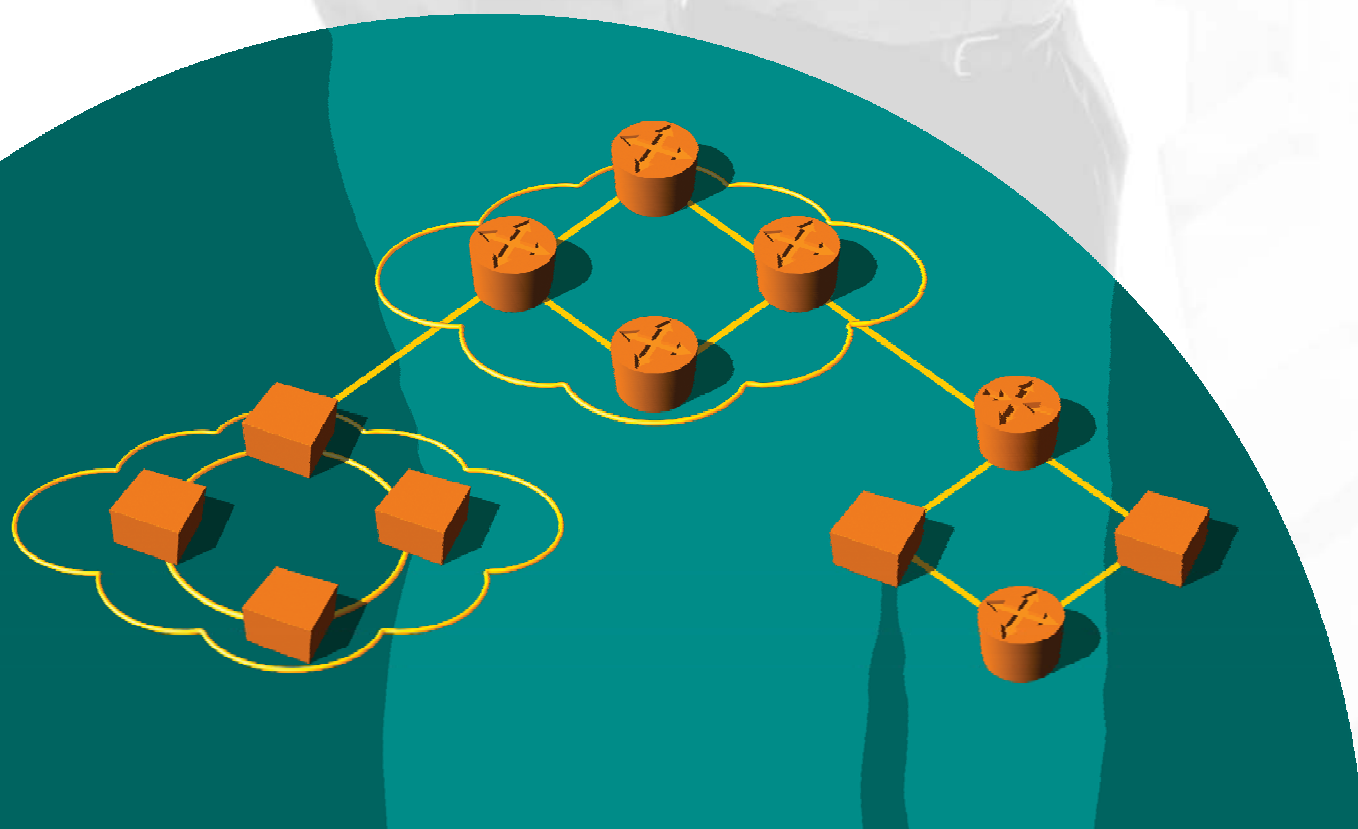


Spirent Communications Test Methodologies

Layer 4-7 Edition



Introduction

What makes Layer 4-7 testing so important?

Besides testing, only a few methods partially support decision-making for network deployment. But these methods — vendor marketing literature, word of mouth, and personal experiences — aren't enough.

Vendors typically publish performance results using best-case scenarios with little relevance to enterprise configurations. Poor test results can be tailored to create an impressive performance result. Then comes word of mouth, which offers varied success since someone else's advice may not be relevant to your installation. Word of mouth also can stem from ulterior motives that are not easily recognized.

Personal experience provides valuable ideas for effective design and deployment of a network. Even then, proper testing ensures performance objectives are met, unforeseen bottlenecks resolved and costs optimized.

As costs rise for technology and implementations, managers seek better returns on investment (ROI) and a reduction in the price of testing. If too much equipment is purchased, the ROI may never be realized. Underprovisioning is equally dangerous. Testing measures the actual performance of the deployment. It discovers bottlenecks, supports increased performance and ensures a company's goals are reached — high ROI, few errors and productivity enhancements.

Networks are proliferating worldwide. Demand for their nonstop functioning has gone beyond simple user expectation. Networks have become critical systems that make the difference between life and death. A 911 dispatch system, a hospital's medical information network and an air-traffic control network rely on uninterrupted traffic flow. Major outages are devastating. Minor outages, while not as dire, cause financial impact and dilute a hard-earned reputation.

Spirent Communications keeps networks functioning. Spirent is a worldwide provider of integrated performance analysis and service assurance systems for next generation network technologies. Spirent also is the leader in comprehensive test methodologies.

If you want to read Layer 4-7 methodologies online or view tests from other technologies, visit Spirent Communications Developers Network (SCDN) at <http://scdn.spirentcom.com>. After registering, you can download free scripts, share information about script development and receive help from a technical community comprising industry members the world over.

We hope you find the Layer 4-7 Edition useful in your daily work.

Spirent Communications

www.spirentcom.com

Table of Contents

Test Methodologies and Supporting Documents

Layer 4-7 Testing Prerequisites with Flowchart.....	1
Workload Validation (TM-0150).....	3
Maximum Connection Rate (TM-0151).....	5
Maximum Connections (TM-0152).....	7
Maximum Bandwidth (TM-0153).....	9
Maximum Simultaneous Users (TM-0154).....	11
Maximum Transaction Rate (TM-0155).....	13
Traffic Surges (TM-0156).....	15
Long-Term Stability (TM-0157).....	17
Failover (TM-0158).....	19

Appendix

Layer 4-7 Resource Data.....	22
Glossary.....	25
Spirent Communications Test Methodologies Information.....	26

Layer 4-7 Testing Prerequisites

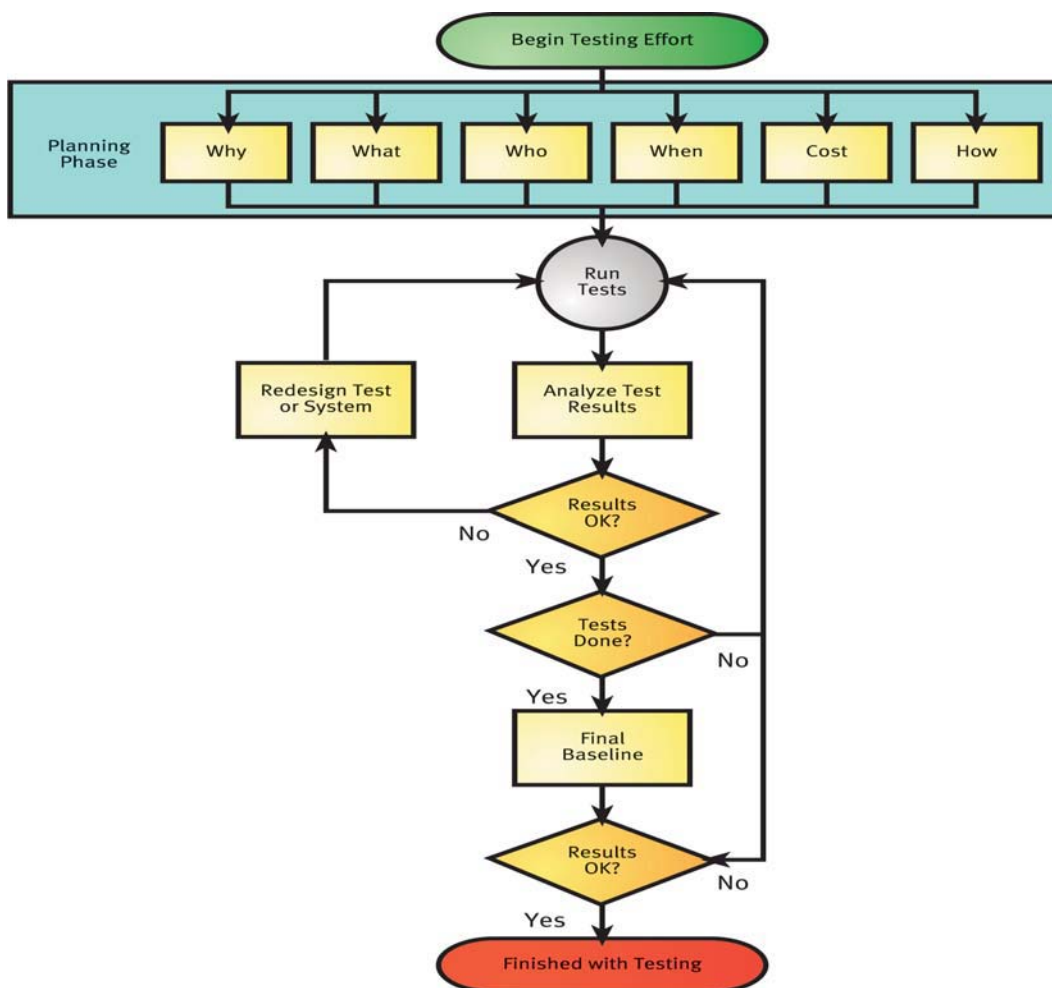
Methodology Assumptions

To gain the most benefit from Spirent's Layer 4-7 test methodologies, make sure you satisfy the following assumptions when using this journal:

- You have a good understanding of the network and the applications on the network.
- You understand how to maintain and configure the system(s) being tested.
- You have access to the system(s) before deployment in the live environment. Testing will cause many systems to fail, which is an important finding in itself.
- You have the ability to generate traffic that realistically models major protocols and applications on your network.
- Sufficient traffic can be generated to exceed peak traffic levels expected by your network.
- For device-in-the-middle testing, you have the ability to generate and respond to realistic traffic (client and server simulators) with sufficient volumes to adequately assess the device.

A Stepped Approach

A testing methodology is a series of steps. The flowchart below depicts suggested steps in a testing methodology for a system.



The flowchart shows a simplified set of steps in testing a system. After the initial planning phase, thorough testing involves a feedback process that improves the system under test with several iterative tests.

Plan Your Test Before Starting

A good methodology always starts with planning. As with any complex project, extra effort at the beginning pays off with improved results. Flowchart items to consider during planning include:

- **Why:** Document the goals of testing. Look for how they map into the overall business objectives and get buy-in from management and other involved parties. Determine the purpose of the system being tested. Document the contributions a testing methodology might have.
- **What:** Determine what should be tested. Look for tools to address the requirements and goals of the test. Consider what the expected results will be, along with what mitigating steps to take to achieve your expected results. Define the success of the testing and application functionality. In qualitative and quantitative terms, what are the levels of reliability, availability, scalability, security and performance?
- **Who:** Ensure enough people are available to conduct and support this assessment and that they have the proper skills and training. Obtain management agreement with the objectives of this testing (along with the benefits above) and the costs that will be incurred.
- **When:** Plan the amount of time that testing will take. Incorporate this time into the overall product delivery plan. Work with the development team to conduct tests early on and to have the group facilitate testing efforts through designs and features that enhance visibility during testing. This can include adding status codes, timers or counters to provide the testing team with valuable information during testing.
- **Cost:** Consider and document anticipated costs for testing, including expertise hired or trained, and tools and equipment acquired.
- **How:** Create a testing methodology that not only determines performance and locates bottlenecks but incorporates flexibility to evolve and respond to different factors and ideas.

Effective Testing Procedures

After the planning step, the most effective testing incorporates an iterative feedback process: Run the test, analyze the results, use the results to improve both the subsequent testing and/or the system(s) being tested, and run again. To test complex, poorly understood systems, a systematic approach of changing only one thing at a time between retests isolates particular issues and problems within the infrastructure.

For example, after a test, analysis of the results shows that the database's lack of indexing is causing performance issues: One could index the database, increase the memory and add extra hard drives before retesting. The more useful methodology involves testing three times, once after each change to quantify the improvements, if any, of making each particular change. After the iterations produce the desired results, conduct a final comprehensive performance analysis of the system before turning it live. This serves as a final "double-check," revealing any unforeseen issues while producing a final baseline result for future comparisons of performance changes.

The Protocol-Application Mix

A network exists to share information and serve particular functions such as Web browsing, an order entry system, e-mail, downloading music or networked game-play. These applications all use protocols that employ specific parts of the network. When considering which protocols/applications to focus on, look at the prominent protocols and the most important applications on your network. Certain applications may take up a large portion of network traffic but have little business value (e.g., peer-to-peer music file sharing).

Other applications take little network traffic yet are critical to the operation of the network and/or the business (e.g., payroll). Careful consideration of these situations should create a protocol and application mix that realistically assesses the performance and stability of the system. This step also assesses the system's handling of network traffic mix that is vital to the business.

After choosing the protocol(s), consider the mix of traffic that makes sense for the system and network(s). For HTTP, it may be enough to include HTTP 1.0 and HTTP 1.1, with a few pages of differing sizes and content. If the system works with SSL, ensure testing incorporates SSL. Results with SSL will probably look different than results without it. (Some systems pass SSL unaltered, like most firewalls and load balancers.) Streaming protocols usually have longer lasting connections, with connections at differing speeds. FTP may have persistent connections with long periods of activity and inactivity.

Workload Validation

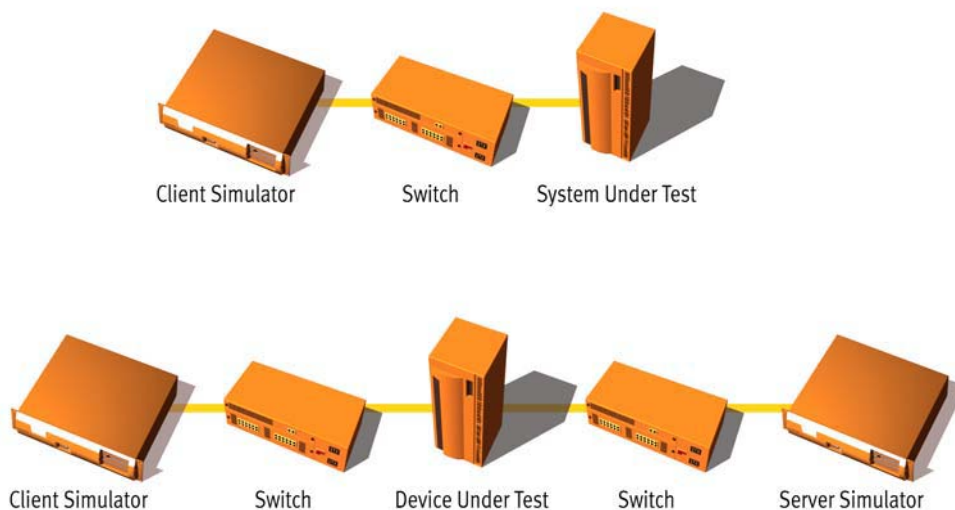
Objective

This test validates workloads, ensuring proper functionality with minimal loads. This is not a performance test, but every other methodology in this journal uses this procedure to validate its particular workload.

Overview

This procedure ensures a candidate workload functions correctly. It also assists with discovery of connectivity issues, syntax errors and other problems before the high-load test is run. Use this validation as a first step before the upcoming test methodologies in this journal.

Setup Diagrams



Methodology Procedure

1. Determine the workload to use based on suggestions in the Appendix (Pgs. 22-23) and/or the methodology being run.
2. Configure the tool to ramp up traffic to 1 user and hold steady for 240 seconds.
3. If the testing tool supports this functionality, have it validate the test first and check for any errors. Some test tools generate a PCAP file during validation, making network troubleshooting much easier. Resources are available on the Web for loading and interpreting PCAP files.
4. Run or validate the test.
5. Use the tool's real-time reporting interface to look for errors that occur during the test.
6. After the test, analyze the detailed statistics to locate errors and system issues.
7. Note application-level errors and connectivity errors. For example, this test may show a connection error exists in the test network. It may also find typographical or unintentional errors in the workload (HTTP 404 — page not found errors).
8. Correct or resolve any errors encountered.
9. Rerun the test until no other unexpected errors occur.

Test Parameters

- Test specification: Users.
- 1 User (per protocol if applicable).
- Type of workload: Depends on test to be run.

Issues To Look For

- Application level errors.
- Application timeouts.
- Unexpected errors, such as page not found, wrong page returned.
- Network level errors.
- Network timeouts.
- System configuration errors.
- Unexpectedly long response times.

Maximum Connection Rate

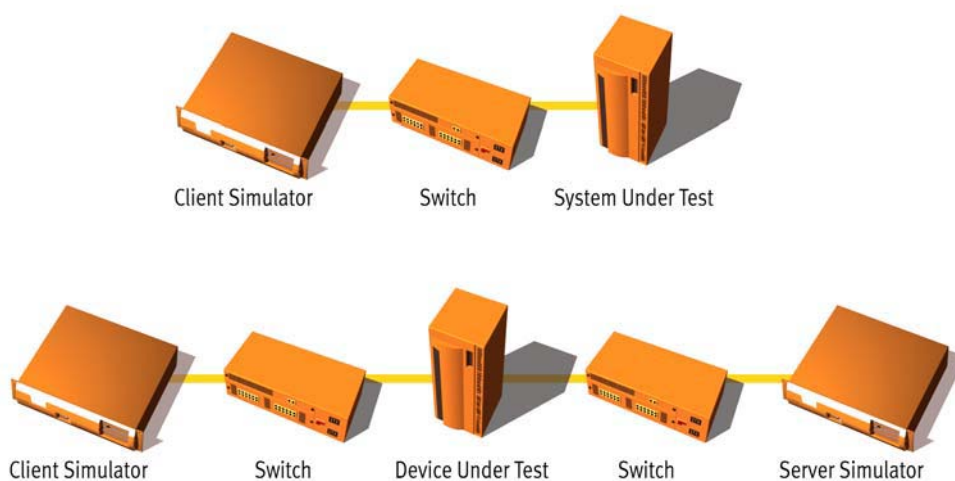
Objective

This test establishes the maximum number of connections per second the system will successfully handle.

Overview

Knowing a system's effective connections/sec performance provides an understanding of the rate of traffic the system can maximally handle. This information also indicates potential system bottlenecks.

Setup Diagrams



Methodology Procedure

1. If possible, obtain the expected handling rate of the system using the system's specifications, denoted here by ExpMaxCR. If ExpMaxCR is not known, start with a "guesstimate" for ExpMaxCR and pay attention during the run in steps 6 and 7 to see if the system reaches maximum capacity. If not, double the first guess for ExpMaxCR and run again.
2. Use a successful workload determined in the workload validation methodology (Pgs. 3-4).
3. Configure the test tool with relatively large step heights of (ExpMaxCR divided by 10). Ramp up each step over a ramp time of 15-30 seconds to ensure the system has the best chance of keeping up with the new load.
4. Configure enough steps to surpass ExpMaxCR by at least 50 percent, about 15-20 steps.
5. Step steady time for each step should be at least 120-240 seconds to maximize the chances the system under test will reach steady state.
6. Run the test.
7. During the test, ensure the system reaches "steady state" during the earlier steps where the traffic has not exceeded ExpMaxCR. If it does not reach steady state, rerun the test with a longer step steady time.
8. After this first test, find the level at which the system cannot keep up with the traffic and errors are encountered, which we'll denote as MacroMaxCR.

9. Using the level MacroMaxCR (determined in the previous step as the new median to ramp to), configure a new test with smaller incremental steps. Configure a starting ramp up to (MacroMaxCR minus 20%) over 120 seconds and hold steady for 240 seconds.
10. Configure ramp heights of (MacroMaxCR divided by 100) with a step steady time of 60 seconds each, for a total of 40 steps and a ramp time of 10 seconds.
11. Run the test with the new parameters.
12. Determine the maximum sustainable connections per second by finding the level at which traffic continues without errors.

Test Parameters

- Load Specification: Connections per second.
- Use a workload with small file sizes, about 1 kilobyte. During the test, the system under test should encounter a high rate of connections opening and closing, with fast retrievals of small objects. For HTTP, use HTTP 1.0 with no keep-alive to maximize connection setups and tear-downs. A small file reduces the chance of encountering bandwidth limitation.

Issues To Look For

- Error conditions as listed in the Appendix, “Layer 4-7 Resource Data,” section entitled Things to Look for During System Failures (Pg. 24).
- Due to application-level timeouts and TCP retransmissions, the actual load at failure is lower than what may be indicated. Look for signs the system has reached steady state when each step is taken.

Maximum Connections

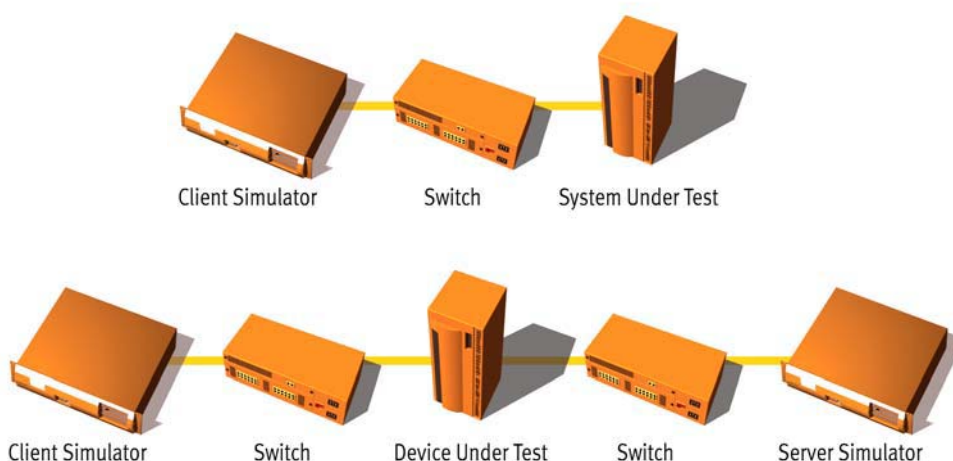
Objective

This test establishes the maximum number of connections the system will successfully handle.

Overview

Having the maximum connections for the system provides an understanding of the total number of network connections the system should handle. It also shows whether the system has a bottleneck.

Setup Diagrams



Methodology Procedure

1. Validate the candidate workload using steps outlined in the workload validation methodology (Pgs. 3-4).
2. If the system under test has an expected maximum simultaneous connections, use that number as a guide, denoted here as ExpMaxConn. If ExpMaxConn is not available, take a "guesstimate" for ExpMaxConn and continue with this methodology. During steps 5 and 6, check whether the system reaches its capacity. If not, double the initial guess for ExpMaxConn and try again.
3. Configure a test to ramp up connections with step heights of (ExpMaxConn divided by 10) for 20 steps. Each step ramp time should be 15-30 seconds.
4. Configure the tool to hold each step for a sufficient amount of time to reach steady state, about 120-240 seconds. At minimum, the step steady time should exceed the think time of the users.
5. Run the test.
6. During the test, look for errors and other failure conditions to appear.
7. Analyze the post-run statistics for behaviors and conditions showing the system is running out of connection resources and obtain the corresponding open connections, denoted as MacroMaxConn.
8. Configure another test to ramp up to (MacroMaxConn minus 20%) over a span of 120 seconds and hold steady for 240 seconds.
9. After the ramp up, configure the tool with step heights of (MacroMaxConn divided by 100) over 40 steps, holding each step for 60 seconds.

10. Run the test with the new parameters.
11. Determine the maximum simultaneous connections by finding the level at which traffic continues without errors.

Test Parameters

- Load Specification: Connections.
- Be sure the connections/sec does not exceed the rate determined in the methodology for maximum connection rate (Pgs. 5-6).
- Method 1:
 - i. Configure the users to abort (click-away) any transaction that does not return valid traffic after 15 seconds.
 - ii. Use, small objects, many persistent connections, and long user think times. For example, use HTTP 1.1 with persistence (on both clients and servers) and a 60-second think time. In this scenario, the simulated user will retrieve an object, hold the connection open for 60 seconds, retrieve the second object and close the connection. This ensures many connections are created and kept open for long periods.
- Method 2:

Configure the server simulator for a long response latency, about 60 seconds. This causes clients to establish a connection, keeping it open while waiting for the response.

Issues To Look For

Pay attention to the results to find timeout errors. Given the nature of this test, do not overlook the possibility that connections, while being kept open, may be stalled. They will pass no useful traffic. This is the reason to have the test tool close the connection after the second object retrieval, so that any stalled connections can be closed out and errors logged.

Maximum Bandwidth

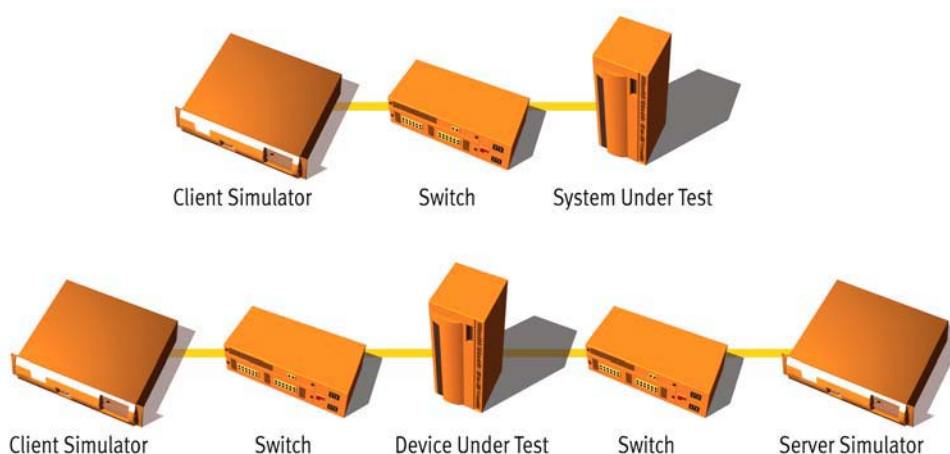
Objective

This test establishes the maximum bandwidth the system will successfully handle.

Overview

Knowing the maximum system bandwidth is vital for understanding how much bandwidth the system should handle. It also indicates whether the system contains a bottleneck. For deployments in which bandwidth and throughput are critical, it is important to be confident that no performance problems will occur.

Setup Diagrams



Methodology Procedure

1. Obtain the expected bandwidth capacity of the system from datasheets, denoted here as ExpBnd. If unavailable, make a “guesstimate” for ExpBnd.
2. Validate the candidate workload using steps outlined in the workload validation methodology (Pgs. 3-4).
3. Configure the test tool to ramp up the bandwidth in heights of (ExpBnd divided by 20), holding each step steady for 120 seconds, for 30 steps.
4. Run the test.
5. During the test, check that steady state is achieved at each step.
6. If a guesstimate was used for ExpBnd, check if bandwidth utilization tops out. If not, double ExpBnd and start over from Step 4.
7. Using the post-test results analysis, obtain the maximum bandwidth, denoted here by MacroEffBnd.
8. Configure another test to ramp up bandwidth to (MacroEffBnd minus 20%) over a span of 120 seconds and hold steady for 240 seconds.
9. After that, increase the bandwidth in steps of (MacroEffBnd divided by 100) for 40 more steps, holding each step for 60 seconds.
10. Run the test with the new parameters.
11. Analyze the post-test results to obtain the maximum effective bandwidth.

Test Parameters

- Load Specification: Bandwidth.
- Be sure the connections/sec does not exceed the rate determined in the methodology for the maximum connection rate (Pgs. 5-6).
- Be sure the total number of connections does not exceed the value determined in the methodology for maximum connections (Pgs. 7-8).
- Ensure the test tool and connectivity devices used in the test do not become bottlenecks. For example, be sure the switch fabric has enough capacity to exceed the bandwidth expected during the test.
- The average size of an HTTP transaction is 8-13 kilobytes. For tools that support dynamic file sizes during a test, use a test that increases the file size, thereby increasing bandwidth utilization. If not, have the tool increase load using connections/sec with file sizes between 8K and 13K. If the tool cannot create enough bandwidth with these average file sizes, then increase the file sizes.
- If raw maximum bandwidth is the goal of this test, have the server simulator return a large file size as its response, helping to maximize bandwidth with fewer requests.

Issues To Look For

As bandwidth utilization increases, the resources of the system under test will decrease. Eventually, errors and timeouts will occur when system resources become scarce.

Maximum Simultaneous Users

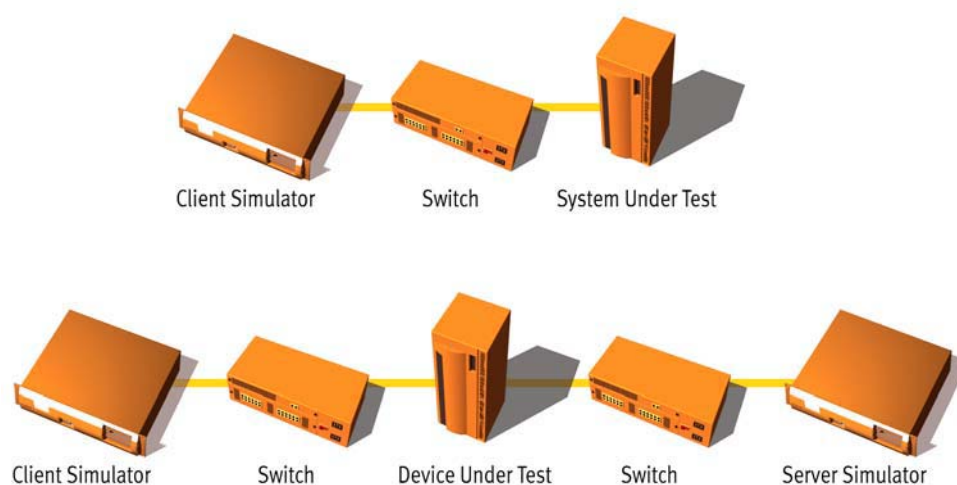
Objective

This test establishes the maximum number of users the system will successfully handle.

Overview

Having the maximum users for the system provides an understanding of the total number of users the system should handle and also shows whether the system has a bottleneck.

Setup Diagrams



Methodology Procedure

1. Validate the candidate workload using steps outlined in the workload validation methodology (Pgs. 3-4).
2. If the system under test has an expected maximum simultaneous users, use that number as a guide, denoted here as ExpMaxU. If ExpMaxU is not available, take a "guesstimate" for ExpMaxU and continue with this methodology. If maximum number of connections is available, divide that number in half to use as the guesstimate. During Step 6, check if the system hits capacity. If not, double the initial value for ExpMaxU and try again.
3. Configure a test to ramp up connections with step heights of (ExpMaxU divided by 10) for 20 steps. Each step ramp time should be 15-30 seconds.
4. Configure the tool to hold each step steady for a sufficient amount of time to reach steady state, about 120 seconds. At minimum, the step steady time should exceed the think time of the users.
5. Run the test.
6. During the test, look for errors and other failure conditions to appear.
7. Analyze the post-run statistics for behaviors and conditions that show the system is running out of connection resources. Obtain the corresponding simultaneous users denoted as MacroMaxU.
8. Configure another test to ramp up to (MacroMaxU minus 20%) over a span of 120 seconds and hold for 240 seconds.

9. After the initial ramp up, configure the tool with step heights of (MacroMaxU divided by 100) over 40 steps, holding each step for 60 seconds.
10. Run the test with the new parameters.
11. Determine the maximum simultaneous users by finding the level at which traffic continues without errors.

Test Parameters

- Load Specification: Users.
- Be sure the connections/sec does not exceed the rate determined in the methodology described in the maximum connection rate test (Pgs. 5-6).
- Do not allow the maximum connections to exceed the amount determined in the maximum connections test (Pgs. 7-8).
- Configure the users to abort (click-away) any transaction that does not return valid traffic after 15 seconds.
- Use, small objects, many persistent connections, and long user think times. For example, use HTTP 1.1 with persistence (on both clients and servers) and a 30-second think time. In this scenario, the simulated user will retrieve an object, hold the connection open for 30 seconds, retrieve the second object and close the connection. This ensures many users are created and remain for long periods.

Issues To Look For

Review results to inspect timeout errors. Given the nature of this test, observe whether the connections, while being kept open, are stalled. If so, they will pass no useful traffic. This is the reason to have the test tool close the connection after the second object retrieval, so that connections can be closed out and errors logged.

Maximum Transaction Rate

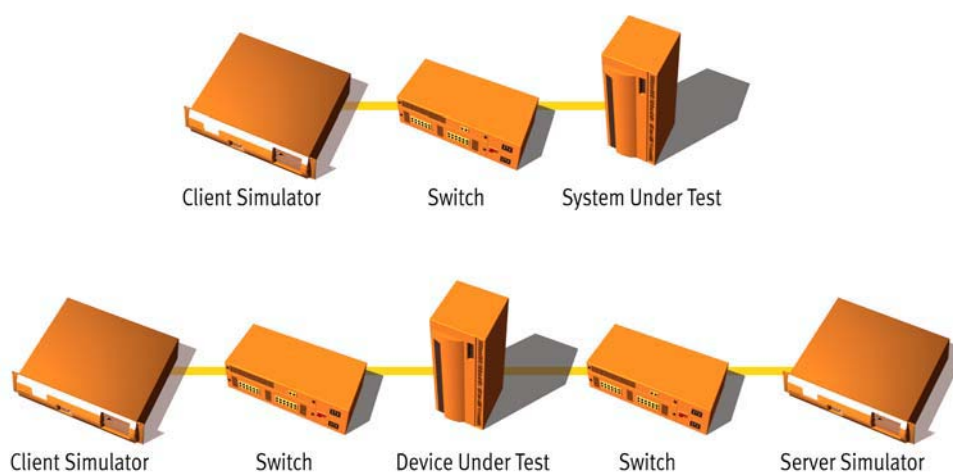
Objective

This test finds the maximum transaction rate that the system under test will successfully handle.

Overview

Transactions, known as “hits,” define the basic unit of performance for most Web-based systems. This number allows us to understand overall system capacity and supports a determination of whether the system has a bottleneck.

Setup Diagrams



Methodology Procedure

1. Validate the candidate workload using steps outlined in the workload validation methodology (Pgs. 3-4).
2. Take the maximum connection rate determined earlier (Pgs. 5-6) and double it to get the initial expected maximum transaction rate, denoted here as ExpMaxTPS.
3. Configure a test to ramp up transactions/sec with step heights of (ExpMaxTPS divided by 10) for 20 steps. Each step ramp time should be 15-30 seconds.
4. Configure the tool to hold each step steady for a sufficient amount of time to reach steady state, about 60 seconds.
5. Run the test.
6. During the test, look for errors and other failure conditions to appear.
7. Analyze the post-run statistics for behaviors and conditions that show the system is running out of resources. Obtain the corresponding successful transaction rate denoted as MacroMaxTPS.
8. Configure another test to ramp up to (MacroMaxTPS minus 20%) over a span of 120 seconds and hold for 240 seconds.
9. After the initial ramp up, configure tool with step heights of (MacroMaxTPS divided by 100) over 40 steps, holding each step for 30 seconds.
10. Run the test with the new parameters.
11. Determine the maximum transaction rate by finding the level at which traffic continues without errors.

Test Parameters

- Load Specification: Transactions per second.
- Do not allow the maximum connections to exceed the amount determined in the methodology for maximum connections test (Pgs. 7-8).
- Use small file sizes to ensure that bandwidth does not become a limiting factor. A 1 KB file size is reasonable. With HTTP, use HTTP 1.1 with persistence and have each simulated user retrieve many files (e.g., 16 files) before closing out the session to minimize overhead from connection setup and teardown. If HTTP 1.0 with no keep-alive is used, transaction rate will equal the connection rate determined in the steps of the maximum connection rate test (Pgs. 5-6).

Issues To Look For

- Error conditions listed in the Appendix, “Layer 4-7 Resource Data,” section entitled Things to Look for During System Failures (Pg. 24).
- Due to application-level timeouts and TCP retransmissions, the actual load at failure is lower than what may be indicated. For this reason, look for signs the system has reached steady state when each step is taken.

Traffic Surges

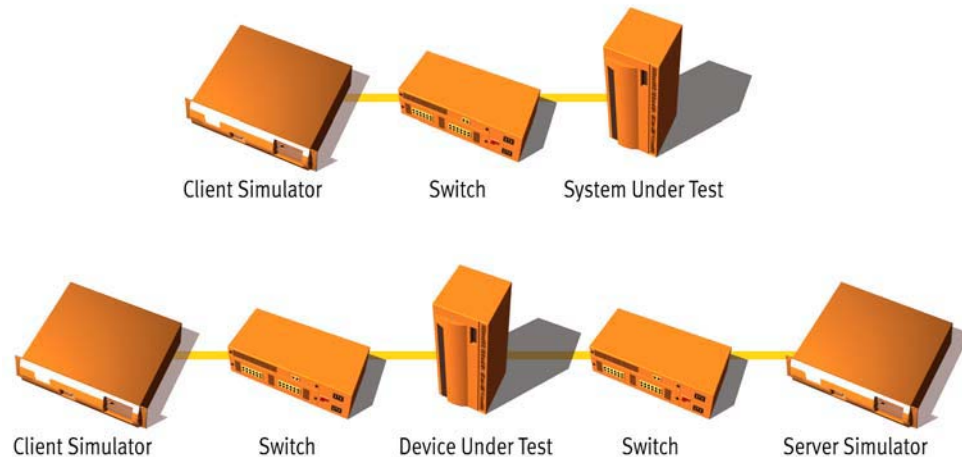
Objective

This test determines a system's ability to handle large surges in traffic.

Overview

Systems often encounter large traffic jumps, planned or unplanned. Examples include surges from a television advertisement, tax-filing deadlines and last-minute holiday purchases. These surges often occur when outages are extremely costly. During a last-minute holiday purchase rush, any outage will cost more because more products are being purchased and the likelihood of customer defection is higher. This test determines the system under test's ability to manage and deal with such traffic surges.

Setup Diagrams



Methodology Procedure

1. Validate the candidate workload using steps outlined in the workload validation methodology (Pgs. 3-4).
2. Choose load specification to apply and use the maximum performance obtained in previous methodologies, denoted here by ExpMaxPerf.
3. Based on think times, download times and other performance issues, determine the maximum time a workload will take to complete, denoted here by WorkTime.
4. Configure the test tool to ramp up the load in heights of (ExpMaxPerf divided by 50), holding each step steady for (WorkTime times 2) seconds, for 50 steps, and ramp times of 15 seconds.
5. Run the test.
6. Analyze the results, looking for performance issues and bottlenecks.
7. Resolve any troublesome issues and repeat this test.

Test Parameters

- Load Specification – System dependent: choose the specification that matters most.
- Be sure the connections/sec does not exceed the rate determined in the maximum connection rate test (Pgs. 5-6).
- Be sure the total number of connections does not exceed the value determined during the maximum connections test (Pgs. 7-8).
- Keep bandwidth utilization below the amount determined in the maximum bandwidth test methodology (Pgs. 9-10).
- Use a workload representing a production network. For more information see Appendix, “Layer 4-7 Resource Data,” section entitled How to Select Loads (Pgs. 22-23).

Issues To Look For

- Unlike other tests, steady state may never be reached during each step, especially with complex workloads. Determine behavior under surge conditions, not steady-state conditions.
- Timeouts, large numbers of open connections and unexpected errors usually indicate an overloaded system.
- The system under test may never complete this test without any errors. To decide whether this matters, consider how critical the system is along with its prevalence of surges, nature of failures (catastrophic or benign feature degradation) and budget constraints.

Long-Term Stability

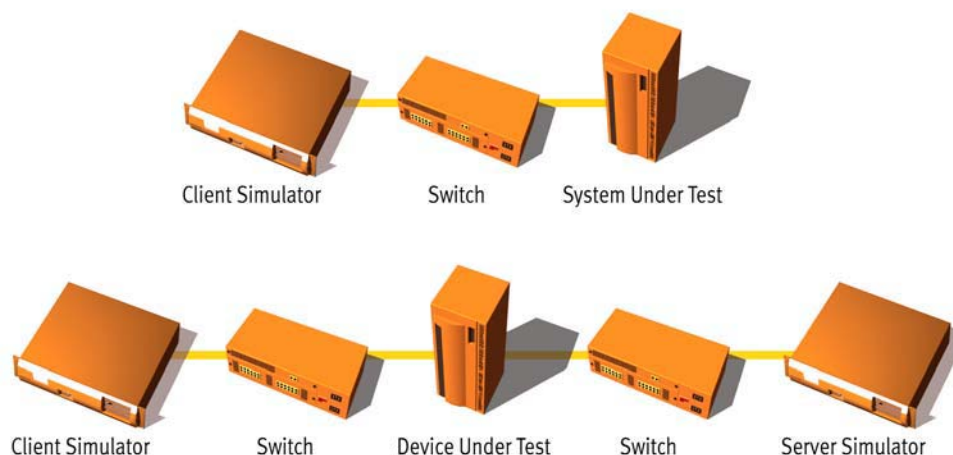
Objective

This test determines the ability of the system to function for a long duration while minimizing the prevalence of memory leaks that cause system failures.

Overview

Systems that function well in short-term tests may still develop problems during long-term operation. The goal of this test is to determine the system's ability to function for long periods of time under high traffic conditions. System availability is a critical measure in planning a network, as the fastest network means little if it constantly fails. This test suggests using a workload that resembles production traffic, but it increases traffic to make any stability issues show up earlier.

Setup Diagrams



Methodology Procedure

1. If required, validate the candidate workload using steps outlined in the workload validation methodology (Pgs. 3-4).
2. Determine the maximum performance of the system under the chosen workload by following the steps outlined in the methodology for maximum connections (Pgs. 7-8). Denote this maximum performance by MaxConn.
3. To keep the system running near the maximum, configure the test to ramp up over 600 seconds to a height of (MaxConn minus 5%). Use a large value for the step steady, several days or weeks if time permits, depending on the available time to run this test and on the minimum acceptable uptime.
4. Run the test.
5. At the beginning of the test, check that unexpected errors do not appear.
6. During the test, periodically check on the health of the system under test. This could include checking on system monitors (CPU utilization, memory utilization, thread pool, etc.) Along with the system under test's own monitoring tools, use the test tool's real-time reporting for help in determining system health. Ensure traffic continues to flow and errors are not growing rapidly.
7. Run the test until the system under test fails or the test ends.

- Analyze the results to determine whether system performance and behavior fit within acceptable parameters.

Test Parameters

- Load Specification: Connections.
- Be sure the connections/sec does not exceed the rate determined during the methodology outlined in the maximum connection rate test (Pgs. 5-6).
- Be sure the total number of connections does not exceed the value determined in the maximum connections test (Pgs. 7-8).
- Keep bandwidth utilization below the amount determined in the maximum bandwidth test (Pgs. 9-10).
- Use a workload that resembles a production network's traffic. For more information, see Appendix, "Layer 4-7 Resource Data," section entitled How to Select Loads (Pgs. 22-23).
- Alternatively, use the workload from the maximum connections test (Pgs. 7-8), which exercises the memory portion of the system under test better than most other tests.

Issues To Look For

- Timeouts, large numbers of open connections and unexpected errors usually indicate an overloaded system.
- For systems that support it, monitor the memory utilization. An increasing amount of memory utilization, while under steady load, usually indicates memory leaks that lead to system failure.

Failover

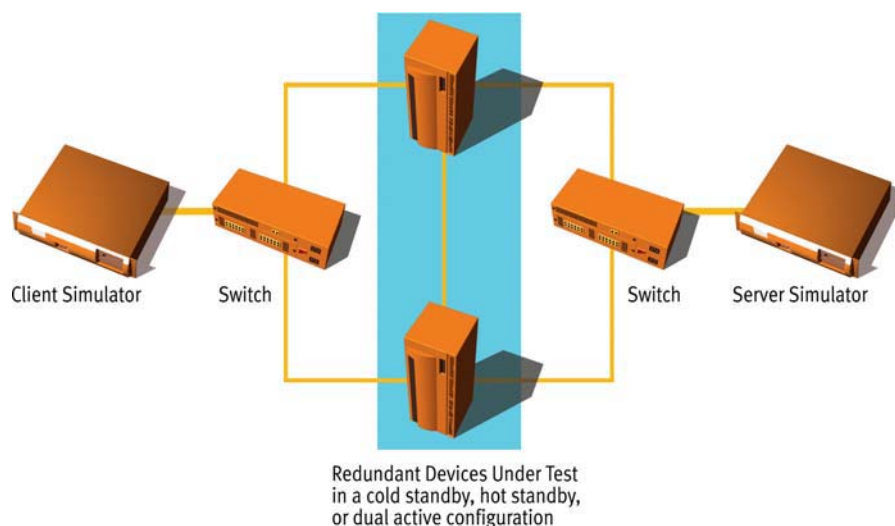
Objective

This test determines the ability of the system to properly and adequately failover.

Overview

Many network systems use high-availability architectures that have redundant systems. These systems fall into three categories in increasing level of failover protection: cold standby, warm standby and dual-active. The best failover schemes completely preserve network activity and transactions, and they are fully undetectable by users. Testing failover without network traffic does little to ensure the network remains viable and users are minimally affected during and after system failover.

Setup Diagram



Methodology Procedure

1. If needed, validate the candidate workload using steps outlined in the workload validation methodology (Pgs 3-4).
2. Determine the expected traffic level of the system, denoted here as ExpTrfc. If ExpTrfc cannot be determined, divide the maximum simultaneous users obtained in the earlier test (Pg. 11-12) by 4 to achieve a baseline level of traffic.
3. Confirm the redundant system under test will properly failover with no traffic. This can be done by confirming the failover settings and initiating a "failure" (e.g., pulling the cable that connects the system to the network).
4. Configure the test tool to ramp up to ExpTrfc over 240 seconds and hold this amount steady for 600 seconds.
5. Start the test and wait for the 240-second ramp up period to complete.
6. Check that the traffic reaches steady state and that there are no failures.
7. Take note of the test time and initiate a failover. Usually, this is as simple as disconnecting the system from the network.
8. Let the test continue to run to completion.

9. Analyze the test results to determine the effect of the failover. The time noted in Step 7 helps guide where/when to delve more deeply into the results.
10. Resolve failover issues that have a fix or workaround, retesting to see if the new fixes improve the failover results.

Test Parameters

- Load Specification: Users.
- Be sure the connections/sec does not exceed the rate determined in the methodology for maximum connection rate (Pgs. 5-6).
- Be sure the total number of connections does not exceed the value determined during the methodology for the maximum connections test (Pgs. 7-8).
- Keep the traffic under the level of simultaneous users found in the methodology for maximum simultaneous users test (Pgs. 11-12).
- Keep bandwidth utilization below the amount determined during testing for maximum bandwidth (Pgs. 9-10).
- If possible, use a workload that resembles a production network's traffic. For more information, see Appendix, "Layer 4-7 Resource Data," section entitled How to Select Loads (Pgs 22-23).
- If the network will support stateful user applications (e.g., Web-site shopping carts, airfare searches, long FTP downloads), be sure the workload incorporates these behaviors so they can be tracked.
- Alternatively, use the workload from the maximum simultaneous users test, which maintains user connections long enough to determine whether they are affected.

Issues To Look For

- Application level errors, network timeouts and network retransmits indicate failovers that may cause user-detectable errors.
- The most troublesome issues are application-level errors in which user state is not maintained. This may cause a user to lose all the contents of the shopping cart.

Appendix

Layer 4-7 Resource Data

The Layer 4-7 Resource Data section contains supporting information for the test methodologies. The main topics are How to Select Loads, Test-Phase Steps, Things to Look for During System Failure, and Statistical Signs of Overloading and Failure.

How to Select Loads

Network and computing devices have particular values for performance. For example, a Web-site operator might want to know how many simultaneous users the site handles during peaks. A fire-wall manufacturer will want to determine the maximum bandwidth its device can handle.

When testing, the test tool should generate traffic that focuses on a metric, such as generating load to see how many users the system under test can effectively handle. Applying the right load ensures the system being tested is assessed in the proper way, and that the test produces results that matter.

Users as a Metric

The user-handling capacity of a system is an important metric. Networks ultimately serve the needs of users, so knowing that a network will properly handle the desired level of user traffic is critical. When testing, a user is the simulation of a person using the network.

For example, a user could be configured to browse, search for and purchase items from a Web site using Netscape Navigator. At the same time, the user could read e-mail, download MP3 songs and catch up on the news using streaming media. Or, a user could be configured to only load one HTML page and leave the Web site.

Two pertinent metrics exist for users: (1) maximum simultaneous users, and (2) maximum sustainable user arrival rate (users per second). Systems that normally focus on users as a metric include Web caches, Web servers and server load balancers.

Network Connections as a Metric

A connection is a channel between two systems that transfer information. Depending on the system, protocol and applications, network connections can be short-lived or long, persistent connections. The HTTP protocol has two versions: HTTP 1.0 and HTTP 1.1, with the major difference being the persistence. HTTP 1.0 was modified to have a persistent mode called keep-alive, but the use of keep-alive is less prevalent.

HTTP 1.1 can initiate a persistent connection between client and server, using the same connection to transfer multiple objects. This reduces overhead devoted to connection setup and teardown. Do not confuse the protocol-level persistence of HTTP 1.1 with application-level persistence, which uses cookies and session IDs to maintain user persistence. Other potentially persistent protocol-level connections include FTP, SMTP and POP3.

A standard connection does not exist with most streaming protocols, DNS nor SNMP, which all run over the connectionless UDP protocol. In this case, the source sends its message with the hope, but no guarantee, that the message will reach its destination.

As with users, two metrics pertain to connections: (1) maximum simultaneous connections, and (2) maximum connections per second. Firewalls and intrusion detection systems use maximum simultaneous connections and maximum connections per second as key metrics.

Internet Transactions as a Metric

A transaction is the initiation and successful completion of a communication request. Transactions include downloading one MP3 file using FTP, sending one e-mail with SMTP or downloading one graphic using HTTP. Browsing the Web and e-mail (when a user sends/receives dozens of messages) usually involve several transactions.

In browsing, a Web page comprises several objects: the HTML page and objects loaded with that page. This includes graphics, java applets, cascading style sheets and Flash animations. While using this journal, consider every object that loads to be a single transaction. On the World Wide Web, this definition equates to a hit.

Two major metrics exist for transactions: (1) maximum total transactions and (2) transactions per second. SSL accelerators, Web sites and Web caches employ these metrics most often.

Page Views as a Traffic Reporting Metric

Web sites now use page views as the major traffic reporting metric, replacing the earlier practice of using hits to denote traffic encountered by the site. This makes sense because it not only matches how the average user experiences a site, but it more accurately portrays traffic on a site for advertising purposes. The page metric is maximum pages per second.

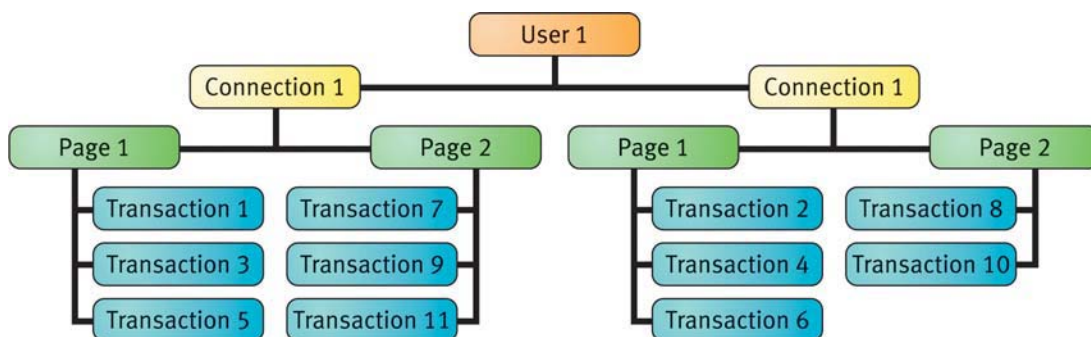
Bandwidth as a Metric

Depending on the network type, a system's ability to handle bandwidth is important in capacity planning decisions. The goal is to ensure a system can handle the needed throughput for a network (such as a firewall adequately handling traffic for a T3 connection to the Internet). Bandwidth usually matters more in lower-level network systems such as routers and switches. When working with more intelligent network systems such as stateful firewalls, Web sites, intrusion detection systems and content switches, bandwidth is usually a by-product of other applied loads. A greater number of transactions or users requires greater bandwidth utilization.

During testing, look for bandwidth utilization while applying one of the above loads and ensure it stays below the bandwidth amounts available in the production network. For example, if bandwidth to the Internet is a T1, make sure during testing that the bandwidth with realistic loads remains under 1.544 Mbps. Finally, the goals of the system/application and of the testing itself determine whether an applied amount of bandwidth becomes an important consideration compared with other metrics discussed in this section.

Relationships between Metrics

Metrics are loosely related to one another. The relationship depends on the protocol(s) and settings used on the system and users' behavior. Take as an example the HTTP protocol. The only time users, connections, transactions and pages are equal is when each user downloads only one HTML page, with no graphics or other objects, and then the user closes out the session. More typically, a single user activates two or more connections to download several pages, generating many transactions per page. Bandwidth utilization increases in proportion to other metrics as well.



A typical relationship between users, connections, pages and transactions on a Web site.

Test-Phase Steps

Some Layer 4-7 test tools automatically (or through scripting) generate progressively larger amounts of load. This automation makes testing easier and more effective, as the tool slowly increases loads by certain amounts, holds for a time and increases to the next load. When the load is held, the user can inspect the system under test for bottlenecks or potential issues.

The tool can be configured to continue increasing load until the system under test reaches its limit. Test-phase steps consist of the following:

- Ramp up — The beginning of load generation when traffic is progressively and linearly increased to reach a particular load. This ramp up phase allows the system(s) being tested to "catch up" (e.g., allocate resources such as memory, connections) with the incoming traffic to ensure the system under test does not experience a sudden jump in traffic that causes it to fail or go into DoS defense mode. This ramp up also allows traffic to reach a certain initial value before continuing with more traffic.

- Step steady time — Amount of time to maintain a certain level of traffic after ramp up or a step.
- Step ramp time — The amount of time to take to increase to the next level of traffic.
- Step height — An increase in traffic to a new level.

Things to Look for During System Failures

Look for the following behaviors on an overloaded system:

- Timeouts for all new incoming connections.
- Resets of incoming connections and while continuing to process current connections.
- Accepts and holds open incoming connections but refuses to forward traffic.
- A complete lockup/failure.

Statistical Signs of Overloading and Failure

System overloads and failures may be indicated by the statistics:

- A topping out of open connections.
- A drop in open connections.
- Increase in time to TCP SYN/ACK.
- Increase in overall response times.
- Topping out of bandwidth.
- Decrease in bandwidth.

Glossary

DDoS – Distributed Denial of Service, defining a subset of network attacks (from the more general denial of service) that seek to prevent legitimate users from accessing a network service. This occurs by leveraging known exploits that cause the systems to misbehave or fail and/or through the sheer overflowing of a network with requests that consume bandwidth and system resources. DDoS attacks combine the resources of many systems to generate traffic that exceeds the capacity of the targeted system(s).

DNS – Domain Name System used to translate common Internet names such as <www.spirentcom.com> to an IP address that network systems can properly work with and forward.

FTP – File Transfer Protocol for transferring large files over a TCP/IP connection.

HTML – HyperText Markup Language used on networks for defining the layout, design, content and functionality for a page displayed within a Web browser.

HTTP – HyperText Transfer Protocol, widely used on the Internet to access and share information, content and interactions between clients and Web servers.

PCAP – Packet CAPture, a file generated by tools configured to capture network conversations. Detailed information contained within this file is used for troubleshooting, ensuring each part of the network behaves properly.

POP3 – Post Office Protocol Version 3, commonly used to access e-mail from an e-mail server.

SMTP – Simple Mail Transfer Protocol for sending e-mails from one system to another.

SNMP – Simple Network Management Protocol, a prevalent standard for network management and monitoring.

SSL – Secure Sockets Layer, used by Web browsers to increase the security of network communications, especially for personal and financial data.

UDP – User Datagram Protocol, a connectionless transport mechanism on IP networks for protocols such as DNS and SNMP with no built-in guaranteed delivery mechanism, but with reduced overhead (often) and faster performance (sometimes).

Spirent Communications

Test Methodologies Journals

The Spirent Communications Test Methodologies Journal series has been developed to make network and equipment testing easier. Spirent provides a Web site for scripts and other resources at <http://scdn.spirentcom.com/welcome.asp>. Registration is required to access scripts and other tools to help you expedite testing.

In addition to Layer 4-7 testing information provided in this edition, Spirent offers test methodology documents and support data for the following technologies, each within its own journal:

- Volume I — Router Performance Testing
- Volume II — IPv6, Wireless, Edge Router, Metro Optical, VoIP, SS7, QoS
- Special IPv6 Edition — IPv6 and IPv4 Test Methodologies
- MPLS Edition — MPLS Network Testing
- Optical Edition — Near End Performance Testing
- Edge Router Edition — Edge Router Testing
- IPSec Edition — IP Security Testing



Analyze | Assure | Accelerate™

Spirent Communications is a worldwide provider of integrated performance analysis and service assurance systems for next-generation network technologies. Our solutions accelerate the profitable development and deployment of network equipment and services by emulating real-world conditions in the lab and assuring end-to-end performance of large-scale networks.

Spirent performance analysis solutions include instruments and systems that measure and analyze the performance of network equipment, particularly the devices that route voice and data messages to their destination. Our service assurance solutions include remote test, fault and performance management systems that let network service providers quickly identify network faults and monitor real-time performance.

Spirent's integrated performance analysis and service assurance solutions enable our customers to more rapidly develop and certify new devices, lowering the cost of widespread deployment and operation of new network services.

Spirent Communications is a wholly owned subsidiary of Spirent plc, an international network technology company.

Spirent Communications

15200 Omega Drive
Rockville, MD USA
20850-3240
Tel: +1 301.417.1224
info@spirentcom.com

Sales and Information

Americas
Tel: +1 800.927.2660
productinfo@spirentcom.com

Europe, Middle East, Africa
Tel: +33 1 6137.2250
salesemea@spirentcom.com

Asia Pacific
Tel: +852.2166.8382
spirentasia@spirentcom.com

Spirent Communications Developers Network
<http://scdn.spirentcom.com>

www.spirentcom.com