Application Note

CPE

Speed Service 1.1 Operation Guide

CPE-AN1801-R

5300 California Avenue • Irvine, CA 92617 • Phone: 949-926-5000 • Fax: 949-926-5203

February 11, 2016

Revision History

Revision	*Date*	*Change Description*
CPE-AN1801-R	02/11/16	**Updated:** • Throughout, for Speed Service version 1.1 • “Speed Service Algorithms” on page 11 • “Command Line Interface” on page 13 • “Client” on page 13 • Table 1: “Client Parameters,” on page 13 • “Server” on page 14 • Table 2: “Server Parameters,” on page 14 • “Examples” on page 15 • “Remote Management Via TR-69” on page 17 • Figure 3: “WebUI Speed Service Screen,” on page 18 **Added:** • “Architecture” on page 7 • “Server Scalability” on page 8 • Figure 1: “Distributed Servers Model,” on page 8 • Figure 2: “Redirect Server Model,” on page 9 • “Software Implementation” on page 10 • “RxRate” on page 15 • “Ramp” on page 15 • “Binary” on page 16 • “Fast” on page 16
CPE-AN1800-R	11/22/15	Initial release

Broadcom Corporation
5300 California Avenue
Irvine, CA 92617

Table of Contents

List of Figures

List of Tables

About This Document

Purpose and Audience

The Broadcom® Speed Service is a tool designed to measure the bandwidth available between a network interface of a Broadcom CPE device and a remote server. This document explains the application and the methods of invoking it. The document is meant for application engineers.

Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use.

For a comprehensive list of acronyms and other terms used in Broadcom documents, go to: http://www.broadcom.com/press/glossary.php.

Document Conventions

The following conventions may be used in this document:

Convention	*Description*
Bold	User input and actions: for example, type **exit**, click **OK**, press **Alt+C**
`Monospace`	Code: `#include <iostream>` HTML: `<td rowspan = 3>` Command line commands and parameters: `wl [-l] <command>`
< >	Placeholders for *required* elements: enter your <username> or `wl <command>`
[ ]	Indicates *optional* command-line parameters: `wl [-l]` Indicates bit and byte ranges (inclusive): [0:3] or [7:0]

Technical Support

Broadcom provides customer access to a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates through its customer support portal (https://support.broadcom.com). For a CSP account, contact your Sales or Engineering support representative.

In addition, Broadcom provides other product support through its Downloads and Support site (http://www.broadcom.com/support/).

Overview

The Broadcom Speed Service is a tool designed to measure the bandwidth available between a network interface of a Broadcom CPE device and a remote server. It utilizes the underlying packet accelerator (when available) to achieve high performance testing with near zero Host CPU utilization.

Architecture

The Speed Service architecture defines four main components: the “Client”, the “Control Server”, the “Data Server”, and the “Unified Server”.

The Control Server, the Data Server, and the Unified Server typically run on a Linux Server located in the service provider network. The Control Server is mainly responsible for coordinating the entire test process with one or more Clients and managing a pool of Data Servers via TCP connections. The Data Server is mainly responsible for providing the data path resources required for conducting bandwidth tests with a Client (test traffic generation and analysis) via UDP connections. The Unified Server is basically a combination of a Control Server and a Data Server that can connect to a single Client at a time.

The Client typically runs on a Broadcom CPE Device, and is responsible for:

- Establishing a TCP connection to the Control Server.
- Requesting the Control Server to assign it a Data Server.
- Establishing a direct UDP connection to the assigned Data Server.
- Running the selected bandwidth measurement algorithm defined by the user.
  - The Client manages the available packet generation and analysis resources of both its own platform, and the assigned Data Server.
- Requesting intermediate test results from the Data Server.
- Measuring the round-trip delay while the bandwidth measurement algorithm is running.
  - The Client sends locally time-stamped UDP Latency packets to the Data Server at regular intervals.
  - The Data Server immediately sends the unmodified Latency packets back to the Client.
  - The Client receives the Latency packets, then calculates the average round-trip delay based on the Latency packet’s time stamp and the current local time.
- Reporting final test results to the user and to the Control Server.

The architecture also defines a proprietary Control Messaging Protocol that is used in all communications between the Client and the Control Server, and between the Client and the Data Server. The communication protocol between the Control Server and a Data Server is left to the user.

Server Scalability

The user is responsible for defining and implementing the Speed Service server model that best fits the scalability requirements and available infrastructure. The following models are provided as reference.

Note: Broadcom's server implementation is for reference only and is not meant to be used for production deployments. Specifically, it lacks scalability.

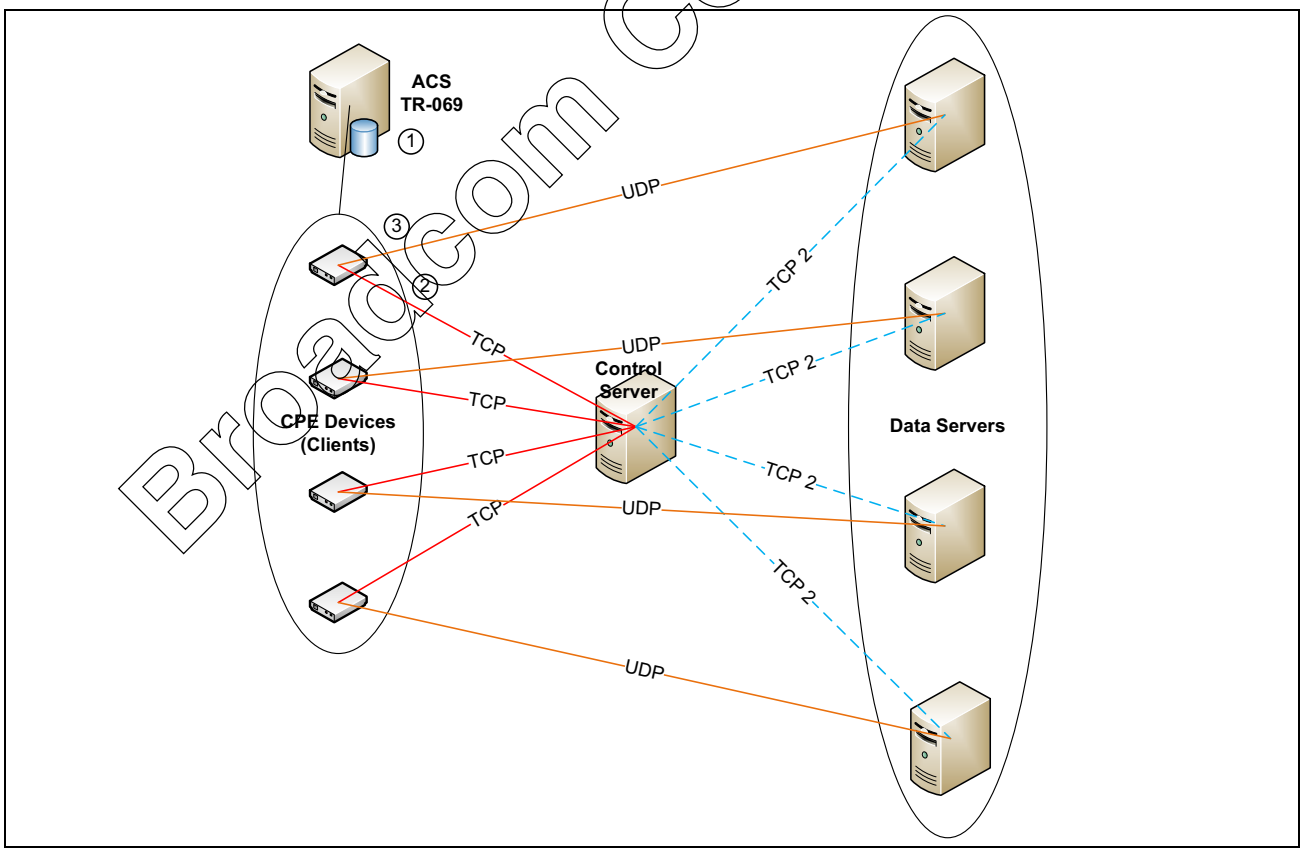
Distributed Servers

In this model, a Control Server accepts connections from multiple Clients and manages a pool of Data Servers. The user is responsible for implementing the Control and Data Servers based on the Unified Server implementation provided by Broadcom.

The test establishment behavior is detailed below.

1. A bandwidth test is initiated, for example, via an ACS call to a CPE.
2. The Client running in the target CPE connects to the specified Control Server and requests a Data Server.
3. The Control Server selects a Data Server with available resources and allocates it to the Client (by providing the Data Server's IP address and UDP Port information). The Client establishes a direct UDP connection to the Data Server and runs the bandwidth test.

Figure 1: Distributed Servers Model



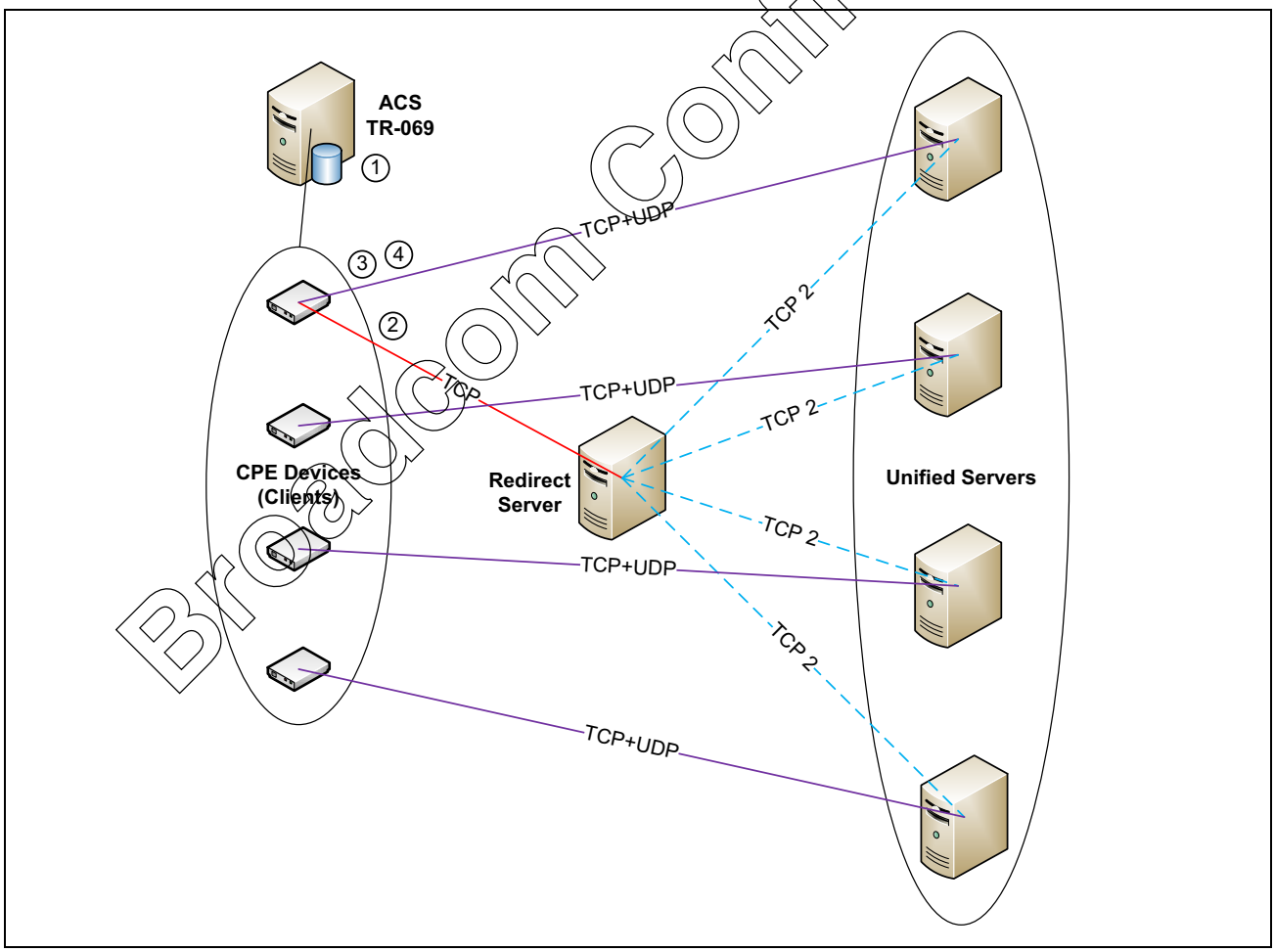
Redirect Server

In this model, a Redirect Server accepts connections from multiple Clients and manages a pool of Unified Servers. There is no need for separate Control Servers and Data Servers. The Broadcom Unified Server reference software can be used directly as the basis for the implementation. The Redirect Server implementation is left to the user.

The test establishment behavior is detailed below.

1. A bandwidth test is initiated, for example, via an ACS call to a CPE.
2. The Client running in the target CPE connects to the specified Redirect Server. The Redirect Server allocates a Unified Server with available resources and tells the Client to Redirect to it (by providing the Unified Server's IP address and TCP Port information).
3. The Client closes its connection to the Redirect Server, and establishes a new TCP connection with the allocated Unified Server. Once the Client requests a Data Server, the Unified Server will assign itself (by providing its own IP address and UDP Port information).
4. The Client establishes a direct UDP connection to the Unified Server and runs the bandwidth test.

Figure 2: Redirect Server Model



Software Implementation

The Speed Service software is implemented as follows.

- Linux Application
  - Client
  - Unified Server
  - Stream Generator and Stream Analyzer
    - Linux Server: Always used.
    - CPE Device: Used when the target interface type is not supported by the Speed Service Linux Driver.
- Linux Driver
  - Stream Generator and Stream Analyzer
    - Linux Server: N/A.
    - CPE Device: Used when the target interface type is not supported by the Network Processor, or when a Network Processor is not available.
- Network Processor (Runner/FAP)
  - Stream Generator and Stream Analyzer
    - Used when the target interface type is supported by the Network Processor

The Stream Generator and Stream Analyzer implementations are functionally equivalent, only differing in performance and portability.

The Stream Generator generates test streams through the UDP connection established between the Client and the Data Server based on the following parameters.

- Duration: The duration of the test stream, in milliseconds.
- Packet length: The UDP payload length of each packet in a test stream. All packets in a test stream will have the same length.
- Bandwidth: The test stream bandwidth (constant), in Kilobits per second (kbps).

The Stream Analyzer receives and discards test stream packets and collects statistics about the test stream. The statistics are used by the Client to measure the forwarding capability of the interface under test.

The Stream Generator and Stream Analyzer implementations in the Linux Driver and the Network Processors are released as binary. The remaining code is available as source under licensing agreement with Broadcom. The software is production ready, whereas the Unified Server code is provided as reference only.

Refer to the build instructions provided with your software release on how to build Speed Service for a Linux Server (README.txt file in the source folder).

Speed Service Algorithms

In order to measure the available bandwidth between the client and the server, a series of test steps are conducted until the result can be determined. A single Test Stream is used for each test step, where the results obtained on a test step may be used to determine if the algorithm has converged, of if a new test step is needed. During each test step, the Client and the Data Server also exchange Latency measurement packets through the UDP connection in order to determine the round-trip delay under load.

Speed Service provides four algorithms to calculate the available bandwidth (Examples are provided in the next section):

- Receive Rate algorithm: Relies on the receive time and received bytes information provided by the Stream Analyzer to compute the available bandwidth. The algorithm runs in two fixed steps. The first step sends a Test Stream at the user specified rate, then measures the Adjusted Receive Rate, which is computed by dividing the total number of received bytes by the receive time. At this point the available bandwidth has been obtained. The goal of the second step is to measure the round trip delay at the obtained Adjusted Receive Rate.
- Ramp algorithm: The goal of this algorithm is to emulate the TCP slow start behavior using UDP Test Streams. It accomplishes that by starting from a low bandwidth Test Stream, and slowly increasing the bandwidth of each following Test Stream until the available bandwidth is obtained. It uses the Adjusted Receive Rate method for bandwidth computation (same as Receive Rate algorithm) at each test step and converges when either packet loss or significant latency increase are detected.
- Binary Search algorithm: The test stream bandwidth at each step is adjusted following a standard binary search progression. The algorithm runs until either the available bandwidth is determined, or a user specified maximum number of steps is reached. A test step is deemed successful when the packet loss information provided by the Stream Analyzer falls below the configured loss threshold, in which case the bandwidth of the next step will be raised. When a test step is not successful, the bandwidth of the next step will be lowered (congestion was detected).
- Fast algorithm: Uses the packet loss percentage of the current test step (measured by the Stream Analyzer) to compute the bandwidth of the next step's Test Stream. For instance, if in the current test step there was a 15% packet loss, the bandwidth of the next test stream will be set to 15% lower than the current rate.

Once the chosen algorithm converges, the following results are provided:

- Bandwidth
- Average round-trip delay
- Overhead
  - Bytes transmitted in addition to the specified payload size, e.g., Ethernet Header, FCS, VLAN Headers, and so on.
- Adjusted Receive Rate
  - The total number of received bytes divided by the receive time.

The Binary Search and Fast algorithms are more suitable for a stable environment where the available bandwidth between the Client and the Data Server will not fluctuate much over the entire duration of the test. The Receive Rate and Ramp algorithms are more robust and should provide accurate results in any circumstance.

The Receive Rate algorithm requires sending one burst slightly above the expected available bandwidth and one burst at the obtained bandwidth, which shortens the test time significantly and provides better immunity to network background fluctuations, so it should be the preferred choice. The Ramp algorithm should be chosen when sending a burst above the expected available bandwidth is not acceptable.

Examples providing more details of each algorithm are provided later in this document.

Command Line Interface

The Speed Service command line interface is shown below.

```
# ss

Broadcom Speed Service 1.1 (Feb  4 2016, 10:33:12)

Usage:
      ss <client/client-sw> send <us/ds> <server_ip> <tcp_port> <duration_msec> <packet_length> <kbps>

       ss <client/client-sw> bw <us/ds> <server_ip> <tcp_port> <duration_msec> <packet_length> <max_kbps> ...
                 bin <max_steps> <loss_percentage> <latency_tolerance_percentage (optional)>
                 fast <max_steps> <loss_percentage>
                 ramp <max_steps> <latency_tolerance_percentage (optional)>
                 rxrate

        ss <server/server-sw> <tcp_port> <udp_port (0 for any port)> <lock_file (optional)>
```

Client

The “client” option allows the best suitable implementation of the Stream Generator and Stream Analyzer to be used. The “client-sw” option forces the Stream Generator and Stream Analyzer implemented in the application to be used, and should only be used in a CPE device where the target interface under test is not supported by the Speed Service driver.

The client has two modes of operation:

- Send (send): A single test stream configured with the specified parameters is transmitted and the obtained results are reported. This mode allows the user to spot check an interface's bandwidth, or to implement its own bandwidth search algorithm if needed.
- Bandwidth (bw): The client and server will autonomously exchange multiple test streams based on the specified parameters and the obtained results are reported.

Table 1: Client Parameters

Parameter	***Description***
<us/ds>	Direction • Upstream (us): Client to Server • Downstream (ds): Server to Client
<server_ip>	Server IP Address
<tcp_port>	Server TCP port
<duration_msec>	Duration of each Test Stream, in milliseconds
<packet_length>	The UDP payload length of each packet in a test stream. All packets in a test stream will have the same length
<kbps>	The test stream bandwidth, in Kilobits per second (send mode only)
<max_kbps>	The maximum expected bandwidth of the interface under test (bandwidth mode only)
rxrate/ramp/fast/bin	Algorithm. See “Speed Service Algorithms” on page 11 for more details.

Table 1: Client Parameters (Cont.)

Parameter	*Description*
<max_steps>	The maximum number of test steps allowed.
<loss_percentage>	The maximum allowed loss of each test stream. When a test stream packet loss is smaller than or equal to <loss_percentage>, it will be considered lossless by the algorithm. A zero value means that no packet loss is tolerated for available bandwidth computation.
<latency_tolerance_percentage>	The maximum allowed latency variation of each test stream. When a Test Stream latency is smaller than or equal to the “minimum obtained latency” + <latency_tolerance_percentage>, the test step is considered valid. Otherwise, its results are ignored. The minimum obtained latency is computed as over each performed test step.

Server

The “server” option allows the best suitable implementation of the Stream Generator and Stream Analyzer to be used. The “server-sw” option forces the Stream Generator and Stream Analyzer implemented in the application to be used, and should only be used in a CPE device where the target interface under test is not supported by the Speed Service driver.

Table 2: Server Parameters

Parameter	*Description*
<tcp_port>	Server TCP port
<udp_port>	The UDP port number used for the test stream UDP connection. When zero is specified, a random port number will be used.
<lock_file>	An optional file is created at the beginning of a client session, then removed when the client session ends. This file can be used by the server to determine whether any sessions are currently in progress.

Examples

Below are a few bandwidth measurement examples taken over a Fast Ethernet WAN interface shaped by an external switch to 60 Mbps upstream, and 100 Mbps downstream (standard).

RxRate

```
# ss client bw us 192.168.1.110 8000 1000 1472 200000 rxrate
Client Speed_Service 1.1
Server Speed_Service 1.1
Server UDP Socket: 192.168.1.110:33527 (0x6E01A8C0)
Client UDP Socket: 192.168.1.1:43394 (0x0101A8C0)
[00] currKbps  200000, adjRxKbps   59984, rxTimeUsec 1106087, packets 16254, loss 10869,
     latency 105736 us (tx 10, rx 3)
[01] currKbps   59984, adjRxKbps   59862, rxTimeUsec 1002158, packets 4875, loss 0,
     latency 1460 us (tx 10, rx 10)
BW: Goodput 59984 Kbps, PayloadRate 57409 Kbps, Loss 0, AvgLatency 1460 us
    AdjRxRate 59984 Kbps, RxTime 1002158 us, Overhead 66 bytes
```

Ramp

```
# ss client bw us 192.168.1.110 8000 300 1472 200000 ramp 20 40
Client Speed_Service 1.1
Server Speed_Service 1.1
Server UDP Socket: 192.168.1.110:41908 (0x6E01A8C0)
Client UDP Socket: 192.168.1.1:51574 (0x0101A8C0)
[00] currKbps   10000, adjRxKbps   10033, rxTimeUsec 298315, packets 243, loss 0, latency 1448/2027 us (tx 3, rx 3)
     bestKbps 0
[01] currKbps   20000, adjRxKbps   20040, rxTimeUsec 299460, packets 487, loss 0, latency 1420/1988 us (tx 3, rx 3)
     bestKbps 10033
[02] currKbps   30000, adjRxKbps   29980, rxTimeUsec 300222, packets 731, loss 0, latency 1323/1852 us (tx 3, rx 3)
     bestKbps 20040
[03] currKbps   40000, adjRxKbps   39988, rxTimeUsec 300194, packets 975, loss 0, latency 1343/1852 us (tx 3, rx 3)
     bestKbps 29980
[04] currKbps   50000, adjRxKbps   49995, rxTimeUsec 300151, packets 1219, loss 0, latency 1314/1839 us (tx 3, rx 3)
     bestKbps 39988
[05] currKbps   60000, adjRxKbps   59961, rxTimeUsec 300344, packets 1462, loss 0, latency 1505/1839 us (tx 3, rx 3)
     bestKbps 49995
[06] currKbps   70000, adjRxKbps   60145, rxTimeUsec 349949, packets 1706, loss 0, latency 31640/1839 us (tx 3, rx 3)
     bestKbps 59961
BW: Goodput 60145 Kbps, PayloadRate 57564 Kbps, Loss 0, AvgLatency 1505 us
    AdjRxRate 60145 Kbps, RxTime 349949 us, Overhead 66 bytes
```

Binary

```
# ss client bw ds 192.168.1.110 8000 1000 1472 150000 bin 8 0
Client Speed_Service 1.1
Server Speed_Service 1.1
Server UDP Socket: 192.168.1.110:52466 (0x6E01A8C0)
Client UDP Socket: 192.168.1.1:58095 (0x0101A8C0)
[00] currKbps  150000, bestKbps       0, loss 3963 (max 12), adjRxKbps  100010 (1013333 us)
     latency 14360/14361 us (tx 10, rx 4), packets 12191
[01] currKbps   75000, bestKbps   75000, loss 0 (max 6), adjRxKbps   75067 (999635 us)
     latency 1770/1771 us (tx 10, rx 10), packets 6095
[02] currKbps  112500, bestKbps   75000, loss 917 (max 9), adjRxKbps   99913 (1013130 us)
     latency 14507/14508 us (tx 10, rx 10), packets 9143
[03] currKbps   93750, bestKbps   93750, loss 0 (max 7), adjRxKbps   93744 (1000012 us)
     latency 1790/1791 us (tx 10, rx 10), packets 7619
[04] currKbps  103125, bestKbps   93750, loss 157 (max 8), adjRxKbps   99988 (1012884 us)
     latency 12376/12377 us (tx 10, rx 10), packets 8381
[05] currKbps   98437, bestKbps   98437, loss 0 (max 8), adjRxKbps   98530 (999715 us)
     latency 1869/1870 us (tx 10, rx 10), packets 8000
[06] currKbps  100781, bestKbps  100781, loss 0 (max 8), adjRxKbps   99970 (1008708 us)
     latency 6354/6355 us (tx 10, rx 10), packets 8190
[07] currKbps  101953, bestKbps  100781, loss 63 (max 8), adjRxKbps   99976 (1012761 us)
     latency 11098/11099 us (tx 10, rx 10), packets 8286
BW: Goodput 100781 Kbps, PayloadRate 96456 Kbps, Loss 0, AvgLatency 6354 us
    AdjRxRate 99970 Kbps, RxTime 1008708 us, Overhead 66 bytes
```

Fast

```
# ss client bw ds 192.168.1.110 8000 2000 1472 150000 fast 2 0
Client Speed_Service 1.1
Server Speed_Service 1.1
Server UDP Socket: 192.168.1.110:49617 (0x6E01A8C0)
Client UDP Socket: 192.168.1.1:36287 (0x0101A8C0)
[00] currKbps  150000, bestKbps       0, loss 8035 (max 24), adjRxKbps   99978 (2013533 us)
     latency 14591/14592 us (tx 20, rx 10), packets 24382
[01] currKbps  100500, bestKbps  100500, loss 0 (max 16), adjRxKbps   99899 (2012233 us)
     latency 7919/7920 us (tx 20, rx 20), packets 16336
BW: Goodput 100500 Kbps, PayloadRate 96187 Kbps, Loss 0, AvgLatency 7919 us
    AdjRxRate 99899 Kbps, RxTime 2012233 us, Overhead 66 bytes
```

Remote Management Via TR-69

The X_BROADCOM_COM_SpeedService proprietary object allows remote management of the Speed Service application. It contains the same parameters and behaves the same way as the Speed Service application. The object also maintains a history log of the results obtained during the past four Speed Service test runs. The test parameters and results associated with the last test run are kept in the main object attributes, while the previous results are stored in the history entries.

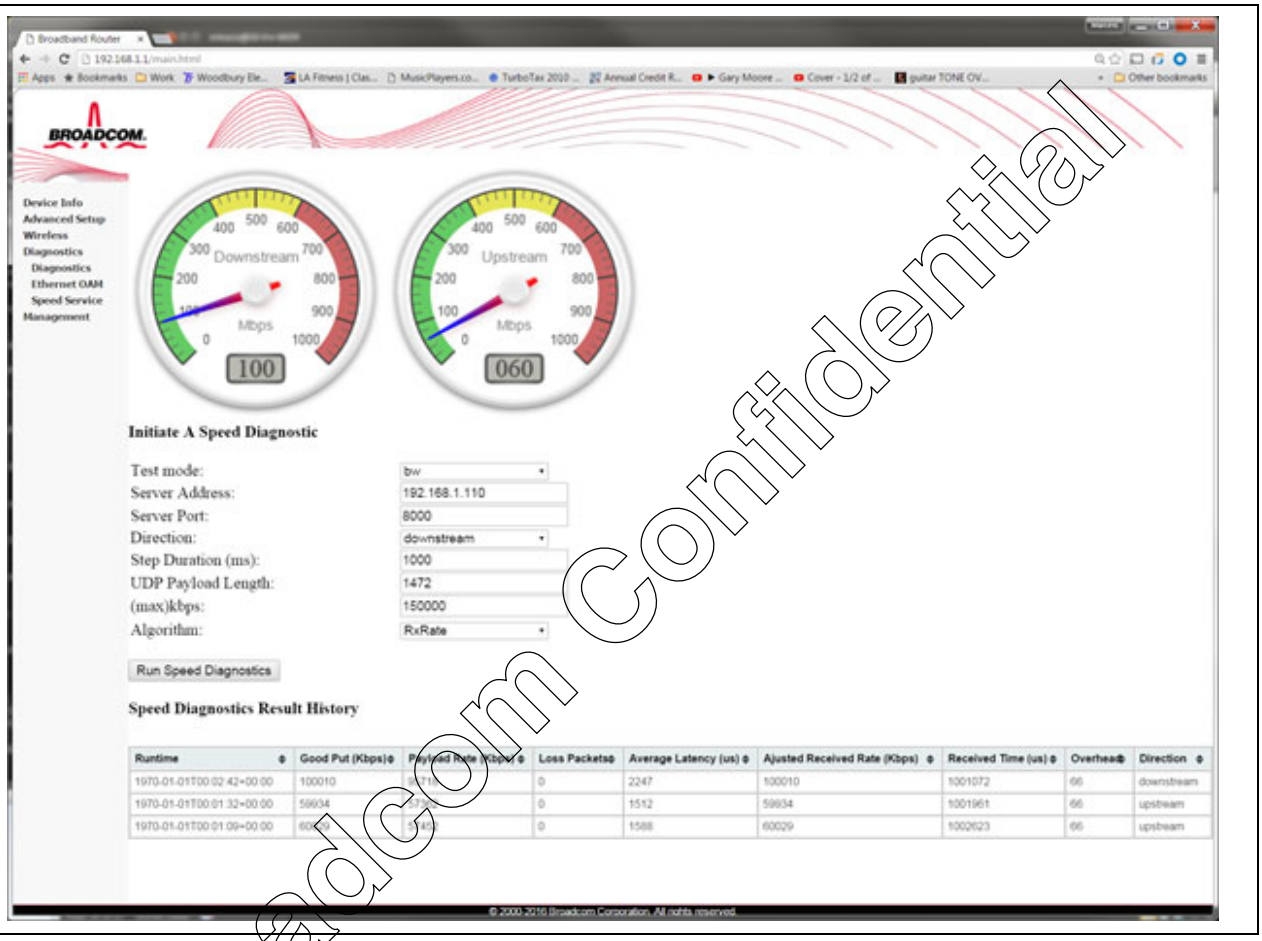
An example of the X_BROADCOM_COM_SpeedService object contents after three test runs is shown below.

```
<X_BROADCOM_COM_SpeedService>
      <DiagnosticsState>Completed</DiagnosticsState>
      <Mode>client_bw</Mode>
      <Direction>downstream</Direction>
      <DataPath>HW</DataPath>
      <ServerIpAddr>192.168.1.110</ServerIpAddr>
      <TcpPort>8000</TcpPort>
      <UdpPort>0</UdpPort>
      <StepDuration>1000</StepDuration>
      <PacketLength>1472</PacketLength>
      <StartingBwKbps>150000</StartingBwKbps>
      <Algorithm>RxRate</Algorithm>
      <MaxSteps>20</MaxSteps>
      <AcceptablePercentageLoss>0</AcceptablePercentageLoss>
      <LatencyTolerancePercentage>-1</LatencyTolerancePercentage>
      <LogBuffer>(null)</LogBuffer>
      <GoodPut>100010</GoodPut>
      <PayloadRate>95718</PayloadRate>
      <PacketLoss>0</PacketLoss>
      <AvgLatency>2247</AvgLatency>
      <AdjustReceivedRate>100010</AdjustReceivedRate>
      <ReceivedTime>1001072</ReceivedTime>
      <Overhead>66</Overhead>
      <LastRunTime>1970-01-01T00:02:42+00:00</LastRunTime>
      <ResultHistoryNumberOfEntries>2</ResultHistoryNumberOfEntries>
      <ResultHistory instance="1" >
        <GoodPut>60029</GoodPut>
        <PayloadRate>57452</PayloadRate>
        <PacketLoss>0</PacketLoss>
        <AvgLatency>1588</AvgLatency>
        <AdjustReceivedRate>60029</AdjustReceivedRate>
        <ReceivedTime>1002623</ReceivedTime>
        <Overhead>66</Overhead>
        <RunTime>1970-01-01T00:01:09+00:00</RunTime>
        <Direction>upstream</Direction>
      </ResultHistory>
      <ResultHistory instance="2">
        <GoodPut>59934</GoodPut>
        <PayloadRate>57362</PayloadRate>
        <PacketLoss>0</PacketLoss>
        <AvgLatency>1512</AvgLatency>
        <AdjustReceivedRate>59934</AdjustReceivedRate>
        <ReceivedTime>1001961</ReceivedTime>
        <Overhead>66</Overhead>
        <RunTime>1970-01-01T00:01:32+00:00</RunTime>
        <Direction>upstream</Direction>
      </ResultHistory>
      <ResultHistory nextInstance="3" ></ResultHistory>
    </X_BROADCOM_COM_SpeedService>
```

Web User Interface

The Speed Service application can also be managed via Broadcom's Web User Interface, which provides access to the X_BROADCOM_COM_SpeedService object.

Figure 3: WebUI Speed Service Screen



Broadcom Corporation
5300 California Avenue
Irvine, CA 92617

Phone: 949-926-5000
Fax: 949-926-5203
E-mail: info@broadcom.com
Web: www.broadcom.com