

How to Detect and Clear Hung TCP Connections using SNMP

Document ID: 61860

Introduction

Prerequisites

Requirements

Components Used

Conventions

Background Information

Details of the MIB Objects Includes Object Identifiers (OIDs)

Use SNMP to Detect if a TCP Connection Hangs

Summary

Step-by-Step Instructions

Use SNMP to Clear a TCP Connection that Hangs

Step-by-Step Instructions

Detailed MIB Object Information

PERL Script to Detect and Clear Hung TCP Connections

NetPro Discussion Forums – Featured Conversations

Related Information

Introduction

This document describes how to use Simple Network Management Protocol (SNMP) to detect and clear hung TCP connections on a Cisco IOS device. The document also explains the SNMP objects that you use for this purpose.

The section entitled, PERL Script to Detect and Clear Hung TCP Connections, provides a link to a PERL script that implements these instructions.

Prerequisites

Requirements

Readers of this document should have knowledge of these topics:

- Understand how to view TCP connection information on Cisco devices
- General use of SNMP **walk**, **get**, **get-next**, and **set** commands
- Understand how to configure SNMP on a Cisco device

Components Used

This document applies to Cisco routers and switches running IOS software supporting the TCP-MIB and the CISCO-TCP-MIB modules.

Note: The CISCO-TCP-MIB module is not loaded by default in NET-SNMP. If the MIB module is not loaded on your system, you must use the OID to reference an object instead of its name.

The information in this document is based on all IOS software and hardware versions.

The information is based upon this version of NET-SNMP:

- NET-SNMP version 5.1.2 available at <http://www.net-snmp.org/>

The PERL script was tested with PERL versions:

- 5.005_03 on FreeBSD
- 5.8.0 on Solaris 5.8
- 5.005_02 shipped as part of CiscoWorks SNMS on Microsoft Windows 2000
- ActivePerl 5.8.4 on Microsoft Windows 2000, available at <http://www.activestate.com/Products/ActivePerl/> .

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Conventions

For more information on document conventions, refer to the Cisco Technical Tips Conventions.

Background Information

Details of the MIB Objects Includes Object Identifiers (OIDs)

These are the objects that you use:

From the CISCO-TCP-MIB module:

- ciscoTcpConnInBytes, OID .1.3.6.1.4.1.9.9.6.1.1.1.1

The number of bytes input on this connection.

- ciscoTcpConnInPkts, OID 1.3.6.1.4.1.9.9.6.1.1.1.2

The number of packets input on this connection.

- ciscoTcpConnOutBytes, OID .1.3.6.1.4.1.9.9.6.1.1.1.3

The number of bytes output on this connection

- ciscoTcpConnOutPkts, OID .1.3.6.1.4.1.9.9.6.1.1.1.4

The number of packets output on this connection.

- ciscoTcpConnRetransPkts, OID .1.3.6.1.4.1.9.9.6.1.1.1.7

The number of packets retransmitted on this connection.

- ciscoTcpConnRto, OID .1.3.6.1.4.1.9.9.6.1.1.1.9

The retransmit timeout value for this connection.

From the TCP-MIB module:

- tcpConnState, OID .1.3.6.1.2.1.6.13.1.1

The status for this connection.

There are more details on these objects in Detailed MIB Object Information.

Use SNMP to Detect if a TCP Connection Hangs

Summary

These steps help you to determine if a TCP connection hangs:

1. In order to determine if the `ciscoTcpConnRetransPkts` and `ciscoTcpConnRto` objects are supported in the device, perform an SNMP **get-next** operation on `ciscoTcpConnRto` and verify if any objects are returned.

Note: You only need to check one object because support for both of them was added at the same time.

Note: Not all Cisco devices support the last two objects (`ciscoTcpConnRetransPkts` and `ciscoTcpConnRto`), but their use can increase the accuracy of the detection.

If the `ciscoTcpConnRetransPkts` and `ciscoTcpConnRto` objects **are** supported, proceed to Step 2.

If the `ciscoTcpConnRetransPkts` and `ciscoTcpConnRto` objects are **not** supported, proceed to Step 3.

2. All objects are supported. For each TCP connection check these:

- ◆ `ciscoTcpConnOutBytes` is 0.
- ◆ `ciscoTcpConnOutPkts` is 0.
- ◆ `ciscoTcpConnRetransPkts` is greater than 0.
- ◆ `ciscoTcpConnRto` is greater than 20,000.

Note: The 20,000 can be reduced to speed up the detection. It takes a minute or so for Rto to reach 20,000 once the connection is hung. However, smaller values may reduce the accuracy of the result.

If all of the previous are true, then this TCP connection is hung and can be cleared. Proceed to [Use SNMP to Clear a TCP Connection that Hangs](#).

3. Only the first four objects are supported. For each TCP connection check these:

- ◆ `ciscoTcpConnInBytes` is greater than 0.
 - ◆ `ciscoTcpConnInPkts` is 0.
 - ◆ `ciscoTcpConnOutBytes` is 0.
 - ◆ `ciscoTcpConnOutPkts` is 0.
- a. Wait a few seconds and **get** the objects again to verify that it was not a TCP connection in the process of being established.

Note: The first two checks (a positive number of input bytes but no input packets) may seem strange, but they were verified against numerous devices and IOS versions.

Note: IOS versions that support all six objects may not exhibit this behavior and, therefore, the test in Step 2 does not include these first two tests.

- b. If all of the objects meet the tests both times then this TCP connection is hung and can be cleared. Proceed to [Use SNMP to Clear a TCP Connection that Hangs](#).

Step-by-Step Instructions

The values in this example are:

- Device hostname a = nms-7206a (supports all objects)
- Device hostname b = nms-1605 (supports only the first four objects)
- Read community = public
- Write community = private

Replace the community strings and the hostname in these commands:

1. Determine if this device supports the `ciscoTcpConnRetransPkts` and `ciscoTcpConnRto` objects:

Perform an **SNMP get-next** operation on `ciscoTcpConnRto`:

```
snmpgetnext -c public nms-7206a ciscoTcpConnRto
```

- ◆ If the objects **are** supported you see a response like this:

```
CISCO-TCP-MIB::ciscoTcpConnRto.14.32.100.75.2065.172.18.86.111.23092 =  
INTEGER: 303 milliseconds
```

Note: The index used for these objects, in this case `14.32.100.75.2065.172.18.86.111.23092`, is a concatenation of the local IP address; `4.32.100.75`, the local TCP port number; `065`, the remote IP address; `72.18.86.111`, and the remote TCP port number; `03092`.

The return is for `ciscoTcpConnRto`. Proceed to Step 2.

- ◆ If the objects are **not** support, you see a response like this:

```
snmpgetnext -c public nms-1605 ciscoTcpConnRto  
CISCO-FLASH-MIB::ciscoFlashDevicesSupported.0 = INTEGER: 1
```

The return is **not** for the `ciscoTcpConnRto` object. The exact object returned is not important. Proceed to Step 3.

2. **Get** information about each TCP connection for devices that support all six objects in the Cisco TCP connection table.

- a. Perform an **SNMP get-next** operation on `ciscoTcpConnOutBytes`, `ciscoTcpConnOutPkts`, `ciscoTcpConnRetransPkts`, and `ciscoTcpConnRto`:

```
snmpgetnext -c public nms-7206a ciscoTcpConnOutBytes  
ciscoTcpConnOutPkts  
ciscoTcpConnRetransPkts  
ciscoTcpConnRto
```

You see a response like this:

```
CISCO-TCP-MIB::ciscoTcpConnOutBytes.14.32.100.75.2065.172.18.86.111.23092 = C  
CISCO-TCP-MIB::ciscoTcpConnOutPkts.14.32.100.75.2065.172.18.86.111.23092 = Co  
CISCO-TCP-MIB::ciscoTcpConnRetransPkts.14.32.100.75.2065.172.18.86.111.23092  
CISCO-TCP-MIB::ciscoTcpConnRto.14.32.100.75.2065.172.18.86.111.23092 = INTEGE
```

- b. Verify these:

◇ `ciscoTcpConnOutBytes` is 0.

- ◇ ciscoTcpConnOutPkts is 0.
- ◇ ciscoTcpConnRetransPkts is greater than 0.
- ◇ ciscoTcpConnRto is greater than 20,000.

Note: The 20,000 can be reduced to speed up the detection. It takes a minute or so for Rto to reach 20,000 once the connection is hung. However, smaller values may reduce the accuracy of the result.

- c. If all of these are true, then this TCP connection is hung and can be cleared. Proceed to Use SNMP to Clear a TCP Connection that Hangs.
- d. Continue to **walk** the TCP connection table. In order to do this, perform an SNMP **get-next** operation repeatedly as you check for hung connections, using the returned objects such as these:

```
snmpgetnext -c public nms-7206a ciscoTcpConnOutBytes.14.32.100.75.2065.172.18.86.111.23092
ciscoTcpConnOutPkts.14.32.100.75.2065.172.18.86.111.23092
ciscoTcpConnRetransPkts.14.32.100.75.2065.172.18.86.111.23092
ciscoTcpConnRto.14.32.100.75.2065.172.18.86.111.23092
```

- e. Check each entry using the previous test until the **get-next** operation returns objects in this manner:

```
CISCO-TCP-MIB::ciscoTcpConnInPkts.14.32.100.75.2065.172.18.86.111.23092 = Counter32: 0
CISCO-TCP-MIB::ciscoTcpConnElapsed.14.32.100.75.2065.172.18.86.111.23092 = Timeticks: (17296508) 2 days, 0:02:45.08
CISCO-TCP-MIB::ciscoTcpConnFastRetransPkts.14.32.100.75.2065.172.18.86.111.23092 = Counter32: 0
CISCO-FLASH-MIB::ciscoFlashDevicesSupported.0 = INTEGER: 5
```

You have now walked all the TCP connections on this device and you are done.

- 3. **Get** information about each TCP connection for devices that only support the first four objects in the Cisco TCP connection table.

- a. Perform an SNMP **get-next** operation on ciscoTcpConnInBytes, ciscoTcpConnInPkts, ciscoTcpConnOutBytes, and ciscoTcpConnOutPkts:

```
snmpgetnext -c public nms-1605 ciscoTcpConnInBytes
ciscoTcpConnInPkts
ciscoTcpConnOutBytes
ciscoTcpConnOutPkts
```

You see a response like this:

```
CISCO-TCP-MIB::ciscoTcpConnInBytes.14.32.6.185.23.14.32.100.33.2249 = Counter32: 0
CISCO-TCP-MIB::ciscoTcpConnInPkts.14.32.6.185.23.14.32.100.33.2249 = Counter32: 0
CISCO-TCP-MIB::ciscoTcpConnOutBytes.14.32.6.185.23.14.32.100.33.2249 = Counter32: 0
CISCO-TCP-MIB::ciscoTcpConnOutPkts.14.32.6.185.23.14.32.100.33.2249 = Counter32: 0
```

- b. Check to see if these are true:

- ◇ ciscoTcpConnInBytes is greater than 0.
- ◇ ciscoTcpConnInPkts is 0.
- ◇ ciscoTcpConnOutBytes is 0.
- ◇ ciscoTcpConnOutPkts is 0.

- c. Wait a few seconds and **get** the objects again. Verify that it was not a TCP connection in the process of being established.
- d. If all of the above **are** true, then this TCP connection is hung and can be cleared. Proceed to Use SNMP to Clear a TCP Connection that Hangs.
- e. Continue to **walk** the TCP connection table. In order to do this, perform an SNMP **get-next** operation repeatedly as you check for hung connections, using the returned objects such as these:

```

snmpgetnext -c public nms-1605 ciscoTcpConnInBytes.14.32.6.185.23.14.32.100.33.4184
ciscoTcpConnInPkts.14.32.6.185.23.14.32.100.33.2249
ciscoTcpConnOutBytes.14.32.6.185.23.14.32.100.33.2249
ciscoTcpConnOutPkts.14.32.6.185.23.14.32.100.33.2249

```

f. Check each entry using the previous test until the **get-next** operation returns objects in this manner:

```

CISCO-TCP-MIB::ciscoTcpConnOutBytes.14.32.6.185.23.14.32.100.33.4184 = Counter32: 2249
CISCO-TCP-MIB::ciscoTcpConnOutPkts.14.32.6.185.23.14.32.100.33.4184 = Counter32: 2249
CISCO-TCP-MIB::ciscoTcpConnInPkts.14.32.6.185.23.14.32.100.33.4184 = Counter32: 2249
CISCO-TCP-MIB::ciscoTcpConnElapsed.14.32.6.185.23.14.32.100.33.4184 = Timeticks: 100000000

```

You have now walked all the TCP connections on this device and you are done.

Use SNMP to Clear a TCP Connection that Hangs

Step-by-Step Instructions

You can use SNMP to clear a hung TCP connection. The SNMP command is equivalent to the **clear tcp local** *<local_ip>* *<local_port>* **remote** *<remote_ip>* *<remote_port>* command. The object that you use to clear a line is **tcpConnState**.

In order to clear a hung TCP connection with SNMP, issue this command:

```

snmpset -c private nms-7206a tcpConnState.14.32.100.75.2065.172.18.86.111.23092 integer de
TCP-MIB::tcpConnState.14.32.100.75.2065.172.18.86.111.23092 = INTEGER: deleteTCB(12)

```

Note: The index used for these objects, in this case 14.32.100.75.2065.172.18.86.111.23092, is a concatenation of the local IP address;4.32.100.75, the local TCP port number;2065, the remote IP address;72.18.86.111, and the remote TCP port number;23092.

Note: You must use the exact index that you determined was hung in Use SNMP to Detect if a TCP Connection Hangs. Be aware that this command disconnects a TCP connection without warning.

Detailed MIB Object Information

```

.1.3.6.1.4.1.9.9.6.1.1.1.1
ciscoTcpConnInBytes OBJECT-TYPE
    -- FROM CISCO-TCP-MIB
    SYNTAX          Counter
    MAX-ACCESS      read-only
    STATUS          Current
    DESCRIPTION     "Number of bytes that have been input on this TCP
connection."
 ::= { ciscoTcpConnEntry 1 }

.1.3.6.1.4.1.9.9.6.1.1.1.2
ciscoTcpConnOutBytes OBJECT-TYPE
    -- FROM CISCO-TCP-MIB
    SYNTAX          Counter
    MAX-ACCESS      read-only
    STATUS          Current
    DESCRIPTION     "Number of bytes that have been output on this TCP
connection."
 ::= { ciscoTcpConnEntry 2 }

.1.3.6.1.4.1.9.9.6.1.1.1.3
ciscoTcpConnInPkts OBJECT-TYPE
    -- FROM CISCO-TCP-MIB

```

```

        SYNTAX          Counter
        MAX-ACCESS      read-only
        STATUS          Current
        DESCRIPTION     "Number of packets that have been input on this TCP
                        connection."
 ::= { ciscoTcpConnEntry 3 }

.1.3.6.1.4.1.9.9.6.1.1.1.4
ciscoTcpConnOutPkts OBJECT-TYPE
    -- FROM CISCO-TCP-MIB
    SYNTAX          Counter
    MAX-ACCESS      read-only
    STATUS          Current
    DESCRIPTION     "Number of packets that have been output on this TCP
                    connection."
 ::= { ciscoTcpConnEntry 4 }

.1.3.6.1.4.1.9.9.6.1.1.1.7
ciscoTcpConnRetransPkts OBJECT-TYPE
    -- FROM CISCO-TCP-MIB
    SYNTAX          Counter
    MAX-ACCESS      read-only
    STATUS          Current
    DESCRIPTION     "The total number of packets retransmitted due to a timeout -
                    that is, the number of TCP segments transmitted containing
                    one or more previously transmitted octets."
 ::= { ciscoTcpConnEntry 7 }

.1.3.6.1.4.1.9.9.6.1.1.1.9
ciscoTcpConnRto OBJECT-TYPE
    -- FROM CISCO-TCP-MIB
    SYNTAX          Integer
    MAX-ACCESS      read-only
    STATUS          Current
    DESCRIPTION     "The current value used by a TCP implementation for the
                    retransmission timeout."
 ::= { ciscoTcpConnEntry 9 }

.1.3.6.1.2.1.6.13.1.1
tcpConnState OBJECT-TYPE
    -- FROM RFC1213-MIB
    SYNTAX          Integer { closed(1), listen(2), synSent(3), synReceived(4),
                            established(5), finWait1(6), finWait2(7), closeWait(8), lastAck(9),
                            closing(10), timeWait(11), deleteTCB(12) }
    MAX-ACCESS      read-write
    STATUS          Mandatory
    DESCRIPTION     "The state of this TCP connection.

                    The only value which may be set by a management
                    station is deleteTCB(12). Accordingly, it is
                    appropriate for an agent to return a `badValue'
                    response if a management station attempts to set
                    this object to any other value.

                    If a management station sets this object to the
                    value deleteTCB(12), then this has the effect of
                    deleting the TCB (as defined in RFC 793) of the
                    corresponding connection on the managed node,
                    resulting in immediate termination of the
                    connection.

                    As an implementation-specific option, a RST
                    segment may be sent from the managed node to the
                    other TCP endpoint (note however that RST segments
                    are not sent reliably)."
```

```
::= { tcpConnEntry 1 }
```

PERL Script to Detect and Clear Hung TCP Connections

This link provides an archive file with a PERL script and the necessary MIB modules. Right click the link and save the file to your system.

- fixTCPhang.tgz

The files in the archive are:

- bin/fixTCPhang.pl
- mibs/CISCO-SMI.my
- mibs/CISCO-TCP-MIB.my

To extract the script and the MIB modules, use a utility such as `gzip` and `tar` on a UNIX-like operating systems. For example, to extract the files to `/tmp` assuming that the archive file is placed in `/tmp`:

```
cd /tmp; gzip -dc fixTCPhang.tgz | tar -xvf -
```

Note: You may need to edit the first line of the script to specify the location of PERL.

Use `winzip` or other utilities on Microsoft Windows operating systems to extract the files. If you extract the files to `c:\tmp` then you do not have to specify the `-m` option when you run the script.

Invoke the files with this command:

```
fixTCPhang.pl -c public -C private -f nms-7206a
```

For each hung TCP connections found you see a line like this output:

```
Found bad TCP connection: Local IP: 14.32.100.75 port 23 Remote IP: 172.18.100.33 port 478
```

As the read-write community string was supplied and the `-f` option was specified, the script cleared the connection. Note the `CLEARED` statement at the end of the output.

The script supports SNMP versions 1, 2c, and 3. If you specify SNMP version 3, you must specify all of the authentication information in the `-v` argument. This is an example of using SNMP v3:

```
fixTCPhang.pl -v "3 -a MD5 -u chelliot -A chelliot -l authNoPriv" -f nms-dmz-ap1200-b
```

The IOS commands to configure SNMP v3 for the previous example are:

```
snmp-server group chelliot-group v3 auth write v1default  
snmp-server user chelliot chelliot-group v3 auth md5 chelliot
```

Note: There appears to be a bug in the Windows version of NET-SNMP used in this testing. The bug does not allow SHA authentication to work properly.

There are several other options that you can use with this script. Some of the script options include where to find the NET-SNMP command-line utilities and where to find the MIB modules if they are not in

/tmp/mibs. You can also view this summary of those options:

fixTCPPhang.pl

```
fixTCPPhang.pl [-dfhV -c <read_community> -C <write_community> -m <mib_directory>
               -p <command_path> -t <timeout> -v <snmp_version>] <device>
```

Version 1.2

Detect hung TCP connections on <device>, optionally clearing them.

Options:

- c Specify read community string. Defaults to public.
- C Specify the readwrite community string. No default.
Must be supplied for the script to clear hung connections.
- d Turn on debug mode.
- f Fix or clear any hung TCP connections found.
- h Print this message.
- m Specify the directory to find CISCO-SMI.my and CISCO-TCP-MIB.my.
Defaults to /tmp/mibs.
- p Where to find the net-snmp utilities.
Optional if the utilities are in the path.
- t SNMP Timeout value. Defaults to 5 sec.
- v Specify SNMP version to use: One of 1, 2c, or 3.
If 3 is specified then this option must include all of the authentication information for SNMPv3. For example:
"3 -a MD5 -u chelliot -A chelliot -l authNoPriv"
Note: NET-SNMP seems to have a bug with SHA authentication on Windows.
See the NET-SNMP documentation for more information.
Defaults to SNMP version 1.
- V Print version number.

NetPro Discussion Forums – Featured Conversations

Networking Professionals Connection is a forum for networking professionals to share questions, suggestions, and information about networking solutions, products, and technologies. The featured links are some of the most recent conversations available in this technology.

NetPro Discussion Forums – Featured Conversations for Network Management
Network Infrastructure: Network Management
Virtual Private Networks: Network and Policy Management

Related Information

- [Technical Support – Cisco Systems](#)

[Contacts & Feedback](#) | [Help](#) | [Site Map](#)

© 2008 – 2009 Cisco Systems, Inc. All rights reserved. [Terms & Conditions](#) | [Privacy Statement](#) | [Cookie Policy](#) | [Trademarks of Cisco Systems, Inc.](#)

Updated: Oct 26, 2005

Document ID: 61860