Contents

Introduction

This document describes how the Layer 2 Tunneling Protocol (L2TP) in StarOS is implemented on the ASR5k and Troubleshoot L2TP Peering - L2TPTunnelDownPeerUnreachable.

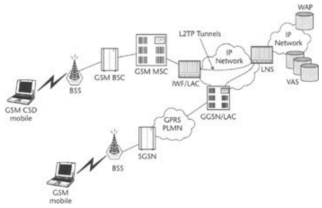
What is L2TP?

L2TP extends the point-to-point nature of PPP. L2TP provides an encapsulation method for the transmit of tunneled PPP frames, which allows the PPP endpoints to be tunneled over a packet-switched network. L2TP is most commonly deployed in remote-access-type scenarios that use the Internet to offer intranet-type services. The concept is that of a Virtual Private Network (VPN).

The two primary physical elements of L2TP are the L2TP Access Concentrator (LAC) and the L2TP Network Server (LNS):

- LAC: The LAC is a peer to the LNS that acts as one side of the tunnel endpoint. The LAC terminates the remote PPP connection and sits between the remote and the LNS. Packets are forwarded to and from the remote connection over the PPP connection. Packets to and from the LNS are forwarded over the L2TP tunnel.
- LNS: The LNS is a peer to the LAC that acts as one side of the tunnel endpoint. The LNS is the termination point for the LAC PPP tunneled sessions. This is used to aggregate the multiple LAC-tunneled PPP sessions and ingress into the private network.

Simplified L2TP setup in Mobile network, as shown in this image.



There are two different message types that L2TP uses:

- Control messages: L2TP passes control and data messages over separate control and data channels. The in-band control channel passes sequenced control connection management, call management, error reporting, and session control messages. Initiation of the control connection is not specific to either the LAC or the LNS but, rather, the tunnel originator and receiver that has relevance in the control connection establishment. A shared-secret challenge authentication method is used between the tunnel endpoints.
- Data messages: Data messages are used to encapsulate the PPP frames that are sent into the L2TP tunnel.

The detailed call flow and tunnel establishment is explained here:

http://www.cisco.com/c/en/us/support/docs/dial-access/virtual-private-dialup-network-vpdn/23980-l2tp-23980.html

Where do we use it in Mobility?

The typical deployment is for corporate users where the GGSN acts as LAC and establishes secure tunnels towards LNS that is operated in the corporate network. Detailed call flows are available in appendix of GGSN configuration guide that can be found, per specific software version, here:

http://www.cisco.com/c/en/us/support/wireless/asr-5000-series/products-installation-and-configuration-guides-list.html

What is ASR5x00 in this setup?

ASR5k can support LAC and LNS functionality.

L2TP LAC Support

L2TP establishes L2TP control tunnels between LAC and LNS before tunneling the subscriber PPP connections as L2TP sessions. The LAC service is based on the same architecture as the GGSN and benefits from dynamic resource allocation and distributed message and data processing. This design allows the LAC service to support over 4000 setups per second or a maximum of over 3G of throughput. There can be a maximum up to 65535 sessions in a single tunnel and as many as 500,000 L2TP sessions using 32,000 tunnels per system.

L2TP LNS Support

The system configured as a Layer 2 Tunneling Protocol Network Server (LNS) supports the termination secure Virtual Private Network (VPN) tunnels between from L2TP Access Concentrators (LACs).

L2TP establishes L2TP control tunnels between LAC and LNS before tunneling the subscriber

PPP connections as L2TP sessions. There can be a maximum of up to 65535 sessions in a single tunnel and up to 500,000 sessions per LNS.

The LNS architecture is similar to the GGSN and utilizes the concept of a de-multiplexer to intelligently assign new L2TP sessions across the available software and hardware resources on the platform without operator intervention.

For more information refer PGW/GGSN configuration guides.

Configuration to enable services on the Cisco Devices

Configuration sample for LAC on ASR5k

```
 apn test-apn
accounting-mode none
     aaa group AAA
     authentication msisdn-auth
     ip context-name destination
     tunnel l2tp peer-address 1.1.1.1 local-hostname lac_l2tp     configure
context destination-gi
lac-service l2tp_service
     allow called-number value apn
     peer-lns 1.1.1.1 encrypted secret pass
     bind address 1.1.1.2
```

Configuration sample for LNS on ASR5k

```
  configure
context destination-gi
lns-service lns-svc
bind address 1.1.1.1
authentication { { [ allow-noauth | chap < pref > | mschap < pref > | | pap < pref > | msid-auth
}
```

> **Note**: Multiple addresses on the same IP interface can be bound to different LNS services. However, each address can be bound to only one LNS service. In addition, the LNS service can not be bound to the same interface as other services such as a LAC service.

Configuration sample for LNS on Cisco IOS device

This can be used as a supporting configuration sample for Cisco IOS configuration and is not subject to this article.

LNS configuration

```
 aaa group server radius AAA
server 2.2.2.2 auth-port 1812 acct-port 1813
ip radius source-interface GigabitEthernet0/1
!  aaa authentication login default local
aaa authentication ppp AAA group AAA
aaa authorization network AAA group AAA
aaa accounting network default
action-type start-stop
```

```
group radius   vpdn-group vpdn
accept-dialin
protocol l2tp
virtual-template 10
l2tp tunnel password pass   interface Virtual-Template10
ip unnumbered GigabitEthernet0/1
peer default ip address pool AAA
ppp authentication pap chap AAA
ppp authorization AAA
```

Troubleshoot Peer Unreachable Event

This section will give some guidelines on how to troubleshoot L2TPTunnelDownPeerUnreachable event in the network. It is explained here with reference to PDSN closed RP but the troubleshoot steps are the same when troubleshooting with GGSN/PGW.

As a reminder, a LAC to LNS tunnel is created in order to contain subscriber sessions while it extends the subscriber connection from a PDSN/HA/GGSN/PGW to the LNS where it is terminated and where an IP address is provided. If on a StarOS chassis, the LNS will get an IP address from a configured IP pool. If on some other LNS, for example at the customer premises, the IP address is provided by the LNS there. In the latter scenario, this could effectively allow for users to connect to their home network through a LAC running on a roaming partner.

A LAC LNS tunnel is first created as the first subscriber session is attempted to be setup, and will stay up as long as there are sessions in the tunnel.

When the last session ends for a given tunnel, that tunnel is closed or shut down. More than one tunnel can be established between the same LAC-LNS peers.

Here is a snippet of output from the command **show l2tp tunnels all** that shows this in this case the chassis hosts both LAC and LNS services (TestLAC and TestLNS). Note that the LAC and LNS tunnels ALL have sessions, while some Closed RP tunnels have no sessions.

```
 [local]1X-PDSN# show l2tp tunnels all | more
|+----State: (C) - Connected       (c) - Connecting
|            (d) - Disconnecting   (u) - Unknown
|
|
v  LocTun ID  PeerTun ID Active Sess Peer IPAddress  Service Name   Uptime
--- ---------- ---------- ----------- --------------- ------------   ---------
.............
C  30         1          511         214.97.107.28   TestLNS         00603h50m
C  31         56         468         214.97.107.28   TestLNS         00589h31m
C  10         105        81          79.116.237.27   TestLAC         00283h53m
C  29         16         453         79.116.231.27   TestLAC         00521h32m
C  106        218        63          79.116.231.27   TestLAC         00330h10m
C  107        6          464         79.116.237.27   TestLAC         00329h47m
C  30         35         194         214.97.107.28   TestLNS         00596h06m
```

Services configuration can be viewed with

```
  show (lac-service | lns-service) name <lac or lns service name>
```

Here is an example of the L2TPTunnelDownPeerUnreachable trap with LAC service 1.1.1.2 and LNS service (peer) 1.1.1.1

```
 Internal trap notification 92 (L2TPTunnelDownPeerUnreachable)  context destination service lac
peer address 1.1.1.1 local address 1.1.1.2
```

Get a count of how many times this trap has been triggered (since reload or last reset of statistics)

using the command **show snmp trap statistics**

The L2TPTunnelDownPeerUnreachable trap is triggered for L2TP when a tunnel setup timeout occurs OR keep-alive (Hello) packets are not responded to. The cause is usually due to the LNS peer not responding to requests from the LAC or transport issues in either direction.

There is no trap to indicate that the peer becomes reachable, which, if it is not understood how to investigate further, can lead to confusion as to whether there is still an issue or not at the time of investigation (feature request submitted).

To proceed, the most important part we need is the peer IP address. The first step is to ensure that there is IP connectivity that can be checked with PING. If there is connectivity you can proceed with the debugs

```
****THIS IS TO BE RUN CAREFULLY and UPON verification of TAC/BU****


Active logging (exec mode) – logs written to terminal window

logging filter active facility l2tpmgr level debug
logging filter active facility l2tp-control level debug
logging active

To stop logging:

no logging active

Runtime logging (global config mode) – logs saved internally

logging filter runtime facility l2tpmgr level debug
logging filter runtime facility l2tp-control level debug

To view logs:

show logs (and/or check the syslog server if configured)
```

Notes:

l2tpmgr tracks specific subscriber session setup

l2tp-control tracks tunnel establishment:

Here is sample debug from this output

Use case: Initial tunnel setup failure due to retry timeouts

```
16:34:00.017 [l2tpmgr 48140 debug] [7/0/555 <l2tpmgr:1> l2tpmgr_call.c:591] [callid 4144ade2]
[context: destination, contextID: 3]  [software internal system] L2TPMgr-1  msid 0000012345
username laclnsuser  service <lac> - IPSEC tunnel does not exist
16:34:00.018 [l2tp-control 50069 debug] [7/0/555 <l2tpmgr:1> l2tpsnx_fsm.c:105] [callid
4144ade2] [context: destination, contextID: 3]  [software internal user] l2tp fsm: state
L2TPSNX_STATE_OPEN event L2TPSNX_EVNT_APP_NEW_SESSION ----------------------
16:34:00.018 [l2tp-control 50001 debug] [7/0/555 <l2tpmgr:1> l2tpsnx_proto.c:1474] [callid
4144ade2] [context: destination, contextID: 3]  [software internal user outbound protocol-log]
L2TP Tx PDU, from 1.1.1.2:13660 to 1.1.1.1:1701 (138)
l2tp:[TLS](0/0)Ns=0,Nr=0 *MSGTYPE(SCCRQ) *PROTO_VER(1.0) *FRAMING_CAP(AS) *BEARER_CAP(AD)
TIE_BREAKER(0706050403020100) FIRM_VER(256) *HOST_NAME(lac) VENDOR_NAME(StarentNetworks)
*ASSND_TUN_ID(10) *RECV_WIN_SIZE(16) *CHALLENGE(dbed79cdc497f266bd374d427607cd52)
16:34:00.928 [l2tp-control 50001 debug] [7/0/555 <l2tpmgr:1> l2tpsnx_proto.c:1474] [callid
```

```
4144ade2] [context: destination, contextID: 3]  [software internal user outbound protocol-log]
L2TP Tx PDU, from 1.1.1.2:13660 to 1.1.1.1:1701 (138)
l2tp:[TLS](0/0)Ns=0,Nr=0 *MSGTYPE(SCCRQ) *PROTO_VER(1.0) *FRAMING_CAP(AS) *BEARER_CAP(AD)
TIE_BREAKER(0706050403020100) FIRM_VER(256) *HOST_NAME(lac) VENDOR_NAME(StarentNetworks)
*ASSND_TUN_ID(10) *RECV_WIN_SIZE(16) *CHALLENGE(dbed79cdc497f266bd374d427607cd52)
16:34:02.943 [l2tp-control 50001 debug] [7/0/555 <l2tpmgr:1> l2tpsnx_proto.c:1474] [callid
4144ade2] [context: destination, contextID: 3]  [software internal user outbound protocol-log]
L2TP Tx PDU, from 1.1.1.2:13660 to 1.1.1.1:1701 (138)
l2tp:[TLS](0/0)Ns=0,Nr=0 *MSGTYPE(SCCRQ) *PROTO_VER(1.0) *FRAMING_CAP(AS) *BEARER_CAP(AD)
TIE_BREAKER(0706050403020100) FIRM_VER(256) *HOST_NAME(lac) VENDOR_NAME(StarentNetworks)
*ASSND_TUN_ID(10) *RECV_WIN_SIZE(16) *CHALLENGE(dbed79cdc497f266bd374d427607cd52)
16:34:06.870 [l2tp-control 50001 debug] [7/0/555 <l2tpmgr:1> l2tpsnx_proto.c:1474] [callid
4144ade2] [context: destination, contextID: 3]  [software internal user outbound protocol-log]
L2TP Tx PDU, from 1.1.1.2:13660 to 1.1.1.1:1701 (138)
l2tp:[TLS](0/0)Ns=0,Nr=0 *MSGTYPE(SCCRQ) *PROTO_VER(1.0) *FRAMING_CAP(AS) *BEARER_CAP(AD)
TIE_BREAKER(0706050403020100) FIRM_VER(256) *HOST_NAME(lac) VENDOR_NAME(StarentNetworks)
*ASSND_TUN_ID(10) *RECV_WIN_SIZE(16) *CHALLENGE(dbed79cdc497f266bd374d427607cd52)
16:34:14.922 [l2tp-control 50001 debug] [7/0/555 <l2tpmgr:1> l2tpsnx_proto.c:1474] [callid
4144ade2] [context: destination, contextID: 3]  [software internal user outbound protocol-log]
L2TP Tx PDU, from 1.1.1.2:13660 to 1.1.1.1:1701 (138)
l2tp:[TLS](0/0)Ns=0,Nr=0 *MSGTYPE(SCCRQ) *PROTO_VER(1.0) *FRAMING_CAP(AS) *BEARER_CAP(AD)
TIE_BREAKER(0706050403020100) FIRM_VER(256) *HOST_NAME(lac) VENDOR_NAME(StarentNetworks)
*ASSND_TUN_ID(10) *RECV_WIN_SIZE(16) *CHALLENGE(dbed79cdc497f266bd374d427607cd52)
-------------------  16:34:22.879 [l2tp-control 50001 debug] [7/0/555 <l2tpmgr:1>
l2tpsnx_proto.c:1474] [callid 4144ade2] [context: destination, contextID: 3]  [software internal
user outbound protocol-log] L2TP Tx PDU, from 1.1.1.2:13660 to 1.1.1.1:1701 (38)
l2tp:[TLS](0/0)Ns=1,Nr=0 *MSGTYPE(StopCCN) *RESULT_CODE(2/0) *ASSND_TUN_ID(10)
16:34:22.879 [l2tp-control 50069 debug] [7/0/555 <l2tpmgr:1> l2tpsnx_fsm.c:105] [callid
4144ade2] [context: destination, contextID: 3]  [software internal user] l2tp fsm: state
L2TPSNX_STATE_WAIT_TUNNEL_ESTB event L2TPSNX_EVNT_PROTO_TUNNEL_DISCONNECTED
```

Here is the resultant SNMP trap triggered to match the above logs for the moment the system determined the failure

```
 16:34:22 2009 Internal trap notification 92 (L2TPTunnelDownPeerUnreachable)  context
destination service lac peer address 1.1.1.1 local address 1.1.1.2
```

Use case: Initial tunnel setup failure due to retry timeouts - Analysis

What we see is that tunnel comes up at 16:34 and it attempts to send the challenge for five times. Apparently, there is no reply and eventually the tunnel disconnects.

Look into the configuration defaults or configured values and see

```
 max-retransmission 5
retransmission-timeout-first 1
retransmission-timeout-max 8
```

This configuration is to be interpeted as first retransmit after 1 second, then exponential increase - doubling each time: 1, 2, 4, 8, 8.

Note the term max-retransmissions (five) includes the first attempt/transmission.
retransmission-timeout-max is maximum amount of time between transmissions after (if) this limit is reached
retransmission-timeout-first is the starting point of how long to wait before the first retransmission.

So, doing the math, in the case of the default parameters, a failure would occur after 1+ 2 + 4 + 8 + 8 seconds = 23 seconds, which is seen exactly as in the output below.

Use case: Initial tunnel setup failure due to keepalives

The other reason for the L2TPTunnelDownPeerUnreachable trap is no response to keepalive-interval messages. These are used during periods where there are no control messages or data being sent over the tunnel, to ensure that the other end is still alive. If there are sessions in the tunnel, but they are not doing anything, this command ensures that the tunnel is still working properly, because by enabling it, keepalive messages are sent after the configured period of no packet exchange (i.e. 60 seconds), and responses are expected. The frequency of sending the keepalive after sending the first one and not getting a response is the same as described above for tunnel setup. So, after 23 seconds of not receiving a response to hello (keepalive) messages, the tunnel will be torn down. See configurable keepalive-interval (default = 60s).

Here are examples of successful keep-alive exchange, both from monitor subscriber and logging. Note the interval of one minute between sets of messages as a result of no user data being transmitted for one minute. In this example, the LAC and LNS services are located in the same chassis, in contexts named **destination** and **lns** respectively.

```
 INBOUND>>>>>  12:54:35:660 Eventid:50000(3)
L2TP Rx PDU, from 1.1.1.1:13660 to 1.1.1.2:13661 (20)
l2tp:[TLS](5/0)Ns=19,Nr=23 *MSGTYPE(HELLO)

<<<<OUTBOUND  12:54:35:661 Eventid:50001(3)
L2TP Tx PDU, from 1.1.1.2:13661 to 1.1.1.1:13660 (12)
l2tp:[TLS](1/0)Ns=23,Nr=20 ZLB

<<<<OUTBOUND  12:55:35:617 Eventid:50001(3)
L2TP Tx PDU, from 1.1.1.2:13661 to 1.1.1.1:13660 (20)
l2tp:[TLS](1/0)Ns=23,Nr=20 *MSGTYPE(HELLO)

INBOUND>>>>>  12:55:35:618 Eventid:50000(3)
L2TP Rx PDU, from 1.1.1.1:13660 to 1.1.1.2:13661 (12)
l2tp:[TLS](5/0)Ns=20,Nr=24 ZLB  12:54:35.660 [l2tp-control 50001 debug] [7/0/555 <l2tpmgr:1>
l2tpsnx_proto.c:1474] [callid 106478e8] [context: lns, contextID: 11]  [software internal user
outbound protocol-log] L2TP Tx PDU, from 1.1.1.1:13660 to 1.1.1.2:13661 (20)
l2tp:[TLS](5/0)Ns=19,Nr=23 *MSGTYPE(HELLO)

12:55:35.618 [l2tp-control 50000 debug] [7/0/555 <l2tpmgr:1> l2tp.c:13050] [callid 106478e8]
[context: lns, contextID: 11]  [software internal user inbound protocol-log] L2TP Rx PDU, from
1.1.1.2:13661 to 1.1.1.1:13660 (20) l2tp:[TLS](1/0)Ns=23,Nr=20 *MSGTYPE(HELLO)
```

Finally, here is an example where, for an EXISTING tunnel, hello messages are not responded to, and the call and tunnel are torn down. Monitor Subscriber output:

```
 <<<<OUTBOUND  14:06:21:406 Eventid:50001(3)
L2TP Tx PDU, from 1.1.1.2:13661 to 1.1.1.1:13661 (20)
l2tp:[TLS](2/0)Ns=4,Nr=2 *MSGTYPE(HELLO)

<<<<OUTBOUND  14:06:22:413 Eventid:50001(3)
L2TP Tx PDU, from 1.1.1.2:13661 to 1.1.1.1:13661 (20)
l2tp:[TLS](2/0)Ns=4,Nr=2 *MSGTYPE(HELLO)

<<<<OUTBOUND  14:06:24:427 Eventid:50001(3)
L2TP Tx PDU, from 1.1.1.2:13661 to 1.1.1.1:13661 (20)
l2tp:[TLS](2/0)Ns=4,Nr=2 *MSGTYPE(HELLO)

<<<<OUTBOUND  14:06:28:451 Eventid:50001(3)
L2TP Tx PDU, from 1.1.1.2:13661 to 1.1.1.1:13661 (20)
l2tp:[TLS](2/0)Ns=4,Nr=2 *MSGTYPE(HELLO)

<<<<OUTBOUND  14:06:36:498 Eventid:50001(3)
L2TP Tx PDU, from 1.1.1.2:13661 to 1.1.1.1:13661 (20)
l2tp:[TLS](2/0)Ns=4,Nr=2 *MSGTYPE(HELLO)
```

```
<<<<OUTBOUND  14:06:44:446 Eventid:50001(3)
L2TP Tx PDU, from 1.1.1.2:13661 to 1.1.1.1:13661 (38)
l2tp:[TLS](2/0)Ns=5,Nr=2 *MSGTYPE(StopCCN) *RESULT_CODE(2/0) *ASSND_TUN_ID(6)
```

Here are the respective logs.

Note the output Control tunnel timeout - retry-attempted five, last-interval 8000 ms for the failed attempts.

```
 14:06:21.406 [l2tp-control 50001 debug] [7/0/9133 <l2tpmgr:2> l2tpsnx_proto.c:1474] [callid
42c22625] [context: destination, contextID: 3]  [software internal user outbound protocol-log]
L2TP Tx PDU, from 1.1.1.2:13661 to 1.1.1.1:13661 (20)
l2tp:[TLS](2/0)Ns=4,Nr=2 *MSGTYPE(HELLO)
14:06:22.413 [l2tp-control 50001 debug] [7/0/9133 <l2tpmgr:2> l2tpsnx_proto.c:1474] [callid
42c22625] [context: destination, contextID: 3]  [software internal user outbound protocol-log]
L2TP Tx PDU, from 1.1.1.2:13661 to 1.1.1.1:13661 (20)
l2tp:[TLS](2/0)Ns=4,Nr=2 *MSGTYPE(HELLO)
14:06:24.427 [l2tp-control 50001 debug] [7/0/9133 <l2tpmgr:2> l2tpsnx_proto.c:1474] [callid
42c22625] [context: destination, contextID: 3]  [software internal user outbound protocol-log]
L2TP Tx PDU, from 1.1.1.2:13661 to 1.1.1.1:13661 (20)
l2tp:[TLS](2/0)Ns=4,Nr=2 *MSGTYPE(HE LLO)
14:06:28.451 [l2tp-control 50001 debug] [7/0/9133 <l2tpmgr:2> l2tpsnx_proto.c:1474] [callid
42c22625] [context: destination, contextID: 3]  [software internal user outbound protocol-log]
L2TP Tx PDU, from 1.1.1.2:13661 to 1.1.1.1:13661 (20)
l2tp:[TLS](2/0)Ns=4,Nr=2 *MSGTYPE(HELLO)
14:06:36.498 [l2tp-control 50001 debug] [7/0/9133 <l2tpmgr:2> l2tpsnx_proto.c:1474] [callid
42c22625] [context: destination, contextID: 3]  [software internal user outbound protocol-log]
L2TP Tx PDU, from 1.1.1.2:13661 to 1.1.1.1:13661 (20)
l2tp:[TLS](2/0)Ns=4,Nr=2 *MSGTYPE(HELLO)
14:06:44.446 [l2tp-control 50068 warning] [7/0/9133 <l2tpmgr:2> l2tp.c:14841] [callid 42c22625]
[context: destination, contextID: 3]  [software internal user] L2TP (Local[svc: lac]: 6
Remote[1.1.1.1]: 2): Control tunnel timeout -  retry-attempted 5 , last-interval 8000 ms, Sr 2,
Ss 5, num-pkt-not-acked 1, Sent-Q-len 1, tun-recovery-flag 0, instance-recovery-flag 0, msg-type
Hello
14:06:44.446 [l2tp-control 50001 debug] [7/0/9133 <l2tpmgr:2> l2tpsnx_proto.c:1474] [callid
42c22625] [context: destination, contextID: 3]  [software internal user outbound protocol-log]
L2TP Tx PDU, from 1.1.1.2:13661 to 1.1.1.1:13661 (38)
l2tp:[TLS](2/0)Ns=5,Nr=2 *MSGTYPE(StopCCN) *RESULT_CODE(2/0) *ASSND_TUN_ID(6)
14:06:44.447 [l2tp-control 50069 debug] [7/0/9133 <l2tpmgr:2> l2tpsnx_fsm.c:105] [callid
42c22625] [context: destination, contextID: 3]  [software internal user] l2tp fsm: state
L2TPSNX_STATE_CONNECTED event L2TPSNX_EVNT_PROTO_SESSION_DISCONNECTED
```

And corresponding SNMP trap

```
 14:06:44 2009 Internal trap notification 92 (L2TPTunnelDownPeerUnreachable)  context
destination service lac peer address 1.1.1.1 local address 1.1.1.2
```

Show output considerations

Running the following command will indicate if there have been peer reachability issues with a specific peer (or for all tunnels in a particular lac/lns service)

```
 show l2tp statistics (peer-address <peer ip address> | ((lac-service | lns-service) <lac or lns
service name>))
```

The Active Connections counter matches the number of existing tunnels for that peer there can be more than one, as seen in the output from show l2tp tunnels all from earlier.

The Failed to Connect counter will indicate how many tunnel setup failures have occurred.
The Max Retry Exceeded counter is probably the most important counter, as it indicates failure to connect due to a timeout (each Retry exceeded results in a L2TPTunnelDownPeerUnreachable trap). This information only tells you the frequency of the problem for a given peer, it does not tell you why the timeout occurred. But knowing the frequency can be helpful in putting together the

pieces in the overall troubleshooting process.

The Sessions section gives detail at the subscriber session level (vs. tunnel level)
The Active Sessions counter matches the sum of (if more than one tunnel for a peer) the Active Sess column output from show l2tp tunnels for the particular peer.
The Failed to Connect counter indicates how many sessions have failed to connect. Note that failed session setups do NOT trigger the L2TPTunnelDownPeerUnreachable trap, only failed tunnel setups do.

There is also a counters version of the show l2tp tunnels command that can be useful.

```
show l2tp tunnels counters peer-address <peer address>
```

Finally, at the session level, all of the subscribers for a given peer can be viewed.

```
show l2tp sessions peer-address <peer ip address>
```

The number of subscribers found should match the number of active sessions as discussed.