P. Johansson
(Editor)
August 1997

IP over IEEE 1394
(High Performance Serial Bus)
Revision 3
August xx, 1997

<draft-ip1394-ipv4-03.txt>

STATUS OF THIS DOCUMENT

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).

EXPIRATION DATE

February, 1998

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 code 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 or removed from the network.

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 range from one byte up to a maximum determined by the transmission speed (at 100 Mbps, named S100, the maximum asynchronous data payload is 512 bytes while at S400 it is 2048 bytes).

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, bytes, 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, bytes, 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.

byte: Eight bits of data.

doublet: Two bytes, or 16 bits, of data.

link fragment header: A quadlet that precedes all IP datagrams (or fragments 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.

IP datagram: An Internet message that conforms to the format specified by RFC 791. It consists of the 20-byte IP header, options (if they are present) and the data that immediately follows.

kilobyte: A quantity of data equal to 2 ** 10 bytes.

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.

node ID: A 16-bit number that uniquely identifies a Serial Bus node within a 1394 subnet. 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 world-wide; also known as the EUI-64 (Extended Unique Identifier, 64-bits).

packet: Any of the 1394 primary packets. 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 bytes, 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.

subnet: Either a single 1394 bus or else two or more 1394 buses with uniquely enumerated bus ID’s connected by bridges. When a subnet consists of only one bus, there is no requirement for the bus ID to be anything other than 0x3FF, the local bus ID.

terabyte: A quantity of data equal to 2 ** 40 bytes.

unit: A component of a Serial Bus node that provides processing, memory, I/O or some other functionality. Routers, terminal servers and workstations are an example of a unit. Once the node is initialized, the unit provides a CSR interface that is typically accessed by device driver software at an initiator. A node may have multiple units, which normally operate independently of each other.

unit architecture: The specification of the interface to and the behaviors of a unit implemented within a Serial Bus node.

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 necessarily capable of ARP or the reception and transmission of 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 bytes. 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 bytes;

- it shall be isochronous resource manager capable, as specified by 1394-1995;

- it shall support both reception and transmission of asynchronous streams as specified by P1394a; and

- it shall implement the BROADCAST_IP_CHANNEL register.

4. BROADCAST_IP_CHANNEL

This register is required for IP-capable nodes. It shall be located at offset 0xFFFF F000 0214 within the node’s address space and shall support quadlet read and write requests, only. The format of the register is 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
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |1|v|                  reserved                     |  channel  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1 - BROADCAST_IP_CHANNEL format

Upon a node power reset or a bus reset, the entire register (with the exception of the most significant bit) shall be cleared to zero.

The most significant bit (a constant one) differentiates the presence of the BROADCAST_IP_CHANNEL register in an IP manager-capable node from the value (all zeros) returned when offset 0xFFFF F000 0214 is read at node(s) that do not implement this register.

The *valid* bit (abbreviated as *v* above), when set to one, indicates that the *channel* field contains meaningful information.

NOTE: IP-capable nodes shall not transmit either ARP or broadcast IP datagrams until one second after the *valid* bit is first set to one subsequent to a bus reset.

The *channel* field shall be initialized by the IP manager (see below) to identify the channel number shared by IP-capable nodes for ARP, IP broadcast and certain IP multicast addresses.

Only the *valid* bit and the *channel* field may be changed by quadlet write requests; the data value in the write request shall be ignored for all other bit positions.

5. IP MANAGER

In order for ARP or broadcast IP datagrams to function on 1394, a prerequisite is the presence of an IP manager. Each Serial Bus has its own IP manager which performs these functions:

- the allocation of a channel number for ARP and broadcast IP; and
- the communication of that channel number to all IP-capable nodes on the same bus.

Without the presence of an IP manager on Serial Bus, IP-capable nodes are unable to use the ARP and broadcast IP methods specified by this document. If other methods (for example, a search of configuration ROM) permit IP-capable nodes to discover each other they may be able to send and receive IP datagrams.

Since more than one IP manager-capable nodes may be present, it is necessary to select one node from the contenders. Subsequent to any Serial Bus reset the new IP manager shall be determined by a distributed algorithm executed by all the IP manager-capable nodes. The algorithm is straightforward: the IP manager-capable node with the largest 6-bit physical ID shall be the IP manager. The steps in the algorithm are as follows:

a) An IP manager-capable node shall also a contender for the role of isochronous resource manager. The contender bit its self-ID packet shall be set to one;

b) Subsequent to a bus reset, isochronous resource manager contention takes place during the self-identification process specified by 1394-1995;
c) If the new isochronous resource manager is also IP manager-capable it is the new IP manager and shall proceed with g);
d) An IP manager-capable node that loses contention for the role of isochronous resource manager shall wait one second before it attempts to become the IP manager. If a write request addressed to the BROADCAST_IP_CHANNEL register is received before one second elapses, another node is the IP manager;
e) Otherwise, the node shall read the BROADCAST_IP_CHANNEL register of any contenders with a larger physical ID (these nodes were identified by their self-ID packets). The candidate IP manager shall read the BROADCAST_IP_CHANNEL register in the contender with the largest physical ID and progress downward. If the register is implemented, the IP manager is determined to be a different node. The losing contender shall ignore the contents of BROADCAST_IP_CHANNEL returned in the read response and shall wait for the *valid* bit of its own register to be set before transmitting any ARP or IP datagrams;
f) If no contender with a physical ID larger than the candidate IP manager’s physical ID also implemented the BROADCAST_IP_CHANNEL register, the search is complete and the candidate becomes the new IP manager;
g) The IP manager shall attempt to allocate a channel number from the CHANNELS_AVAILABLE register (note that the IP manager may also be the isochronous resource manager). If a channel number is allocated, the IP manager shall write this value, along with a *valid* bit of one, to the BROADCAST_IP_CHANNEL register of all the IP-capable nodes on the bus. Either a broadcast write request or a series of directed write requests may be used to propagate the information. Otherwise, if no channel number is available, the IP manager shall take no action (all *valid* bit(s) were cleared by the bus reset);

When the IP manager is unable to allocate a channel for ARP and broadcast IP, a warning should be communicated to a user that IP initialization cannot complete because of a lack of Serial Bus resources. The user should be advised to reconfigure or remove other devices if she wishes to make use of IP.

6. 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 bytes, varies with 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; this is specified by *max_rec* in the bus information block.

For either of these reasons, the minimum capabilities between any two IP-capable nodes may be less than the 2024 byte MTU specified by this document. This necessitates 1394 link level fragmentation of IP datagrams, which provides for the ordering and reassembly of link fragments when necessary.

NOTE- Link fragmentation is orthogonal to the question of whether or not datagrams should be preceded with an additional header, such as LLC/SNAP. The usage of LLC/SNAP in the first (or only) fragment of a datagram is an open issue that has a bearing on the charter of the IP / 1394 working group and is not yet resolved.

6.1. Link fragment header

All datagrams transported over 1394 are prefixed by a link fragment 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	IP
0x806	ARP

In cases where the length of the datagram exceeds the maximum data payload between any two nodes, the datagram shall be broken into link fragments; the first quadlet of the data payload for each link fragment shall follow 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 |    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.

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 and the value of *dgl* shall wrap from 63 back to zero. This field shall be assigned a monotonically increasing sequence number by the sender of the datagram.

signature: The sender shall assign a signature value to all of its link fragments such that *signature* is reasonably expected to be unique among all the IP-capable nodes on the bus. Unless the sender has knowledge (beyond the scope of this standard) that a particular *signature* value is unique, the sender shall initialize this field to the most significant 16-bits of its own NODE_IDS register.

The value 0xFFFF is reserved and shall not be transmitted in the *signature* field.

NOTE- Although the most common *signature* value may be the sender’s own node ID, the recipient of a fragmented datagram shall not impute any such meaning and shall use *signature* only for link fragment reassembly.

All datagrams sent by block write requests or stream packets shall have one of the above described link fragment headers as the first quadlet of their data payload. This permits uniform software treatment of datagrams without regard to the mode of their transmission.

6.2. Fragment reassembly

The recipient of a fragmented datagram shall use both *signature* and *dgl* for orderly link fragment reassembly. Together, *signature* and *dgl*, are intended to uniquely all the fragments from a single datagram. The recipient shall verify the IP header checksum of the reassembled datagram.

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 either by a) the MTU size of 2024 bytes specified by this standard, b) the total length of the datagram as specified by the IP header or c) by other means outside of the scope of this standard.

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 and a fresh reassembly 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 packet header.

The *dgl* field is part of the unique identification of an IP datagram; it may also invalidate partially reassembled datagram(s) from the same sender. For all reassembly buffers whose *signature* value is equal to the *signature* of a received fragment, the following procedure is followed to discard partial datagrams:

a) the two’s complement of *dgl* is added to the value of *dgl* for each partially reassembled datagram and the result is truncated to six bits; and
b) if the most significant bit of the result is one, the partially reassembled datagram shall be discarded.

This algorithm eliminates the need for a reassembly time-out on individual reassembly buffers. Partially reassembled datagrams whose *dgl* is offset by more than 32 from a newly arrived *dgl* are discarded.

7. ARP

Address Resolution Protocol (ARP) is accomplished on 1394 by means of asynchronous stream packets transmitted with the channel number specified by all of the IP-capable nodes’ BROADCAST_IP_CHANNEL register. The data payload of an ARP datagram is 48 bytes and 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 (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_IP_address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+---                     target_unique_ID                    ---+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        target_node_ID         |     target_unicast_FIFO_hi    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    target_unicast_FIFO_lo                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       target_IP_address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4 - ARP datagram format

The first quadlet shown above is the link fragment header already described in section 6. Field usage in the remainder of an ARP datagram 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 bytes, of the 1394-dependent hardware address associated with an IP address and shall have a value of 16.

IP_addr_len: This field indicates the size, in bytes, 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 8. 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_IP_address: This field shall specify the IP address of the sender.

target_unique_ID: In an ARP request, this field shall be set to ones. 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, this field shall be set to ones. 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, these fields shall be set to ones. 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_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.

Both ARP requests and responses shall be transmitted with the same channel number in their stream packet header. This permits nodes other than the requester to eavesdrop ARP responses and cache the information.

NOTE: The 16-bit node ID's of any IP-capable nodes are volatile and likely to change each time Serial Bus is reset. This could provoke a storm of ARP requests and responses subsequent to each bus reset---which is also a time when it is extremely desirable for all 1394 applications to hold their bus utilization to a minimum. Bus resets are also likely to occur in pairs. Because of this, implementers are strongly encouraged to make judicious use of ARP requests, both in their timing and frequency. Although some strategies may be self-evident, the possible impact upon 1394 bus utilization is important enough that they bear mention below:

- Flush the ARP cache when a bus reset is observed but defer updates until a particular IP address is requested by an application;

- Eavesdrop ARP responses; and

- If OS support for 1394 permits, build the ARP cache for the IP service layer on the basis of node unique ID (EUI-64) and permit a (presumably lower level) 1394 service layer to reestablish the correlation between EUI-64 and 16-bit node ID after each reset. Because all 1394 applications share the need to maintain a mapping between EUI-64 and node ID it is probable that operating systems will provide this facility.

8. 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. This provides a maximum transmission speed between any two nodes on the local Serial Bus and is derived from the self-ID packets transmitted by all 1394 nodes subsequent to a bus reset;

- the self-ID packets themselves, in the case where no 1394 bus manager is present;

- the *max_rec* field in the target’s bus information block. This document requires a minimum value of 8 (equivalent to a data payload of 512 bytes). Nodes that operate at S200 and faster are encouraged but not required to implement correspondingly larger values for *max_rec*; or

- other methods beyond the scope of this standard.

8.1. 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. The first quadlet of the data payload of the block write request shall be the link fragment header specified by section 6.

If the IP datagram cannot be encapsulated within a single 1394 packet, it shall be split into multiple link fragments for transmission in a separate 1394 packet, also specified in section 6.

If no acknowledgement is received in response to a unicast block write request, the state of the target is ambiguous. The sender of the IP datagram or fragment should either abandon transmission of the datagram or retransmit from the first (or only) link fragment. If the datagram is retransmitted the sender shall use a different value for *dgl*.

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.

8.2. Isochronous IP unicast

Unicast IP datagrams that require quality of service shall be contained within the data payload of 1394 isochronous stream packets. The first quadlet of the data payload of the stream packet shall be the link fragment header specified by section 6.

NOTE: Isochronous IP unicast is a degenerate case of IP multicast that requires quality of service: a multicast group in which only two nodes participate as talker and listener.

The details of coordination between the two nodes with respect to allocation of channel number(s) and bandwidth is beyond the scope of this standard. The channel number used for isochronous IP unicast shall be different from the *channel* field in the BROADCAST_IP_CHANNEL register.

9. IP BROADCAST

Broadcast IP datagrams are encapsulated and fragmented according to the specifications of section 6 and are transported by asynchronous stream packets. There is no quality of service provision for IP broadcast over 1394. The channel number used for IP broadcast is specified by the BROADCAST_IP_CHANNEL register.

The channel number specified by BROADCAST_IP_CHANNEL is intended for datagrams that the sender wishes to transmit to all IP-capable nodes on the local bus or subnet. Broadcast addresses include all of the following:

- TBD: Add list of IP addresses valid for broadcast

All IP datagrams addressed to one of the preceding addresses shall use asynchronous stream packets whose channel number is equal to the *channel* field from the BROADCAST_IP_CHANNEL number.

10. 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. Unless the address of a multicast group is one specified in section 9, the channel number used to identify a multicast group shall be different from the *channel* field in the BROADCAST_IP_CHANNEL register.

The coordination of multicast groups, for example, "join" and "leave" requests and their affect on the channel number allocation, are unresolved.

11. SECURITY CONSIDERATIONS

This document raises no security issues.

12. 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, that have advanced the technical content.

13. 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)

14. EDITOR'S ADDRESS

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

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