NAME

ng_l2tp - L2TP protocol netgraph node type

SYNOPSIS

#include <sys/types.h>
#include <netgraph/ng_l2tp.h>

DESCRIPTION

The **l2tp** node type implements the encapsulation layer of the L2TP protocol as described in RFC 2661. This includes adding the L2TP packet header for outgoing packets and verifying and removing it for incoming packets. The node maintains the L2TP sequence number state and handles control session packet acknowledgment and retransmission.

HOOKS

The **l2tp** node type supports the following hooks:

lower	L2TP frames.
ctrl	Control packets.
session_hhhh	Session 0xhhhh data packets.

L2TP control and data packets are transmitted to, and received from, the L2TP peer via the lower hook. Typically this hook would be connected to the inet/dgram/udp hook of an ng_ksocket(4) node for L2TP over UDP.

The ctrl hook connects to the local L2TP management entity. L2TP control messages (without any L2TP headers) are transmitted and received on this hook. Messages written to this hook are guaranteed to be delivered to the peer reliably, in order, and without duplicates.

Packets written to the ctrl hook must contain a two byte session ID prepended to the frame (in network order). This session ID is copied to the outgoing L2TP header. Similarly, packets read from the ctrl hook will have the received session ID prepended.

Once an L2TP session has been created, the corresponding session hook may be used to transmit and receive the session’s data frames: for the session with session ID 0xabcd, the hook is named session_abcd.

CONTROL MESSAGES

This node type supports the generic control messages, plus the following:

NGM_L2TP_SET_CONFIG (**setconfig**)

This command updates the configuration of the node. It takes a *struct ng_l2tp_config* as an argument:

```
/* Configuration for a node */
struct ng_l2tp_config {
    u_char     enabled;       /* enables traffic flow */
    u_char     match_id;      /* tunnel id must match 'tunnel_id' */
    uint16_t   tunnel_id;     /* local tunnel id */
    uint16_t   peer_id;       /* peer's tunnel id */
    uint16_t   peer_win;      /* peer's max recv window size */
    uint16_t   rexmit_max;    /* max retransmits before failure */
    uint16_t   rexmit_max_to; /* max delay between retransmits */
};
```

The *enabled* field enables packet processing. Each time this field is changed back to zero the sequence number state is reset. In this way, reuse of a node is possible.

The *tunnel_id* field configures the local tunnel ID for the control connection. The *match_id* field determines how incoming L2TP packets with a tunnel ID field different from *tunnel_id* are handled. If *match_id* is non-zero, they will be dropped; otherwise, they will be dropped only if the tunnel ID is non-zero. Typically *tunnel_id* is set to the local tunnel ID as soon as it is known and *match_id* is set to non-zero after receipt of the SCCRP or SCCCN control message.

The peer's tunnel ID should be set in *peer_id* as soon as it is learned, typically after receipt of a SCCRQ or SCCRP control message. This value is copied into the L2TP header for outgoing packets.

The *peer_win* field should be set from the "Receive Window Size" AVP received from the peer. The default value for this field is one; zero is an invalid value. As long as *enabled* is non-zero, this value may not be decreased.

The *rexmit_max* and *rexmit_max_to* fields configure packet retransmission. *rexmit_max_to* is the maximum retransmission delay between packets, in seconds. The retransmit delay will start at a small value and increase exponentially up to this limit. The *rexmit_max* sets the maximum number of times a packet will be retransmitted without being acknowledged before a failure condition is declared. Once a failure condition is declared, each additional retransmission will cause the **l2tp** node to send a NGM_L2TP_ACK_FAILURE (**ackfailure**) control message back to the node that sent the last NGM_L2TP_SET_CONFIG. Appropriate action should then be taken to shutdown the control connection.

NGM_L2TP_GET_CONFIG (**getconfig**)

Returns the current configuration as a *struct ng_l2tp_config*.

NGM_L2TP_SET_SESS_CONFIG (**setsessconfig**)

This control message configures a single data session. The corresponding hook must already be connected before sending this command. The argument is a *struct ng_l2tp_sess_config*:

```
/* Configuration for a session hook */
struct ng_l2tp_sess_config {
    uint16_t    session_id;     /* local session id */
    uint16_t    peer_id;        /* peer's session id */
    u_char      control_dseq;   /* whether we control data sequencing */
    u_char      enable_dseq;    /* whether to enable data sequencing */
    u_char      include_length; /* whether to include length field */
};
```

The *session_id* and *peer_id* fields configure the local and remote session IDs, respectively.

The *control_dseq* and *enable_dseq* fields determine whether sequence numbers are used with L2TP data packets. If *enable_dseq* is zero, then no sequence numbers are sent and incoming sequence numbers are ignored. Otherwise, sequence numbers are included on outgoing packets and checked on incoming packets.

If *control_dseq* is non-zero, then the setting of *enable_dseq* will never change except by another NGM_L2TP_SET_SESS_CONFIG control message. If *control_dseq* is zero, then the peer controls whether sequence numbers are used: if an incoming L2TP data packet contains sequence numbers, *enable_dseq* is set to one, and conversely if an incoming L2TP data packet does not contain sequence numbers, *enable_dseq* is set to zero. The current value of *enable_dseq* is always accessible via the NGM_L2TP_GET_SESS_CONFIG control message (see below). Typically an LNS would set *control_dseq* to one while a LAC would set *control_dseq* to zero (if the Sequencing Required AVP were not sent), thus giving control of data packet sequencing to the LNS.

The *include_length* field determines whether the L2TP header length field is included in outgoing L2TP data packets. For incoming packets, the L2TP length field is always checked when present.

NGM_L2TP_GET_SESS_CONFIG (**getsessconfig**)

This command takes a two byte session ID as an argument and returns the current configuration for the corresponding data session as a *struct ng_l2tp_sess_config*. The corresponding session hook must be connected.

NGM_L2TP_GET_STATS (**getstats**)
: This command returns a *struct ng_l2tp_stats* containing statistics of the L2TP tunnel.

NGM_L2TP_CLR_STATS (**clrstats**)
: This command clears the statistics for the L2TP tunnel.

NGM_L2TP_GETCLR_STATS (**getclrstats**)
: Same as NGM_L2TP_GET_STATS, but also atomically clears the statistics as well.

NGM_L2TP_GET_SESSION_STATS (**getsessstats**)
: This command takes a two byte session ID as an argument and returns a *struct ng_l2tp_session_stats* containing statistics for the corresponding data session. The corresponding session hook must be connected.

NGM_L2TP_CLR_SESSION_STATS (**clrsessstats**)
: This command takes a two byte session ID as an argument and clears the statistics for that data session. The corresponding session hook must be connected.

NGM_L2TP_GETCLR_SESSION_STATS (**getclrsessstats**)
: Same as NGM_L2TP_GET_SESSION_STATS, but also atomically clears the statistics as well.

NGM_L2TP_SET_SEQ (**setsequence**)
: This command sets the sequence numbers of a not yet enabled node. It takes a *struct ng_l2tp_seq_config* as argument, where *xack* and *nr* respectively *ns* and *rack* must be the same. This option is particularly useful if one receives and processes the first packet entirely in userspace and wants to hand over further processing to the node.

SHUTDOWN

This node shuts down upon receipt of a NGM_SHUTDOWN control message, or when all hooks have been disconnected.

SEE ALSO

netgraph(4), ng_ksocket(4), ng_ppp(4), ng_pptpgre(4), ngctl(8)

W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter, *Layer Two Tunneling Protocol L2TP*, RFC 2661.

HISTORY

The **l2tp** node type was developed at Packet Design, LLC, *http://www.packetdesign.com/*.

AUTHORS

Archie Cobbs <*archie@packetdesign.com*>