



DELIVERABLE
AC-018
SMASH

Deliverable #12
a0018/dtb/r&d/ds/p/012/b1
February 1998

**SPECIFICATION OF
THE INTERCONNECT SYSTEM**

Project number	: AC018
Project title	: SMASH
Deliverable Type	: Public

CEC Deliverable Number	: a0018/dtb/r&d/ds/p/012/b1
Internal Project Number	: SMS-DTB-808-1
Contractual Deliverable Date	: 28 February 1998
Actual Date of Deliverable	: 28 February 1998
Title of Deliverable	: Specification of the interconnect system
Contributing Workpackages	: WP310
Nature of Deliverable	: Specification
Authors	: Dieter Haupt (Deutsche Thomson Brandt) Hans Hermann Hake (DTB) Ronald Tol (Philips Research Eindhoven) Marisa Farusi (Olivetti) Paolo Spinelli (Olivetti) Cristina Tani (Olivetti) Edwin Montie (Philips Research Eindhoven) Marijan Leban (University of Ljubljana)

Abstract:
The SMASH storage device is a complete remotely controlled system without any user interface. Both data and control have to be transferred through the Interconnect System. The deliverable specifies the hardware and software interface under consideration of the applicable standards.

Keywords:
Digital Interface, API, Layered Structure, IEEE1394, IEC61883, MPEG2, DVC, packet format, isochronous transmission, asynchronous transmission, Function Control Protocol, plug, IP

Specification of the Interconnect System

1. INTRODUCTION	4
2. STANDARDIZED INTERCONNECT SYSTEM	4
2.1 Applicable Standards	4
2.2 Layered Structure of the Interface	4
2.3 Specification of the Physical Layer	5
2.4 Specification of the Link Layer Capabilities	6
2.5 Transaction Layer	7
2.6 Serial Bus Management Capability	8
2.7 Audio and Video Layer	9
2.8 Internet Protocol (IP) over IEEE1394	14
3. APPLICATION PROGRAMMABLE INTERFACE	15
3.1 Introduction of the Application Programmable Interface	15
3.2 Concepts, notation, conventions	15
3.3 Low level API	17
3.4 File system API	18
3.5 Streams API	18
3.6 Operational Management API	19
4. ANNEX	20
4.1 The Web-VCR	20
4.2 The usage of the <i>SMASH</i> API by the DVB-VCR application	24
4.3 The Usage of the <i>SMASH</i> API by the Remote Education Application	26

1. Introduction

The SMASH storage device is drawn up as a server for multimedia applications in the home. A lot of different data type like text, graphics, digital audio and video sequences can be stored on this device. The combo itself has no user interface and will be complete remote controlled. All these information has to be transported with help of the Interconnection System of the SMASH Combo, which is specified below.

2. Standardized Interconnect System

2.1 Applicable Standards

The interface for the SMASH storage device is based fundamentally on two standards:

IEC 61883/FDIS CONSUMER AUDIO/VIDEO EQUIPMENT- DIGITAL INTERFACE

IEEE 1394-1994 Standard for a High Performance Serial Bus

The characteristics and behavior of the open interface for the SMASH storage device shall comply to these standards if there are no qualifications and limitations described in this document.

2.2 Layered Structure of the Interface

The interface of the SMASH storage device can be divided in different layers. The different layers of the interconnect system are depicted in figure 1.

The Bus Management, Transaction, Link and Physical Layer are standardized in IEEE 1394-1995. The AV Layer is subject of the IEC 61886 standard.

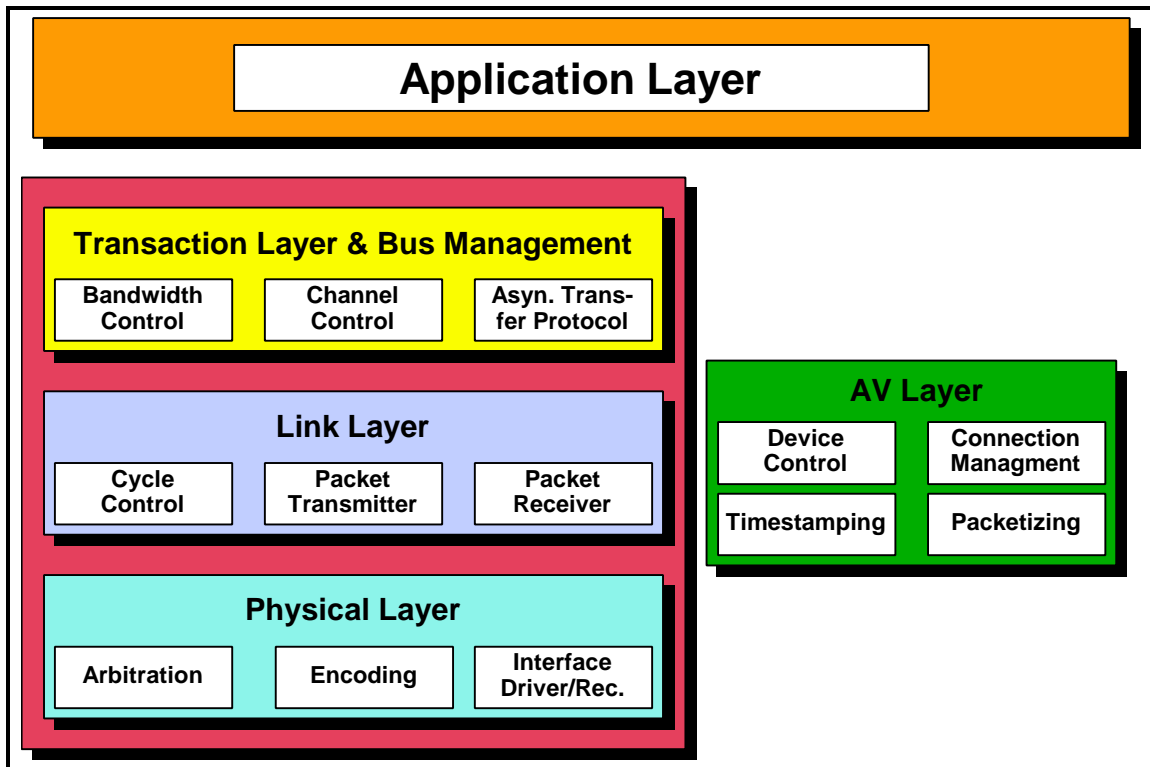


Figure 1 Layered Structure of the SMASH Interconnect System

2.3 Specification of the Physical Layer

The specification of the Physical Layer for the SMASH storage device refers to the “Cable Phy Specification” described in chapter 6 of the IEEE 1394-1995 standard.

2.3.1 Number of Ports

The SMASH device shall have one node with a minimum number of two ports. Additional ports are optional.

2.3.2 Connector socket type

The two mandatory ports shall have 6-pin connectors in accordance to chapter 4.2.1.1.3 of IEEE 1394-1995. Additional ports may have either 6 pin connectors or 4-pin connectors described in Annex A of the IEC 61883-1/FDIS.

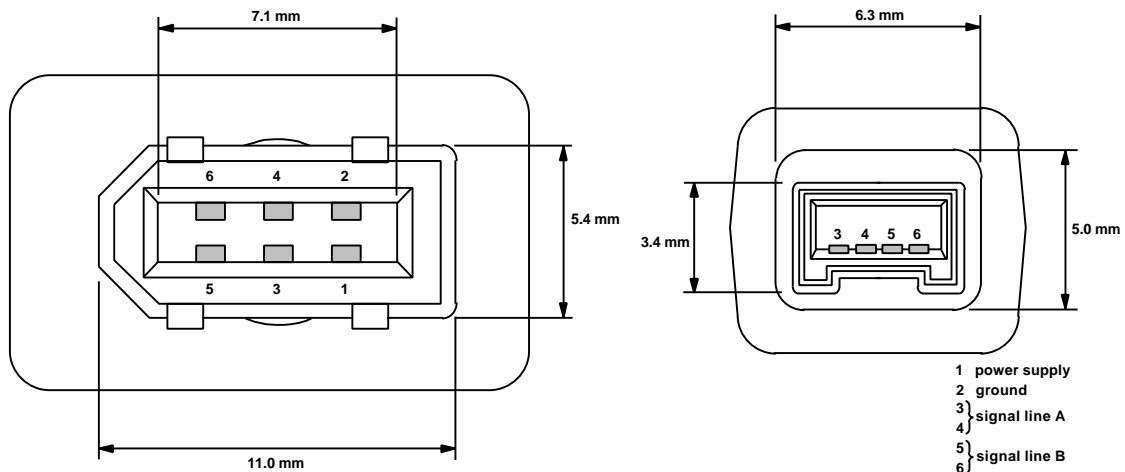


Figure 2 6-pin connector IEEE 1394-1995

4 pin-connector IEC 61883 Annex A

2.3.3 Cable Power Requirements

The SMASH device shall not operate as a power sink. The operation as power source is optional. The corresponding `POWER_CLASS` has to be attached in the `Self-ID` packet.

2.3.4 Data Rate

The interface has to support the S100 cable media data rate of 98.304 Mbit/s \pm 100ppm. The recommended data rate is S200 (196.608 Mbit/s \pm 100ppm).

2.3.5 Isolation Requirements

An galvanic isolation between signal ground and power distribution ground is required to avoid leakage currents greater than 50 μ A.

2.3.6 Cable Phy Packets

The Physical Layer of the SMASH device shall send and react of the receiving of `SELF-ID` PACKETS, `LINK ON` PACKETS and `PHY CONFIGURATION` PACKETS.

2.4 Specification of the Link Layer Capabilities

2.4.1 Packet Types

In accordance to the standard IEEE 1394-1995 chapter 6.2 shall the Link Layer of the SMASH storage device be able to process following packet types:

- Write response asynchronous packets
- Read request for data quadlet asynchronous packets
- Read request for data block asynchronous packets
- Write request for data block asynchronous packets
- Read response for data quadlet asynchronous packets
- Write request for data block asynchronous packets
- Read response for data block asynchronous packets
- Lock request asynchronous packets
- Lock response asynchronous packets
- Cycle start packets
- Isochronous packets

2.4.2 Cycle Time Register

To ensure the “Cycle Master” capability (see chapter 2.6), the Cycle Time Register has to be implemented. Additionally the Link Layer must be able to provide the time stamp, which is needed in the Source Packet Header for MPEG 2 Transport Stream transmission (see chapter 2.7.1.1) and the time stamp used in the SYT field of the CIP header for DVC transmission (see chapter 2.7.1.2). Both time stamps are derived from the Cycle Time Register contents.

2.4.3 Isochronous Channel Support

The SMASH storage device shall support two isochronous channels (one outgoing, one incoming) simultaneously. Additional isochronous channel support is optional.

2.4.4 Maximum Isochronous Channel Bandwidth Requirements

Each channel of the interface shall support maximal 24.064 Mbit/s MPEG transport stream transmission or one DVC SD channel. The support of HD DVC and SDL DVC data transmission in accordance to IEC 61882 part 3 and part 5 are optional.

2.5 Transaction Layer

Every node should be transaction capable. The maximal size of a asynchronous packet should be 512 byte payload for both S100 and S200 speed. The value `max_rec` in the Configuration ROM is therefore set to 1000_2 .

For write transaction is the use of a unified transaction recommended. For the Function Control Protocol (FCP) it is mandatory to use only the unified transaction.

Split transaction shall be supported.

Only the single phase retry mechanism is required to be supported.

2.6 Serial Bus Management Capability

The interface of the SMASH storage device shall have Isochronous Resource Manager capability.

The Interconnect System shall be able to provide Cycle Master capability.

Following Serial Bus node registers shall be implemented:

- CYCLE_TIME register
- BUS_TIME register
- BUS_MANAGER_ID register
- BANDWIDTH_AVAILABLE register
- CHANNELS_AVAILABLE register

The STATE_CLEAR.cmstr has to be implemented.

The NODE_UNIQUE_ID shall be present in the BUS_INFO_BLOCK.

Following entries shall be implemented in the ROOT_DIRECTORY :

- MODULE_VENDOR_ID
- NODE_CAPABILITIES
- NODE_UNIQUE_ID_OFFSET
- UNIT_DIRECTORY_OFFSET

2.6.1 Configuration ROM

BUS_INFO_BLOCK

Name	Value
irmc	1 ₂
cmc	1 ₂
isc	1 ₂
bmc	0 ₂ (1 ₂ optional)
cyc_clk_acc	100 ppm
max_rec	1000 ₂
node_vendor_id	24 bit company ID
chip_id	40 bit node unique ID

Table1 Assignment of Values in BUS_INFO_BLOCK

ROOT_DIRECTORY

The ROOT_DIRECTORY shall contain all mandatory data as described in chapter 5.3.3.2 of IEC 61883 part 1.

UNIT_DIRECTORY

The UNIT_DIRECTORY shall contain all mandatory data as described in chapter 5.3.3.3 of IEC 61883-1.

2.7 Audio and Video Layer

2.7.1 Real Time Data Transmission Protocol

In case of real time data transmission from or to the interface of the SMASH storage device, especially the transmitting of MPEG transport streams or DVC data, the “Real Time Data Transmission Protocol” in accordance to chapter 6 of IEC 61883 part 1 shall be used. The data has to be transmitted in isochronous packets described in chapter 6.2.3.1 of the IEEE 1394-1995 standard. The TAG field in the isochronous header shall be set to 01₂ as an indication that a Common Isochronous Packet (CIP) header will follow.

The structure of a CIP header is shown below:

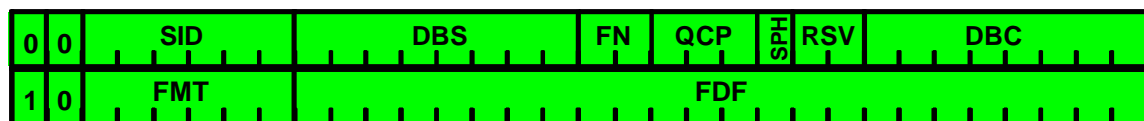


Figure 3 Format of Isochronous Common Packet Header

SID	Source Node ID
DBS	Data Block Size in quadlets
FN	Fraction Number
QPC	Quadlet Padding Count
SPH	Source Packet Header present
DBC	Data Block Continuity counter
FMT	Format ID
FDF	Format Dependent Field

Table 2 Common Interface Packet Header Contents

2.7.1.1 Structure of Isochronous Packets for MPEG Transmission

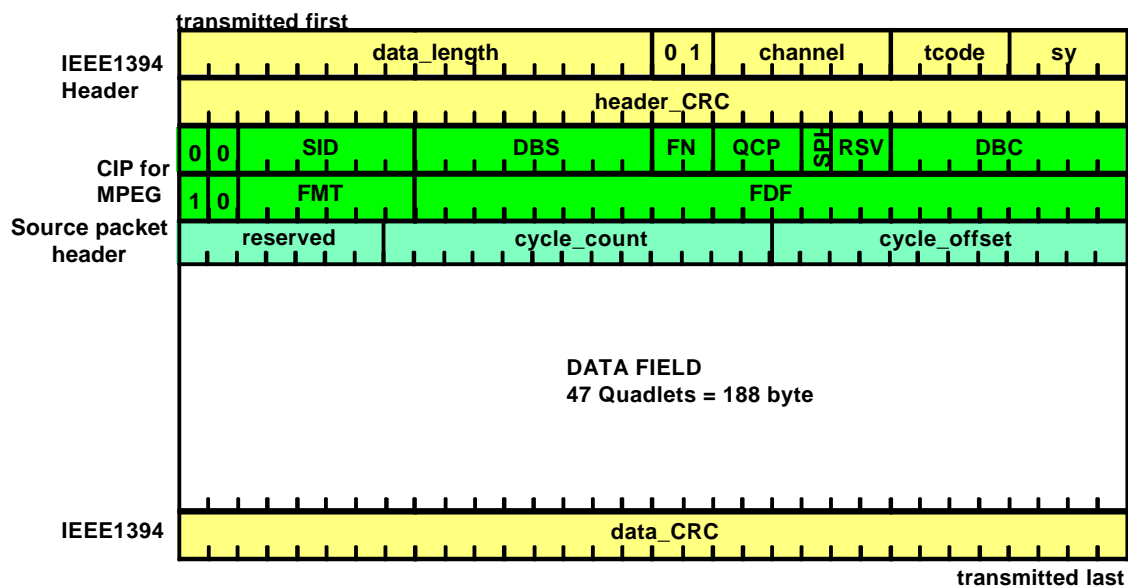


Figure 4 Format of Isochronous Packts for MPEG TS Transmission

SID	Source Node ID	
DBS	Data Block Size in quadlets	00000110 ₂
FN	Fraction Number	XX ₂
QPC	Quadlet Padding Count	XX ₂
SPH	Source Packet Header present	1 ₂
DBC	Data Block Continuity counter	0 ... 255
FMT	Format ID	100000 ₂
FDF	Format Dependent Field	reserved

Table 4 Common Interface Packet Header Contents for MPEG TS

Fractions of MPEG TS packets per 1394 packet are allowed with:

1/8 of a packet	FN = 11 ₂	DBC increments with 1
1/4 of a packet	FN = 10 ₂	DBC increments with 2, LSB = 0
1/2 of a packet	FN = 01 ₂	DBC increments with 4, LSBs = 00 ₂
N packets N is an integer	FN = 11 ₂	DBC increments with 8, LSBs = 000 ₂

Table 5 Allowed Fractioning for MPEG Transport Strams

The time stamp in the SOURCE_PACKET_HEADER indicates the time of the CYCLE_TIME register, when the first bit on a transmitted MPEG transport packet has to be delivered to Transport Stream decoder. The SOURCE_PACKET_HEADER shall be generated in the sender of the interface by adding a appropriate value to current content of the CYCLE_TIME register. The low

order 25 bits are used for the time stamp. The time stamp is necessary to eliminate the jitter added from the interface in the receiver.

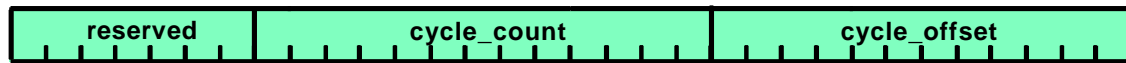


Figure 5 Source packet Header for MPEG TS transmission

The interface shall transmit empty packets in those cycles if not enough data is available to transmit a Common Interface Packet in this cycle.

An overall view of transporting MPEG transport packets over the SMASH interface is shown in figure 6.

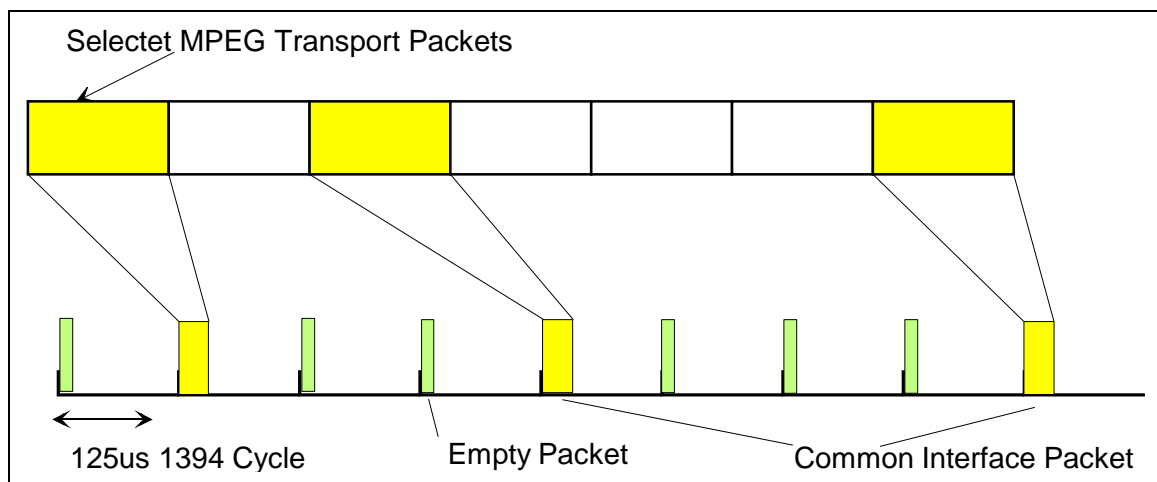


Figure 6 Mapping of Transport Packets on Bus Packets

MPEG transport packets, which could not reach the target node within the in the source packet header indicated time shall be discarded at the transmitter (late packets).

In accordance to the in chapter 2.4.4 described maximum bandwidth for MPEG transmission, the interface shall provide minimal 3264 byte of buffer memory.

2.7.1.2 DVC Data Transmission

The transmission of SD DVC data from and to the *SMASH* storage device shall be compliant to IEC 61882 part 2. The appropriate CIP isochronous packet is shown in figure 7.

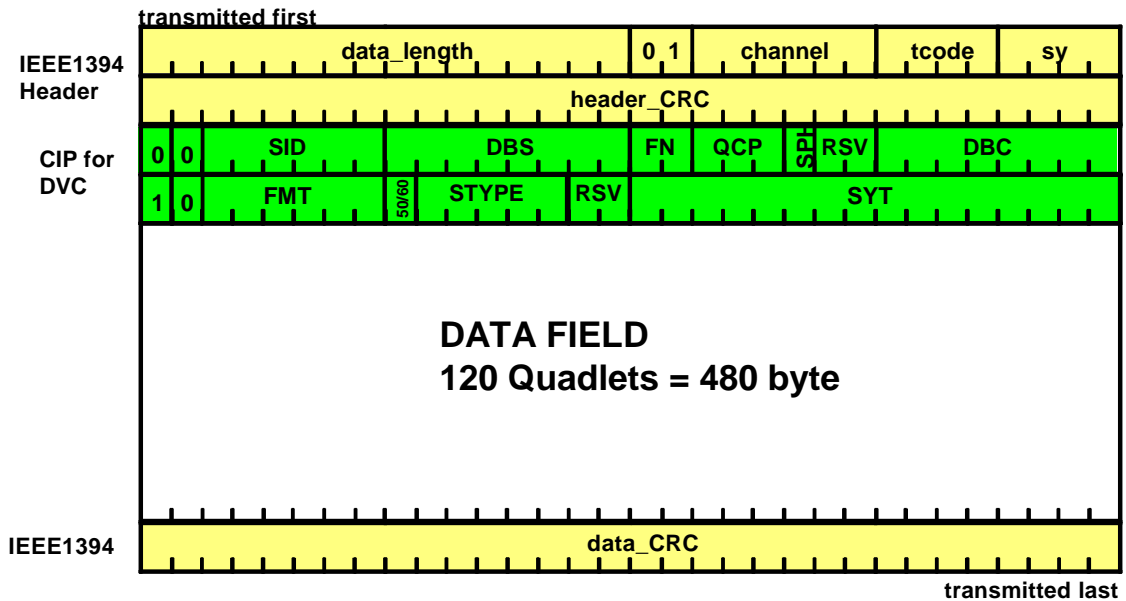


Figure 7 Common Interface Packet Header Contents for DVC

SID	Source Node ID	
DBS	Data Block Size in quadlets	01111000 ₂
FN	Fraction Number	00 ₂
QCP	Quadlet Padding Count	000 ₂
SPH	Source Packet Header present	0 ₂
DBC	Data Block Continuity counter	0 ... 255
FMT	Format ID	000000 ₂
50/60	Fields System	50Hz = 1
STYPE	Video Signal Type ("HD Flag")	00000 ₂ f. 525/625 00010 ₂ f. 1125/1250
SYT	Video Frame Sync Time Stamp	Cyc. Time f. frame start

Table 6 Content of DVC CIP Header

2.7.1.3 Audio and Music Data Transmission

The support of Raw Audio, MIDI conformant and IEC 958 conformant audio data by the SMASH interconnect system is optional. If these kind of data is used by the storage device, the data format should be compliant to IEC 61883 part 6.

2.7.2 Isochronous Data Flow Management

The control of isochronous data flows on the bus shall be done by the concept of plugs and their associated registers which are defined in chapter 7 of IEC 61883-1.

The requirement for the SMASH device are the implementation of an output master plug register and an input master plug register. Since the *SMASH* device shall support at least one isochronous input stream or one output stream it is necessary to implement at least one output plug control register and one input plug control register.

2.7.3 Connection Management Procedures (CMP)

The management of Isochronous data flow shall be done by procedures which are defined in IEC 61883-1.

Following basic procedures shall be implemented:

- establishing a connection
- overlaying a connection
- breaking a connection

All three types of connections shall be supported:

- point-to-point connections
- broadcast-out connections
- broadcast-in connections

Which type of connection is used is beyond this specification.

2.7.4 Function Control Protocol (FCP)

The FCP is required to send and receive commands for AV devices. It shall be according to IEC 61883 -1 chapter 9. A command and a response register at following addresses are required:

Top address of command register: FFFF F000 0B00₁₆

Top address of response register : FFFF F000 0D00₁₆

The Command/Transaction Set (CTS) code value shall be 0000_b for AV devices.

2.7.4.1 AV/C Command language

The AV/C command language is currently in the draft standard under development at the 1394 Trade Association, but will be submitted to the IEC. Following general commands shall be supported:

SUB UNIT INFO
UNIT INFO

Following commands to control a digital VCR (subunit type = 4) shall be supported. These commands perform only basic functionality.

OUTPUT SIGNAL MODE
INPUT SIGNAL MODE

Following values shall be supported

Value (hex)	Signal mode
80	SD 625-50
10	MPEG 25 Mbit/s-60
14	MPEG 12,5 Mbit/s-60
18	MPEG 6,25 Mbit/s-60
90	MPEG 25 Mbit/s-50
94	MPEG 12,5 Mbit/s-50
98	MPEG 6,15 Mbit/s-50

Table 7 Values and Coding for Signal Mode Command

RECORD

The recording mode with the value 75 (recording speed = x1) shall be supported.

PLAY

Only the playback speed x1 (value = 38₁₆) is required

WIND

Following subfunctions are required:

Value (hex)	subfunction
60	STOP
65	REWIND
75	FAST FORWARD

Table 8 Values and Coding for Wind Subfunctions

2.8 Internet Protocol (IP) over IEEE1394

Some *SMASH* applications are using the Internet Protocol as a Network Layer. This protocol layer shall be kept on the Interconnect System.

The standardization on Internet Protocol over IEEE1394 has not been finalized yet. A working group of the "Internet Engineering Task Force" (IETF) is developing a draft. This encapsulation shall be used when the Internet standard (RFC) becomes available.

3. Application Programmable Interface

3.1 Introduction of the Application Programmable Interface

This document describes the *SMASH* Home Storage APIs. It takes into account the requirements of the three applications currently foreseen for the *SMASH* storage device: DVB-VCR, Web-VCR, and Remote Education Application (REA).

There is no such thing as a single API. As a matter of fact, there are a number of high-level interfaces: a table of contents (TOC) a VCR-like interface with streams, and a more traditional file system interface based on Samba. These interfaces are built on low level interfaces dealing with the storage media directly.

To provide an efficient implementation of the (high-level) APIs, information about the data to be stored is needed. More specifically, one needs to know how the data is to be used in each application. As an example, consider the DVB-VCR. Playing a MPEG-2 video with a certain bit rate requires that the video file is stored contiguously on the storage medium. As another example, suppose that we are dealing with a lot of relatively small files on a slow storage medium. If we know which files are related, e.g., the files are always accessed in a certain order, we can utilize this by caching related files on the faster storage medium.

3.2 Concepts, notation, conventions

The *SMASH* COMBO can be seen as a file server for a client application. Basically the *SMASH* server should handle the following types of files: text, graphics, animation, audio and digital video. Also, certain operations on files need to be supported: create, delete, open, close, read, write, etc. Finally, we need to be able to retrieve the files using some Table Of Contents (TOC) of the stored data.

Each of the applications currently defined has different requirements with respect to files and TOC. For example, DVB-VCR deals mainly with very large files with certain real-time requirements. Meta-data, information about the DVB programs provided by the content provider, as well as the storage location of the programs is stored in a multimedia database (MMDB). The REA has many small (HTML) files and a simple TOC. Web-VCR deals both with small and larger files, the TOC can be similar to the one for the REA.

For simplicity reasons, we assume that the *SMASH* server will be used for one particular application at a time. The interaction with the server is called a session.

The following figure shows how the application PC and the *SMASH COMBO* are connected via some network (currently Ethernet, in the future IEEE 1394).

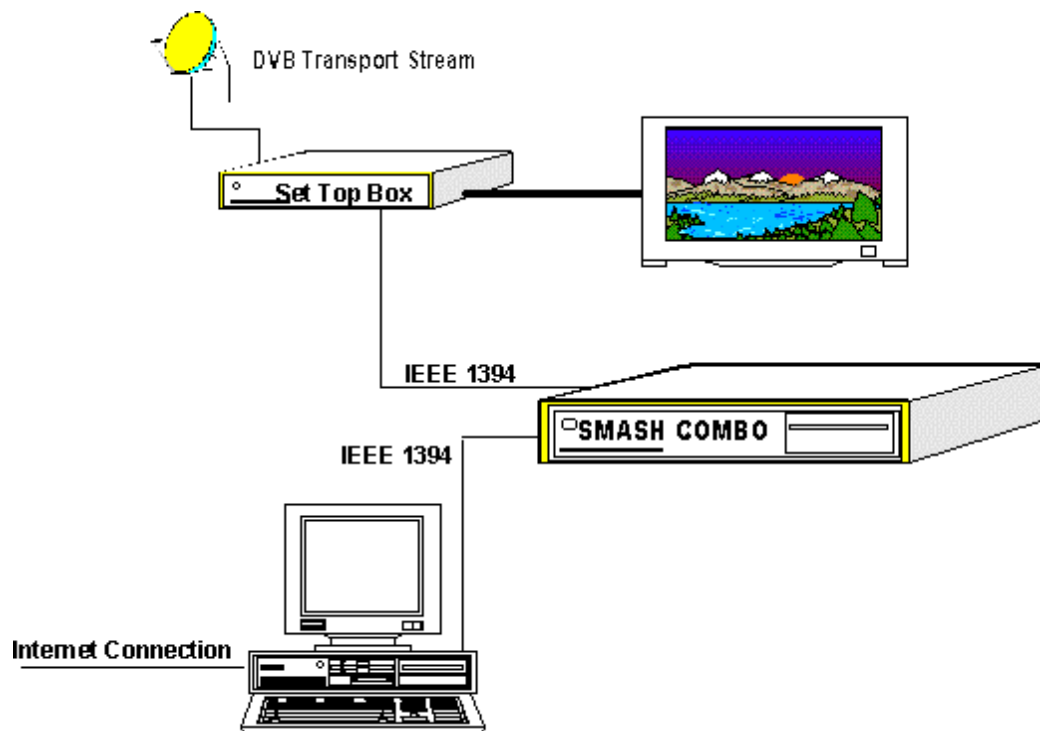


Figure 8 Residential Network

Typically an application on a client makes use of the functionality as provided by the API. The underlying hardware is hidden for the application programmer. On the *SMASH* server, a controller deals explicitly with the peculiarities of the current hardware architecture.

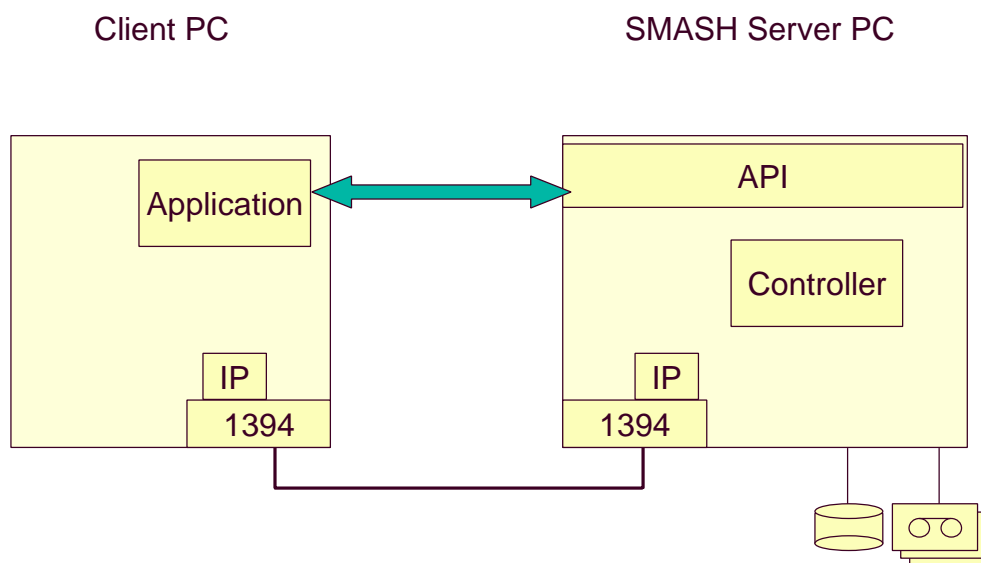


Figure 9 Client Server Relation

If we zoom in on a Web-based application (like the Web-VCR, or the REA), we see the following interactions:

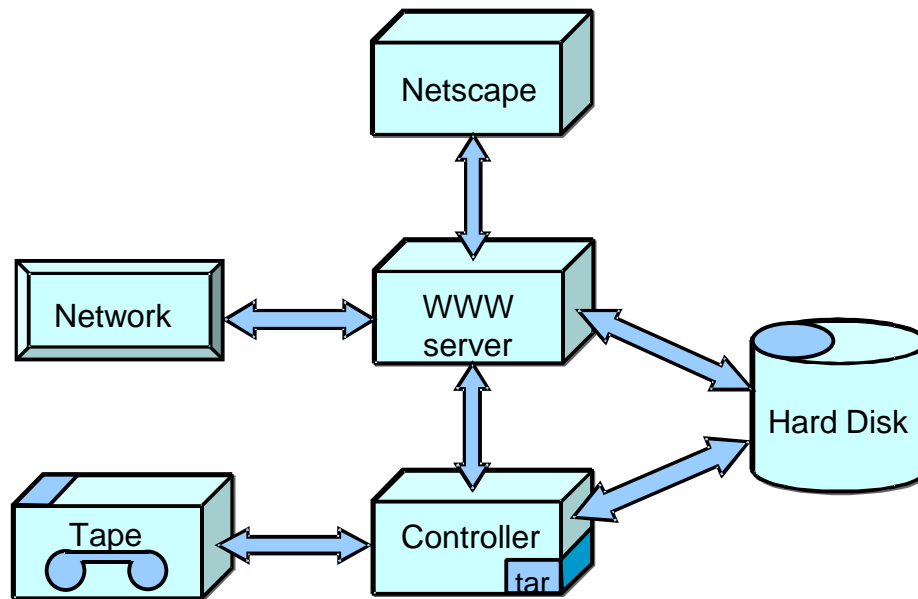


Figure 10 Model of WEB based Applications

The functionality needed to implement the interactions between controller and the tape drive and hard disk drive is typically described by the low level API.

3.3 Low level API

The low level API is aware of the storage media. In the current implementation we need to deal with tapes, tape partitions, and a disk for caching tape partitions. This disk can be accessed using the native file system of the client using the Samba protocol, which supports open, close, read, write of files. Information on files stored on tape can be found in the TOC, which is copied upon session initialization to disk. The implementation of the TOC is application dependent. We only describe the commands from the low-level API that are relevant for the end user application.

In principle, files are retrieved partition-wise. Thus, we need to be able to find out in which partition(s) a particular file resides.

Lookup(*Filename*)

Returns the partition in which *Filename* resides or the first partition if the file occupies more than one partition.

Once we know in which partition a particular file resides, we can copy that partition to disk:

Restore(*TapePartition*)

Copy the files in *TapePartition* to disk.

Vice versa, files can be written to tape:

Archive()

Copy all the files on disk to tape. Update TOC.

Archive(*FileList*)

Copy the list of files identified by *FileList* to tape. Update TOC.

3.4 File system API

The interface available for files is provided by the classes in the `java.io` package. At the moment storage space is limited by the size of the exported server disk. The commands in the low level API can be used to enlarge that by copying tape partitions back and forth to the exported disk at the appropriate time.

3.5 Streams API

Streams are a combination of a file and a port, where the file must either send or receive a stream of bits from the specified port in the communication network at a specified sustained bit rate. Streaming functionality can be handled using the file system interface. It is the responsibility of the application to read (write) the data from (to) a file at the appropriate rate. In the current configuration we can handle two of such streams at the same time.

If needed play, pause/resume, stop, and record can be implemented on top of general file system commands. Obviously, these commands are file type dependent, since playing MPEG-1 is different from playing AVI, QuickTime, or MPEG-2 single program transport stream. (At the moment we have an implementation of the commands for the DVB-VCR application). The signature of these commands is as follows:

Play(*Port*, *Filename*)

Allocates the resources and starts playback of a stream from the file identified by *Filename* to the *Port*. A reference to the stream object for further stream commands is returned.

Pause(*Handle*)

Interrupt a stream identified by *Handle*. During the interrupt all the resources (bandwidth, ports, etc.) will remain reserved.

Resume(*Handle*)

Resumes a previously interrupted stream.

Stop(*Handle*)

Stops recording of playback of a stream and releases all resources.

Record(*Port, Filename*)

Allocates the resources and starts the recording of a stream coming from *Port* to the file identified by *Filename*. A reference to the stream object for further stream commands is returned.

Status(*Handle*)

Returns current values of stream attributes.

3.6 Operational Management API

Recall that we introduced sessions to deal with application-specific implementations of generic commands. Each session bracketed by

OpenSession - **CloseSession** pairs:

OpenSession(*SessionType*)

Start a session of type *SessionType*. Returns a reference to the session for further commands. Initializes on-disk TOC, if needed. Perform other initializations depending on *SessionType*.

CloseSession(*SessionId*)

Update TOC, if needed. Release resources.

Further we have some commands dealing explicitly with tapes:

List()

Returns the files that reside on the current tape.

List(*TapePartition*)

Returns the files that reside on *TapePartition* of the current tape.

Free()

Returns the amount of free space of the current tape.

Eject()

This function ejects the tape from the tape drive.

Insert()

This function inserts a tape into the system. If the tape was not used before, it can be initialized, otherwise the TOC should be read.

Only the REA needs to deal with several tapes explicitly. Namely, a course resides on a particular tape. We assume that such a tape is only temporary in the system. The tapes, which are permanent in the system, as is the case for the DVB-VCR and the Web-VCR applications, are handled by a tape autoloader. Here the application programmer need not worry about individual tapes. However, manually changing a tape can be done using **Eject** and **Restore** commands.

4. ANNEX

4.1 The Web-VCR

The Web-VCR is an application which can be dynamically loaded into an appropriate Web browser such as the Netscape one, which enhances the browser with VCR-like capabilities, allowing to play video streams coming from the network, recording them on the Combo tape with the option of recording text and graphics on the associated Web.

The main benefits of this application for the user can be summarized as follows:

- recording of the Web pages on a site and all the related content allows to visit them multiple times without paying for the connection;
- hosts may sometimes be down or inaccessible; content can change over time; it is often useful to store locally the content of an interesting site, when found;
- the Combo provides a large capacity and fast access compared to normal linear tape drives or normal hard disks;
- the Combo allows to record streams at variable bit rates, making possible to record text, audio and graphics together with the video streams;
- the Combo allows to change tapes while recording without loss of data;
- the 'garbage collection' and the inevitable removal of unwanted content are causes of a highly inefficient use of storage space, due to fragmentation on the tape. The combination of tape unit and hard disk provides the possibility to rearrange and defragment the valid content, resulting in a more efficient use of storage space and faster access to the stored content.

The development environment for this application is the JAVA programming language under Windows95.

The application has been designed with a layered architecture using the JAVA paradigm as shown in the following figure:

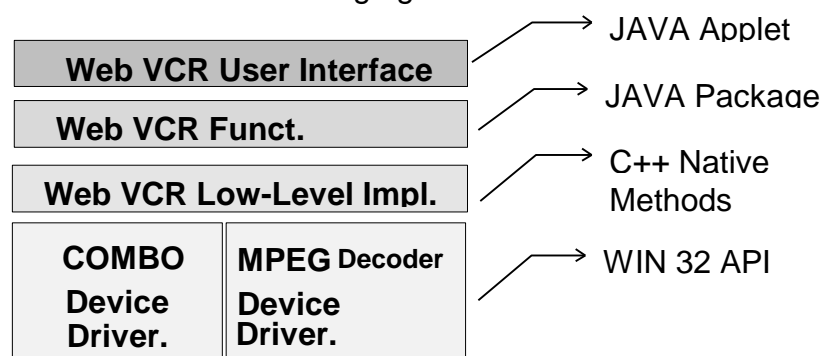


Figure 11 Layered Structure using JAVA

The JAVA applet implements the Graphical User Interface of the application while the software functionality has been implemented in a JAVA "Package"

called *StreamVideo*. A JAVA Package is a way to categorize and group classes .

Due to the fact that the application have to interface with a Win32 API of functions either to interact with the Combo or with the MPEG decoder card, the methods of the *StreamVideo* package have been implemented as “JAVA native methods”, that is a method whose code is written in a different language from JAVA as C or C++.

Hereafter follows a brief description of how the WEB-VCR application works.

The application can have two different types of interaction depending from the context where it works:

- *Internet*: when the video stream referred is stored on a remote Internet site
- *Combo*: when the video stream referred is stored on the Combo

INTERNET INTERACTION

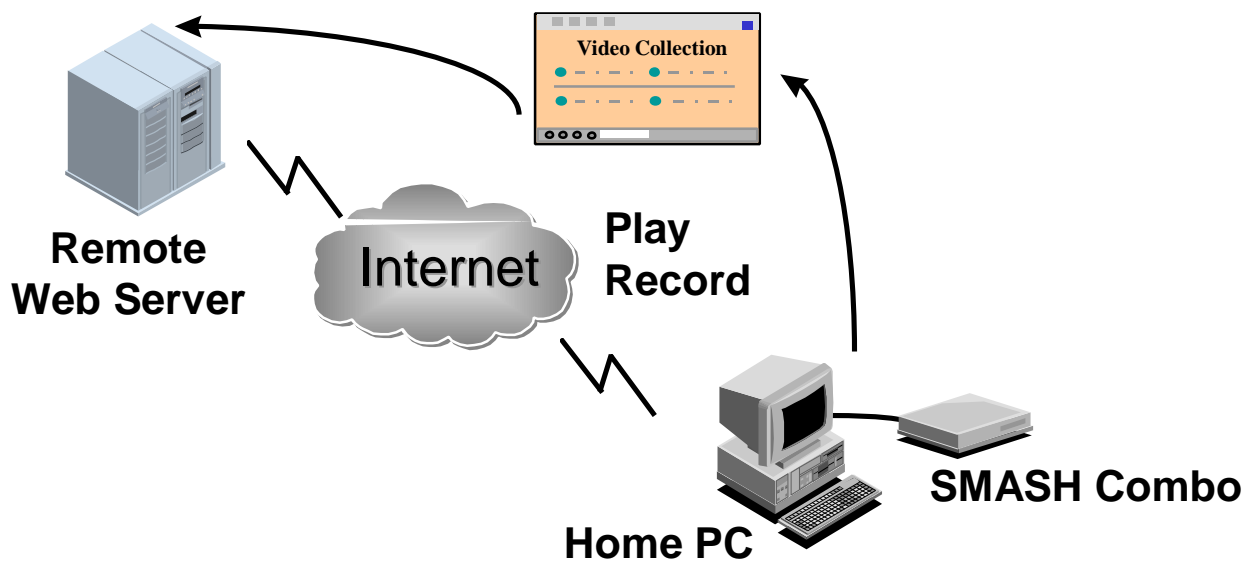


Figure 12 WEB-VCR Internet Interaction

COMBO INTERACTION

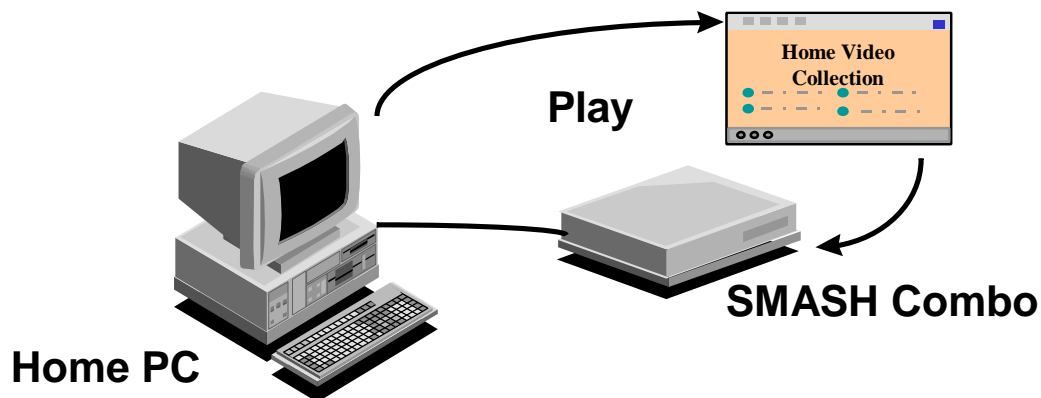


Figure 13 *WEB-VCR Combo Interaction*

The standard VCR interface, simulated by the applet JAVA, provides different commands for the two environments.

During the so called “*Internet Interaction*” the user can select a video link thus enabling the activation of the JAVA applet which simulates a VCR user-interface with the functionality of PLAY, PAUSE, STOP, RECORD and RECORD PAGE. In particular the PLAY function starts a connection between the remote server and the local PC and allows the transfer and play-back of the selected MPEG file from the WEB server to the PC. The RECORD function acts in a similar way as the PLAY function, but the stream video beside being played is also stored in a Combo file and a reference is added to the “ComboIndex” page.

The RECORD PAGE function allows to record in the COMBO system an entire HTML page with all its first level video-links.

During the “*Combo Interaction*”, the VCR user-interface shown by the JAVA applet, provides the functionality of PLAY, PAUSE, STOP and DELETE. In particular the PLAY function starts a buffered connection between the Combo device and the local PC to enable the buffered data flow from the Combo towards the decoder board. In a first release of the application prototype the video data are transferred from the Combo to the application PC using a Ethernet connection, and then, for the final demonstration, the 1394 Bus connection will be used. The DELETE function allows canceling a video stream from the COMBO together all the references to it.

As regards to the “*Internet Interaction*”, the RECORD function is not available because it is not meaningful in a Combo context.

The following figure shows the links between the Home PC running the WEB-VCR application and the SMASH COMBO device.

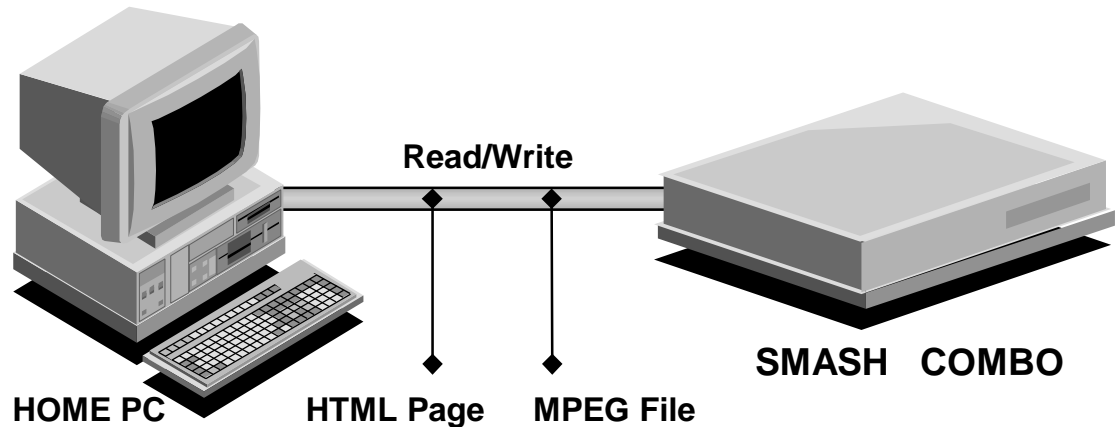


Figure 14 WEB-VCR PC/ COMBO connections

The exchanged information between the two systems belongs to different data type, and for this the WEB-VCR application needs to ask the COMBO for the following operation:

- I/O of small files like HTML pages, and then some “File System” functionality should be provided.
- Buffered I/O of MPEG files.
- Management of COMBO system itself (i.e. Open/Close session, Load / Eject Tape ...)
- URL addressing mechanism to access HTML information stored in the COMBO system.

4.2 The usage of the *SMASH* API by the DVB-VCR application

The DVB-VCR (Digital Video Broadcasting – Video Cassette recorder) uses the *SMASH* Combo as an advanced video recorder, that can e.g. record and play back simultaneously. As such, it utilises the *SMASH* streaming API to handle the video streams.

The DVB-VCR would be operated by a consumer through its graphical user interface (GUI) displayed on a television. This TV is connected to the digital home network, to which a wide range of other equipment may be connected (see figure 15 below). In our case, the TV will communicate with the Combo server, which need not necessarily be in the same room. In fact, more TV's in the same home may use one Combo.

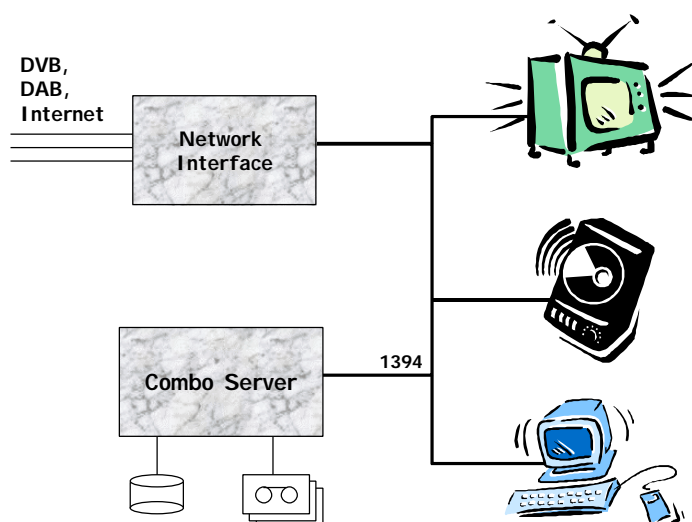


Figure 15: *SMASH* Combo server in a home network environment.

The consumer will be presented a GUI that offers possibilities to select future programs for recording, or select previously recorded programs for viewing. The information on available programs, both future broadcasts and recordings, is retrieved from a 'multimedia database' (MMDB). The MMDB content and access is not currently contained in the API, because it is very application-specific, and is expected to be subjected to many changes in the near future.

After the consumer has made his selection from the menu, say to play back the movie *Jurassic Park*, appropriate commands are sent to the Combo to start the playback process. The MMDB contains all information required to issue the correct streaming API calls, e.g. correct reference to the *Jurassic Park* file(s) on the available tapes.

During playback, the consumer may issue further commands, e.g. to pause and later resume the playback, using his remote control. Again, this will result in the appropriate API calls to achieve the desired action.

In a similar fashion other equipment connected to the network may utilise the Combo. One could imagine PC's accessing the Combo to retrieve (parts of) movies for playback or editing. Similar to video, audio data could be retrieved to be played anywhere in the house, and many other possibilities exist.

4.3 The Usage of the *SMASH* API by the Remote Education Application

The Remote Education Application (REA) is based on the WWW to use the network connection and to be supported by a local mass storage device as well. A graphical user interface is built under the a WWW browser (Netscape or Microsoft Internet Explorer) because the idea is to keep the application as general as possible. This means that the same concept of the educational tool can be used with or without a local mass storage device. To achieve such goal, the REA should a clear interface between the application level and the storage device to hide the storage system from the application. A complete concept of the interactions between the REA, the *SMASH* Combo as a mass home storage system and the network access has been shown in Figure 15.

All data requests from the REA are sent to the simple WWW server, which is running on the same computer as the REA. Local file requests are checked by the WWW server, all other requests are passed to the network. The same situation is when data comes through the server back to the REA.

When the simple WWW server receives a file request from the REA (WWW browser), it tries to get the file from a hard disk of the Combo, which is mounted as a local disk on the client PC. If the file is on the hard disk, the WWW server sends it to the WWW browser. However, if the file is not on the hard disk, the WWW server receives an error message and after this, it sends a request for a file to the tape controller running on the Combo system. When the file is on a hard disk, the WWW server sends the file to the Netscape. If the tape controller for any reason cannot retrieve the file, it sends an error message to the WWW server.

At the moment, the client PC is connected to the Combo system by the Ethernet using the TCP/IP protocol. This means that all commands and all data are exchanged between the client PC and the Combo using IP packets. The communication between the REA and the Combo is defined by the *SMASH* API. Because the REA is based on the WWW it needs only API commands that can be used with the WWW browser. This is the reason that the stream API, defined as a part of the *SMASH* API, is not used in the REA.

The communication between the client PC and the hard disk of the Combo is performed by the file system API, which is the same as for any other NFS and it will probably remain the same in the future. The communication between the simple WWW server on the client PC and the tape controller on the Combo system is performed over Internet and it is also based on the TCP/IP protocol, but it can be changed in any other better connection supported by Java in the future. The simple WWW server communicate with the tape controller using low level and operational management API defined in the Deliverable 8.