Network Working Group
Internet-Draft
<draft-ietf-ip1394-ipv4-05.txt>
Expires: May, 1998

P. Johansson
Congruent Software, Inc.

IPv4 over IEEE 1394

STATUS OF THIS DOCUMENT

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

To view the entire list of current Internet-Drafts, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), ftp.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

ABSTRACT

This document specifies how to use IEEE Std 1394-1995, Standard for a High Performance Serial Bus (and its supplements), for the transport of Internet Protocol Version 4 (IPv4) datagrams. It defines the necessary methods, data structures and code for that purpose and additionally defines a standard method for Address Resolution Protocol (ARP).

TABLE OF CONTENTS

1. INTRODUCTION

This document specifies how to use IEEE Std 1394-1995, Standard for a High Performance Serial Bus (and its supplements), for the transport of Internet Protocol Version 4 (IPv4) datagrams. It defines the necessary methods, data structures and codes for that purpose and additionally defines a standard method for Address Resolution Protocol (ARP).

The group of IEEE standards and supplements, draft or approved, related to IEEE Std 1394-1995 is hereafter referred to either as 1394 or as Serial Bus.

1394 is an interconnect (bus) that conforms to the CSR architecture, ISO/IEC 13213:1994. Serial Bus implements communications between nodes over shared physical media at speeds that range from 100 to 400 Mbps. Both consumer electronic applications (such as digital VCR’s, stereo systems, televisions and camcorders) and traditional desktop computer applications (*e.g.*, mass storage, printers and tapes), have adopted 1394. Serial Bus is unique in its relevance to both consumer electronic and computer domains and is expected to form the basis of a home or small office network that combines both types of devices.

The CSR architecture describes a memory-mapped address space that Serial Bus implements as a 64-bit fixed addressing scheme. Within the address space, ten bits are allocated for bus ID (up to a maximum of 1,023 buses), six are allocated for node physical ID (up to 63 per bus) while the remaining 48 bits (offset) describe a per node address space of 256 terabytes. The CSR architecture, by convention, splits a node’s address space into two regions with different behavioral characteristics. The lower portion, up to but not including 0xFFFF F000 0000, is expected to behave as memory in response to read and write transactions. The upper portion is more like a traditional IO space: read and write transactions to the control and status registers (CSR’s) in this area usually have side effects. Registers that have FIFO behavior customarily are implemented in this region.

Within the 64-bit address, the 16-bit node ID (bus ID and physical ID) is analogous to a network hardware address---but 1394 node ID's are variable and subject to reassignment each time one or more nodes are added to or removed from the bus.

The 1394 link layer provides a datagram service with both confirmed (acknowledged) and unconfirmed datagrams. The confirmed datagram service is called "asynchronous" while the unconfirmed service is known as "isochronous." Other than the presence or absence of confirmation, the principal distinction between the two is quality of service: isochronous datagrams are guaranteed to be delivered with bounded latency. Datagram payloads vary with implementations and may range from one octet up to a maximum determined by the transmission speed (at 100 Mbps, named S100, the maximum asynchronous data payload is 512 octets while at S400 it is 2048 octets).

NOTE: Extensions underway in IEEE P1394b contemplate additional speeds of 800, 1600 and 3200 Mbps; engineering prototypes are planned for early 1998.

2. DEFINITIONS AND NOTATION

2.1 Conformance

Several keywords are used to differentiate levels of requirements and optionality, as follows:

expected: A keyword used to describe the behavior of the hardware or software in the design models assumed by this standard. Other hardware and software design models may also be implemented.

ignored: A keyword that describes bits, octets, quadlets, octlets or fields whose values are not checked by the recipient.

may: A keyword that indicates flexibility of choice with no implied preference.

reserved: A keyword used to describe objects-bits, octets, quadlets, octlets and fields-or the code values assigned to these objects in cases where either the object or the code value is set aside for future standardization. Usage and interpretation may be specified by future extensions to this or other standards. A reserved object shall be zeroed or, upon development of a future standard, set to a value specified by such a standard. The recipient of a reserved object shall not check its value. The recipient of a defined object shall check its value and reject reserved code values.

shall: A keyword that indicates a mandatory requirement. Designers are required to implement all such mandatory requirements to assure interoperability with other products conforming to this standard.

should: A keyword that denotes flexibility of choice with a strongly preferred alternative. Equivalent to the phrase "is recommended."

2.2 Glossary

The following terms are used in this standard:

address resolution protocol: A method for a requester to determine the hardware (1394) address of an IP node from the IP address of the node.

bus ID: A 10-bit number that uniquely identifies a particular bus within a group of bridged buses. The bus ID is the most significant portion of a node’s 16-bit node ID.

IP datagram: An Internet message that conforms to the format specified by RFC 791.

link fragment: A portion of an IP datagram transmitted within a single 1394 packet. The data payload of the 1394 packet contains both a link fragment header and its associated link fragment. It is possible to transmit datagrams without fragmentation.

link fragment header: A structure that precedes all IP datagrams (or each fragment thereof) when they are transmitted over 1394. See also link fragment.

local bus ID: A bus ID with the value 0x3FF. A node shall respond to transaction requests addressed to its 6-bit physical ID if the bus ID in the request is either 0x3FF or a bus ID explicitly assigned to the node.

node ID: A 16-bit number that uniquely identifies a Serial Bus node . The most significant 10 bits are the bus ID and the least significant 6 bits are the physical ID.

node unique ID: A 64-bit number that uniquely identifies a node among all the Serial Bus nodes manufactured worldwide; also known as the EUI-64 (Extended Unique Identifier, 64-bits).

octet: Eight bits of data.

packet: Any of the 1394 primary packets; these may be read, write or lock requests (and their responses) or stream data. The term "packet" is used consistently to differentiate 1394 packets from ARP or IP datagrams, which are also (generically) packets.

physical ID: On a particular bus, this 6-bit number is dynamically assigned during the self-identification process and uniquely identifies a node on that bus.

quadlet: Four octets, or 32 bits, of data.

stream packet: A 1394 primary packet with a transaction code of 0x0A that contains a block data payload. Stream packets may be either asynchronous or isochronous according to the type of 1394 arbitration employed.

2.3 Abbreviations

The following are abbreviations that are used in this standard:

ARP	Address resolution protocol
CSR	Control and status register
CRC	Cyclical redundancy checksum
EUI-64	Extended Unique Identifier, 64-bits (essentially equivalent to names used elsewhere, such as global unique ID or world-wide unique ID)
IP	Internet protocol (within the context of this document, IPv4)

3. IP-CAPABLE NODES

Not all 1394 devices are capable of the reception and transmission of ARP or IP datagrams. An IP-capable node shall fulfill the following minimum 1394 requirements:

- the *max_rec* field in its bus information block shall be at least 8; this indicates an ability to accept write requests with data payload of 512 octets. The same ability shall also apply to read requests; that is, the node shall be able to transmit a response packet with a data payload of 512 octets;

4. LINK ENCAPSULATION AND FRAGMENTATION

All IP datagrams (broadcast, unicast or multicast), as well as ARP requests and responses, that are transferred *via* 1394 block write requests or stream packets shall be encapsulated within the packet's data payload. The maximum size of data payload, in octets, is constrained by the speed at which the packet is transmitted.

Table 1 - Maximum data payloads

Speed	Asynchronous	Isochronous
S100	512	1024
S200	1024	2048
S400	2048	4096
S800	4096	8192
S1600	8192	16384
S3200	16384	32768

The maximum data payload may also be restricted by the capabilities of the sending or receiving node(s); this is specified by *max_rec* in either the bus information block or ARP response.

For either of these reasons, the minimum capabilities between ~~any two~~ IP-capable nodes may be less than the ~~2024~~1500 octet maximum transmission unit (MTU) specified by this document. This necessitates 1394 link level encapsulation of IP datagrams, which provides for the ordering and reassembly of link fragments as necessary.

4.1 Link encapsulation header

All datagrams transported over 1394 are prefixed by a link encapsulation header with one of the two formats illustrated below.

If an entire IP datagram may be transmitted within a single 1394 packet, it is unfragmented and the first quadlet of the data payload shall conform to the format illustrated below.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         0         |  reserved |          ether_type           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2 - Unfragmented datagram header format

The *ether_type* field shall specify the nature of the datagram that follows, as specified by the following table.

ether_type	Datagram
0x800	IPv4
0x806	ARP

In cases where the length of the datagram exceeds the maximum data payload supported by the sender and all recipients, the datagram shall be broken into link fragments; the first two quadlets of the data payload for each link fragment shall conform to the format shown below.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|m| fragment_offset |rsv| bsize |          ether_type           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              dgl              |           signature           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3 - Fragmented datagram header format

The definition and usage of the fields is as follows:

more: If there are other link fragments for the IP datagram whose offset value(s) are greater than *fragment_offset*, the *more* bit (abbreviated as *m* above) shall be one. When the *more* bit is zero this is the last fragment of the datagram.

For each IP datagram, there shall be exactly one link fragment whose *more* bit is zero.

fragment_offset: This field shall specify the offset, in quadlets, of the fragment from the beginning of the IP datagram. The first quadlet of the datagram (the start of the IP header) has an offset of zero.

bsize: The size of the buffer, expressed as (*bsize* + 1) * 128 bytes, necessary for the recipient to reassemble the datagram fragments.

ether_type: This field shall have a value of 0x800, which indicates an IP datagram.

NOTE- Other network protocols, identified by different values of *ether_type*, may use the encapsulation format defined above but such use is outside of the scope of this document.

dgl: The value of *dgl* shall be the same for all fragments of an IP datagram. The sender shall increment the value of *dgl* for successive, fragmented datagrams; the incremented value of *dgl* shall wrap from 65,535 back to zero.

signature: The sender shall set this field to the most significant 16-bits of its own NODE_IDS register.

All datagrams, regardless of the mode of transmission (block write requests or stream packets) shall be preceded by one of the above described link encapsulation headers. This permits uniform software treatment of datagrams without regard to the mode of their transmission.

4.2 Fragment reassembly

The recipient of a fragmented datagram shall use both *signature* and *dgl* to identify all the fragments from a single datagram. Subsequent to reassembly, the recipient shall verify the IP header checksum of the datagram.

NOTE- The use of *signature* for any purpose other than datagram reassembly is fraught with error and is strongly discouraged.

Upon receipt of a datagram fragment, the recipient may place the data payload (absent the link fragment header) within an IP datagram reassembly buffer at the quadlet offset specified by *fragment_offset*. The size of the reassembly buffer may be determined from *bsize*.

If a datagram fragment is received that overlaps another fragment for the same *signature* and *dgl*, the fragment(s) already accumulated in the reassembly buffer shall be discarded. A fresh reassembly may be commenced with the most recently received fragment. Fragment overlap is determined by the combination of *fragment_offset* from the link fragment header and *data_length* from the 1394 packet header.

Upon detection of a Serial Bus reset, recipient(s) shall discard all fragments of all partially reassembled datagrams and sender(s) shall discard all not yet transmitted fragments of all partially transmitted datagrams.

5. ADDRESS RESOLUTION PROTOCOL (ARP)

ARP requests and responses shall be transmitted by the same means as broadcast IP datagrams. The data payload of an ARP request/response is 56 octets and shall conform to the format illustrated below.

NOTE- The first quadlet of the ARP request/response is the link encapsulation header for an unfragmented datagram describe in 4.

~~NOTE- The contents of the ARP request/response are agreed but its exact format remains subject to change.~~

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0 |           reserved            |       ether_type (0x0806)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    hardware_type (0x0018)     |    protocol_type (0x0800)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  hw_addr_len  |  IP_addr_len  |            opcode             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+---                    sender_unique_ID                     ---+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         sender_node_ID        |     sender_unicast_FIFO_hi    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     sender_unicast_FIFO_lo                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| sender_max_rec|      sspd     |           reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       sender_IP_address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+---                    target_unique_ID                     ---+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         target_node_ID        |     target_unicast_FIFO_hi    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     target_unicast_FIFO_lo                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| target_max_rec|      tspd     |           reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       target_IP_address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4 - ARP request/response format

Field usage in an ARP request/response is as follows:

hardware_type: This field indicates 1394 and shall have a value of 0x0018.

protocol_type: This field shall have a value of 0x0800; this indicates that the protocol addresses in the ARP request or response conform to the format for IP addresses.

hw_addr_len: This field indicates the size, in octets, of the 1394-dependent hardware address associated with an IP address and shall have a value of 20.

IP_addr_len: This field indicates the size, in octets, of an IP version 4 (IPv4) address and shall have a value of 4.

opcode: This field shall be one to indicate an ARP request and two to indicate an ARP response.

sender_unique_ID: This field shall contain the node_unique_ID of the sender and shall be equal to that specified in the sender’s bus information block.

sender_node_ID: This field shall contain the most significant 16 bits of the sender’s NODE_IDS register.

sender_unicast_FIFO_hi and *sender_unicast_FIFO_lo*: These fields together shall specify the 48-bit offset of the sender’s FIFO available for the receipt of IP datagrams in the format specified by section 6. The offset of a sender’s unicast FIFO shall not change, either as a result of a bus reset, power reset or other circumstance, unless the new FIFO offset is advertised by an unsolicited ARP response datagram.

sender_max_rec: This field shall be equal to the value of *max_rec* in the sender’s configuration ROM bus information block.

sspd: This field shall be set to the lesser of the sender’s link speed and PHY speed. The link speed is the maximum speed at which the link may send or receive packets; the PHY speed is the maximum speed at which the PHY may send, receive or repeat packets. The encoding used for *sspd* is specified by the table below; all values not specified are reserved.

Value	Speed
0	S100
1	S200
2	S400
3	S800
4	S1600
5	S3200

sender_IP_address: This field shall specify the IP address of the sender.

target_unique_ID: In an ARP request, the value of this field is not specified~~shall be all ones~~; it shall be ignored by the recipient. In an ARP response, it shall be set to the value of *sender_unique_ID* from the corresponding ARP request.

target_node_ID: In an ARP request, the value of this field is not specified~~shall be all ones~~; it shall be ignored by the recipient. In an ARP response, it shall be set to the value of *sender_node_ID* from the corresponding ARP request.

target_unicast_FIFO_hi and *target_unicast_FIFO_lo*: In an ARP request, the value of these fields is not specified~~shall be all ones~~; they shall be ignored by the recipient. In an ARP response, they shall be set to the value of *sender_unicast_FIFO_hi* and *sender_unicast_FIFO_lo* from the corresponding ARP request.

target_max_rec: In an ARP request, the value of this field is not specified~~shall be zero~~; it shall be ignored by the recipient. In an ARP response, it shall be equal to the value of *max_rec* from the corresponding ARP request.

tspd: In an ARP request, the value of this field is not specified; it shall be ignored by the recipient. In an ARP response, it shall be equal to the value of *sspd* from the corresponding ARP request.

NOTE- This above descriptions of unspecified values for *target_* fields in an ARP request are subject to confirmation by a consensus poll in progress at the time of publication.

target_IP_address: In an ARP request, this field shall specify the IP address from which the responder desires a response. In an ARP response, it shall be set to the value of *sender_IP_address* from the corresponding ARP request.

6. IP UNICAST

IP unicast may be transmitted to a recipient within a 1394 primary packet that has one of the following transaction codes:

tcode	Description	Arbitration
0x01	Block write	Asynchronous
0x0A	Stream packet	Isochronous
0x0A	Stream packet	Asynchronous

Block write requests are suitable when 1394 link-level acknowledgement of the datagram is desired but there is no need for bounded latency in the delivery of the packet (quality of service).

Isochronous stream packets provide quality of service guarantees but not 1394 link-level acknowledgement.

The last method, asynchronous stream packets, is mentioned only for the sake of completeness. This method should not be used, since it provides for neither 1394 link-level acknowledgment nor quality of service---and consumes a valuable resource, a channel number.

NOTE: Regardless of the IP unicast method employed, asynchronous or isochronous, it is the responsibility of the sender of a unicast IP datagram to determine the maximum data payload that may be used in each packet. The necessary information may be obtained from:

- the SPEED_MAP maintained by the 1394 bus manager and provides a maximum transmission speed between any two nodes on the local Serial Bus. The speed in turn implies a maximum data payload (see Table 1).

NOTE: The SPEED_MAP is derived from the self-ID packets transmitted by all 1394 nodes subsequent to a bus reset. An IP-capable node may observe the self-ID packets directly;

- the *target_max_rec* field in an ARP response. This document requires a minimum value of 8 (equivalent to a data payload of 512 octets). Nodes that operate at S200 and faster are encouraged but not required to implement correspondingly larger values for *target_max_rec*; or

- other methods beyond the scope of this standard.

The maximum data payload shall be the minimum of the largest data payload implemented by the sender, the recipient and the PHYs of all intervening nodes.

6.1 Asynchronous IP unicast

Unicast IP datagrams that do not require any quality of service shall be contained within the data payload of 1394 block write transactions addressed to the *target_node_ID* and *target_unicast_FIFO* obtained from an ARP response packet.

If no acknowledgement is received in response to a unicast block write request, the state of the target is ambiguous.

NOTE: An acknowledgment may be absent because the target is no longer functional, may not have received the packet because of a header CRC error or may have received the packet successfully but the acknowledge sent in response was corrupted.

6.2 Isochronous IP unicast

Unicast IP datagrams that require quality of service shall be contained within the data payload of 1394 isochronous stream packets.
The details of coordination between nodes with respect to allocation of channel number(s) and bandwidth is beyond the scope of this standard.

7. IP BROADCAST

The 1394 facilities, whether asynchronous stream packets or block write requests with a *destination_ID* of 0xFFFF, have yet to be determined by the working group.

The use of asynchronous streams for IP broadcast requires some method for the allocation of a channel number and the communication of this channel number to all IP-capable nodes. The method has yet to be agreed by the working group.

On the other hand, if block write requests with a *destination_ID* of 0xFFFF are used, it will be necessary to propose a fixed 48-bit *destination_offset* to IEEE P1394a.

8. IP MULTICAST

Many of the details of multicast remain outside the scope of this draft in its present form (but are expected to be added by the working group as the draft is advanced).

IP multicast shall use stream packets, either asynchronous or isochronous, according to the quality of service required.

9. SECURITY CONSIDERATIONS

This document raises no security issues.

10. ACKNOWLEDGEMENTS

This document represents work in progress by the IP / 1394 Working Group. The editor wishes to acknowledge the contributions made by all the active participants, either on the reflector or at face-to-face meetings, which have advanced the technical content.

11. REFERENCES

[1] IEEE Std 1394-1995, Standard for a High Performance Serial Bus

[2] ISO/IEC 13213:1994, Control and Status Register (CSR) Architecture for Microcomputer Buses

[3] IEEE Project P1394a, Draft Standard for a High Performance Serial Bus (Supplement)

[4] IEEE Project P1394b, Draft Standard for a High Performance Serial Bus (Supplement)

12. EDITOR’S ADDRESS

Peter Johansson
Congruent Software, Inc.
3998 Whittle Avenue
Oakland, CA 94602

(510) 531-5472
(510) 531-2942 FAX
pjohansson@aol.com