Macintosh OS Ethernet Driver
Design Specification Proposal

Alan B. Oppenheimer

May 18, 1987

The following is a proposal for a Macintosh Ethernet driver. It is envisioned to be useable for any Ethernet implementation (card, SCSI device, etc.). The interface is such that it should also be applicable to other networks (e.g. Token Ring) with little or no changes. This proposal is for a general-purpose Ethernet driver, and does not address the issue of AppleTalk protocols running on Ethernet hardware. It conforms to IEEE 802.2 Type 1 service.

Macintosh network developers are familiar with the AppleTalk driver interface; this interface is patterned directly after that. See the AppleTalk Manager chapter in volume 2 of *Inside Macintosh* for data structure details.

<u>Opening the Ethernet driver:</u> The Ethernet driver is opened through a device manager Open call, indicating the slot (for a Mac II) in which the Ethernet card is installed. The driver is initially opened in "AppleTalk" mode. In this mode, packets sent by the driver are restricted to a maximum of 768 bytes. This is big enough to encapsulate AppleTalk packets, and allows more of a shared buffer pool to be allocated for packet reception. The driver can be changed to "general" mode, where it will transmit any valid Ethernet packet, through a control call defined below.

The name of the Ethernet driver is '.ENET'.

<u>Commands to the Ethernet driver:</u> Commands to the Ethernet driver are specified by means of Device Manager Control calls, with arguments passed in the queue element starting at CSParam. The following is a list of commands:

EAttachPH: attach a "protocol handler" to the driver. Arguments are a two-byte protocol type and a handler address. The handler will be called (see "protocol handlers" below) when a packet of its type is received. If the protocol handler address is zero, a "default" protocol handler will be supplied by the driver which will enable the caller to issue standard read calls for that protocol type (see the ERead call).

Note: to attach (or detach) a handler for 802.3 (which use protocol types 0 through $5DC), specify protocol type zero.

EDetachPH: detach (remove) the protocol handler for the given protocol type. All pending reads are aborted with an error.

EWrite: write out a packet on the Ethernet. The only argument is a WDS (write data structure) pointer. The WDS is a series of length/pointer pairs, terminated by a zero length. The data is "gathered" from each of the WDS entries, in the order provided. The first entry must start with the destination address (6 bytes for Ethernet), and then contain 6 unused bytes (in the Ethernet case) followed by the two-byte protocol type field. Data may then follow if desired.

If the total length of the packet is too big (greater than 1514 bytes if "general", or 768 bytes if "AppleTalk" mode) an error is returned. If the total length of the packet is too small (less than 60 bytes for Ethernet), the packet is padded with zeroes to the minimum length.

ERead: read in a packet. The ERead call can only be used if an EAttachPH with a zero handler address has been issued for the desired protocol type. ERead takes as arguments the protocol type and a buffer pointer and size, and returns the actual size of the packet read. The driver dequeues the ERead call from the system queue, so more than one ERead call can be active concurrently (they will be queued internally). The entire packet, including headers, is placed in the buffer. If there is not enough room for the packet, as much as will fit is placed in the buffer, and an error is returned.

ERdCancel: cancel a particular ERead call. The only argument is the queue element pointer of the ERead to be cancelled. The ERead will be completed with an error if it is still active.

EGetInfo: get driver info. Takes as argument a buffer pointer and size. Returns as the first six bytes in the buffer the Ethernet address for the node on which the driver is running. Also can return driver-specific information such as statistics (none is currently defined).

ESetGeneral: changes modes. No arguments. Changes the driver from "AppleTalk" to "general" mode, if not already in "general" mode. Once this is done, the driver can not be changed back. Note that this operation may involve a reset of the hardware, and could cause loss of an incoming packet.

<u>The protocol handler:</u> the driver's protocol handler structure mimics that of the AppleTalk driver. Upon reception of a packet for a given protocol type, the LAP-level header is read in to internal driver space and the handler is called "on the fly" to process the rest of the packet. The handler calls a driver routine to read in a specified number of bytes into a specified location. It may do this as many times as it wants, until it has processed the entire packet.

Register setup and restrictions, summarized below, will be essentially the same as for AppleTalk protocol handlers. See *Inside Mac* for details. Note that there will not be the same timing constraints as for AppleTalk, as hardware buffering will generally be provided.

The protocol handler is called as follows:

A0, A1 : for internal use by the driver; must be preserved until ReadRest
A2 : free
A3 : pointer past data link header bytes (i.e. byte after the Type field)
A4 : pointer to "ReadPacket" routine
A5 : free
D0, D2, D3 : free
D1 : number of bytes in packet left to read (preserve, except as below)

The protocol handler must examine the data link header and call one of two routines.

To read in a specific number of bytes, it should call "ReadPacket", which is pointed to by A4. It can then process these bytes and call "ReadPacket" again, any number of times. To read in all remaining bytes in the packet, it should call "ReadRest", which begins two bytes past ReadPacket. ReadPacket and ReadRest are called as follows. Note that after calling ReadRest, standard interrupt conventions apply, i.e. the protocol handler can freely use only D0-D3 and A0-A3.

On entry:

A3: buffer to read into
D3 : number of bytes to read (or max buffer size for ReadRest)

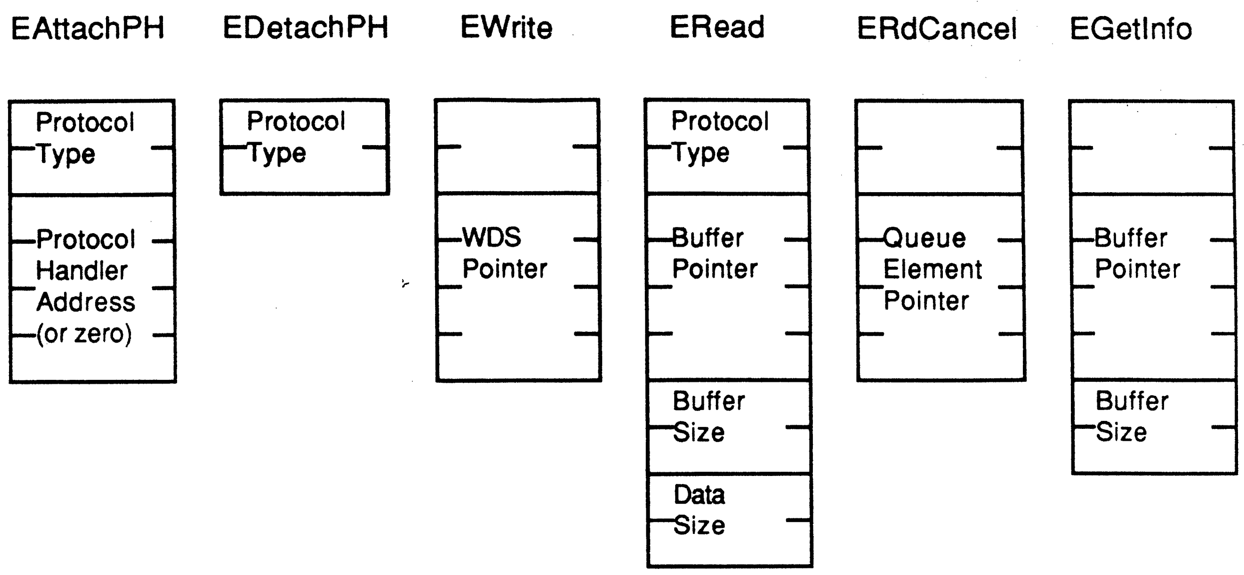
On exit:

D0 : modified
D1 : number of bytes left to read
D2 : preserved
D3 : = 0 if requested number of bytes read
< 0 if packet was (-D3) bytes to large to fit in buffer (ReadRest)
> 0 if D3 bytes weren't read (ReadRest)
A0-A2 : preserved
A3 : one byte past last byte read

Note that the protocol handler may at any time reduce D1 (the number of bytes left in the packet) to eliminate bytes that it has determined to be "pad" bytes through examining parts of the packet. Other than this, D1 should not be modified until ReadRest is called.

<u>Closing the Ethernet driver:</u> the Ethernet driver is closed through the Device Manager Close call. All pending reads are aborted.

<u>Queue element formats:</u> queue element formats are specified below, starting at offset CSParam in the queue element.



Queue Element Formats