Open Transport LocalTalk Developer Note

Revision 1
2/29/96

Table of Contents

Revision History

2/29/96	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 implementation of an Open Transport LocalTalk driver or any other non-extended AppleTalk link layer.

LocalTalk Driver

The LocalTalk driver should be implemented as a STREAMS module providing 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 LocalTalk or other AppleTalk non-extended link layer drivers that will interoperate with the Open Transport AppleTalk implementation should use the information given in this section to guide the implementation.

Supported DLPI Primitives

The following DLPI primitives should be supported by the Open Transport LocalTalk driver. The ones marked with a † are not required by Appletalk stacks, but should be implemented to provide a robust implementation

DL_INFO_REQ

DL_INFO_ACK

DL_BIND_REQ

DL_BIND_ACK

DL_UNBIND_REQ

DL_SUBS_BIND_REQ

DL_SUBS_BIND_ACK
DL_SUBSUNBIND_REQ
DL_SUBSUNBIND_ACK
DL_OK_ACK
DL_ERROR_ACK
DL_UNITDATA_REQ
DL_UNITDATA_IND

DL_PHYS_ADDR_REQ
DL_PHYS_ADDR_ACK
DL_GET_STATISTICS_REQ †
DL_GET_STATISTICS_ACK †
DL_PROMISCON_REQ †
DL_PROMISCOFF_REQ†

Address Formats

Addresses used by the Open Transport LocalTalk driver vary depending on the message and configuration. There are 1-byte, 2-byte, and 3-byte address formats. The formats to support are described under the various functions outlined later in this document.

Binding

The information passed in a Bind Request is a function of the type of packets to be handled by this stream. The first bind to a LocalTalk device should initiate address acquisition. This should never be done in the `Open`routine of your driver.

There are two types of bind requests. The first specifies both a protocol type and an address. The protocol type is the lower 8 bits of the `DLSAP` value, and the address is the 2nd 8 bits (i.e. the address requested is (dlsap >> 8) & 0xff) and the protocol type is dlsap & 0xff). If the address has not already been obtained, then the requested address should attempt to be obtained, and if an address collision is detected, the bind should be failed. If the address has already been obtained, then attach the stream to the requested protocol type, if possible, and complete the bind.

The second specifies only a protocol type (i.e. (dlsap & 0xff00) == 0). In this case, if an address has not already been obtained, LocalTalk should attempt to obtain any address that it can, trying all 254 legal addresses using a random number generator.

In order to make it easier to implement AppleTalk, a bind to either protocol type 1 or 2 should imply the other. This way, AppleTalk does not have to open two different streams to get both long and short header packets.

When acknowledging the bind, the ack structure should be filled out as:

`dl_primitive`	= `DL_BIND_ACK`
`dl_sap`	= protocol type requested (no address information)
`dl_addr_length`	= 1
`dl_addr_offset`	= `DL_BIND_ACK_SIZE`
`dl_max_conind`	= 0
`dl_xidtest_flg`	= 0

The actual one-byte address of the link should be stored immediately after the end of the `dl_bind_ack_t` structure (of course, if you change what you store in the `dl_addr_offset` field, you can store the one-byte address anywhere in the message).

If the requested address is already in use, either locally or remotely, the bind should be failed with the error

DL_NOADDR. If the dl_subs_sap_length is not 1, or the requested address is 0 or 0xff, the bind should be failed with the error DL_BADADDR.

Clients unbind using the DL_UNBIND_REQ message. There are no parameters to this message. Remember that a DL_UNBIND_REQ implies that all multinodes for the targeted stream are removed as well (see the next section).

Adding and RemovingMultinodes

A LocalTalk implementation needs to be able to support multiple nodes. The node acquired during the bind operation is the primary node of the machine. The DL_SUBS_BIND_REQ message is used to request a multinode from the LocalTalk driver. The fields of the DL_SUBS_BIND_REQ should be filled out as follows by the client:

dl_primitive	= DL_SUBS_BIND_REQ
dl_subs_sap_offset	= some offset into the message
dl_subs_sap_length	= 1
dl_subs_bind_class	= DL_PEER_BIND

The actual one-byte address for the requested multinode should be stored at the offset in the message indicated by the dl_subs_sap_offset field.

If the requested address is already in use, either locally or remotely, the subsBind should be failed with the error DL_NOADDR. If the dl_subs_sap_length is not 1, or the requested address is 0 or 0xff, the subsBind should be failed with the error DL_BADADDR.

Multinode requestors are responsible for all packet dispatch, so incoming packets of all protocol types that match the multinode address are sent to the client.

A DL_SUBS_UNBIND_REQ must be supported for removing multinodes. The fields of this request should be filled out as follows:

dl_primitive	= DL_SUBS_UNBIND_REQ
dl_subs_sap_offset	= some offset into the message
dl_subs_sap_length	= 1

The error code DL_BADADDR should be used if the address is either in an improper format, or it is not bound to.

Answering DL__INFO_REQ requests

LocalTalk must respond to all DL_INFO_REQ requests from clients. The appropriate responses are:

dl_primitive	= DL_INFO_ACK

dl_max_sdu	= Maximum bytes in a packet, normally 603
dl_min_sdu	= Minimum bytes in a packet, normally 8
dl_addr_length	= 1 (0 if you have not yet acquired an address)
dl_mac_type	= DL_OTHER (for now)
dl_reserved	= 0
dl_current_state	= whatever your current DLPI state is
dl_sap_length	= 1
dl_service_mode	= DL_CLDLS
dl_qos_length	= 0
dl_qos_offset	= 0
dl_qos_range_length	= 0
dl_qos_range_offset	= 0
dl_provider_style	= DL_STYLE1
dl_addr_offset	= some offset into the message (or 0 if unbound)
dl_version	= DL_CURRENT_VERSION
dl_brdcst_addr_length	= 1
dl_brdcst_addr_offset	= some offset into the message
dl_growth	= 0

If you are currently bound, you should place the one-byte node address at the offset indicated by the dl_addr_offset field.

Multicasts

LocalTalk does not support multicast, so any request for multicast addressing should be return an error.

Sending And Receiving

Sending Packets

Packets are sent with the DL_UNITDATA_REQ message. In order to fully support multinodes and AppleTalk, LocalTalk must support three formats for the destination address. The first is destination address of 1 byte, which is suitable for non-multinode packets, where the protocol type is unambiguous (which is only true for non-AppleTalk types, since AppleTalk binds to 2 protocol types with a single bind). In this case, the protocol type is inferred from the bind information associated with the current instance of the module. For AppleTalk packets, a 2-byte format is specified for the destination - the first byte is the destination node, and the second byte is the protocol type (1 or 2). For multi-node packets, a 3-byte format is specified for the destination - the first byte is the destination node, the second byte is the src node, and the third byte is the protocol type.

In addition, LocalTalk drivers should accept M_DATA messages. For M_DATA messages, the data packet is preceded by the 3-byte LocalTalk header. If you are a non-extended driver that needs some other format of header, you will need to remove the 3-byte header, and use the information from it to create the new header.

Receiving Packets

Incoming packets are passed to the client in DL_UNITDATA_IND messages. The fields should be set as:

dl_primitive	= DL_UNITDATA_IND
dl_dest_addr_length	= 1 or 2
dl_dest_addr_offset	= some offset in message
dl_src_addr_length	= 1
dl_src_addr_offset	= some offset in message

`dl_group_address` = 0 if directed packet, 1 if broadcast

The source address is the one-byte address indicating the source node of the packet. The destination address length may be set to 1 or 2. It may only be set to 1 (indicating just the node address of the packet) where not specifying the incoming protocol type is unambiguous (i.e. this is not a multi-node packet, and the protocol type is a specific number). For AppleTalk (where it's ambiguous because of the "implied" protocol type 1 and 2 binding), and for multinode delivery, you need to use the 2-byte format specified in the bind, where the first byte is the node address and the second byte is the protocol type.

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. The following sections define the workings of promiscuous mode for LocalTalk:

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 protocol types.

If the DLPI provider is in DL_IDLE state, the DLPI user receives all traffic on the wire destined for the bound protocol type, 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 or broadcast address) which match any protocol type 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 or broadcast address) and which match the bound protocol type.

DL_PROMISC_MULTI

This mode of promiscuous is not supported for LocalTalk, since LocalTalk does not support multicast addressing.

Support for the DL_PROMISCON_REQ/DL_PROMISCOFF_REQ pair is not required for normal operation of the driver. However, for your driver to be usable by network peek programs, their support is necessary. 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. The most important variant is the DL_PROMISC_PHYS in the DL_UNBOUND state.

Statistics

In order to support SNMP, the driver should support the `DL_GET_STATISTICS_REQ` and `DL_SET_STATISTICS_REQ` calls.

The `DL_GET_STATISTICS` _REQ call should return a `DL_GET_STATISTICS_ACK` that contains the complete LocalTalk MIB as defined in RFC# xxxxx (to be filled in later).

The implementation of the `DL_SET_STATISTICS_REQ` call is still being investigated.

Index