Design and Implementation of Firewire device driver on FreeBSD

Katsushi Kobayashi
Communication Research Laboratory

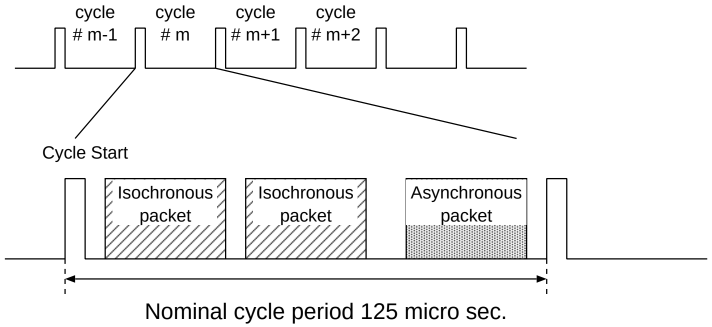
What's Firewire, IEEE1394, iLink

- High speed media and capable to connect heterogenous speed device (from 100Mbps up to 3.2Gbps)
- Hot pluggable, Dynamic Configuration
- Support up to 63 devices without bridge, up to almost 2^{16} device with bridge
- Both support bandwidth guaranteed and best effort communication within single media
- Power supply within cable
- Every device is mapped into unified memory space. memory-like device access (Bus image)

What's Firewire, IEEE1394, iLink (cont'd)

- All transmission action is brought with 125 μ time slice

- 3 type communication mode is defined Asynchronous request, Isochronous stream, Asynchronous stream



Application of Firewire

- Storage device Interface instead of SCSI

- Peripheral Interface (Printer, Digital still camera)

- Network media instead of Ethernet

- A/V digital interface, integrate complicated red, white, yellow RCA cables and control line (DV VCR, DV camcorder)

- Processor Bus

Application of Firewire(cont'd)

Firewire std. does not define application protocols. To use it, higher layer protocol is required as

- To define adaptation layer of legacy protocol
  IP, SBP(SCSI over Firewire)

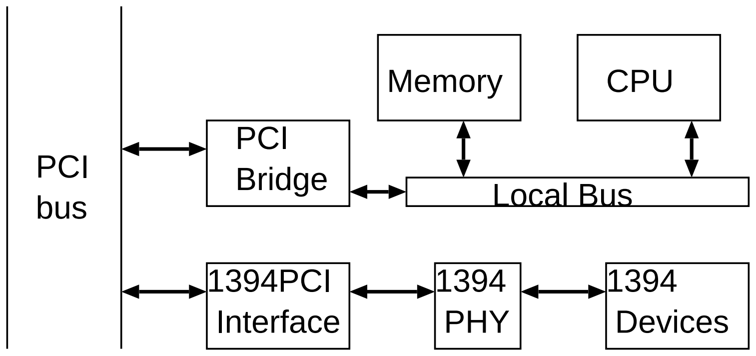
- To build the protocol all over
  DV steam, AV/C

IP World	Digital Video SD,HD,SD-L		SCSI
IP over Firewire	IEC61883	AV/C	SBP-2
IEEE1394 (100 - 3,200Mbps)			

Firewire support on other OS

SGI-IRIX	Support in default with API
Windows 98	Support in default with SDK
MacOS 8.5	Support in default with SDK
Linux	Driver Development is ongoing `http://eclipt.uni-klu.ac.at/ieee1394/`

Firewire H/W block



Functions we implemented

- IP stack (subset)
- Native Firewire socket API
- IEC61883 (DV stream support)

Functions we implemented, How to use it
IP stack

- No special operation is required same as Ethernet.

```
# ifconfig lynx0 10.0.0.1 netmask 255.255.255.0
```

Our implementation is not compliance to IETF standard as following:

- Isynchronous stream mode is used instead of asynchronous stream, because of the chipset does not support it.

- MCAP (Multicast Channel Allocation Protocol) is not implemented, every multicast packet is sent in broadcast channel.

- Not follow latest specification, only accord to `draft-ietf-ip1394-ipv4-8.txt`

Functions we implemented, How to use it
Native Firewire socket.
User can access Firewire with lowest programming cost in familiar socket API.
Firewire socket structure as follows:

```
struct sockaddr_n1394 {
    u_int8_t  sn1394_len;          /* length                 */
    u_int8_t  sn1394_family;       /* AF_N1394               */
    char      sn1394_if[IFNAMSIZ];/* interface name         */
    u_int8_t  sn1394_fch;          /* isochronous channel No. */
    u_int8_t  sn1394_lch;          /* parameter unuse        */
    u_int8_t  sn1394_ftag;         /* isochronous tag        */
    u_int8_t  sn1394_ltag;         /* parameter unuse        */
    u_int8_t  sn1394_mode;         /* parameter unuse        */
    u_int8_t  sn1394_flags;        /* specify socket status  */
    u_int8_t  sn1394_sped;         /* transmission speed.    */
};
```

Functions we implemented, How to use it
Native Firewire socket, asynchronous request

- When using asynchronous request, user must ensure packet format oneself.
- The response of request will arise from same socket
- `tl`(transaction label) field is managed by kernel
- More than one asynchronous request cannot be sent on the same socket until response of previous request

```
/*
 * 0x0001ffff f0000018 represents address offset in unified memory
 * network = 0x0,
 * node    = 0x1,
 * address = 0xfffff0000018(SPLIT_TIME_OUT_HI register)
 */
addr =  0x0001fffff0000018ull;      /* 64 bit */
/* To make a packet for Read request for data quadret */
sendbuf[0] = htonl(0x00000040 |((addr >> 32) & 0xffff0000 ));
sendbuf[1] = htonl(((addr >> 32) & 0x0000ffff ));
sendbuf[2] = htonl(addr & 0xffffffff);

send(s, sendbuf, 12, 0);
len = recv(s, recvbuf, 512, 0);
```


Functions we implemented, How to use it
Native Firewire socket, isochronous stream

- User can use it as connected UDP socket

```
sfw.sn1394_family = AF_N1394;
sfw.sn1394_lch = 63;
sfw.sn1394_ltag = 0x1;
bcopy(ifname, sfw.sn1394_if, IFNAMSIZ);

if(connect(s, (struct sockaddr *)&sfw, sizeof(sfw)) < 0){
        perror("connect");
        exit(1);
}
while(1){
        len = recv(s, recvbuf, 512, 0);
        ......
}
```

Functions we implemented, How to use it
Problem in isochronous stream socket

- IEC 61883 protocol requires certain 24.576 MHz based clock time stamp push into specific packet. How to treat it ?

- When bandwidth guaranteed transmission, packet must be sent every 125 μ second time slice without a break. Can OS send stream under such condition with per packet control ? Also, too many interrupt will be raised.

We prepare "dv" device interface for stable transmission of IEC61883 stream.

Functions we implemented, How to use it
To use `write()` on "dv" device

- least programming effort, only `write()` stream data buffer
- Every packet is arranged in transmission order and into 512 bytes boundary
- DMA action must to be kicked by application
- `read()` call does not implemented yet

```
d = open("/dev/dv0", O_RDWR);
        ......
/* kick playout */
if( ioctl(d, LYNX_DV_TXSTART, &dummy) < 0 ) {
      err(1, "LYNX_DV_TXSTART");
}
while(1){
.....
        if( write(d, sendbuf[frame], datalen[frame] * 512) < 0 ){
        perror("write");
}
```

Functions we implemented, How to use it
To use `mmap()` on “dv” device

- Direct manipulate buffer space allocated “dv” device
- Every packet is arranged in transmission order and into 512 bytes boundary same as `write()`
- DMA action must to be kicked by application
- Capable to use for both send/receive actions

Supported device and related feature

Following chipsets may be working. Both chips support up to 200Mbps speed and not support asynchronous Stream on chip level. If programmer spent a few of effort, capable to use asynchronous stream.

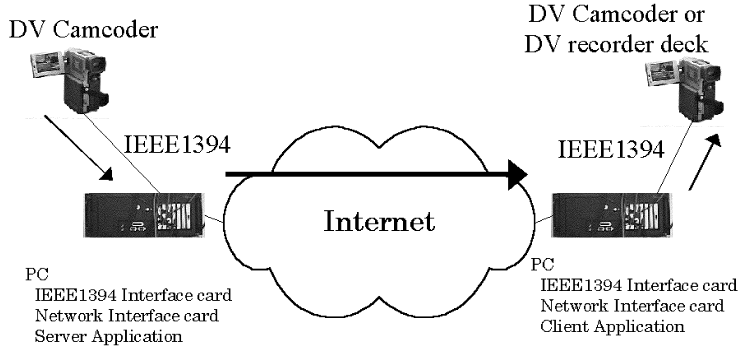
- Texas Instrument PCILynx chipset
  Digital Origin(Radius) MotoDV, PhotoDV
- Adaptec AIC 5800 (Derived Apple chipset)
  Adaptec AHA5940 (Give me a spec. sheet)

Following are planning. Support up to 400Mbps speed and also ready for asynchronous stream.

- PCILynx2
- OHCI
  Some people believe it a sure thing.

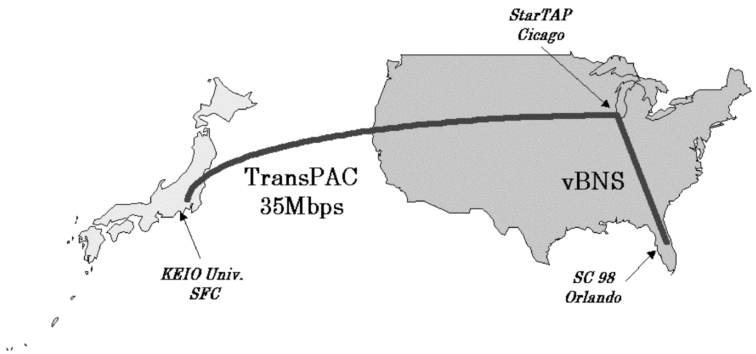
DV Application

Communication tool using DV video equipment on IP protocol. Application consumes about 35Mbps bandwidth including header overhead.



DV Application (cont'd)

We have demonstrated DV communication between US and JAPAN on SC98 iGrid held in Orlando. Network link are provided vBNS, Startap, TransPAC and APAN.
Quality of video was not good due to network troubles.
Now another communication effort is in progress.



Acknowledge

Thanks for Akimichi Ogawa developing DV communication and a lot of cooperator in SC98 demonstration especially StarTAP and APAN NOC team.

Related URLs

- You can obtain our driver code from the following:
  `ftp://www.uec.ac.jp/pub/firewire`
- If you have interest in DV communication tool we presented.
  `http://www.sfc.wide.ad.jp/DVTS`