Kbps]	Tx Throughput [Kbps]	Rx Throughput [Kbps]	TCP Throughput [Kbps]
1	BBC iPlayer1	78,498.928	76,452.840	2,046.088	78,498.928
2	BitTorrent Service Provider1	38,972.224	37,995.584	976.640	38,972.224
3	Citrix1	49,842.864	49,258.352	584.504	49,842.864
4	Exchange-Outlook Send Mail	6,491.712	3,885.032	2,606.680	6,491.712
5	Facebook1	69,484.400	68,061.408	1,422.992	69,484.400
6	Farmville1	98,924.632	98,526.816	397.816	98,924.632
7	Gnutella1	95,753.304	95,672.688	80.616	95,753.304
8	Google Play Search, View, and Download1	85,124.720	81,090.656	4,034.064	85,124.720
9	HTTP Service Provider1	72,951.856	72,765.680	186.184	72,951.856
10	Netflix Service Provider1	101,288.944	100,954.656	334.288	101,288.944
11	Oracle Database1	7,348.608	4,248.392	3,100.216	7,348.608
12	Yahoo Mail1	50,772.672	43,657.088	7,115.584	50,772.672
13	YouTube Service Provider1	90,454.632	90,032.880	421.752	90,454.632
14	iTunes Mobile App Store1	108,087.912	108,067.568	20.344	108,087.912

Figure 135. AppMix – Per Flow – Responder View

Using both above mentioned statistic views (**AppMix Per Flow Initiator** and **AppMix Per Flow Responder**) you can validate that the traffic management polices configured on the DUT are applied correctly:

- Blocked applications: flows initiated/successful/failed, transactions initiated/successful/failed, connections initiated/successful/failed.
- Traffic shaping (if enabled on the DUT): throughput (Rx/Tx, UDP/TCP)

Additionally, you can investigate a number of per-application key performance indicators in order to:

- Assess DUT performance in the given conditions (correctly identifying and blocking applications) in terms of: throughput (TCP/UDP), connection rates, transaction rates, and flow rates.
- Estimate the QoE (Quality of Experience) impact of the traffic management policies on a per-application basis: connect time and flow lifetime.

You can also view most of the above mentioned statistics in an aggregated manner, at the application mix level in the **AppMix** statistic view:

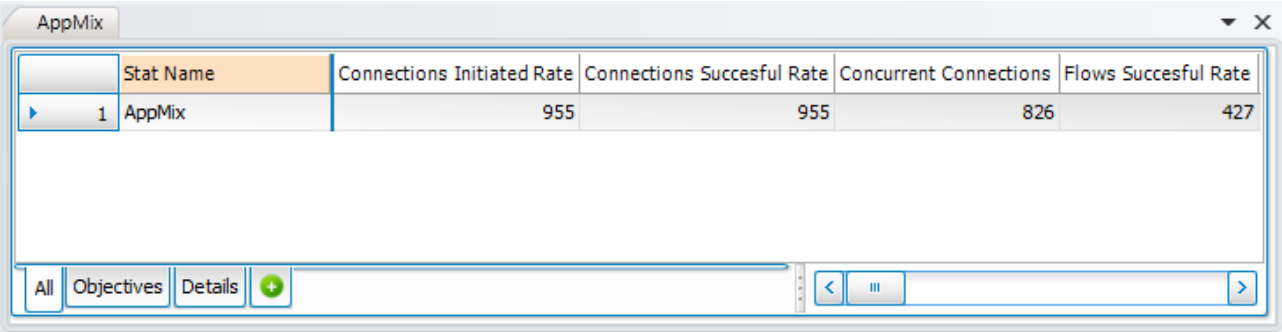


Figure 136. AppMix Aggregated Statistics View

Below figure is the real-time view of the Application Mix Throughput, which is equivalent to the good put of the DUT:

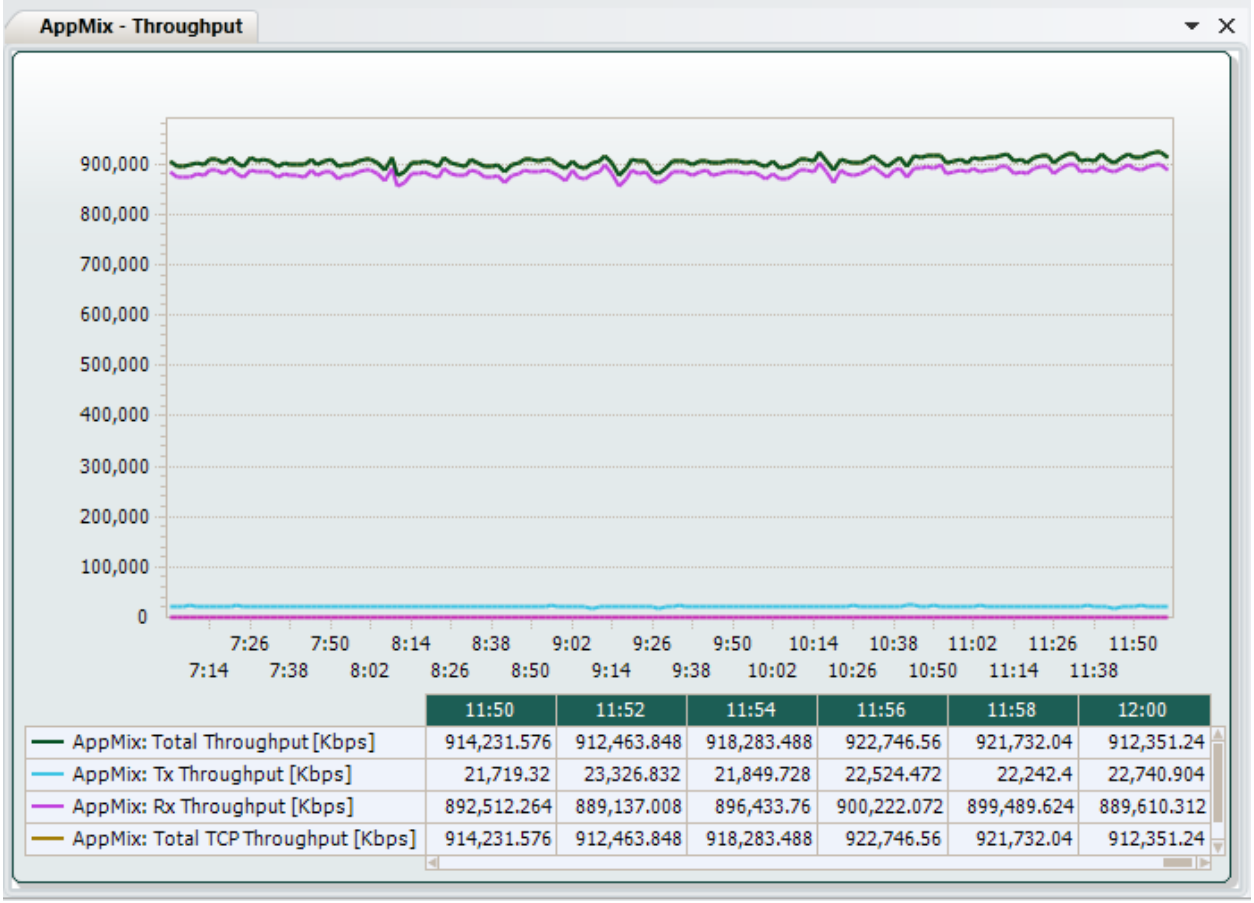


Figure 137. AppMix Application Throughput (Goodput) View

You can view the L2 throughput by reviewing the L2-3 Stats. This indicates the device's forwarding performance that was sustained.

	9:30	9:32	9:34	9:36	9:38	9:40	9:42
Client Bits Sent Rate (Kbps)	49,332.304	47,677.144	47,645.368	48,762.688	48,359.752	50,354.776	47,142.928
Client Bits Received Rate (Kbps)	928,353.544	935,116.704	929,342.84	944,509.352	909,390.056	966,335.432	930,099.856
Server Bits Sent Rate (Kbps)	928,059.672	935,086.024	929,560.616	943,956.936	910,275.008	965,473.096	930,094.408

Figure 138. Layer 2-3 Throughput View

Troubleshooting and Diagnostics

Table 70. Troubleshooting and diagnostics

Issue	Diagnosis, Suggestions
What are the optimal statistics that can be used to verify that the maximum capacity has been reached?	The relatively quick way to know if the device is at maximum capacity is to incrementally add new test ports, and see if the overall concurrent connections, connections per second, and throughput metrics increase. If TCP failures occur and new connections are failing, then it is an indication that the limit is reached. Other indications include a high latency, including *Connect Time* and *Flow Lifetime*. Concurrent flows and number of simultaneous subscribers that can be supported is a function of system memory. Refer to the memory usage on the device, which should indicate that it's near its high water mark.

Test Case: Measure Data Reduction Performance of WAN Optimization Devices

Overview

Today's enterprise networks are reaching saturation point. The rising demands of mobility, big data, and the cloud have increased bandwidth demands, application needs and security risks, threatening to push networks to the brink. But strengthening and future proofing the WAN networks is moving beyond the traditional mindset of simply adding bandwidth. A truly Intelligent WAN takes a holistic approach. It uses bandwidth efficiently and providing tools for agile management and monitoring of applications to ensure a consistent and positive experience for end users.

WAN Optimization Controllers or WAN Accelerators are dedicated devices, deployed on either end of a WAN link to improve application response degradation caused by bandwidth constraints and latency or protocol limitations such as chattiness of applications. WAN optimization technologies generally work to prevent network latency by using protocol optimization techniques, reduce data traveling over the WAN through compression or caching, and prioritize traffic streams according to business needs.

The primary function of WAN Optimization is to improve the response times of business-critical applications over WAN links, but they can also help to maximize return on investment (ROI) in WAN bandwidth, and sometimes avoid the need for costly upgrades. With more and more applications now moving to the cloud, WAN Optimization devices are quickly becoming indispensable within many enterprises.

How do WAN Optimization devices work?

WAN Optimization or Accelerators are in-line devices used as a pair between two sites to accelerate delivery of content, usually large files. It works by:

1. Using TCP optimization and application optimization between the devices at the two sites.
2. Data caching, which reduces the data that must be sent across a WAN by suppressing information that has been 'seen' by the device and cached such that only the incremental information is sent to the remote peer. Optimizations are achieved at different rates for three types of data patterns:
    - **Hot** data is 100 percent cached and can be accessed right away, from memory.
    - **Warm** data is partially cached and must be accessed from storage/disk.
    - **Cold** data is new data, not seen or cached at all by the WAN optimizer.

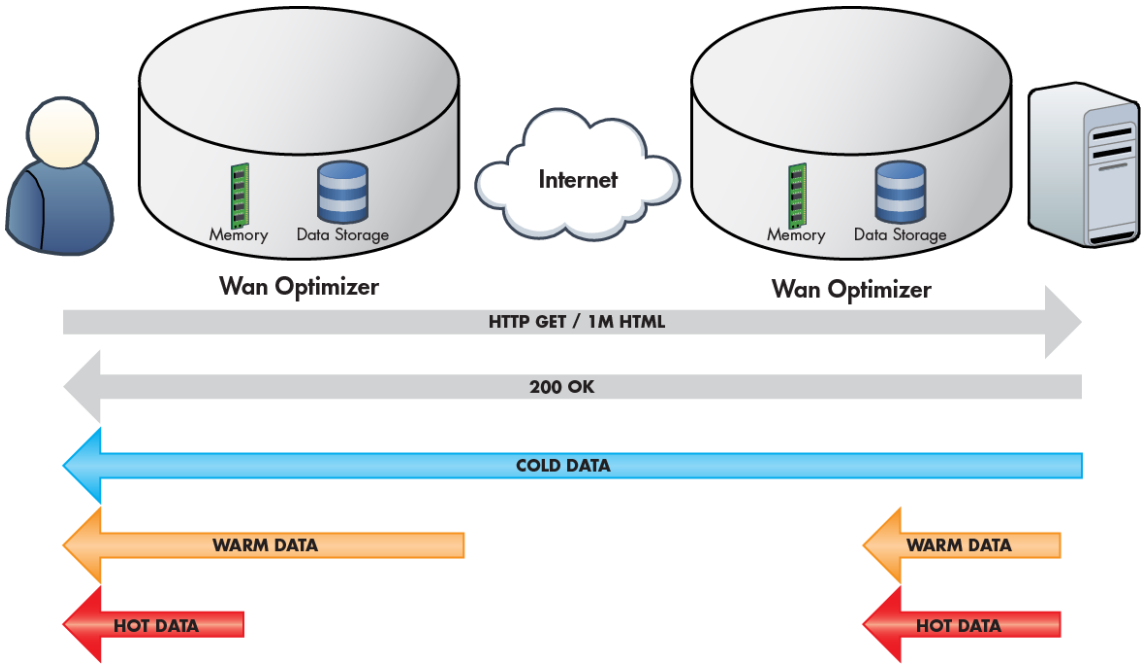


Figure 139. Figure 5 Typical WAN Optimization Deployment

3. Compression, which shrinks the size of data to limit bandwidth use; compressing data by using onboard CPU/memory. Compression involves the usage of WAN links and improves the overall speed of delivering content from site to site.

Objective

Determine the scalability and capacity limits of a WAN Optimization or Acceleration devices with a realistic blend of traffic of varying randomness and compressibility.

The Random Data Generation Engine or RDGE introduced in IxLoad 6.10 will enable users to configure tests to generate appropriate random content in accordance with the memory and data storage capacity of the WAN Optimization device.

For this test, we will add a pair of WAN Optimizer devices between the Ixia client port and server port. The WAN Optimizers will be entirely transparent to the Ixia client and servers and hence, network configuration will be same as that of a back to back test.

In this test methodology, we will create a Random Data Profile to generate 100 percent Hot data and run a test with the random data profile. We will set compressibility to minimum and disable compression on the DUT to demonstrate the Data Reduction because of Data Redundancy.

To characterize the performance of the WAN Optimization devices, repeat this methodology for different page sizes and random data profiles that are representative of various applications.

Setup

This test requires a server and a client/subscriber port to emulate HTTP users. In this test, we will configure an HTTP Client to request pages of size 4k from the server. The content of the pages will be initially 100 percent Cold. As the cache gets filled up, IxLoad will start repeating the data sequence and data will be considered 100 percent Hot, because it is a repeating pattern as seen by the DUT.

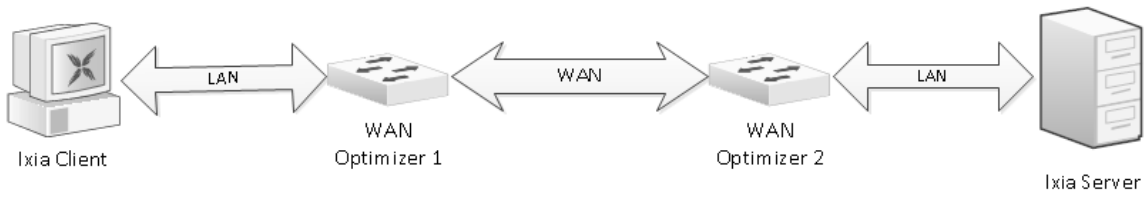


Figure 140. Ixia Test Topology

Test Variables

Test Tool Variables

The following test configuration parameters provide the flexibility to create the traffic profile that a device would experience in a production network.

Table 71. Test Tool Variables

Parameter	Description
HTTP clients	100 IP addresses or more, use sequential or use all IP addresses.
HTTP client parameters	HTTP 1.1 20 TCP connections per user Maximum transactions per TCP connection
HTTP client pages to request	1 GET command – payload of varying sizes
HTTP page content	Configure a test with various patters of Hot, Warm, Cold data, or a mix of them.
TCP Parameters	Client TCP - RX 32768 bytes, TX 4096 bytes Server TCP – RX 4096 bytes, TX 32768 bytes
MSS	1460 bytes
HTTP servers	1 per Ixia test port, or more
HTTP server parameters	Random response delay – 0 – 20 ms Response timeout – 300 ms

DUT Test Variables

The primary goal of this test is to find the maximum throughput achieved on the WAN Optimizer for 100 percent Hot data profile. The device is added in transparent mode so we have configured clients and servers in the same network.

Set the DUT mode such that it only achieves WAN Optimization by Data Deduplication or skipping transfer of duplicate data over the WAN link. Some vendor calls it SDR-only.

Table 72. WAN Optimization features

WAN Optimizations	Test Description
Mode	Enable Data Reduction by data Deduplication. Disable data Reduction by Compression.
Protocol	Disable any protocol specific data reduction.

Step-by-Step Instructions

Configure a baseline test, which is an Ixia port-to-port test to verify the test tool's performance. After you have obtained the baseline performance, set up the DUT and the test tool in accordance with the Setup section earlier. Refer to the preceding Test and DUT Variables sections for recommended configurations.

Fill the following table with the baseline performance to use as a reference of the test tool's performance, based on the quantity and type of hardware available.

Table 73. Reference baseline performance form

Performance indicator	Value per port pair	Load module type
100% Hot Data Throughput		
100% Warm Data Throughput		
100% Cold Data Throughput with min compression		
100% Cold Data Throughput with max compression		

1. Open IxLoad. In the main window, all of the test configurations will be made by using the **Scenario Editor** window.

   To get familiar with the IxLoad GUI, see the Getting Started Guide section.

2. Follow the steps of **Maximum Throughput** test case to create a throughput test between two Ixia Ports through the DUT (Page 33 of this Application Delivery Backbook).

   In this throughput test, IxLoad HTTP Server sends the same content in all /1024k.html responses. So after the first page is transferred, all subsequent pages contain identical data. As a result for all subsequent pages instead of sending the entire page, WAN Optimizers only exchange a reference to the page and the page is served from memory cache of local WAN Optimizer.

   The unique Random Data Generation Engine (REDG) functionality in IxLoad will allow users to configure the randomness of data content in each page in Server's response.

3. In IxLoad, click **Profiles** on the ribbon under the **Home** tab on the menu.

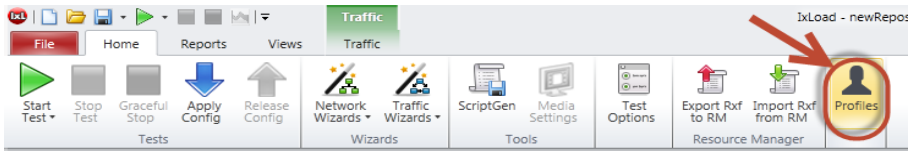


Figure 141. Profiles button

This will add a new **Profiles** section in the **Test Navigation** pane. Select **Random Data** and click the **+** sign to create a random profile.

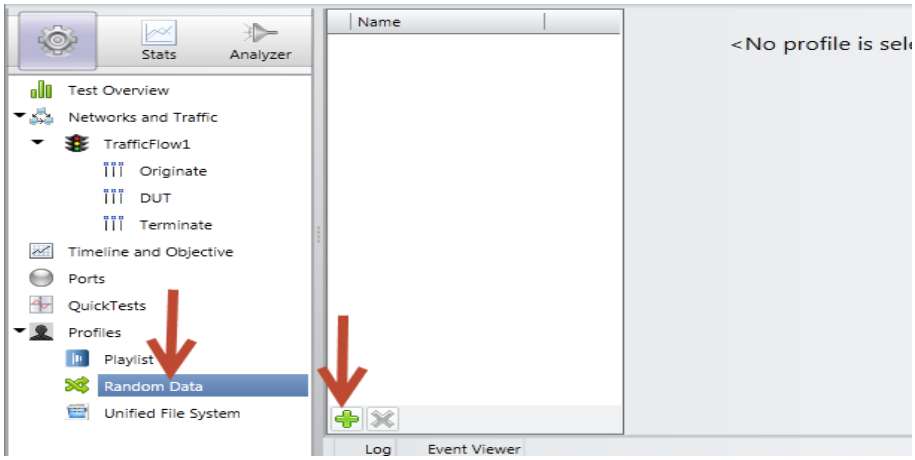


Figure 142. Creating a random profile

4. Create a Random Data Profile with the following configuration and rename the profile **Hot Data**.

   You need to configure the **Hot Data Pool** and **Warm Data Pool** depending on the Memory and Disk Storage size of the DUT and the Traffic Distribution.

   Example 1: Consider that the DUT has 1G Memory and 5G Data Storage and you want to generate 100 percent Hot Data. So you will need to configure the Hot Data Pool as 2G and the Warm Data Pool as 5G.

Table 74. Configurations for a new profile

Parameter	Description
Hot Data Pool	1 GB
Warm Data Pool	5 GB
Hot%	100%
Warm%	0%
Cold%	0%
Compressibility	None

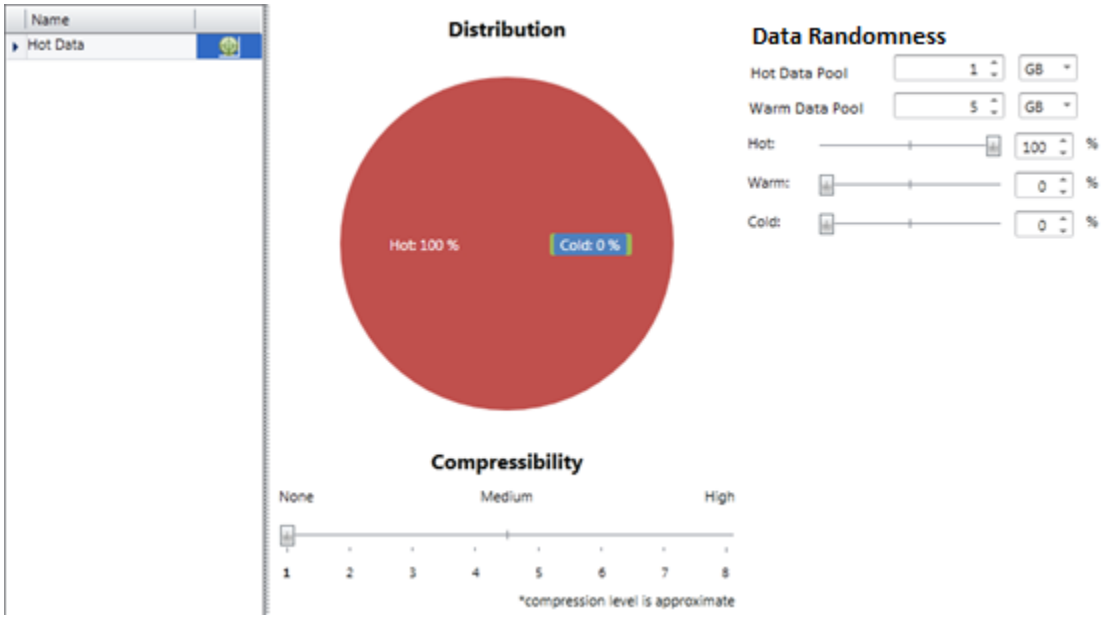


Figure 143. Random data profile configuration

Example 2: Consider that the DUT has 1G Memory and 5G Data Storage and you want to generate 30 percent Hot Data, 30 percent Cold Data and 40 percent Warm Data. This situation is slightly complex and the configuration will depend on how the DUT processes data. In usual cases, when the DUT receives data, it looks for a match in memory and data storage and if it does not find a match, it considers the data as cold

and pushes data to the WAN after updating/pushing the data into memory and cache. So at sustain time, both memory and data storage will have hot, warm, and cold data in the ratio 30:30:40.

Hence, Hot data will occupy 30 percent of 1G of memory; so we need to configure Hot Data Pool as 300 MB.

This 300 MB of hot data will also occupy 300 MB in Data Storage. The remaining 4.7 GB of Storage is occupied by Warm Data and Cold Data in the ratio 30:40. So we need to configure Warm Data Pool as 2014 MB (30/70 * 4700 MB).

Table 75. Profile Configurations

Parameter	Description
Hot Data Pool	300 MB
Warm Data Pool	2014 MB
Hot%	30%
Warm%	40%
Cold%	40%
Compressibility	None

5. On HTTP Server, create a new page in the **Web Pages** tab. Rename the page as **/RDGE.html** and select **Payload Type** as **Random Data.** This will make the File/Name Payload field available. Select the profile **Hot Data** from the list of previously defined profiles.

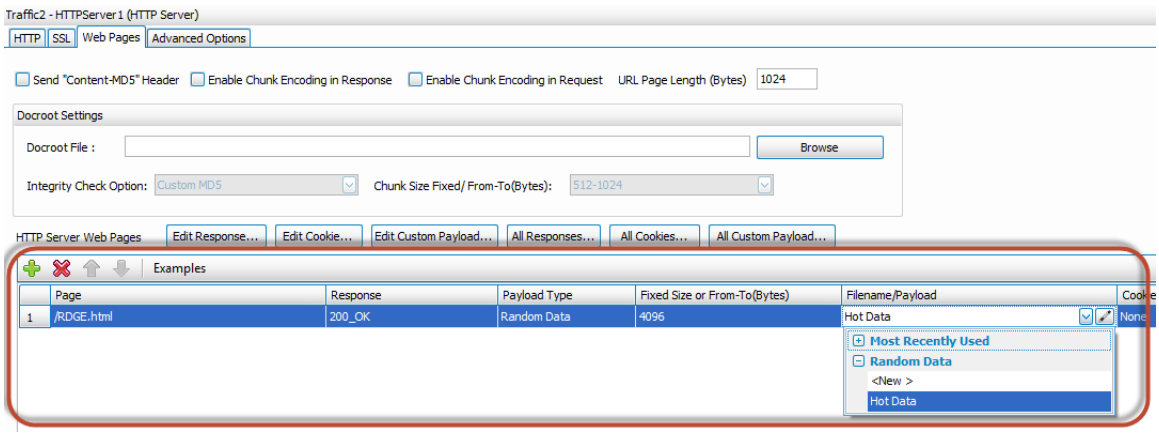


Figure 144. RDGE profile selections

6. On HTTP Client, select **RDGE.html** as **Page/Object** of the GET command. This completes the test configuration on IxLoad.

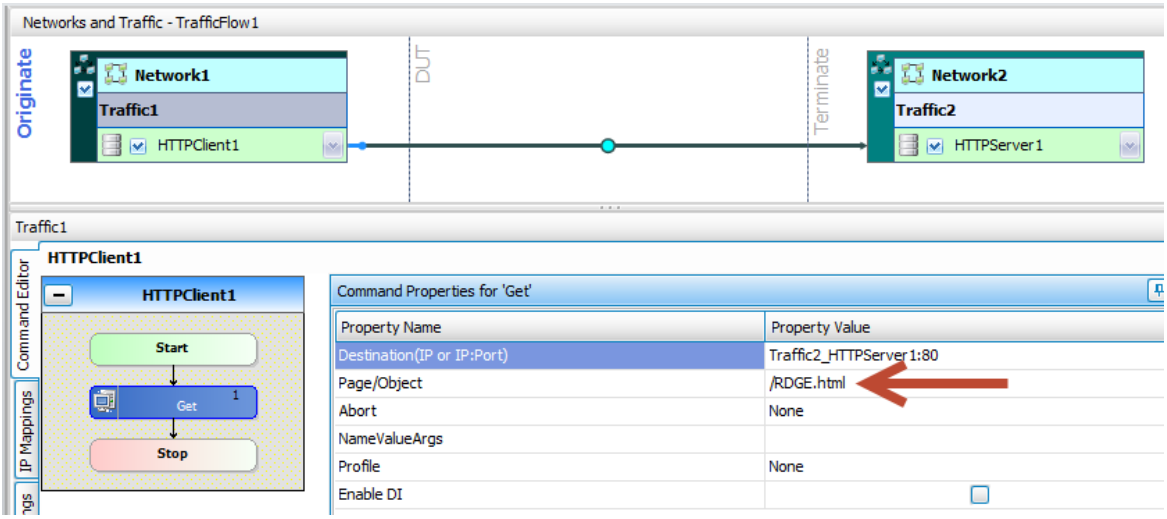


Figure 145. HTTP GET command configuration

In this mode, IxLoad will generate 512 MB of fresh data to fill up the Memory of the DUT. During this time, the WAN Optimizer will see it as 100 percent Cold data. Then, IxLoad will repeatedly generate the same 512 MB data sequence. This content will appear as 100 percent Hot to the WAN Optimizer.

Initially, the throughput will be small because the Data is new to the WAN Optimizer and has to be transmitted over the WAN. After the server starts sending repeated data, the WAN Optimizer near the Server will detect a Page cache hit and send the pointer to the page to the remote WAN Optimizer. The WAN optimizer will respond to Client with the page in its memory cache. This will drastically increase the throughput because minimal data is transmitted over the WAN.

Results Analysis

Enabling the Ixload Random Data Generation engine (RDGE) will not provide any new statistics on IxLoad. The two statistics that will be of importance in these tests are:

1. **HTTP Throughput**: The Layer 7 throughput statistics on **HTTP->HTTP Client – Throughput Objective** statview shows the HTTP throughput on LAN side. The throughput will increase as the DUT encounters Hot Data and reduces data transfer over the WAN.

2. **Latency**: The stats under **HTTP->HTTP Client – Latency** will provide the Connect Time, HTTP TTFB (Time To First Byte), and TTLB (Time To Last Byte) numbers. The Latency numbers will decrease as Data Reduction % increases over WAN.

We need to rely also on the DUT Statistics for analysis. The following statistics are the primary ones to analyze on the WAN Optimization device:

1. Data Reduction%: The Data Reduction report summarizes the percent reduction of data transmitted by an application. This report may contain Data Reduction percentage and volume of data sent over the LAN and WAN.

2. Data Store Status: The Data Store Status report summarizes the current status and state of the data store synchronization process.

3. Data Store Hit Rate: The Data Store Hit Rate report summarizes how many times the data-store disk and memory have seen a data segment. A hit is a data segment that has been seen before by the datastore in the system. When a hit occurs, the system sends the reference to the data segment rather than the actual data over the WAN.

Table 76. Key AppReplay performance statistics to monitor

Data Type	LAN Throughput	WAN Throughput	Data Reduction	Data Store Hit Rate
100% Hot				
30% Hot, 40% Warm and 40% Cold				

Real-Time Statistics

The following figure**Error! Reference source not found.** provides a real-time view of the easured performance for a test run. The **Throughput Objective** statistic shows that initially the throughput was 8 Mbps. During this time, IxLoad was generating the initial 512 MB of fresh data. After IxLoad starts repeating the data pattern, throughput increases to 60 Mbps because minimal data is getting transferred over the WAN and the Wan Optimizer near the client is responding to most of the data from its memory.

The Latency statistic shows that initially traffic was facing congestion on WAN because of 8Mbps WAN throughput, which was causing TTFB and TTLB to increase. As IxLoad started repeating the data pattern, latency dropped because the responses were getting served by the local memory cache.

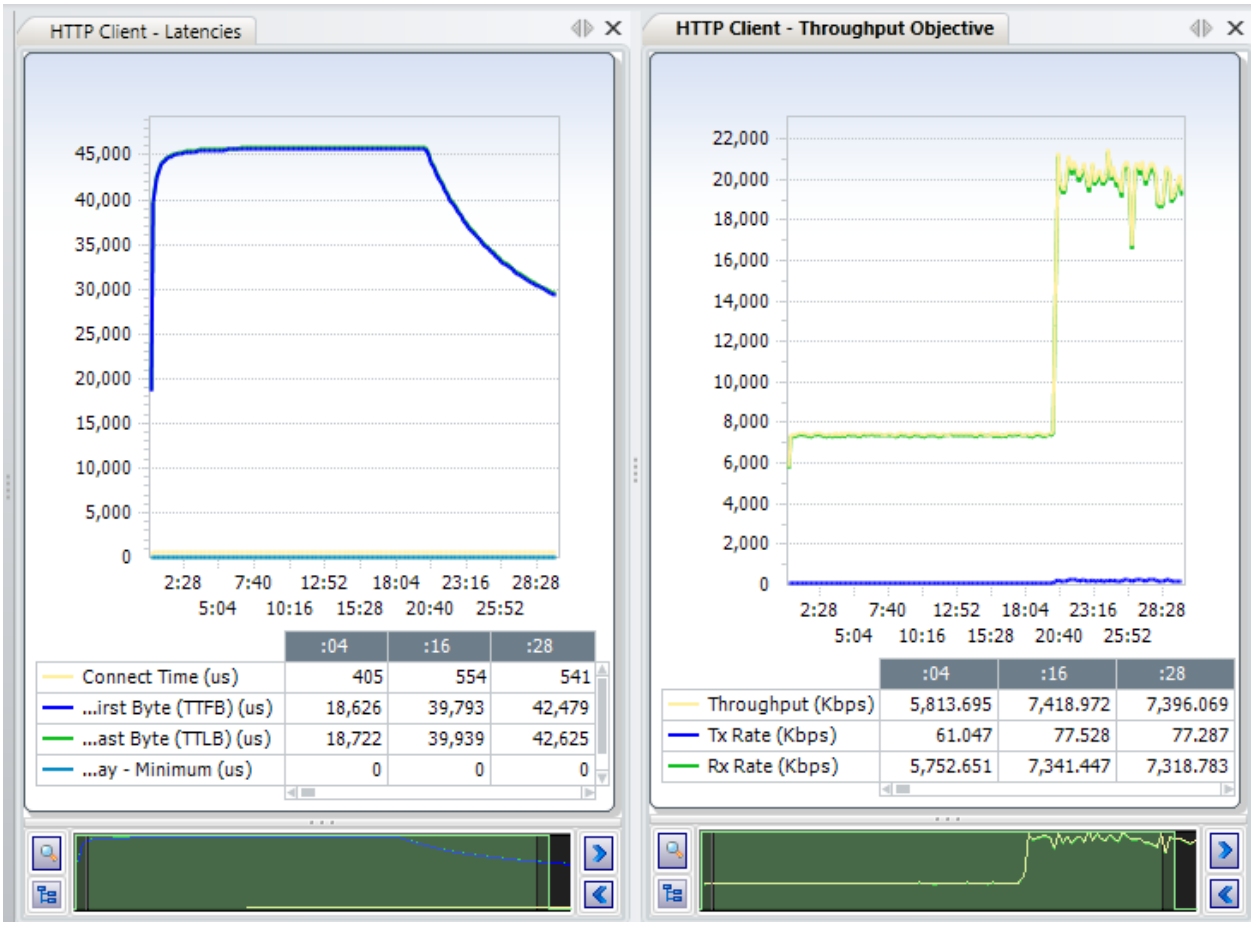


Figure 146. IxLoad HTTP Throughput and Latency Statistics

The following figure is the real-time view of the **Bandwidth Optimization** statistic on the DUT. This shows that initially, when data was Cold, nearly 100 percent of data was transferred over the WAN. Later, as IxLoad started sending Hot data, throughput increased and very negligible amount of data was transferred over the WAN.

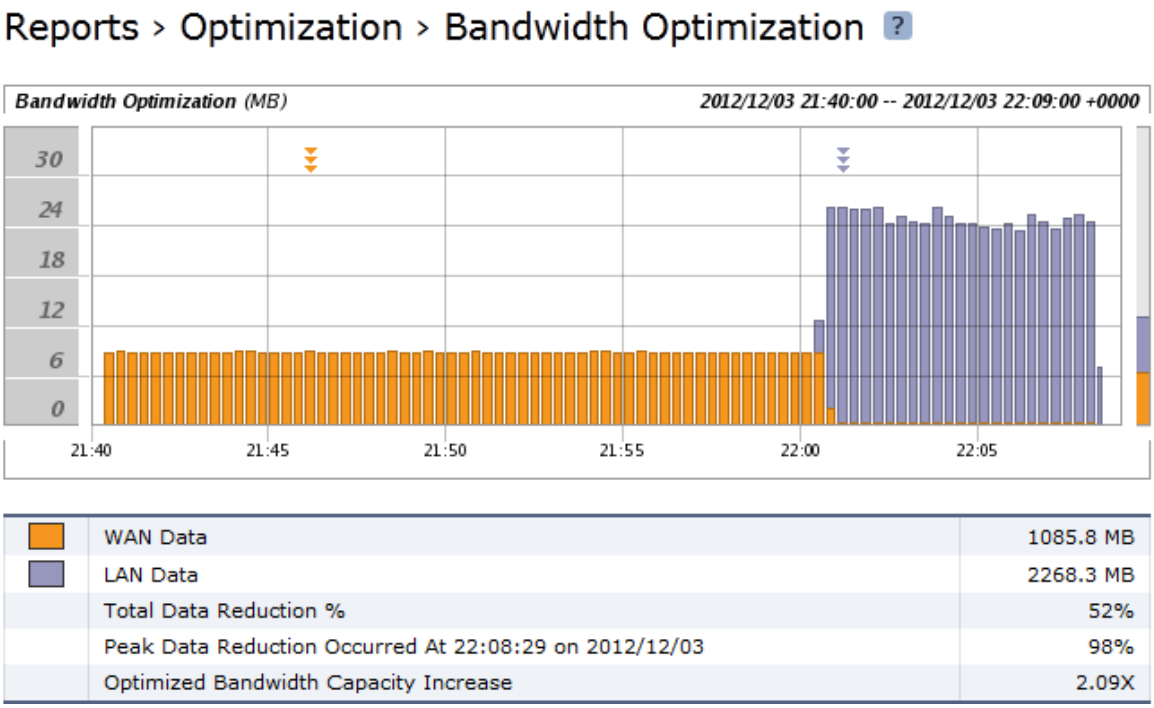


Figure 147. Bandwidth optimization statistics

The following figure is the real-time view of the **Data Store Hit Rate** statistic on the DUT. This shows that initially, when data was Cold, nearly 100 percent of the lookups was resulting in misses. Later, as IxLoad started sending Hot Data, almost all the lookups were resulting in Hits and the near end Wan Optimizer is serving the response from its local cache.

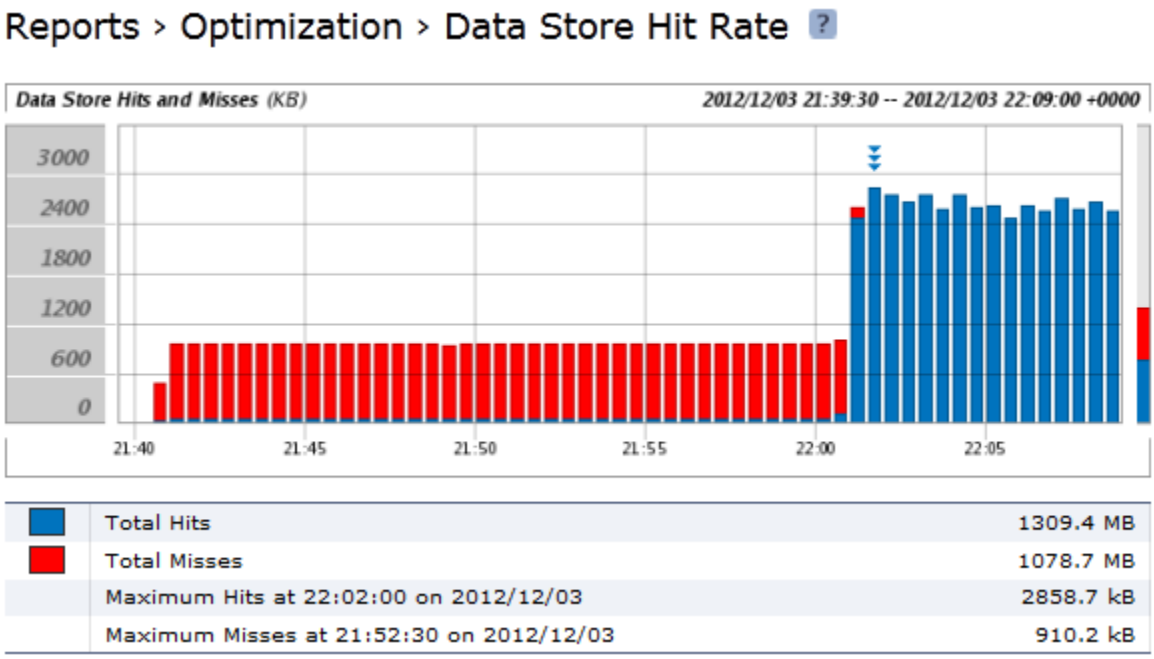


Figure 148. Data Store Hit Rate statistics

Troubleshooting and Diagnostics

Table 77. Troubleshooting and diagnostics

Issue	Diagnosis, Suggestions
What are the optimal statistics that can be used to certify that the optimal capacity has been reached?	The relatively quick way to know if the device is at maximum capacity is to incrementally add new test ports, and see if the overall concurrent connections, connections per second and throughput metrics increase. If TCP failures occur and new connections are failing, it is an indication that the limit has been reached. Other indications include a high latency, including connect Time, TTFB, and TTLB. Concurrent flows and number of simultaneous subscribers that can be supported is a function of system memory. Refer to the memory usage on the device, which should indicate that it is near its high water mark.

Test Case: Controlling Secondary HTTP Objective While Maintaining Predictable Concurrent Connection Levels.

Overview

Many devices have a restriction on the maximum concurrent users that can be supported. For these devices, it is very important to maintain predictable Concurrent Connections, but at the same time achieve predictable connections per second with the ability to control other objectives like transactions.

Objective

This test case aims to achieve a particular Concurrent Connection value at a determined CPS rate. Once the concurrent connection is stabilized, the transaction rate is also consistent, thus creating predictable CPS and TPS for concurrent connection objective. [Note: This is the most effective with lower byte page sizes and the accuracy falls as the page size increases]

Setup

This test requires a server and a client/subscriber port to emulate HTTP users. The inline device is a simple switch, but it can be replaced by any other inline device like a Firewall, SLB, DPI device, and so on.

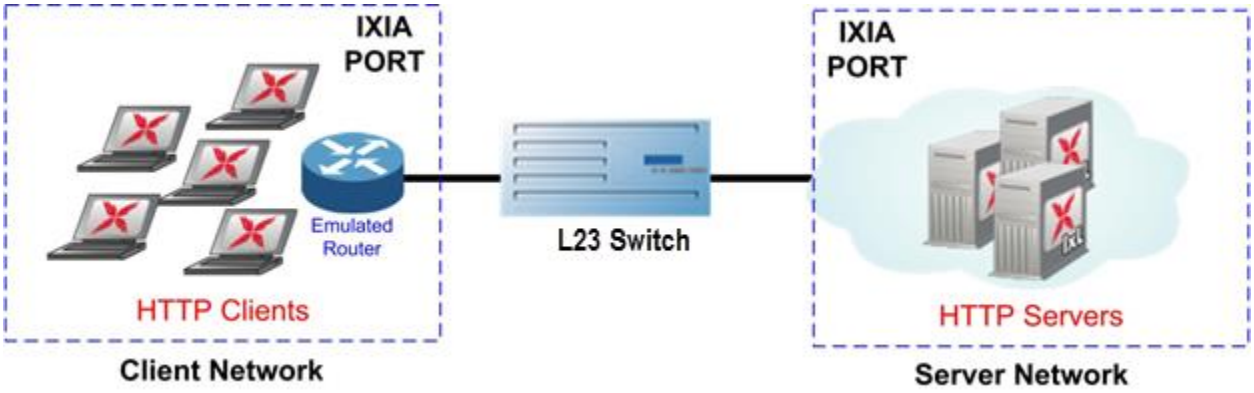


Figure 149. Ixia Test Topology

Test Variables

Test Tool Variables

The following test configuration parameters provide the flexibility to create the traffic profile that a device experiences in a production network.

Table 78. Test Tool Variables

Parameter	Description
HTTP clients	100 IP addresses or more, use sequential or use all IP addresses.
HTTP client parameters	HTTP 1.1 or Http1.0 with 'Keep Alive' Disabled (1 TCP transaction per connection)
HTTP client pages to request	1 GET command – payload of varying sizes from 1 Byte to 1024bytes
TCP Parameters	Client TCP - RX 32768 bytes, TX 4096 bytes Server TCP – RX 4096 bytes, TX 32768 bytes
MSS	1460 bytes
HTTP servers	1 per Ixia test port, or more

DUT Test Variables

Table 79. DUT test variables

Device(s)	Variation	Description
Server load balancer	Activate packet filtering rules	• Configure SLB engine for stickiness • Change the algorithm for load balancing • Use Bridged or Routed Mode for servers
Firewall	Activate access control rules	• Configure Network Address Translation or Disable for Routed Mode
Firewall security device	Enable deep content inspection (DPI) rules	• Advanced application-aware inspection engines enabled • IDS or threat prevention mechanisms enabled

Step-by-Step Instructions

1. Add the client **NetTraffic** object. Configure the **Client** network with total IP count, gateway and VLAN, if used.

   Add the server **NetTraffic**. Also configure the total number of servers that are used. For performance testing, use *1* server IP per test port.

   For a step-by-step workflow, see Appendix A.

Figure 150. IxLoad Scenario Editor View with Client and Server Side NetTraffics And Activities

2. The TCP parameters used for a specific test type are important when optimizing the test tool. Refer to the **Test Tool Variables** section to set the correct TCP parameters.

There are several other parameters that can be changed. Leave them at their default values unless you need to change them for testing requirements.

   For a step-by-step workflow, see Appendix B.

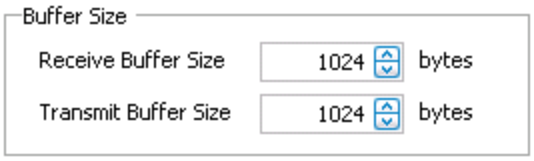


Figure 151. TCP Buffer Settings Dialogue

3. Configure the **HTTP server**. Add the **HTTP Server Activity** to the server **NetTraffic**. The default values are sufficient for this testing.

   For a step-by-step workflow, see Appendix C.

4. Configure the **HTTP client**. Add the **HTTP Client Activity** to the client **NetTraffic**. Refer to **Test Variables** section to configure the HTTP parameters.

   You can use advanced network mapping capabilities to use sequence IPs or use all IPs configured.

   For a step-by-step workflow, see Appendix D.

5. In the **Settings** tab change the HTTP version to *HTTP1.1*

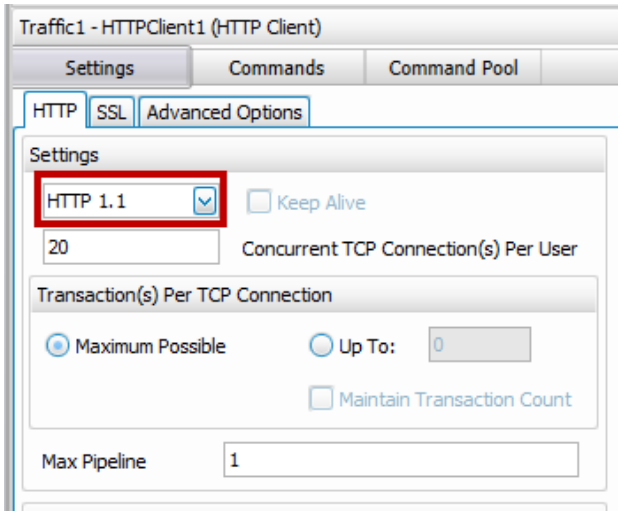


Figure 152. HTTP Client Protocol Settings dialogue

Based on the above numbers, the total time taken in achieving 200K CC must be 200K / 10K = 20 Seconds.

In the **Commands** tab, add a Think for *20 Seconds*. This ensures that the get requests are sent in batches of 20K. Also after the end of the first set of commands, the subsequent commands will be sent 20 seconds later. This ensures a constant Get request rate of 20K.

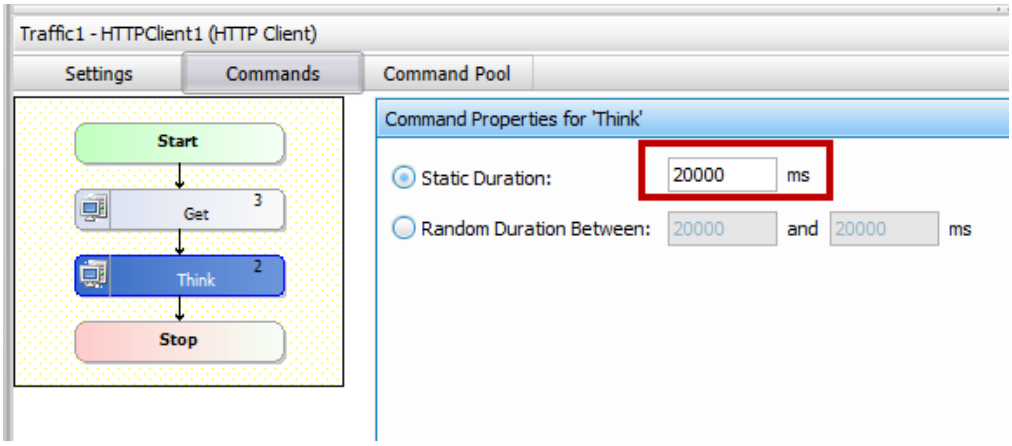


Figure 153. Test Objective Settings dialogue

6. Having setup the client and server networks and the traffic profile, the test objective can now be configured. Go to the **Timeline and Objectives** view and select **Switch to advanced timeline.**

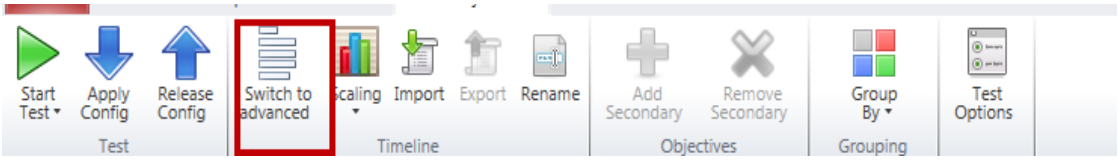


Figure 154. Switch to Advanced tab

7. Set the **Objective** as **Concurrent Connections** with value as 200000*.*

8. Configure the Linear Segment 1 in Timeline with Duration as 20 seconds and End Objective Value as` 200000.

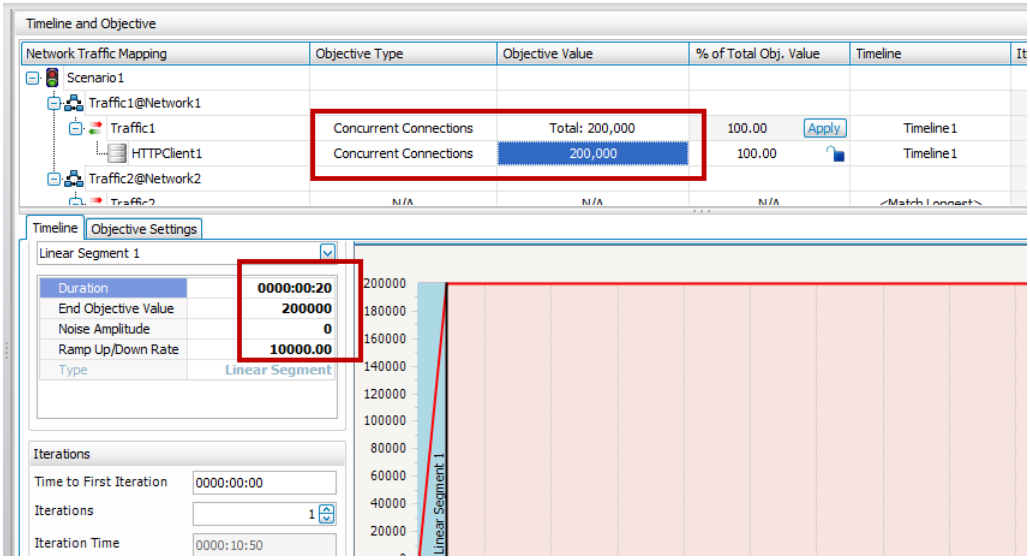


Figure 155. Test Objective Settings dialogue

9. Set the **Linear Segment 2** as **10 minutes** with end objective value set to **200000.**

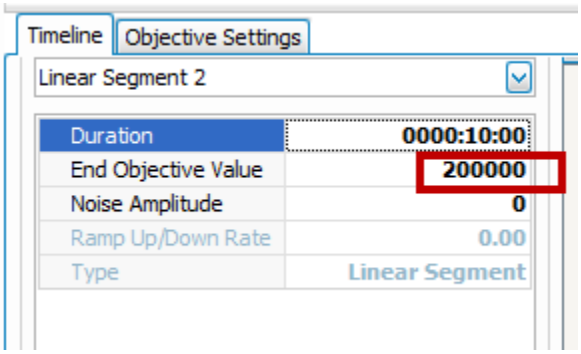


Figure 156. Set the linear Segment 2 to 10 minutes to Ensure the Test Runs Sufficient Time in Steady State

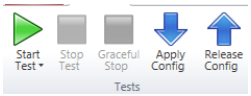
10. Once the Test Objective is set, the **Port CPU** on the bottom indicates the total number of ports that are required. See the Appendix below on adding the ports to the test.

    For a step-by-step workflow, see Appendix F.

    Run the test for few minutes for the performance to attempt to reach a steady-state. Steady state is referred as Sustain duration in the test. Continue to monitor the DUT for the target rate and any failure/error counters. See the Results Analysis section for important statistics and diagnostics information.

    In most cases, interpretation of the statistics is non-trivial, including what they mean under different circumstances. The **Results Analysis** section that follows provides a diagnostics-based approach to highlight some common scenarios, the statistics being reported, and how to interpret them.

11. Iterate through the test setting **TARGET_MAX_CONN** to the steady value attained during the previous run. To determine when the DUT has reached its **MAX_CONN**, see the **Results Analysis** section on interpreting results before making a decision.

    The test tool can be started and stopped in the middle of a test cycle, or wait for it to be gracefully stopped using the test controls shown here. 

    For a step-by-step workflow, see Appendix F.

Results Analysis

Different devices have different requirements of Concurrent Connections and Connections per second or Transactions per second. The key is to add a sleep (THINK) command after the GET with the duration of the THINK being the **Total Concurrent Connections Objective** / **Connections per second**.

Using the help of **Advanced Timeline**, set the CC ramp up to be equal to **Total Concurrent Connections Objective** / **Connections per second**.

Table 80. Key performance statistics to monitor

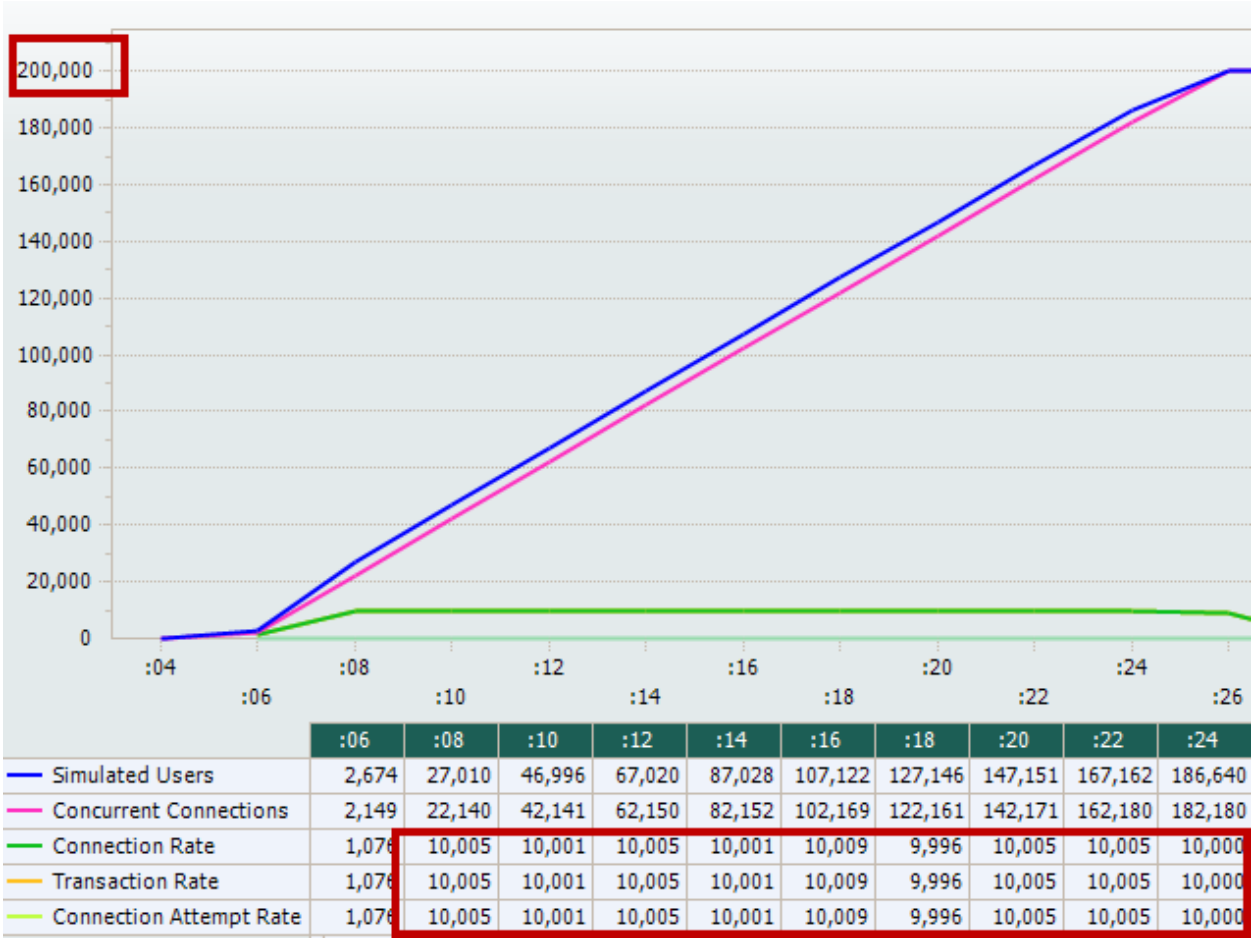
Metric	Key Performance Indicators	Statistics View
Performance metrics	Connections/sec Total connections, Number of Simulated Users, Throughput	HTTP Client – Objectives HTTP Client – Throughput
Application level transactions Application level failure monitoring	Requests Sent, Successful, Failed Request Aborted, Timeouts, Session Timeouts Connect time, 4xx, 5xx errors	HTTP Client – Transactions HTTP Client – HTTP Failures HTTP Client – Latencies

Metric	Key Performance Indicators	Statistics View
TCP Connection Information TCP Failure monitoring	SYNs sent, SYN/SYN-ACKs Received RESET Sent, RESET Received, Retries, Timeouts	HTTP Client – TCP Connections HTTP Client – TCP Failures
Other Indicators	Per URL statistics, Response Codes	HTTP Client – Per URL HTTP Client – xx Codes

Real-Time Statistics

The graph below provides a view of the real-time statistics for the test. Real-time statistics provide instant access to key statistics that should be examined for failures at the TCP and HTTP protocol level.

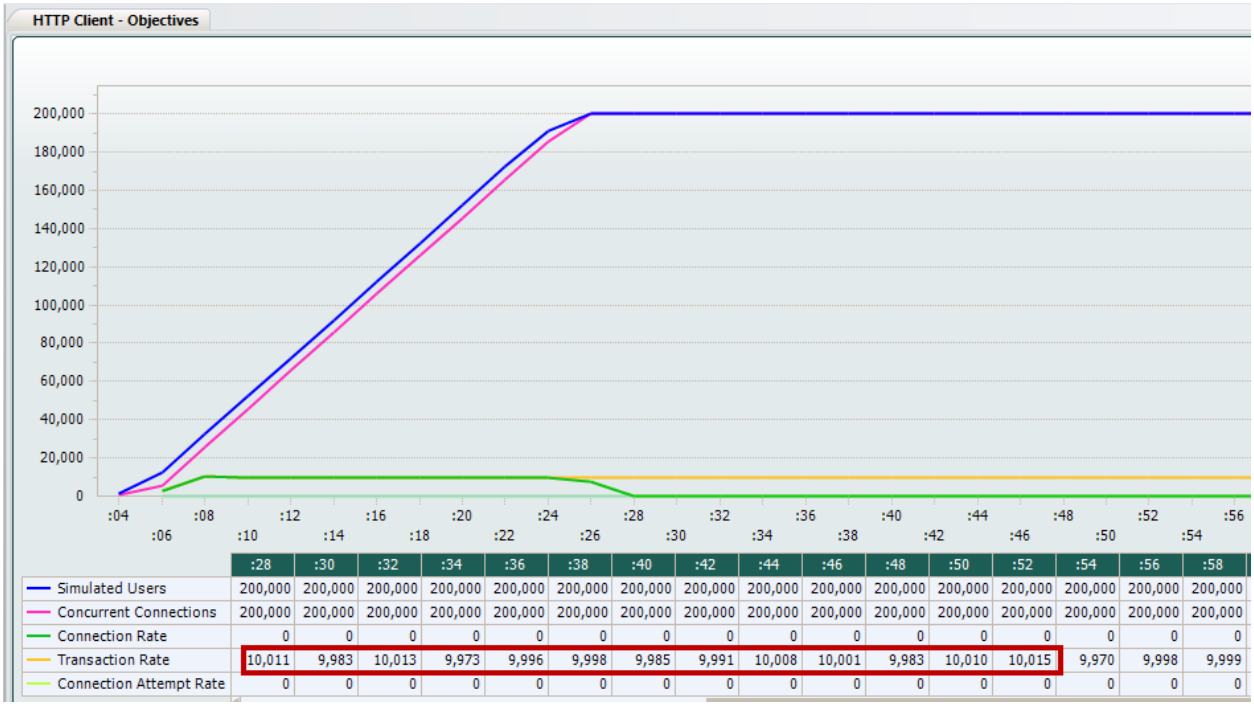
The first statistics view (HTTP Client – Objectives) shows a steady Connection Rate **of10000,** while trying to achieve the Concurrent Connections objective of **200000**



	:06	:08	:10	:12	:14	:16	:18	:20	:22	:24
Simulated Users	2,674	27,010	46,996	67,020	87,028	107,122	127,146	147,151	167,162	186,640
Concurrent Connections	2,149	22,140	42,141	62,150	82,152	102,169	122,161	142,171	162,180	182,180
Connection Rate	1,076	10,005	10,001	10,005	10,001	10,009	9,996	10,005	10,005	10,000
Transaction Rate	1,076	10,005	10,001	10,005	10,001	10,009	9,996	10,005	10,005	10,000
Connection Attempt Rate	1,076	10,005	10,001	10,005	10,001	10,009	9,996	10,005	10,005	10,000

Figure 157. HTTP Client Statistics view – showing concurrent connections results

In the second statistics view (HTTP Client – Objectives), the test has ramped up completely to 200000 CC, while maintaining a Transaction Rate of 10000.



	:28	:30	:32	:34	:36	:38	:40	:42	:44	:46	:48	:50	:52	:54	:56	:58
Simulated Users	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000
Concurrent Connections	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000	200,000
Connection Rate	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Transaction Rate	10,011	9,983	10,013	9,973	9,996	9,998	9,985	9,991	10,008	10,001	9,983	10,010	10,015	9,970	9,998	9,999
Connection Attempt Rate	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 158. HTTP client – consistent transaction rate after ramp up is completed

Rerun the test with multiple server (host) count.

The predictable transaction count might not be achievable when the test requires a sever IP count greater than 1. For that we need to use the Custom Mesh concept.

Step-by-Step Instructions

1. Set the **Client side IP** count to **10000** IPs and the **server side net traffic** IP to **100**.

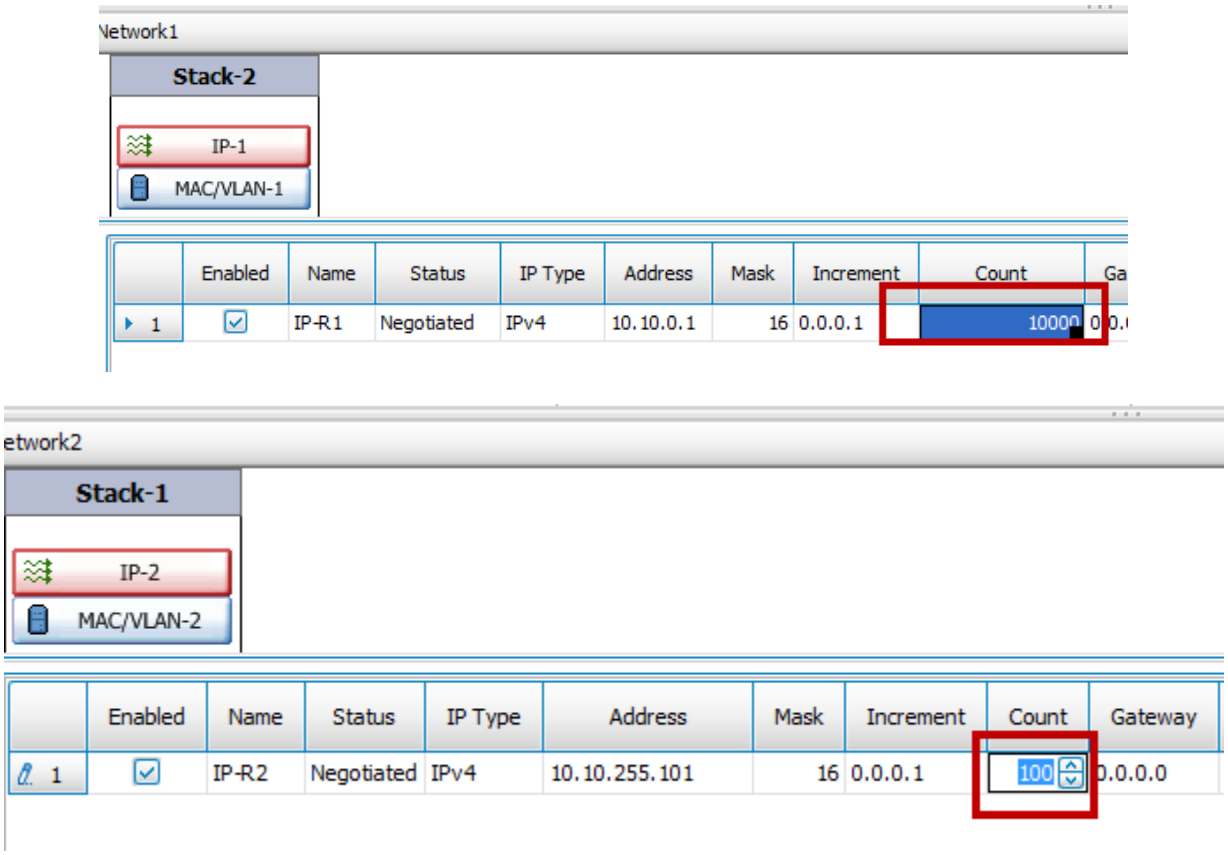


Figure 159. HTTP client and server network with multiple IPs

2. Click the **Lollypop** connector that connects the client to server.
3. Select **Traffic Map** as **Custom.**

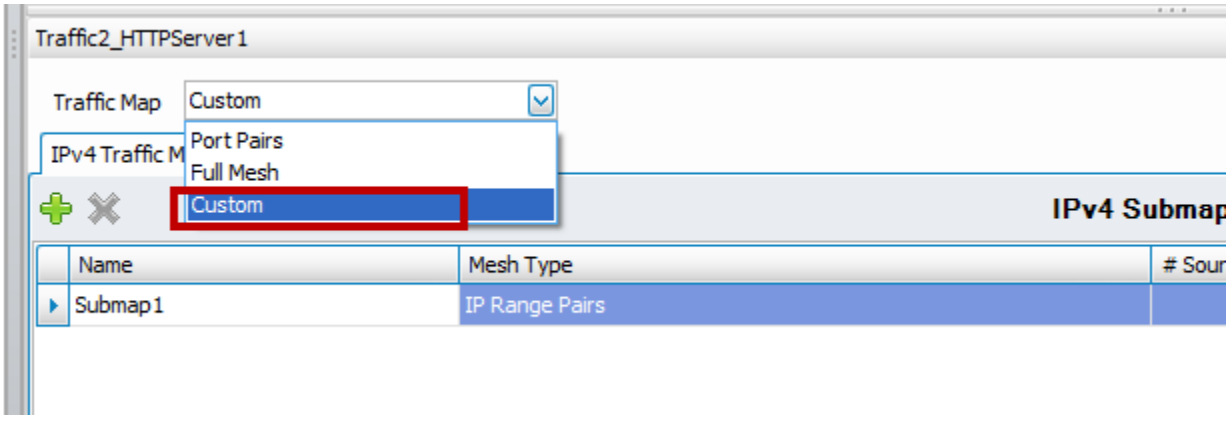


Figure 160. Select custom mesh

4. Right click the source ip and click **Split**. Split it exactly into **100 ranges** as100 is the number of IP's at the server side.

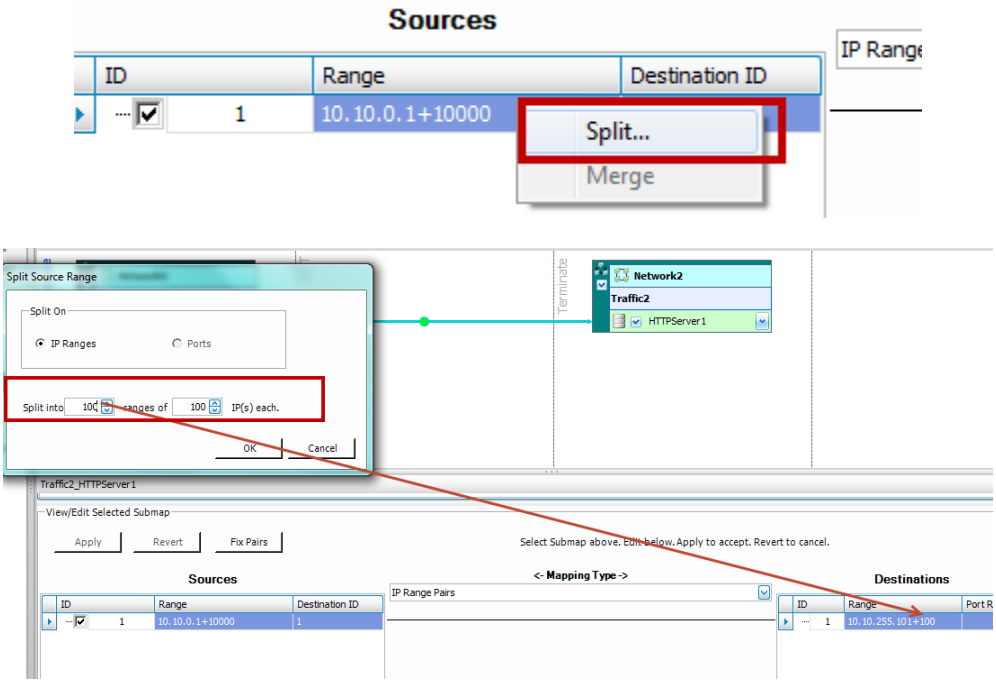


Figure 161. Divide the client side IP's into ranges equal to the number of IP's at server side. Since Cserver has 100 IP's hence clients 10000 IP's are divided into 100 ranges

5. Repeat the same for the ranges at the destination side. Split them into **100** ranges of**1** IP each.

This way, every 100 source IPs associate with one single destination IP. This ensures HTTP1.1 behavior even with several different server IP's.

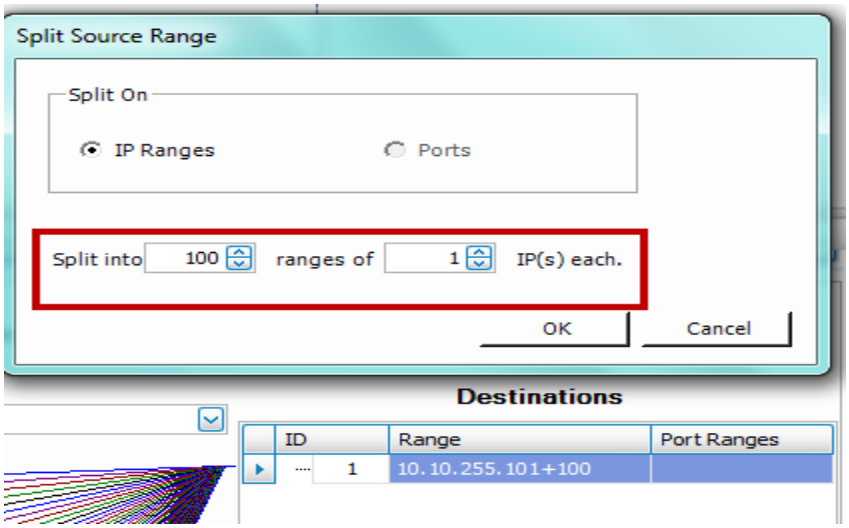


Figure 162. HTTP server side IPs are divided into 100 ranges with each range having one IP

6. Select the **Mapping Type** as **IP Range Pairs** and click **Fix Pairs**.

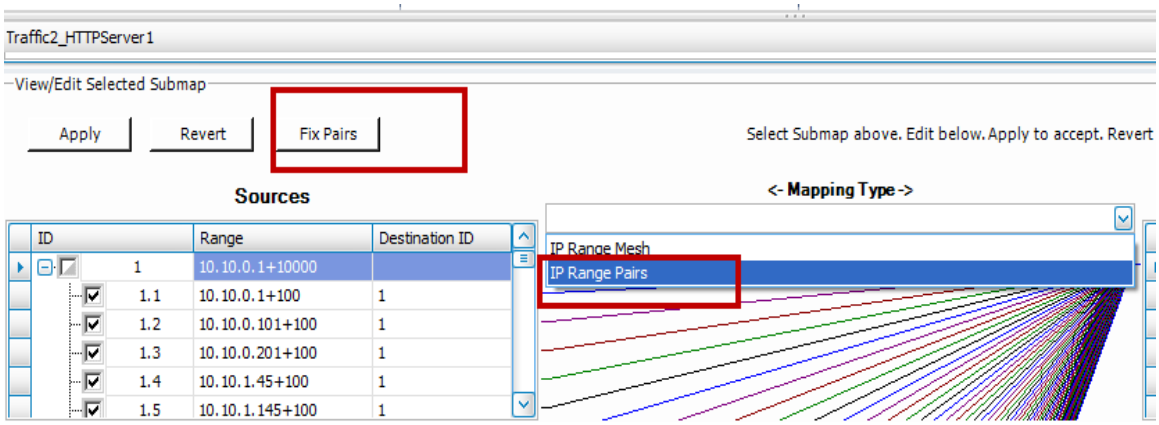


Figure 163. Creating IP-range pairs to have each 100 client IP connect to only 1 server IP

This results in perfect range mapping as shown in the below figure, where each 100 ip's at client side connect to only one ip at the target side.

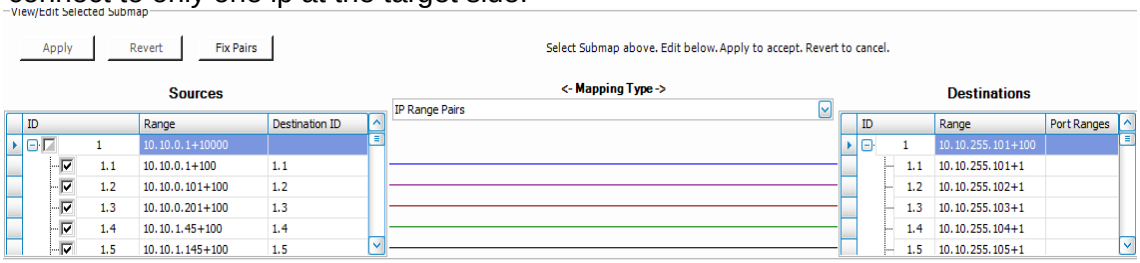


Figure 164. Client and server IP range mapping

7. Rerun the test with rest of the parameters remaining same.

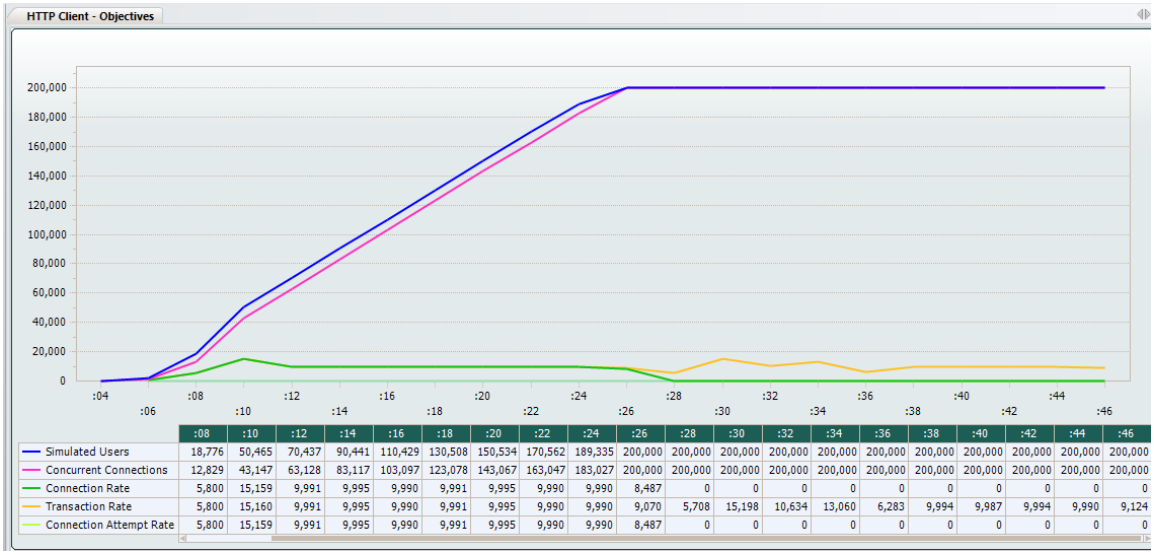


Figure 165. Similar performance as shown in case 1

Troubleshooting and Diagnostics

Table 81. Troubleshooting and diagnostics

Issue	Diagnosis, Suggestions
The CC never reaches steady state; it is always moving up and down and the variation is reasonably high.	If the device is not reaching steady-state, check the Connect time, and TCP failures. If the Connect time keeps increasing, then it is an indication that the device may not be able to service or maintain any connections. Check the Simulated Users count – if it is very high then it is an indication that the test tool is attempting to reach the target and its system resources are depleting. Add more test ports or check the configuration to determine any possible issue.
What are the optimal statistics that can be used to certify that the optimal concurrent connections (CC) metric has been reached?	The relatively quick way to know if the device is at maximum capacity is to incrementally add new test ports, and see if the overall CC increases. If TCP failures occur and new connections are failing, then it is an indication that the limit is reached. Other indications include a high latency, including Connect Time, TTFB and TTLB. Concurrent connections metric is a function of system memory; look at the memory usage on the device, must indicate it is near its high water mark.

Test Case: URL Filtering

Overview

There is often a requirement to restrict a user's access to objectionable content. These restrictions can be enforced by various agencies and at various levels and can range from a concerned parent restricting access of impressionable children at home to school/university-level restrictions, company/work-level restrictions, and even ISP- and government-level restrictions.

Some techniques of content filtering have already been discussed earlier in this Black Book: application filtering with access control lists, content inspection, web security filtering, etc. This chapter deals with another content filtering mechanism known as URL filtering.

URL filtering essentially entails a device that either has a database of blocked words/strings or a database of allowed keywords. Such a device looks at the URLs being requested by the users and only allows those requests to go through that either don't contain a blocked word/string or contains only permissible/allowed keywords, respectively.

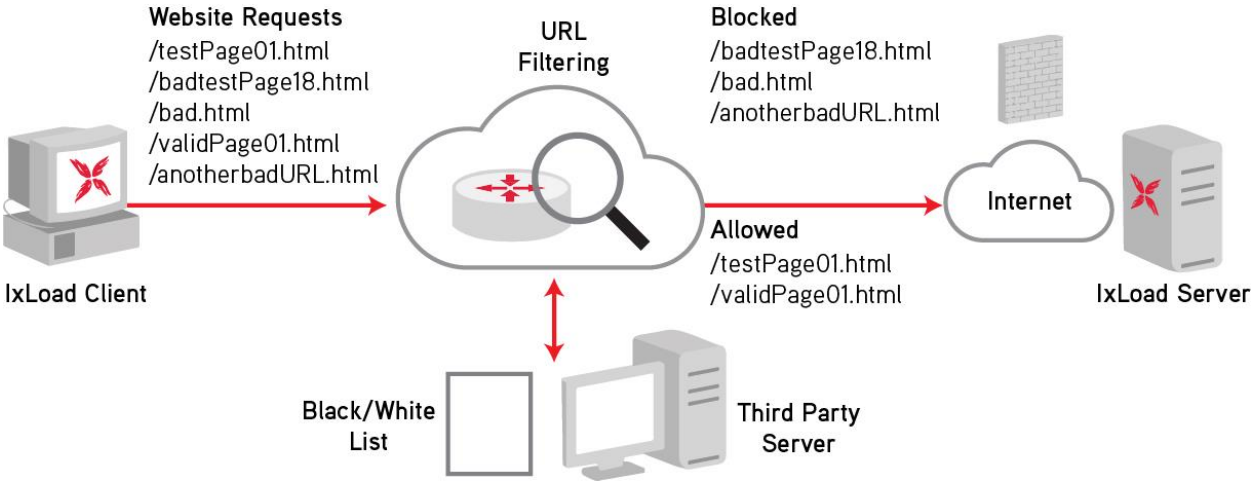


Figure 166. URL filtering

Objective

The purpose of this test case is to demonstrate how to configure HTTP clients (simulated users) in IxLoad that are requesting multiple URLs, some of which consist of blocked words and strings using playlist csv files on the HTTP client.

The test then also demonstrates if any of the blocked URLs were successfully served by the emulated IxLoad HTTP server by means of the server's 'Per URL' stat view. Any URL with blocked words that were successfully served by the server indicates a failure of the DUT, whereby the DUT failed to catch the bad URL and failed to take action against it by blocking it.

Setup

The firewall should be configured with the basic web filtering/security feature enabled and tested. Additional web security features should be turned on and tested until all web security protection mechanisms are enabled and tested.

Here is an example firewall configuration:

Allow TCP/80 traffic from INT network to EXT network.
Allow ICMP traffic from INT network to EXT network.
Block all ICMP traffic from EXT network to INT network.
Enable one after another
Web security feature 1
Web security feature
…
Web security all features

Figure 167. Firewall configuration with content inspection

The test tool setup includes at least one server and one client port.

Test Variables

Test Tool Configuration

The test tool configuration profile is shown below. Configure one NetTraffic pair, with HTTP client and server protocol activities as specified below.

HTTP client and server configuration

Table 82. HTTP configuration for test case

Parameter	Description
HTTP clients	100 IP addresses or more, use sequential or use all IP addresses.
HTTP client parameters	HTTP 1.1 20 TCP connections per user Maximum transactions per TCP connection
HTTP client pages to request	1 GET command – payload of varying sizes
HTTP page content	Configure a test with various patters of Hot, Warm, Cold data, or a mix of them.
TCP parameters	Client TCP - RX 32768 bytes, TX 4096 bytes Server TCP – RX 4096 bytes, TX 32768 bytes
MSS	1460 bytes
HTTP servers	1 per Ixia test port, or more
HTTP server parameters	Random response delay – 0 – 20 ms Response timeout – 300 ms

DUT Test Variables

URL filtering can be configured not only on the various different devices like ADCs and firewalls, these policies can also be configured on the client machine itself, either on the browser or email client or on the workstation itself. For the purpose of this test case we will only focus on the devices in the middle like ADCs and firewalls. Configuration of these policies on the client machine is beyond the scope of this test case.

Table 83. HTTP configuration for test case

Device(s)	Variation	Description
Application Delivery Controller	Activate application-layer security	• Configure Black and White lists
Firewall	Activate access control rules	• Configure Black and White lists

Step-by-Step Instructions

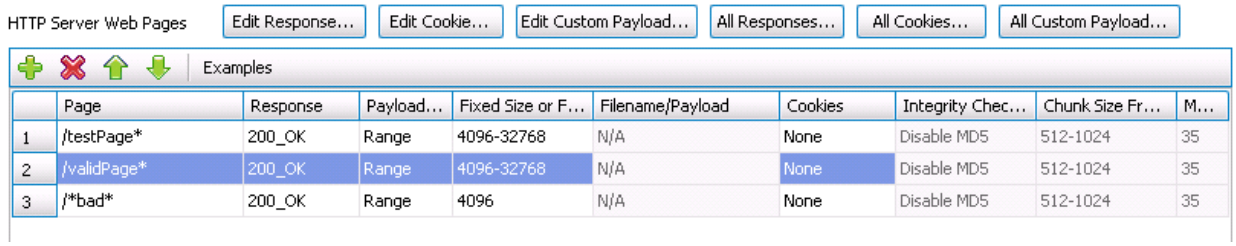
1. Add the client **NetTraffic** object. Configure the **Client** network with total IP count, gateway and VLAN, if used.

   Add the server **NetTraffic**. Also configure the total number of servers that are used. For performance testing, use *1* server IP per test port.

   For a step-by-step workflow, see Appendix A.

Figure 168. IxLoad Scenario Editor View with Client and Server Side NetTraffics And Activities

2. Configure the **HTTP server**. Click on the **HTTP Server Activity** in the server **NetTraffic**. Remove all the default HTTP Server Web Page and add three new web pages:
   - /testPage* with a size range of 4096-32768
   - /validPage* with a size range of 4096-32768
   - /*bad* with a fixed size of 4096



	Page	Response	Payload...	Fixed Size or F...	Filename/Payload	Cookies	Integrity Chec...	Chunk Size Fr...	M...
1	/testPage*	200_OK	Range	4096-32768	N/A	None	Disable MD5	512-1024	35
2	/validPage*	200_OK	Range	4096-32768	N/A	None	Disable MD5	512-1024	35
3	/*bad*	200_OK	Range	4096	N/A	None	Disable MD5	512-1024	35

Figure 169. HTTP Server Web Pages

Note: The "*" used in the page name above is a wildcard character. As long as the text string (before or) after the "*" is a match, IxLoad will automatically match the full string. As an example, "/testPage*" will match all of the following pages: "/testPage01.html", "/testPage99.txt", and so on.

For a step-by-step workflow, see Appendix C.

3. For the purpose of this test we will create a csv playlist file that contains a list of webpages that the client will request.

   Playlists are lists of servers and the resources stored on them, such as web pages and video streams. You can create playlists outside of IxLoad and import them into IxLoad. This enables you to create playlists using applications other than IxLoad, and that contain servers and resources compiled from many different sources.

   Create a CSV file using Microsoft Excel (or another text editor) with the following content:

Webpage
/testPage01.html
/testPage02.html
/validPage01.html
/testPage03.html
/validPage01.html
/testPage04.html
/testPage05.html
/validPage04.html
/testPage06.html
/testPage07.html
/validPage99.html
/testPage08.html
/testPage09.html
/testPage10.html
/testPage11.html
/testPage12.html
/testPage13.html
/testPage14.html
/testPage15.html
/testbadPage16.html
/testPage17.html
/badtestPage18.html
/testPage19.html
/testPage20.html
/bad.html

/anotherbadURL.html
/anotherbadURL1.html
/anotherbadURL2.html
/anotherbadURL3.html

You can edit the names of the webpages to match the blocked and allowed key-words configured on the DUT.

4. Upload the above playlist file into the IxLoad Playlist Profile.

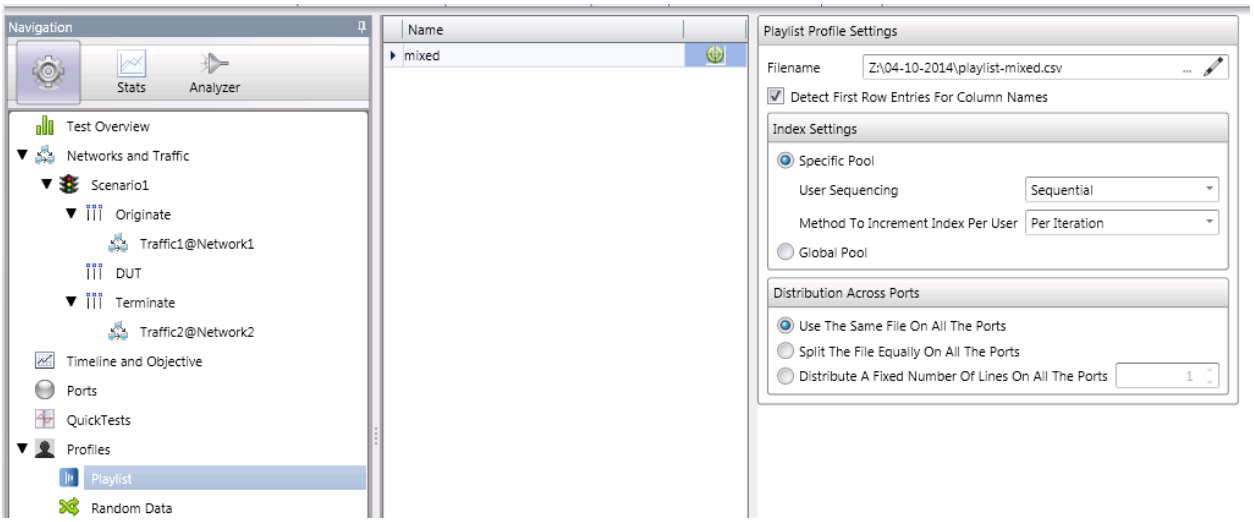


Figure 170. Uploading a playlist

For a step-by-step workflow, see Appendix G.

5. Configure the **HTTP client**. Click on the **HTTP Client Activity** to the client **NetTraffic**.

   For a step-by-step workflow, see Appendix D.

6. In the **Settings** tab change the HTTP version to *HTTP1.0* and number of Concurrent TCP Connection(s) Per user to 3

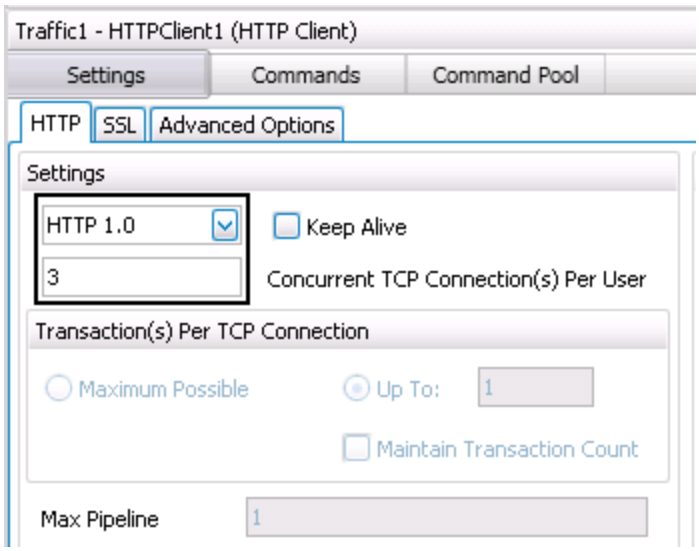


Figure 171. HTTP Settings tab

In the **Commands** tab, add a Get command. Ensure that playlist column is selected as the value of the Page/Object field.

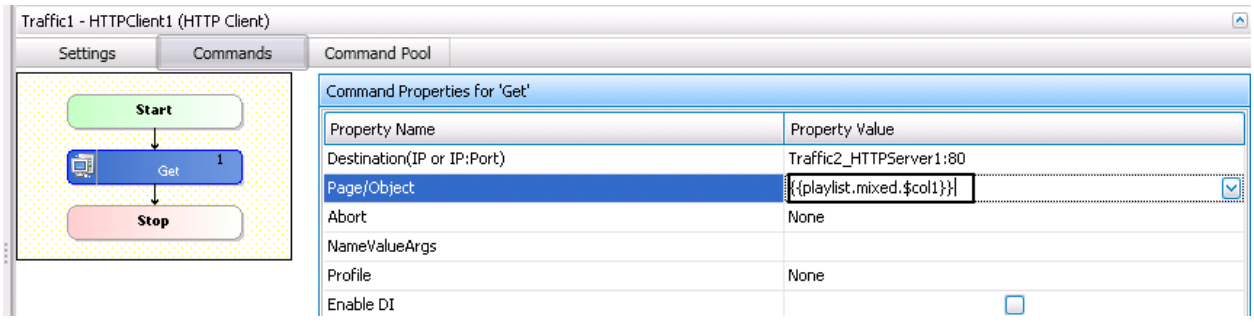


Figure 172. Commands tab

7. Having setup the client and server networks and the traffic profile, the test objective can now be configured. Go to the **Timeline and Objective.**

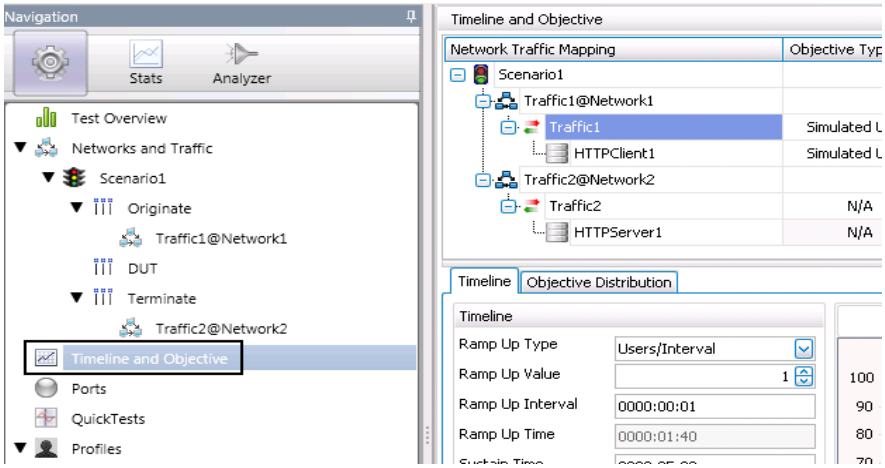


Figure 173. Timeline and Objective

8. Set the **Objective** as **Simulated Users** with value of 10,000**.**

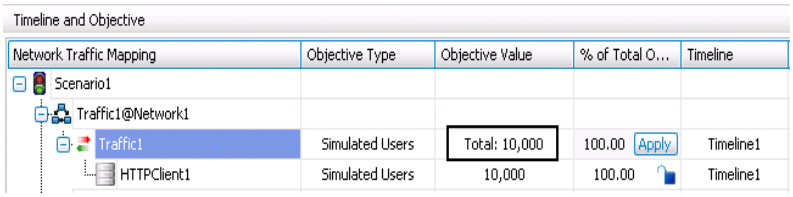


Figure 174. Objective: Simulated Users

9. Set the Ramp Up Value as 100 and Sustain Time as 10 minutes

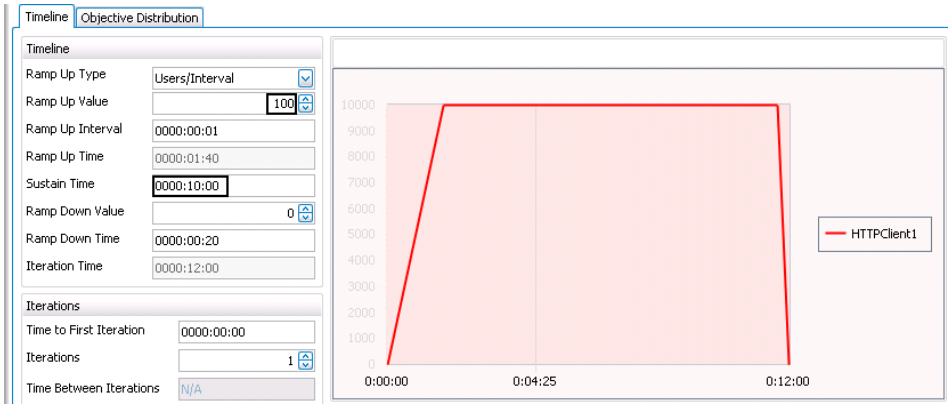


Figure 175. Timeline

10. Once the Test Objective is set, the **Port CPU** on the bottom indicates the total number of ports that are required. See the Appendix below on adding the ports to the test.

    For a step-by-step workflow, see Appendix F.

11. Run the test

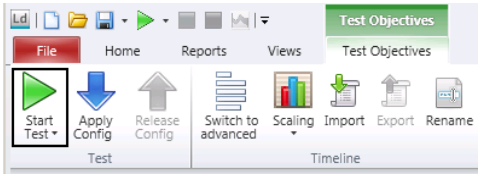


Figure 176. Start Test

For a step-by-step workflow, see Appendix F.

Wait for the test to reach Steady State. Steady State is referred as Sustain duration in the test. Continue to monitor the DUT for any failure/error counters and to also see it blocking the blacklisted URLs. See the Results Analysis section for important statistics and diagnostics information.

Results Analysis

The expectation from the device in the middle is that it should successfully be able to block the black-listed URLs while allowing the requests with the permitted URLs to go through.

IxLoad provides key stats here to help the user determine if the DUT was successfully able to block all the black-listed URLs. The two most important stat views of interest here are the **HTTP Server – Per URL** and the **HTTP Client – Per URL** views.

If there are any requests from the black-list that gets past the DUT, these will be reported in the “HTTP Server – Per URL” stat view. On the other hand the “HTTP Client – Per URL” view will report all the URLs that the client is requesting. If the DUT is configured properly, the blocked/black-listed URLs should not be successful in this view.

If the device is sending HTTP error responses for the black-listed URLs then those will also be reported in the **HTTP Client – HTTP Failure** stat view. Another stat that can indicate TCP level errors is the **HTTP Client – TCP Failures** stat view

Real-Time Statistics

HTTP Client – Per URL

HTTP Client - Per URL

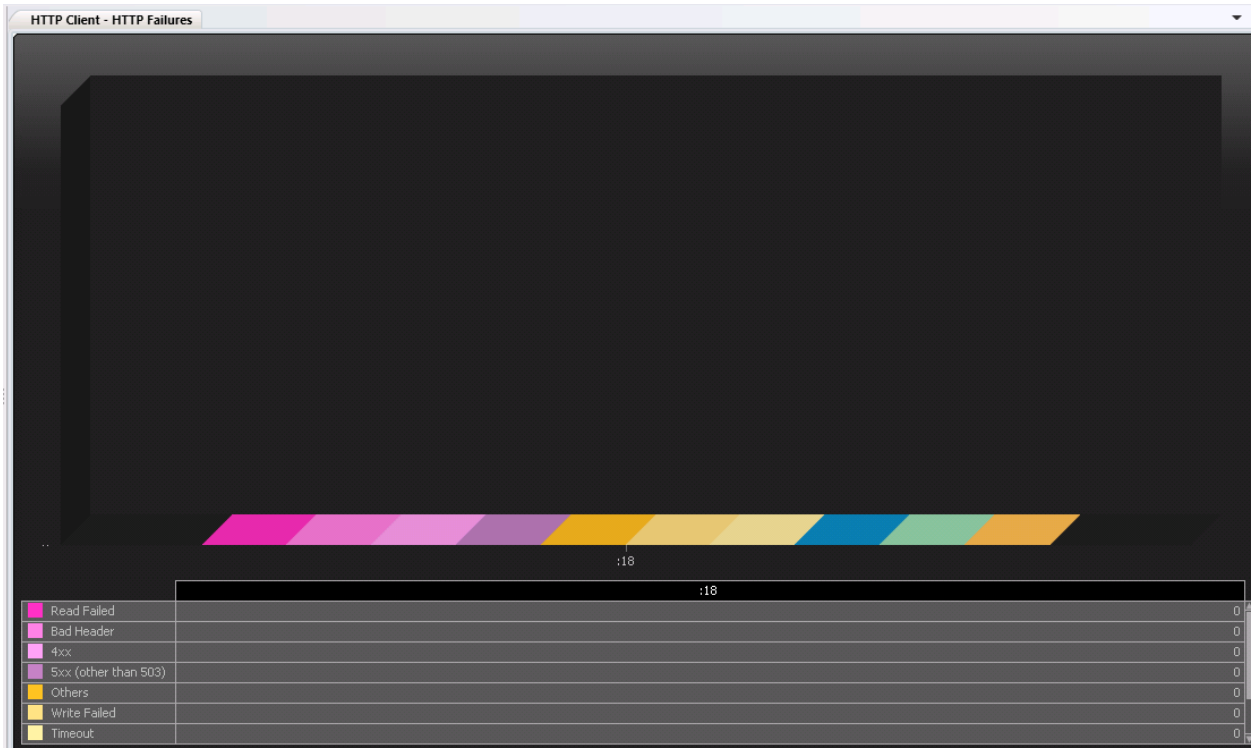
	Stat Name	Requests Sent	Requests Successful	Requests Successful (Provisional)	Inter
1	URL anotherbadURL.html@10.200.134.149/Card1/Port2/HTTPCli ...	21,067	20,755	0	
2	URL anotherbadURL1.html@10.200.134.149/Card1/Port2/HTTPClient1	21,250	20,940	0	
3	URL anotherbadURL2.html@10.200.134.149/Card1/Port2/HTTPClient1	21,043	20,738	0	
4	URL anotherbadURL3.html@10.200.134.149/Card1/Port2/HTTPClient1	21,083	20,788	0	
5	URL bad.html@10.200.134.149/Card1/Port2/HTTPClient1	20,793	20,487	0	
6	URL badtestPage18.html@10.200.134.149/Card1/Port2/HTTPClient1	21,049	20,742	0	
7	URL testPage01.html@10.200.134.149/Card1/Port2/HTTPClient1	21,321	20,281	0	
8	URL testPage02.html@10.200.134.149/Card1/Port2/HTTPClient1	21,097	20,026	0	
9	URL testPage03.html@10.200.134.149/Card1/Port2/HTTPClient1	20,950	19,811	0	
10	URL testPage04.html@10.200.134.149/Card1/Port2/HTTPClient1	21,099	20,033	0	
11	URL testPage05.html@10.200.134.149/Card1/Port2/HTTPClient1	21,116	20,099	0	
12	URL testPage06.html@10.200.134.149/Card1/Port2/HTTPClient1	21,137	20,126	0	
13	URL testPage07.html@10.200.134.149/Card1/Port2/HTTPClient1	20,822	19,787	0	
14	URL testPage08.html@10.200.134.149/Card1/Port2/HTTPClient1	21,285	20,241	0	
15	URL testPage09.html@10.200.134.149/Card1/Port2/HTTPClient1	21,346	20,306	0	
16	URL testPage10.html@10.200.134.149/Card1/Port2/HTTPClient1	21,052	20,058	0	
17	URL testPage11.html@10.200.134.149/Card1/Port2/HTTPClient1	20,962	19,891	0	
18	URL testPage12.html@10.200.134.149/Card1/Port2/HTTPClient1	21,137	20,047	0	
19	URL testPage13.html@10.200.134.149/Card1/Port2/HTTPClient1	21,050	19,963	0	
20	URL testPage14.html@10.200.134.149/Card1/Port2/HTTPClient1	21,068	20,041	0	
21	URL testPage15.html@10.200.134.149/Card1/Port2/HTTPClient1	21,232	20,107	0	
22	URL testPage17.html@10.200.134.149/Card1/Port2/HTTPClient1	21,184	20,180	0	
23	URL testPage19.html@10.200.134.149/Card1/Port2/HTTPClient1	21,085	19,996	0	
24	URL testPage20.html@10.200.134.149/Card1/Port2/HTTPClient1	21,023	19,992	0	
25	URL testbadPage16.html@10.200.134.149/Card1/Port2/HTTPClient1	21,146	20,835	0	
26	URL validPage01.html@10.200.134.149/Card1/Port2/HTTPClient1	42,403	40,252	0	
27	URL validPage04.html@10.200.134.149/Card1/Port2/HTTPClient1	21,075	20,036	0	

HTTP Server – Per URL

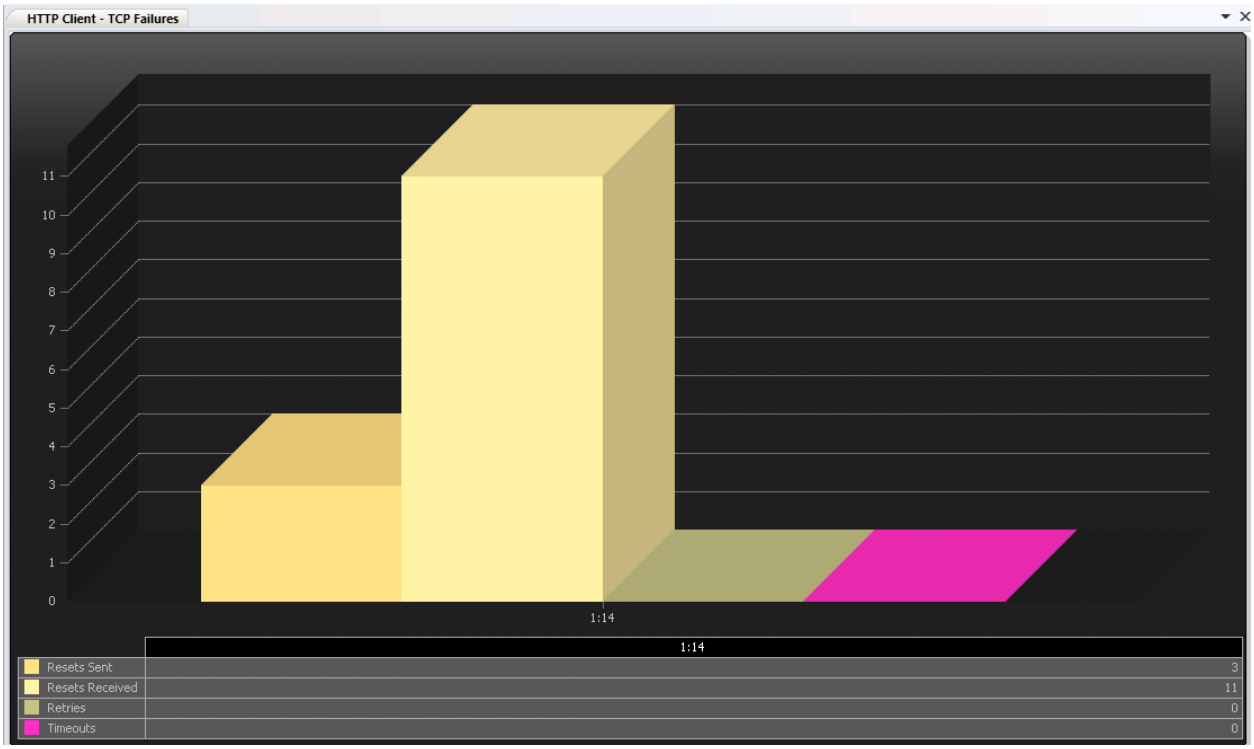
HTTP Server - Per URL

	Stat Name	Requests Received	Responses Sent	Responses Sent (1xx)	Responses Sent (2xx)	Responses Sent (3xx)	Responses Sent (4xx)	Responses Sent (5xx)
1	URL _bad_@10.200.134.149/Card2/Port2	157,080	157,080	0	157,080	0	0	0
2	URL testPage_@10.200.134.149/Card2/Port2	403,878	403,878	0	403,878	0	0	0
3	URL validPage_@10.200.134.149/Card2/Port2	90,018	90,018	0	90,018	0	0	0

HTTP Client – HTTP Failures



HTTP Client – TCP Failures



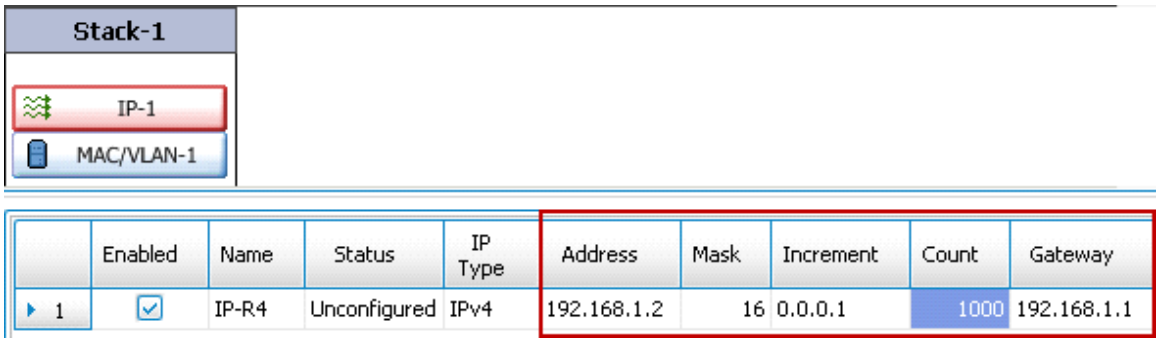
Appendix A: Configuring IP and Network Settings

1. In the **Scenario Editor**, add a **NetTraffic** Object. This object can contain network configurations and **Activities** (Protocols).

Figure 177. New NetTraffic Object

2. Click on the **Network1** to configure the IP, TCP and other network configuration parameters.

   On the **IP stack**, configure the desired number of static IP addresses. If VLANs are required, configure it by selecting the **MAC/VLAN** stack and configuring the details.



	Enabled	Name	Status	IP Type	Address	Mask	Increment	Count	Gateway
1	☑	IP-R4	Unconfigured	IPv4	192.168.1.2	16	0.0.0.1	1000	192.168.1.1

Figure 178. IP Stack

VLAN

	Enabled	Name	Status	First ID	Increment every # addresses	Increment By	Unique Count
1	☑	VLAN-R4	Unconfigured	101	1	1	1000

Figure 179. VLAN Settings

Appendix B: Configuring TCP Parameters

The TCP settings shown below should be configured as per the test tool **Input Parameters** for the specific test case.

1. Select the **NetTraffic** for the TCP configurations. Select the **Network** object to open the **Stack Manager** window pane on the bottom.

2. Click the **TCP/IP** button. Configure the **Receive and Transmit Buffer Size** based on what is set in the **Input Parameters** for TCP/IP settings.

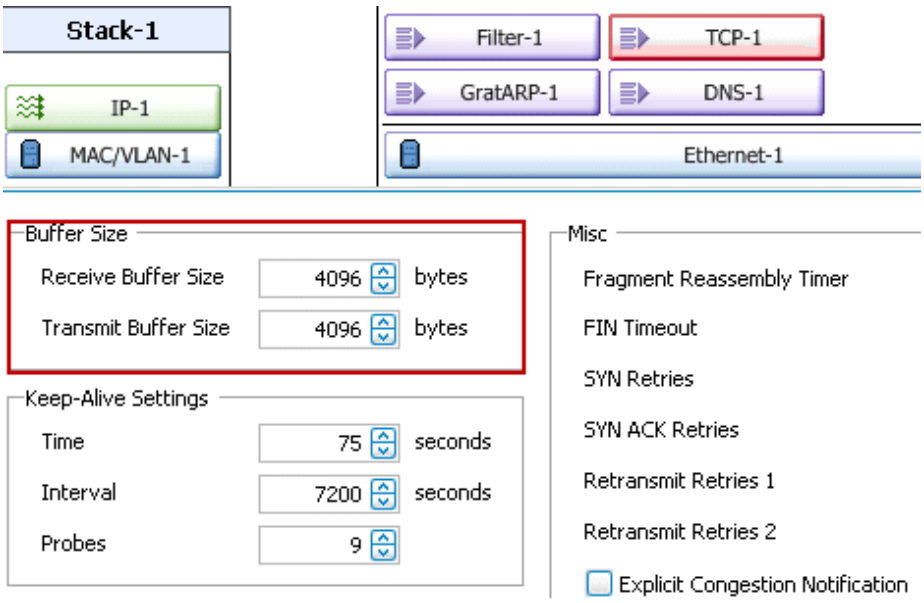


Figure 180. Buffer Size Settings

3. Other TCP/IP configurations should not be changed from their defaults.

Appendix C: Configuring HTTP Servers

1. Add the **HTTP server Activity** to the server **NetTraffic** object.

2. To configure the HTTP server parameters, pages and responses, select the **HTTPServer1** object to open the configuration pane on the bottom.

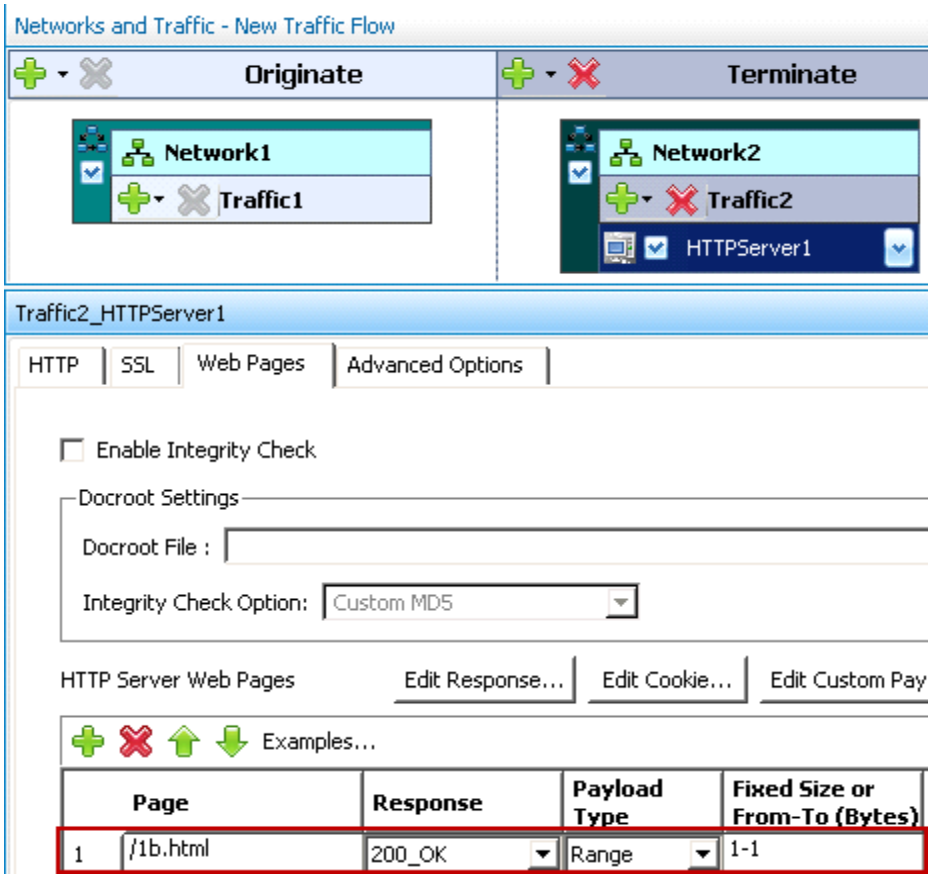


Figure 181. HTTP Server Configuration

3. Configure the HTTP server as outlined in the Input Parameters section.

Appendix D: Configuring HTTP Clients

1. Add the **HTTP client Activity** to the client **NetTraffic** object.
2. To configure the HTTP parameters, and pages to request, select the “HTTPClient1” object to open the configuration pane on the bottom.
3. Configure the HTTP behavior on the “HTTP” tab. Refer to the Input Parameters section.

   The version of HTTP to use, TCP connections per user and transactions per TCP are configured here.

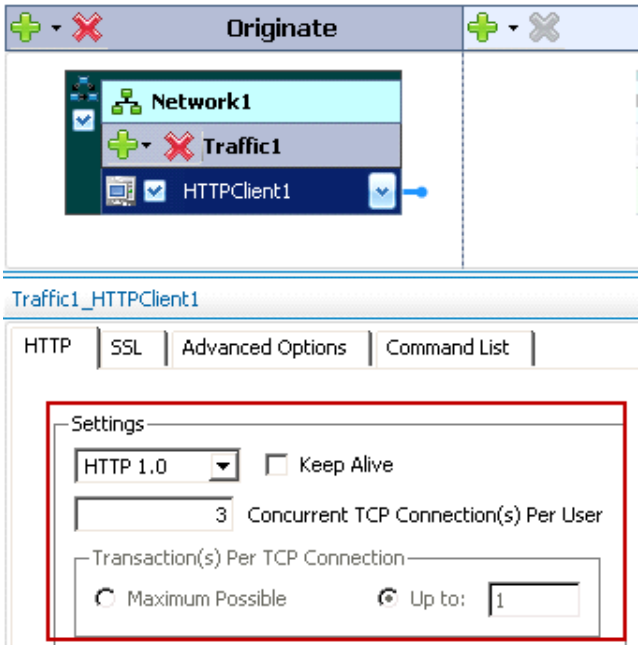


Figure 182. HTTP Client Configuration

4. Go to the **Command List** to configure the list of HTTP commands to use. The specific commands that should be used for the specific test objective type are outlined in the **Input Parameters**.

   For example, when testing for CPS, the page size is *1b.html*.

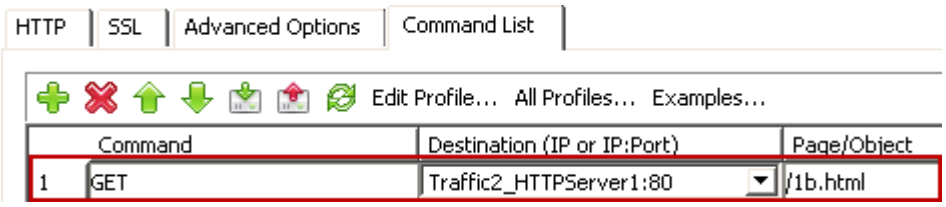


Figure 183. HTTP Client Command List

   For throughput testing, the page size is *1024k.html*.

Note the **Traffic2_HTTPServer1:80** in the **Destination** field. IxLoad supports this symbolic destination to associate traffic to the server object. It allows dynamic configuration of traffic distribution from the clients to servers across several ports, without manual configuration.

5. The use of source IP addresses for a test can depend on the test requirements. To maximize the test tool's performance, this is set to the default configuration of **Use Consecutive IPs**. This means that every simulated user will use the IPs as needed from the network configuration.

   To change it, click the **Traffic1** object and change the **Use Source IP Rule (per port)** setting.

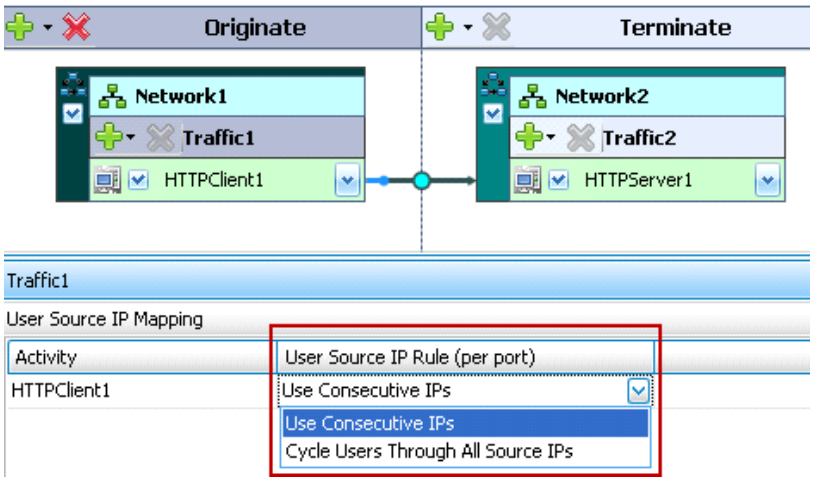


Figure 184. User Source IP Rule (per port)

Using **Cycle Users Through All Source IPs** allows all IP addresses to be used. This may be desirable, however note that performance from the test tool may vary. Consider running a baseline test port-to-port to determine the test tool's limit before performing a test with this feature.

Appendix E: Setting the Test Load Profile and Objective

1. Open the **Timeline and Objective** window from the **Test Configuration** left pane.

   Each **NetTraffic** will be listed, with one or more activities under it. Set the **Objective Value** to the desirable value, based on what the target performance is.

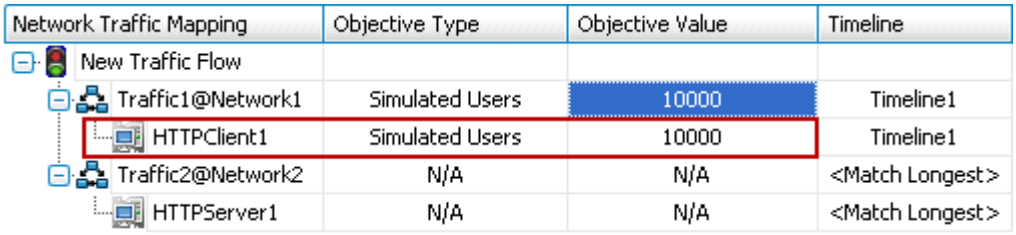


Figure 185. Settings the Objective Value

2. Set the ramp up and the overall test duration. Use the **Input Parameters** section.

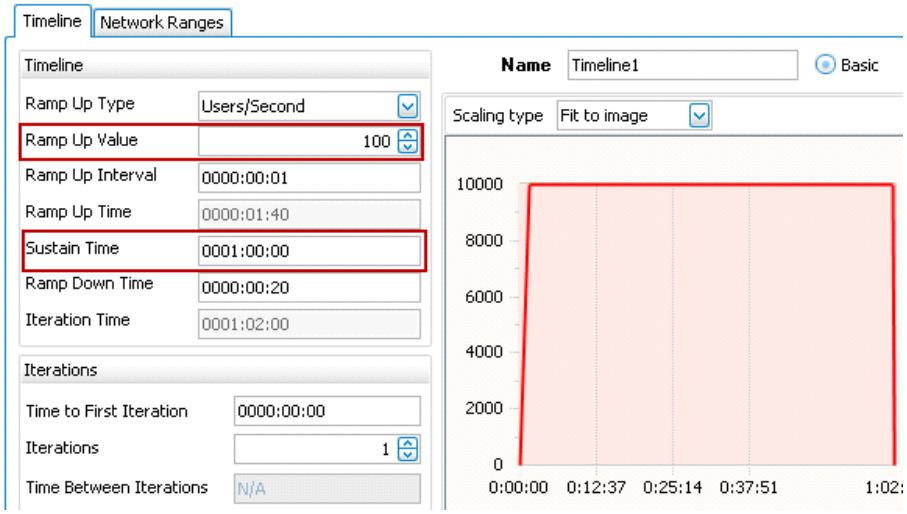


Figure 186.

Figure 187. Setting the Ramp Up and Overall Test Duration

3. The **Number of Ports** required highlights the ports required to accomplish the test objective. The computations here are conservative and its recommended that Ixia's guidance be used in addition to the ports required listed here.

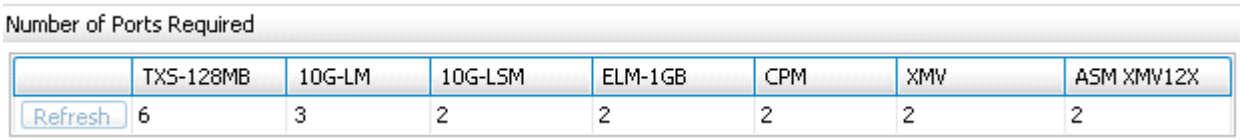


Figure 188. Number of Ports Required

Appendix F: Adding Test Ports and Running Tests

1. Go to the **Port Assignments** window, and add the chassis that will be used.

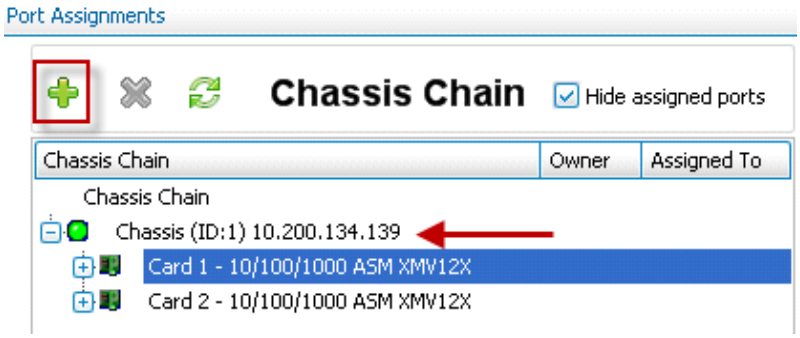


Figure 189. Adding a Chassis

2. The test ports are assigned at the **NetTraffic** level. In the simplest case in which HTTP client and server traffic is being emulated, there will be two **NetTraffic**s. Add the **Required** number of ports to each **NetTraffic** object. Use the arrows to add or remove the available ports to the Assigned Ports.

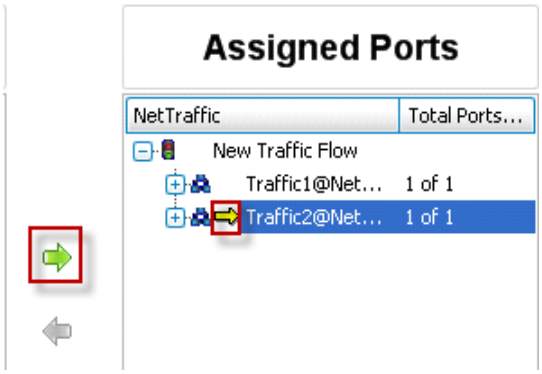


Figure 190. Assigning Ports

3. At this point, the test is ready to be executed.

Figure 191. Executing a Test

The PLAY button starts the test. The RED button stops the test. The DOWN arrow downloads the configuration to the test ports, but does not initiate any traffic. The **UP** arrow de-configures the test ports.

Appendix G: Adding a Playlist

1. In the ribbon, click Home | Profiles to display the list of profiles in the configuration panel.

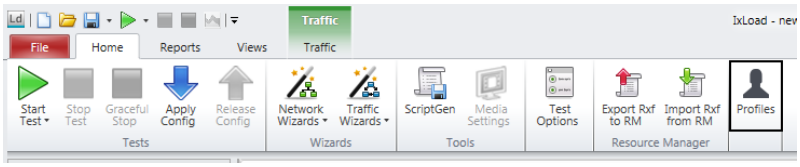


Figure 192. Profiles configuration

2. In the list of profiles, click Playlist, then click Add ().

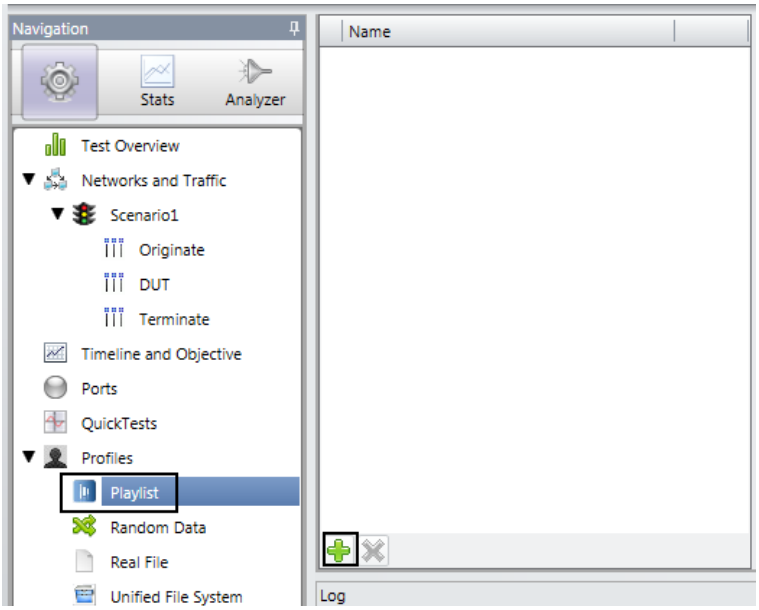


Figure 193. Adding a playlist

3. IxLoad adds a new empty playlist.

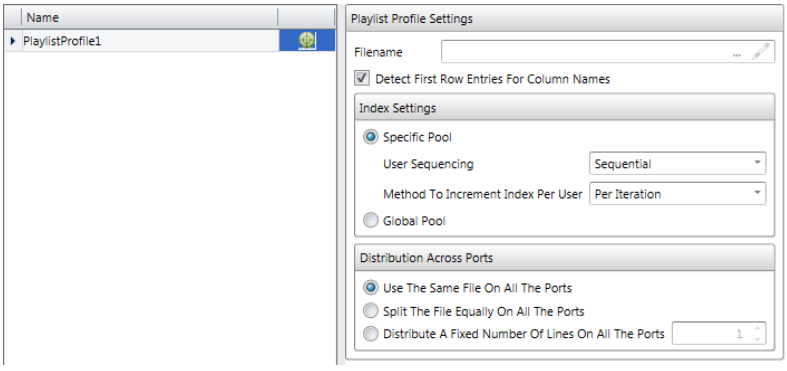


Figure 194. Playlist Profile Settings

Figure 195.

4. Rename the playlist if required.

Figure 196. Renaming a playlist

5. In the Filename field, click Browse, and select a CSV file, or enter the full path of the CSV file.

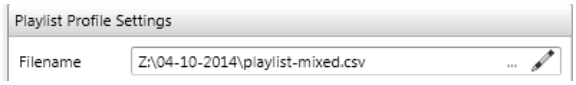


Figure 197. Adding a CSV file

6. If you want to display or edit the CSV file (for example, to make sure that it is the one that you want to use or to change it), click View/Edit CSV

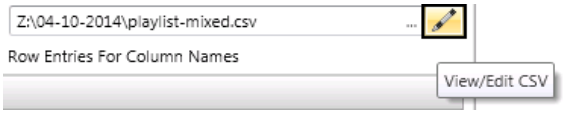


Figure 198. View/Edit a CSV file

7. If you want IxLoad to use the first row in the CSV file as the column headings, check Detect First Row Entries for Column Names. If you do not check this box, IxLoad uses generic column headings, such as Col1, Col2, etc.

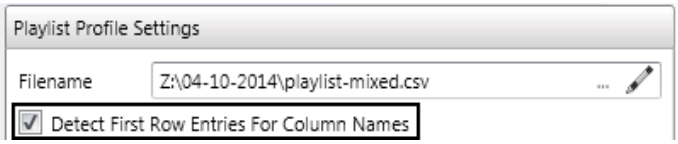


Figure 199. Using the first row in a CSV

8. In the Index Settings area, select the method used to select the entry in the playlist each user begins executing with:

    a. Specific Pool allows you to select a fixed, repeatable method for distributing the resources among the users. If you select this option, choose the method used to initially distribute the resources among the users (the User Sequencing parameter), and the method used to select the second and subsequent resources that are retrieved (the Method to Increment Index per User parameter).

    b. Global Pool causes the playlist to be accessed exactly in the order that it is laid out, from first entry to last, without regard for which user accesses an entry. Users will access the playlist as they come online, and as they complete their commands. Global Pool is the option to select if you want to ensure that the playlist is accessed in order, and you are not concerned about which user accesses a particular entry.

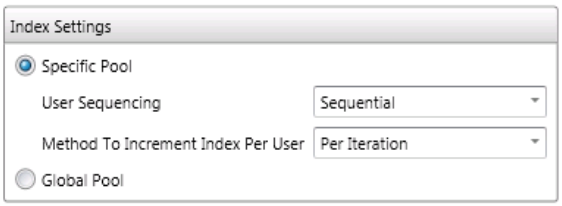


Figure 200. Index Settings

9. 'Distribution Across Ports' determines how the entries in the playlist are distributed among the ports in the test, if you run the client across more than one port in the test. You can choose from the following methods:

    a. Use the Same File on all the Ports: If you choose this option, the entire playlist is duplicated on each port. Each client that runs on a port accesses the same entries as the other clients running on all the other ports.

    b. Split the file Equally on all the Ports: If you choose this option, IxLoad divides the playlist equally among the ports, placing the first entry on the first port, then the second entry on the second port, the third entry on the third port, and so on. If the playlist cannot be divided evenly among the ports, the entries are distributed evenly among the ports using the same distribution method until there are no more entries left. Therefore, one or more ports will have fewer entries than the others.

    c. Distribute a Fixed Number of Lines on all the Ports: This option divides the playlist into chunks containing the number of entries you specified, and distributes one chunk to each port. Any leftover entries are ignored.

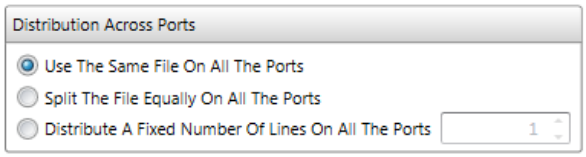


Figure 201. Selecting Distributions Across Ports

Contact Ixia

Corporate Headquarters
Ixia Worldwide Headquarters
26601 W. Agoura Rd.
Calabasas, CA 91302
USA
+1 877 FOR IXIA (877 367 4942)
+1 818 871 1800 (International)
(FAX) +1 818 871 1805
sales@ixiacom.com

Web site: www.ixiacom.com
General: info@ixiacom.com
Investor Relations: ir@ixiacom.com
Training: training@ixiacom.com
Support: support@ixiacom.com
+1 877 367 4942
+1 818 871 1800 Option 1 (outside USA)
online support form:
http://www.ixiacom.com/support/inquiry/

EMEA
Ixia Technologies Europe Limited
Clarion House, Norreys Drive
Maiden Head SL6 4FL
United Kingdom
+44 1628 408750
FAX +44 1628 639916
VAT No. GB502006125
salesemea@ixiacom.com

Renewals: renewals-emea@ixiacom.com
Support: support-emea@ixiacom.com
+44 1628 408750
online support form:
http://www.ixiacom.com/support/inquiry/?location=emea

Ixia Asia Pacific Headquarters
21 Serangoon North Avenue 5
#04-01
Singapore 5584864
+65.6332.0125
FAX +65.6332.0127
Support-Field-Asia-Pacific@ixiacom.com

Support: Support-Field-Asia-Pacific@ixiacom.com
+1 818 871 1800 (Option 1)
online support form:
http://www.ixiacom.com/support/inquiry/