NAME

gld – Generic LAN Driver

SYNOPSIS

#include <sys/stropts.h>
#include <sys/stream.h>
#include <sys/dlpi.h>
#include <sys/ethernet.h>
#include <sys/gld.h>

AVAILABILITY

x86

DESCRIPTION

gld is a multi-threaded, clonable Generic LAN Driver support module which resides in a loadable kernel module, **/kernel/misc/gld**. It implements most of the STREAMS functions and DLPI functionality required of a LAN driver. Several Solaris network drivers are implemented using GLD.

Any Solaris network driver implemented using GLD is divided into two distinct parts: a generic part that deals with STREAMS and DLPI interfaces, and a device-specific part that deals with the particular hardware device. The device-specific module indicates its dependency on the GLD module and registers itself with GLD from within the driver's **attach**(9E) function. After the driver has been successfully loaded, it is a fully DLPI-compliant driver. The device-specific part of the driver calls GLD functions when it receives data or needs some service from GLD. GLD makes indirect calls into the device-specific driver through pointers provided to GLD by the device-specific driver when it registered itself with GLD.

DLPI is an implementation of a standard that defines the services provided by the data link layer. The data link interface is the boundary between the network and the data link layers of the OSI Reference Model. The network layer entity is the user of the services of the data link interface (DLS user). The data link layer entity is the provider of those services (DLS provider). The DLS user accesses the provider using **open**(9E) to establish a STREAM to the DLS provider. The STREAM acts as a communication channel between a DLS user and a DLS provider.

gld and Ethernet V2 and 802.3

Ethernet V2 service and 802.3 mode are provided by GLD. V2 enables a data link service user to access and use any of a variety of conforming data link service providers without special knowledge of the provider's protocol. A Service Access Point (SAP) is the point through which the user will communicate with the service provider. SAP values in the range [0-1500] are treated as equivalent and represent a desire by the user for 802.3 mode. If the value of the SAP field of the **DL_BIND_REQ** is within this range, GLD computes the length, not including initial **M_PROTO** message blocks, of all subsequent **DL_UNITDATA_REQ** messages and transmits 802.3 frames having this value in the MAC frame header type field. All frames received from the media having a type field in this range are assumed to be 802.3 frames and are routed up all open STREAMS that are bound to any SAP value within this range. If more than one STREAM is in 802.3 mode, the frame will be duplicated and routed up multiple STREAMS as **DL_UNITDATA_IND** messages.

gld and Style 1 and 2 Providers

GLD implements both Style 1 and Style 2 providers. The Style 1 provider assigns a physical point of attachment (PPA) based on the major/minor device that has been opened and bound. A PPA is the point at which a system attaches itself to a physical communication medium. All communication on that physical medium funnels through the PPA. The Style 2 provider requires the DLS user to explicitly identify the desired PPA using **DL_ATTACH_REQ**. **open**(9E) creates a STREAM between GLD and the device and **DL_ATTACH_REQ**, then associates a particular PPA with that STREAM. Style 2 is denoted by a minor number of zero. If the minor number is not zero, it denotes the PPA number plus one. In both Style 1 and Style 2 opens, the device is cloned.

GLD implements a connectionless-mode service. Once the STREAM is initialized, connectionless data transfer can begin. Because there is no established connection, the DLS user must identify the destination of each data unit to be transferred.

Implemented DLPI Primitives

GLD implements the following DLPI primitives:

The **DL_INFO_REQ** primitive requests information about the DLPI STREAM. The message consists of one **M_PCPROTO** message block. GLD-based drivers return device-dependent values in the **DL_INFO_ACK** response to this request. However, all GLD-based drivers return the following values:

- The version is **DL_VERSION_2**.
- The service mode is **DL_CLDLS**.
- The provider style is **DL_STYLE1** or **DL_STYLE2**.
- No optional Quality Of Service (QOS) support is present, so the QOS fields are zero.

The **DL_ATTACH_REQ** primitive is called to associate a PPA with a STREAM. This request is needed for Style 2 DLS providers to identify the physical medium over which the communication will transpire. This request may not be issued when using the driver in Style 1 mode. Upon completion, the state changes from **DL_UNATTACHED** to **DL_UNBOUND**. The message consists of one **M_PROTO** message block.

The **DL_DETACH_REQ** primitive requests detach from the physical point of attachment from a STREAM.

The **DL_BIND** and **DL_UNBIND** primitives bind and unbind a DLSAP to the STREAM. The PPA associated with each STREAM will be initialized upon completion of the processing of the **DL_BIND_REQ**. Multiple STREAMS may be bound to the same SAP; each STREAM receives a copy of any packets received for that SAP.

The **DL_ENABMULTI_REQ** and **DL_DISABMULTI_REQ** primitives enable and disable reception of individual multicast group addresses. A set of multicast addresses may be iteratively created and modified on a per-STREAM basis using these primitives. These primitives are accepted by the driver in any state following **DL_ATTACHED**.

The **DL_PROMISCON_REQ** and **DL_PROMISCOFF_REQ** primitives enables and disables promiscuous mode on a per-STREAM basis, either at a physical level or at the SAP level. The DL Provider will route all received messages on the media to the DLS User until either a **DL_DETACH_REQ** or a **DL_PROMISCOFF_REQ** is received or the STREAM is closed.

The **DL_UNITDATA_REQ** primitive is used to send data in a connectionless transfer. Because this is an unacknowledged service, there is no guarantee of delivery. The message consists of one **M_PROTO** message block followed by one or more **M_DATA** blocks containing at least one byte of data.

The **DL_PHYS_ADDR_REQ** primitive returns the 6-octet Ethernet address currently associated (attached) to the STREAM in the **DL_PHYS_ADDR_ACK** primitive. When using style 2, this primitive is only valid following a successful **DL_ATTACH_REQ**.

The **DL_SET_PHYS_ADDR_REQ** primitive changes the 6-octet Ethernet address currently associated (attached) to this STREAM. The credentials of the process which originally opened the STREAM must be superuser or an **EPERM** error is returned in the **DL_ERROR_ACK**. This primitive is destructive in that it affects all other current and future STREAMS attached to this device. An **M_ERROR** is sent up all other STREAMS attached to this device when this primitive on this STREAM is successful. Once changed, all STREAMS subsequently opened and attached to this device will obtain this new physical address. The new physical address will remain in effect until this primitive is used to change the physical address again or the system is rebooted, whichever occurs first.

The **DL_UNITDATA_IND** type is used when a packet is received and is to be passed upstream. The packet is put into an **M_PROTO** message with the primitive set to **DL_UNITDATA_IND**.

The interface between GLD and GLD-based drivers is an internal interface not currently published for external use.

FILES

/kernel/misc/gld　　loadable kernel module

SEE ALSO

dlpi(7P), **attach**(9E), **open**(9E)

WARNINGS

Contrary to the DLPI specification, GLD returns the device's correct address length and broadcast address in **DL_INFO_ACK** even before the device has been attached.