Open Transport CSMA/CD Developer Note

Revision 1.1b14
1/12/96

Table of Contents

Revision History

1/12/96	Corrected som figures
9/15/95	Added Future extensions
3/14/95	Added support for raw mode.
1/20/95	Corrected IPX stuff. Now allow multiple binds to the same SAP.
11/28/94	Creation

Related Documents

Data Link Provider Interface Specification Unix International, OSI Workgroup

Streams Modules and Drivers Unix® SVR4.2 UNIX Press

Apple Shared Library Manager Developer's Guide, by ESD Publications, October 4, 1993, Apple Computer, Inc.

Open Transport Client Developer Note

Logical Link Control, ANSI/IEEE Standard 802.2–1985.

Carrier Sense Multiple Access with Collision Detection, ANSI/IEEE Standard 802.3–1989

Designing PCI Cards and Drivers for Power Macintosh Computers preliminary draft, Apple Computer, Inc.

Introduction

This document describes the Open Transport CSMA/CD driver. It is divided into two parts: The first part describes the driver as it exists in Open Transport version 1.1. The second part describes extensions which will be made in the near future.

Current CSMA/CD Driver

The current Open Transport CSMA/CD driver is a Streams driver that presents a Data Link Provider Interface (DLPI) to its clients. It is based on Revision 2.0.0 of the DLPI Specification, and is a Style 1 provider, supporting the Connectionless Mode primitives. Developers who wish to write CSMA/CD drivers that will interoperate with the Open Transport AppleTalk and TCP/IP implementations should use the information given in this section to guide the implementation.

Supported DLPI Primitives

The following DLPI primitives are supported by the Open Transport CSMA/CD driver. The ones marked with a † are not required by either the Appletalk or TCP/IP stacks:

DL_INFO_REQ

DL_INFO_ACK

DL_BIND_REQ

DL_BIND_ACK

DL_UNBIND_REQ

DL_SUBS_BIND_REQ

DL_SUBS_BIND_ACK

DL_ENABLEMULTI_REQ

DL_DISABLEMULTI_REQ

DL_OK_ACK

DL_ERROR_ACK

DL_UNITDATA_REQ

DL_UNITDATA_IND

DL_TEST_REQ †

DL_TEST_IND †

DL_TEST_RES †

DL_TEST_CON †

DL_XID_REQ †

DL_XID_IND †

DL_XID_RES †

DL_XID_CON †

DL_PHYS_ADDR_REQ

DL_PHYS_ADDR_ACK

Future versions of the driver will also support these additional primitives:

DL_GET_STATISTICS_REQ †

DL_GET_STATISTICS_ACK †

DL_PROMISCON_REQ †

DL_PROMISCOFF_REQ†

Extensions to DLPI

In addition to supporting the DLPI primitives listed above, the Open Transport CSMA/CD driver includes extensions to support Mentat's Fast Path mode (described later in this document). This includes the handling of M_IOCTL messages with a type of DL_IOC_HDR_INFO, and special handling of M_DATA messages. It also defines several special M_IOCTL messages to enable the reception of raw packets and to inform it what kind of framing the client expects.

Packet Formats

The Open Transport CSMA/CD driver recognizes three packet formats. They are Ethernet, 802.2, and Novell "Raw 802.3". The details of the packet format are largely hidden from the client by the driver

The type of packets the driver will handle is specified at bind time.

In all three packet formats, the first six bytes are the destination hardware address, the next six bytes are the source hardware address. After this is a protocol dependent section, and finally the packet data.

Ethernet Packets

In Ethernet packets, the protocol dependent section consists of a two byte protocol type field. This field has a value in the range 1501_{10} – $65535_{10.}$ ($5DD_{16}$ – $FFFF_{16}$).

802.2 Packets

In 802.2 packets, the protocol dependent section consists of a two byte length word, a one byte Destination SAP (DSAP), a one byte Source SAP (SSAP), a Control byte and an optional five byte SNAP field. Thus this section of the packet can be either five or ten bytes long.

IPX Packets

IPX payloads may be carried in any one of three frames. In addition to Ethernet and 802.2, an IPX packet may be framed in what Novell calls a "Raw 802.3" packet. In this case, the protocol dependent section consists only of a two byte length word. However, to make it possible to distinguish these packets from 802.2 packets, Novell specifies that the first two bytes of the data section are always set to FF_{16}.

For brevity we will refer to these packets as IPX packets in the rest of this document.

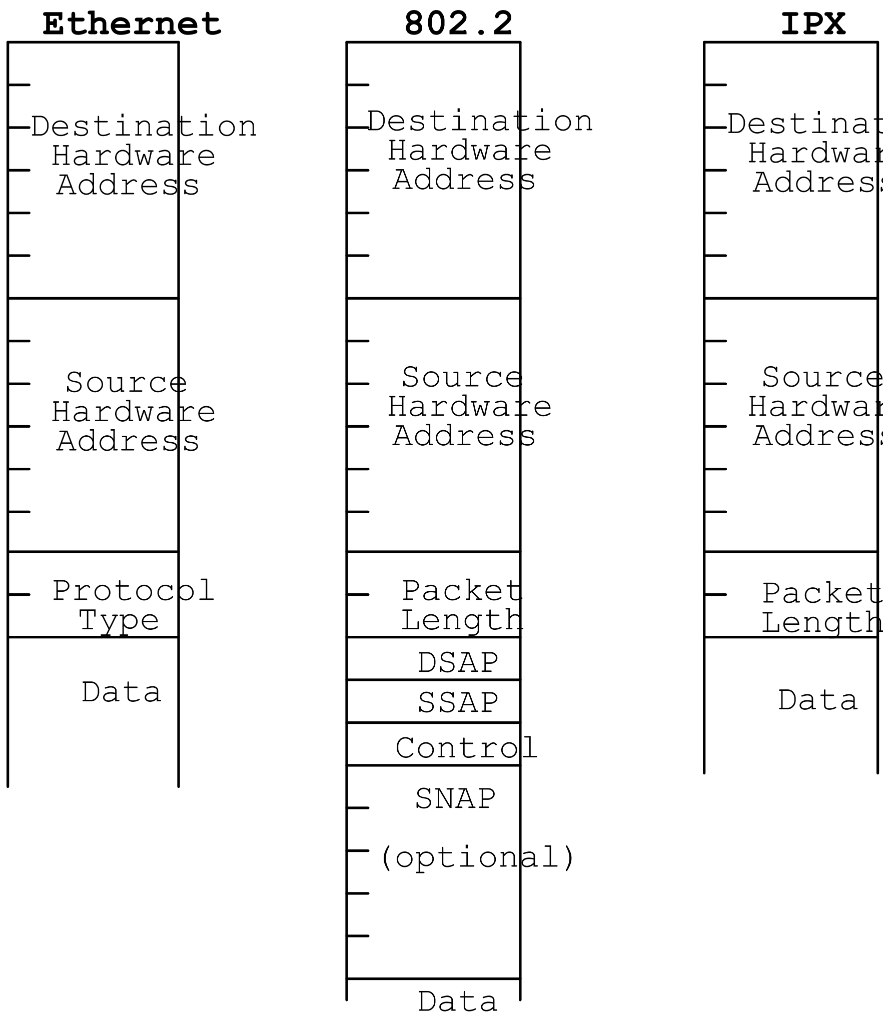

Figure 1 - Packet Formats Recognized by the CSMA/CD Driver

Address Formats

Addresses used by the Open Transport CSMA/CD driver consist of two parts, a hardware address, and a protocol dependent field. The hardware address is a six byte physical address. A hardware address of all ones is the broadcast address. If a hardware address is not all ones but the low bit of the first (leftmost) byte is set, then the address is a multicast address. The protocol address consists of a two byte value called a DLSAP (since it corresponds to the DLSAP defined in the DLPI specification) optionally followed by a five byte SNAP. The protocol address, when present, is appended to the hardware address.

Ethernet

In Ethernet the DLSAP corresponds to the protocol type field.

802.2

In 802.2 packets, the DLSAP corresponds to the SSAP (in a DL_BIND_REQ, DL_BIND_ACK or in the source address field of a DL_UNITDATA_IND), or the DSAP (in a DL_UNITDATA_REQ or in the destination address field of a DL_UNITDATA_IND). If the DLSAP is AA_{16}, then it must be followed by a five byte SNAP.

IPX

In IPX packets, the DLSAP is always $00FF_{16}$.

Binding

The information passed in a Bind Request is a function of the type of packets to be handled by this stream.

Ethernet

To bind to an Ethernet protocol, the client sends a DL_BIND_REQ with the *dl_sap* field set to the protocol type. This is a value in the range 1501_{10} – $65535_{10.}$ ($5DD_{16}$ – $FFFF_{16}$). The *dl_xidtst_flg* field is ignored.

802.2

To bind to an 802.2 address, the client sends a DL_BIND_REQ with the *dl_sap* field set to the SSAP. This is an even value in the range 0_{10} – $254_{10.}$ (0_{16} – FE_{16}). The *dl_xidtst_flg* field may optionally have either or both of the DL_AUTO_XID or DL_AUTO_TEST bits set.

If the SSAP is AA_{16}, then the client should follow the acknowledgment of the bind with a DL_SUBS_BIND_REQ with a five byte SNAP. The *dl_subs_bind_class* field should be set to DL_HIERARCHICAL_BIND.

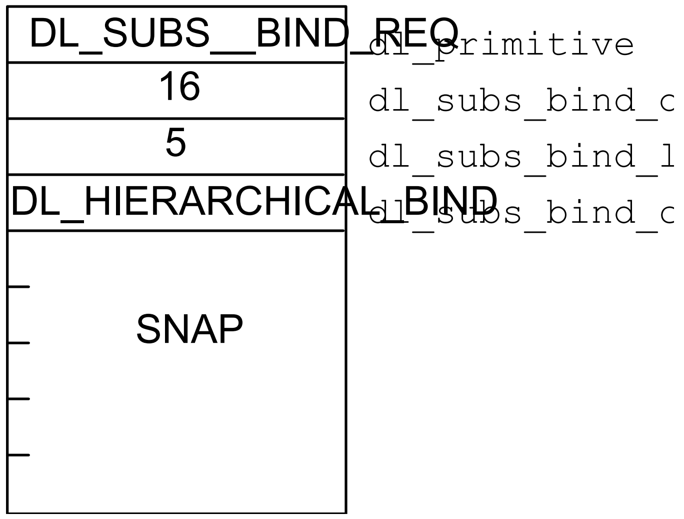

Figure 2 - Subsbind Message for enabling a SNAP

After successfully binding to an 802.2 SAP, the client may enable a Group SAP by sending a DL_SUBS_BIND_REQ with a two byte DLSAP containing the Group SAP. Valid Group SAPs are odd numbers in the range 1_{10} – $253_{10.}$ (1_{16} – FD_{16}). In this case, the *dl_subs_bind_class* field should be set to DL_PEER_BIND. Note that SAP $255_{10.}$ (FF_{16}) is the Global (Broadcast) SAP, and is always enabled.

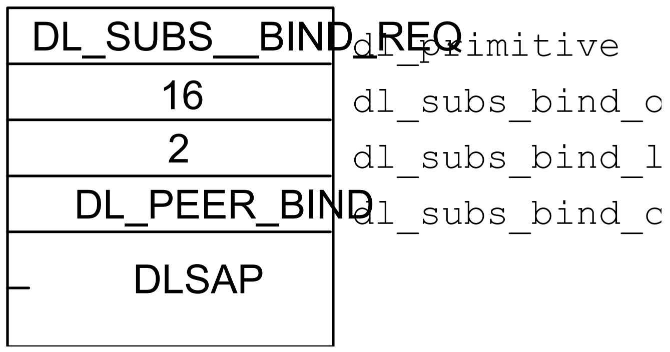

Figure 3 - Subsbind Message for enabling a Group SAP

As a special case, a client may request that it receive all 802.2 packets that come in. It does so by sending a DL_SUBS_BIND_REQ with a two byte DLSAP set to 0. In this case, the *dl_subs_bind_class* field should be set to DL_PEER_BIND

IPX

To bind to an IPX protocol, the client sends a DL_BIND_REQ with the *dl_sap* field set to 255_{10} (FF_{16}). The *dl_xidtst_flg* field is ignored.

In all three cases, the *dl_max_conind* field should be set to 0 and the *dl_service_mode* field must be set to the constant DL_CLDLS.

Note that the DLPI specification leaves open the possibility that several streams on the same hardware port could be bound to a single DLSAP. This feature is *explicitly* supported by the Open Transport CSMA/CD Driver. If a packet arrives addressed to two or more streams simultaneously, each stream receives a copy of the packet.

Multicasts

A multicast address may be enabled on a driver with the DL_ENABMULTI_REQ message. The value must be a valid multicast address as defined in the section *Address Formats*, above.

Similarly a multicast address may be disabled on a driver with the DL_DISABMULTI_REQ message. The value must be a valid multicast address that was enabled on that particular stream with a prior DL_ENABMULTI_REQ.

Sending And Receiving

Sending Packets

Packets are sent with the DL_UNITDATA_REQ message. If the destination has the same protocol address as the sender, it is only necessary to supply the hardware address of the destination, otherwise the full address must be used. Note that only a stream bound to the IPX SAP can send to another IPX stream.

“Fast Path” Packets

In order to support Mentat’s “Fast Path” mode, the Open Transport CSMA/CD driver treats M_DATA messages as fully formed ("Raw") packets, including all addresses and headers. The only modification made before sending the packet to the hardware is to check for a zero in the 802.2 length field, and if found setting it to the appropriate value.

Receiving Packets

Incoming packets are passed to the client in DL_UNITDATA_IND messages. The *dl_group_address* field is set to zero if the packet was addressed to a standard Ethernet Address, *keaMulticast* if the packet was addressed to a multicast address and *keaBroadcast* if the packet was addressed to a broadcast address, where *kaeMulticast* and *kaeBroadcast* are constants, currently set to 1 and 2 respectively.

The data portion of the message consists of everything following the protocol–dependent section.

Test and XID Packets

The driver includes support for 802.2 Test and XID packets

If the client requested automatic handling of Test (XID) packets by setting the DL_AUTO_TEST (DL_AUTO_XID) bit in the *dl_xidtest_flag* field of the Bind request when binding to an 802.2 DLSAP, then the driver will respond to incoming Test (XID) packets without notifying the client. If automatic handling has been requested, the client may not send Test (XID) packets.

If the client did not request automatic handling of Test (XID) packets, then incoming Test (XID) packets will be passed up to the client as DL_TEST_IND (DL_XID_IND) messages. The client should respond to them with a DL_TEST_RES (DL_XID_RES) message.

If automatic handling has not been requested, the client may send Test (XID) packets with a DL_TEST_REQ (DL_XID_REQ) message. Any responses are passed back to the client as DL_TEST_CON (DL_XID_CON) message.

Attempts by non-802.2 streams to send DL_TEST_REQ, DL_XID_REQ, DL_TEST_RES or DL_XID_RES messages will be quietly ignored.

Promiscuous Mode

The DLPI specification defines three levels of promiscuous mode: DL_PROMISC_PHYS, DL_PROMISC_SAP and DL_PROMISC_MULTI. The specification is notably vague as to exactly what these levels mean. Working with other Streams developers, we have come up with the following definitions

DL_PROMISC_PHYS

If the DLPI provider is in DL_UNBOUND state, the DLPI user receives all traffic on the wire regardless of MAC address or SAP.

If the DLPI provider is in DL_IDLE state, the DLPI user receives all traffic on the wire destined for the bound SAP(s), regardless of MAC address.

DL_PROMISC_SAP

If the DLPI provider is in DL_UNBOUND state, the DLPI user receives all traffic destined for this interface (physical address match, multicast address match or broadcast address) which match any SAP bound by any DLPI user of this interface.

If the DLPI provider is in DL_IDLE state, the DLPI user receives all traffic destined for this interface (physical address match, multicast address match or broadcast address) and which match the bound SAP.

DL_PROMISC_MULTI

If the DLPI provider is in DL_UNBOUND state, the DLPI user receives all multicast packets on the wire, regardless of   SAP.

If the DLPI provider is in DL_IDLE state, the DLPI user receives all multicast packets on the wire destined for the bound SAP(s).

Support for the DL_PROMISCON_REQ/DL_PROMISCOFF_REQ pair is not required for normal operation of the driver. If you do not support them, be sure to reply to the request with a DL_ERROR_ACK with the error code set to DL_NOTSUPPORTED.

Extensions to DLPI

The driver includes several optional extensions to the DLPI definition.

Fast Path Mode

"Fast Path" is an optional optimization wherein the driver supplies the client with pre-computed packet header for a given destination, The client caches the header, and copies it directly into packets addressed to that destination before passing them to the driver.

The client first requests a header by sending the driver an M_IOCTL message with its ioc_cmd field set to DL_IOC_HDR_INFO and its chained data block containing the *dl_unitdata_req_t* structure that the client would normally use to send packets to that particular destination. If the driver does not support fast path, it simply responds with an M_IOCNAK message. If the driver supports fast path, it responds with a M_IOCACK message with the chained data block containing the precomputed header. (Note that in the case of 802.2 packets, the length field of the precomputed header is set to zero).

The client then prepends the header to outgoing packets and passes them to the driver as M_DATA messages. The driver then sends the packet as is, filling in the 802.2 length field if necessary.

Note that the data block returned in the M_IOCACK should not be modified by the client, and it should always be copied with copyb rather than dupb, since the driver may modify it before sending the packet.

Once the driver has acknowledged a DL_IOC_HDR_INFO message, its handling of incoming packets changes as well. If an incoming packet is a directed packet, its header is stripped off and the data portion is delivered to the client as an M_DATA message. Non-directed packets (*e.g.* packets sent to a Multicast or Broadcast address) are not affected: they continue to be delivered to the client as DL_UNIT_DATA_IND messages.

Raw Packets

Occasionally, a client may wish to send or receive "Raw" packets, *i.e.* packets with the link and protocol headers attached. To send raw packets the client merely sends them as M_DATA messages, as described in the "Fast Path" section above.

A client that wishes to receive raw packets may send an `M_IOCTL` message with the `ioc_cmd` field set to *kOTSetRawMode* and its chained data block containing a UInt32 value. The value can be either *kOTRawRcvOn kOTRawRcvOnWithTimeStamp* or *kOTRawRcvOff*, to turn the reception of raw packets on or off, respectively. If the driver supports the delivery of raw packets, it will respond with an M_IOCACK message. If not, it will respond with M_IOCNAK message.

Any raw packets received will have the *keaRawPacketBit* set in the *dl_group_address* field of the corresponding `dl_unitdata_ind_t`.

If the constant *kOTRawRcvOnWithTimeStamp* is set, then the first 64 bits (8 bytes) of the data will be a 64-bit timestamp obtained with the `OTGetTimeStamp` function. Packets received with this timestamp will have the *keaTimeStampBit* set in the *dl_group_address* field of the corresponding `dl_unitdata_ind_t`.

Fast Path and Raw Mode

If the client enables Raw Mode and Fast Path, directed packets will not have their header stripped before being delivered to the client as M_DATA's. Because there is no header block associated with the packet, there is no way to designate that it is a raw packet, or that a timestamp is appended to the front of the packet (if *kOTRawRcvOnWithTimeStamp* was used).

Framing and DL_INFO_REQ

In order to support the TCP/IP stack available with Open Transport, CSMA/CD drivers must support both Ethernet and 802.2 framing (including full SAP/SNAP binding). Because the DLPI specification does not allow for a driver supporting multiple kinds of framing, it is ambiguous how to fill out the *dl_mac_type* field of a `dl_info_ack_t`. Open Transport has defined that the default value of this field should be `DL_ETHER`. Clients may send an `M_IOCTL` message with the *ioc_cmd* field set to *kOTSetFramingType* and its chained data block containing a UInt32 value with a single bit set. If this value is the constant *kOTFraming8022*, then subsequent `DL_INFO_ACK` requests should set the *dl_mac_type* field to `DL_CSMACD`. If the value is not that constant, then subsequent `DL_INFO_REQ` requests should set the *dl_mac_type* field to `DL_ETHER`. The return value of this IOCtl is the current value of the framing type. If the value is the constant *kOTGetFramingValue*, then the driver should not changed the framing type, but merely return the current value of the framing type.

It should be emphasized that the only thing this M_IOCTL affects is the contents of the `DL_INFO_ACK`. The framing that is actually used by the driver is specified in the bind.

Token Ring and FDDI Drivers

The Open Transport Token Ring and FDDI Drivers are identical to the CSMA/CD Driver with the following exceptions:

Packet Types

Only 802.2 packets are supported.

Address Formats

Only 802.2 addressing is supported.

For Token Ring, a hardware multicast is a hardware address with the two high order bits of the leftmost byte set to one.

Future Extensions

There are a number of extensions planned for the Open Transport CSMA/CD and related drivers. These include

- Defining and supporting the DL_GET_STATISTICS_REQ/ACK messages.
- Defining a Style 2 DLPI which would allow multiple hardware address.
- Miscellaneous extensions to the DLPI standard to support different transports.

Statistics

In order to properly support SNMP, the DL_GET_STATISTICS messages will have to be supported. This requires that the format of the information returned be defined and documented.

Style 2 DLPI

In order to support systems where we need multiple hardware addresses on a single interface, we will expand the definition of the DLPI to allow Style 2 providers. In this case the DL_ATTACH_REQ message will be used to indicate which interface is being used.

In the *dl_attach_req_t*, the *dl_ppa* field will contain an ordinal indicating the interface to use. A value of zero will indicate the default interface and an non-zero value (TBD) will indicate alternate interfaces.

Optionally, the DL_SET_PHYS_ADDR_REQ message may be used to set the actual hardware address recognized by the alternate interfaces.

Other Extensions

We plan to define the following additional values for the *dl_mac_type* field in the DL_INFO_ACK.

DL_FC	Fibre Channel
DL_ATM	ATM AAL5/Raw
DL_IPATM	ATM Classical IP
DL_X25	X.25
DL_ISDN	ISDN
DL_HIPPI	HIPPI
DL_100VG	100 Based VG Ethernet
DL_100VGTPR	100 Based VG TokenRing
DL_ETH_CSMA	802.3 and Ethernet

Index

Note that the 802.3 specification guarantees that this length will be less than 1501_{10}, so that it is always possible to differentiate Ethernet and 802.2 packets based on this field.

If we were pedantic, we would point out that these are really 802.3 packets with an 802.2 payload. However, demonstrating our aversion to pedantry, we will continue to refer to these as 802.2 packets.

A legal 802.2 packet will never have both the SSAP and DSAP fields set to FF_{16}.

Attempting to do a hierarchical subsbind to any SAP other than AA_{16} will result in an error.

For a discussion of Group and Global SAPs, refer to the 802.2 specification listed in the *Related Documents* section.

When sending packets to DLSAP FF_{16} there is an ambiguity as to whether it is destined for an 802.2 Global SAP or to an IPX SAP. This resolved by declaring that only an IPX endpoint can send to another IPX endpoint, and an IPX endpoint cannot send to a Global SAP.

This feature is necessary to support Appletalk backward compatibility.

For a description of Test and XID packets, refer to the 802.2 specification. In general, most clients should merely enable automatic handling and let the driver deal with them.

Streams drivers are defined to NAK any IOCTLs that they do not understand.

It is strongly recommended that drivers support this feature since Appletalk backwards compatibility, among other things, requires it.