

Software Architecture of Mass Local Storage Supported Remote Education Application

Andrej Košir, Marijan Leban, Jurij F. Tasič

University of Ljubljana

Faculty of Electical Engineering

Tržaška 25, SI-1000 LJUBLJANA

Abstract

Development of computer technology, like multimedia and the Internet, enables new possibilities for remote education. Many professions nowadays require permanent education what is accompanied with travelling. In order to reduce time consumption and to protect environment from pollution an Remote Education Application can be used. Lessons involved are prepared from different type of materials to give learners a better insight into the topic and to enable them to be more creative in the educational process. This means that learners have possibilities to study from their home or their jobs without loosing the quality of the education. Proposed remote education application is based on the World Wide Web (WWW) to be platform independent and available to the most learners. Remote education can be performed on the server side from the learner's computer over the network. The problem is, however, that most learners usually do not have high bandwidth network connections at their homes or at the campus. A possible solution is usage of a local mass storage device to store almost all multimedia data locally and only a small amount of material is accessed remotely over the network. The purpose of this article is to present the basic idea and the concept of the distance education, where a local mass storage device is used to store all multimedia data locally. A COMBO [1,2], which combines a linear tape and a hard disk into one storage system, is used as a local mass storage device.

1 Introduction

The overall need in the remote education system identified recently is to enable the usage of the Remote Education Application (REA) terminal at the site with the narrow-bandwidth links. A mass storage device, which is the goal of the project ACTS AC-018 SMASH, can help to solve the bottleneck problem. We investigated the possibility of usage a mass storage unit combined with a low-speed connection like a modem to improve the efficiency of the multimedia education application, which could use the on-line connection to the teacher's server. The REA can also be run in the off-line mode, where only a local mass storage device is used as a source of the multimedia educational material. To avoid slow response time of the system caused by a transfer a huge amount of data via the narrow-bandwidth link we suggest a configuration where a local mass storage device is located not only on a teacher side but also on a client side. A network link between a teacher and a learner is then used to bear an interactive connection, which allows sending messages, comments and other information needed to be available on-line. To establish such application we need a two-way client server

connection between both sides. From this we can see that a Java programming language and an Internet connection, as a network link should be applied here. The main objective of this paper is to describe an approach to Java environment supported client-server application as a base for Remote Education Application.

2 Remote Education Application

Distance education or distant learning has great potential to aid education, which is of critical interest due to the convergence of trends in education. First, occupational and personal success highly depends on education and formal education is becoming a lifelong endeavour for those in the professions. Second, education has become increasingly expensive and there is now increasing pressure to re-engineer the educational process to be significantly more cost effective. One of possible solutions is distance education, which will allow learners to study from their homes or from their jobs. Educational lessons can be very efficient if they are carefully prepared using multimedia, which enables usage of text, images, video, and audio materials together. To avoid repeated transfer of educational material via network connection, a mass storage device can be used store materials locally.

2.1 Structure of the Remote Education Application

An important idea is to keep the remote application as general as possible. This means that the same educational tool can be used with or without a local mass storage device. To achieve such goal, the distance education application is built as a general education application based on a WWW browser with an additional interface between the application level and the storage device to hide the storage system from the application. The remote education application can be divided into three software layers:

- application layer as a main education tool with the graphic user interface,
- interface layer for the connection between the application layer and the storage device, and
- management of the storage device.

On the application layer, the file requests are generated and transferred through the interface layer to the management of the local mass storage device, where the file requests are processed. Structure of the remote education application is shown in Figure 1. The WWW browser presents the application layer, the WWW server is the interface layer and the controller is used for the file management of the storage system.

The interface between the application and the storage unit is a simple WWW server, which is used to redirect file requests to the controller of the storage system and to send files back to the WWW browser. A program called Controller is at the lower software layer. It communicates with the simple WWW server on the upper layer and controls the operation of the storage system. The tasks of this software layer are:

- organisation and storage of the multimedia files into appropriate structures on the tape (defined by hierarchically optimised structure of the educational material according the organisation of the tape),
- whole management of the data files (on the disk and magnetic tape),
- copying of files from the tape to the disk and vice versa, and

management of the table of contents with the information about files stored on the tape.

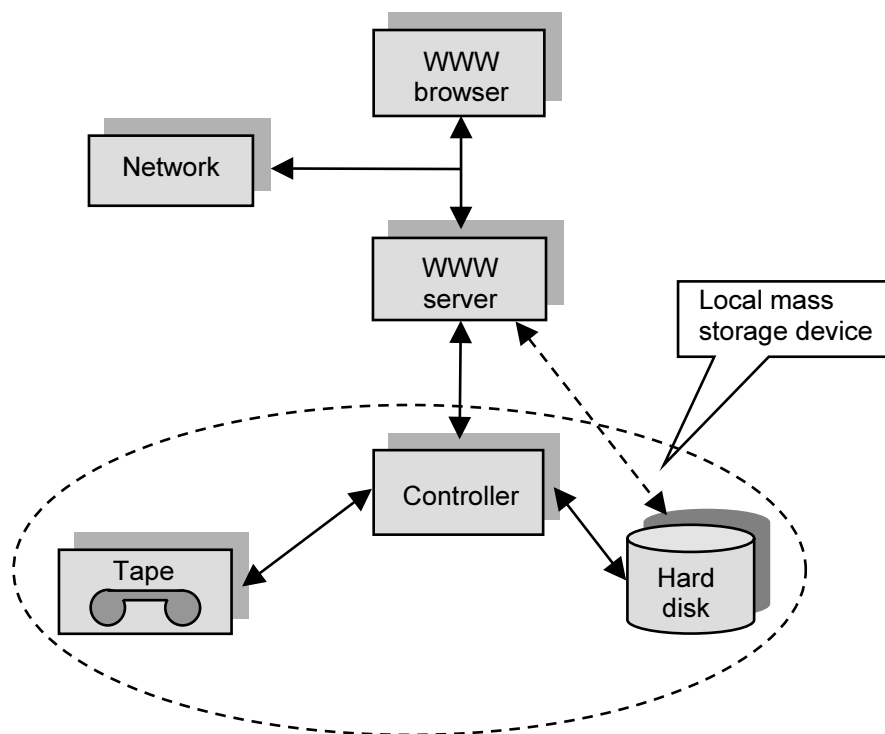


Figure 1 Structure of the remote education application.

2.2 COMBO in the Remote Education Application

A local mass storage device COMBO [1,2] consists of a linear tape with huge capacity 13 GB [3,4] and a 2 GB hard disk for caching data from the tape. At the moment, the Ethernet connects the local mass storage device with the computer, but in the future, a fast connection over the IEEE 1394 port will be available.

The main objective of the Combo system in Remote Education Application- REA is to present the effectiveness and attractiveness of the massive storage device. Typically for the application running in Windows NT environment supported by Internet communication possibilities is that it may extract some educational materials or from the network or from the incoming streams of data stored as data files on massive storage device (tape). The REA application may consist of a large amount of small files and some much larger multimedia files, like movies, virtual environment applications, etc. The Combo unit consists of two different storage units, magnetic tape providing large storage capacity, supporting exchanging of the units and the other with much faster access and smaller storage capacity. All of the files have to be stored on the multimedia tape, where the whole REA application has to be stored and during the application of the REA the 3G to 4G bytes hard disk has to be occupied by the sequences of 361 Mbytes blocks of partitions. The maximum occupied storage could be up to 2Gbytes.

For the organisation of the data stored on the massive storage media the special approach has to be in used. This approach has to enable the access of the user to the storage unit in software written drivers.

When a learner requires a local file, the request comes from the WWW browser to the simple WWW server, which tries to obtain the file from the hard disk of the COMBO. If the file is on the disk, it is sent to the browser, otherwise, the request is sent to the controller of the tape. The controller copies the whole partition (360 MB) with the required file on the hard disk. When the file is on the disk, the simple server sends it to the browser. The tape controller also takes care of writing files on the tape, which support only writing of streams. Files are grouped together into one stream which is then written on the tape. The concept of the interaction between the simple WWW server and the local mass storage device is shown in Figure 2. The direct interaction between the simple server and the hard disk of the local mass storage device is performed if the hard disk can be mounted as a local hard disk of the computer. If this is not possible, the hard disk is available only to the tape controller, which sends files to the simple WWW server.

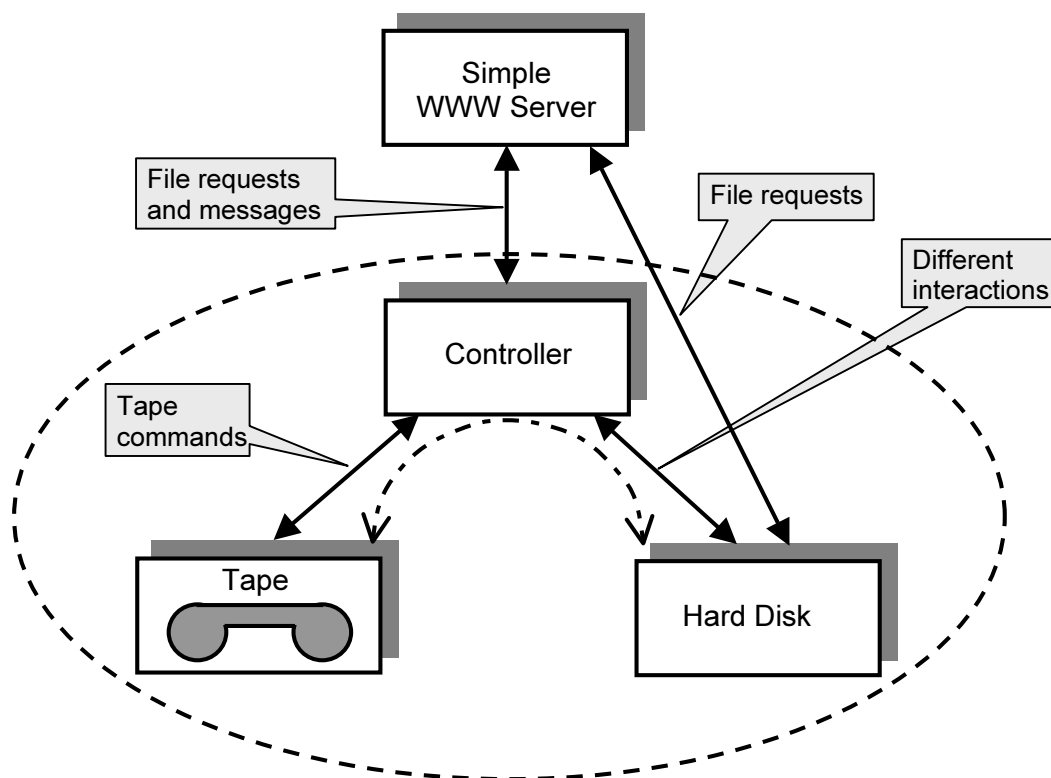


Figure 2 Interaction between the simple WWW server and the local mass storage device.

3 Java Programming Environment

Java's appearance in the right place at the right time with the answers to hard problems is earning it a place in the toolkits of developers world-wide. From a developer's standpoint, Java's cross-platform, secure, object-oriented, and network-centric features make it useful. Its syntax makes it relatively easy to use. Most importantly, its status as the only high-level cross-platform language with native Web browser support makes it a must for serious Web- and intranet-oriented development tool. It changes the passive nature of Internet and WWW capable of processing text only to architecturally neutral environment. Java environment

offers an opportunity to support network control functions, file management as well as any other computer algorithm to be evolved and implemented inside the same programming environment. This feature of Java environment is of crucial importance for our application in order to achieve platform independent application.

3.1 Class Java.net

The most important part that makes Java distinguished from other programming tools is a hierarchy of classes of package Java.net. We list its classes here with brief comments in order to introduce the programming approach we have chosen in our application.

- **InitAddress**
It represents an IP address. Its methods allow us to make inquiries about domain names and aliases.
- **URL**
It represents given source in Internet. It offers a wide range methods for parsing to make connection to a given resource. It takes absolute or relative URL.
- **URLConnection**
An abstract class for establishing network connections.
- **URLEncoder**
It brings a method for encoding strings to a MIME (Multiple Internet Mail Extension).
- **URLStreamHandler**
An abstract class which methods compose a frame of classes for handling data between different Internet protocols like HTTP, FTP or Gopher.
- **ContentHandler**
An abstract class on the top of the hierarchy of all classes for handling data accessed from the network.
- **DatagramPacket**
A class for datagram handling. It allows us to communicate directly to a network layer.
- **DatagramSocket**
A class for dealing width socket in order to communicate to datagrams.
- **ServerSocket**
A class intended for establishing server programs that scan a given port and respond to client requests.
- **Socket**
A class for dealing width sockets from client point of view.

- **SocketImpl**

An abstract class on the top of hierarchy for dealing with network connections established by two previously listed classes.

For obvious reason in our application we use classes Socket and ServerSocket. The Socket class in the Java.net package is a platform-independent implementation of the client end of a two-way communication link between a client and a server. The Socket class sits on top of a platform-dependent implementation, hiding the details of any particular system from your Java program. By using the Java.net Socket class instead of relying on native code, your Java programs can communicate over the network in a platform-independent fashion. In client-server applications, the server provides some service, such as processing database queries or sending out current stock prices. The client uses the service provided by the server to some end: displaying database query results to the user or making stock purchase recommendations to an investor. The communication that occurs between the client and the server must be reliable--no data can be dropped and it must arrive on the client side in the same order that it was sent by the server.

TCP provides a reliable, point-to-point communication channel, which client-server applications on the Internet use to communicate. The Socket and ServerSocket classes in Java.net provide a system-independent communication channel using TCP.

A socket is one end-point of a two-way communication link between two programs running on the network. Socket classes are used to represent the connection between a client program and a server program. The Java.net package provides two classes--Socket and ServerSocket--that implement the client side of the connection and the server side of the connection, respectively.

A server application normally listens to a specific port waiting for connection requests from a client. When a connection request arrives, the client and the server establish a dedicated connection over which they can communicate. During the connection process, the client is assigned a local port number, and binds a socket to it. The client talks to the server by writing to the socket and gets information from the server by reading from it. Similarly, the server gets a new local port number (it needs a new port number so that it can continue to listen for connection requests on the original port). The server also binds a socket to its local port and communicates with the client by reading from and writing to it.

This client program is straightforward and simple. The client sends text to the server, the server echoes it back. When your client programs are talking to a more complicated server such as an http server, your client program will also be more complicated. However, the basics are much the same as they are in this program:

1. Open a socket.
2. Open an input stream and output stream to the socket.
3. Read from and write to the stream according to the server's protocol.
4. Close streams.
5. Close sockets.

Only the step 3 differs from client to client, depending on the server. The other steps remain largely the same. When you write Java programs that communicate over the network, you are programming at the application layer. Typically, you don't need to concern yourself with the TCP and UDP layers--instead you can use the classes in the Java.net package. These classes provide system-independent network communication. However you do need to understand the difference between TCP and UDP to decide which Java classes your programs should use. When two applications want to communicate to each another reliably they establish a connection and send data back and forth over that connection. You send data back and forth

over the connection by speaking to one another over the phone lines. Like the phone company, TCP guarantees that data sent from one end of the connection actually gets to the other end and in the same order it was sent (otherwise an error is reported).

Applications that require a reliable, point-to-point channel to communicate, use TCP to communicate. Hyper Text Transfer Protocol (HTTP), File Transfer Protocol (ftp), and Telnet (telnet) are all examples of applications that require a reliable communication channel. The order that the data is sent and received over the network is critical to the success of these applications-when using HTTP to read from a URL, the data must be received in the order that it was sent otherwise you end up with a jumbled HTML file, a corrupt zip file, or some other invalid information.

For many applications this guarantee of reliability is critical to the success of the transfer of information from one end of the connection to the other. However, other forms of communication don't require such strict communications and in fact are hindered by them either because of the performance hit from the extra overhead, or because the reliable connection invalidates the service altogether.

4 COMBO Server/Client application

Let us focus on client/server application connecting COMBO PC as a server with learner's PC as a client. A client side Java program named COMBO controller is based on a class Socket, server side program named Simple Java WWW server is based on a class ServerSocket. Both sides are connected via Internet Connection. User interface is Java Applet program running in a Web Browser window. The whole situation is depicted in Figure 3.

In order to explain its functioning the best way is to follow steps of all involved programs. To run the remote educational application, the Combo controller must be running on the Combo and the tape with the courses must be inserted in the tape drive on the Combo. On the client side, which is a learner's PC, the simple Java WWW server is started first. After this, the learner starts a web browser (Netscape or Microsoft Internet Explorer) supported with multimedia plug-ins. From the web browser, the learner connects to the simple WWW server, which is configured for the Internet connection with the Combo, with the URL "http://localhost:port_number/". The simple WWW server returns a starting page of the courses and the learner can start the learning.

When a file is required from the user interface by the web browser, the simple WWW accepts a connection from the web browser and start a thread to obtain the file from the Combo hard disk mounted as a local disk of the PC. If the file is not on the hard disk, this thread starts another thread for the Internet communication with the Combo controller running on the Combo. First, the WWW server checks on which tape partition the required file is and then it sends a command to copy this tape partition on the Combo hard disk. To minimise the delay, the simple WWW server starts each 10 seconds a new thread to check if the required file has been copied on the hard disk. When the file is on the hard disk, the WWW server sends it to the web browser.

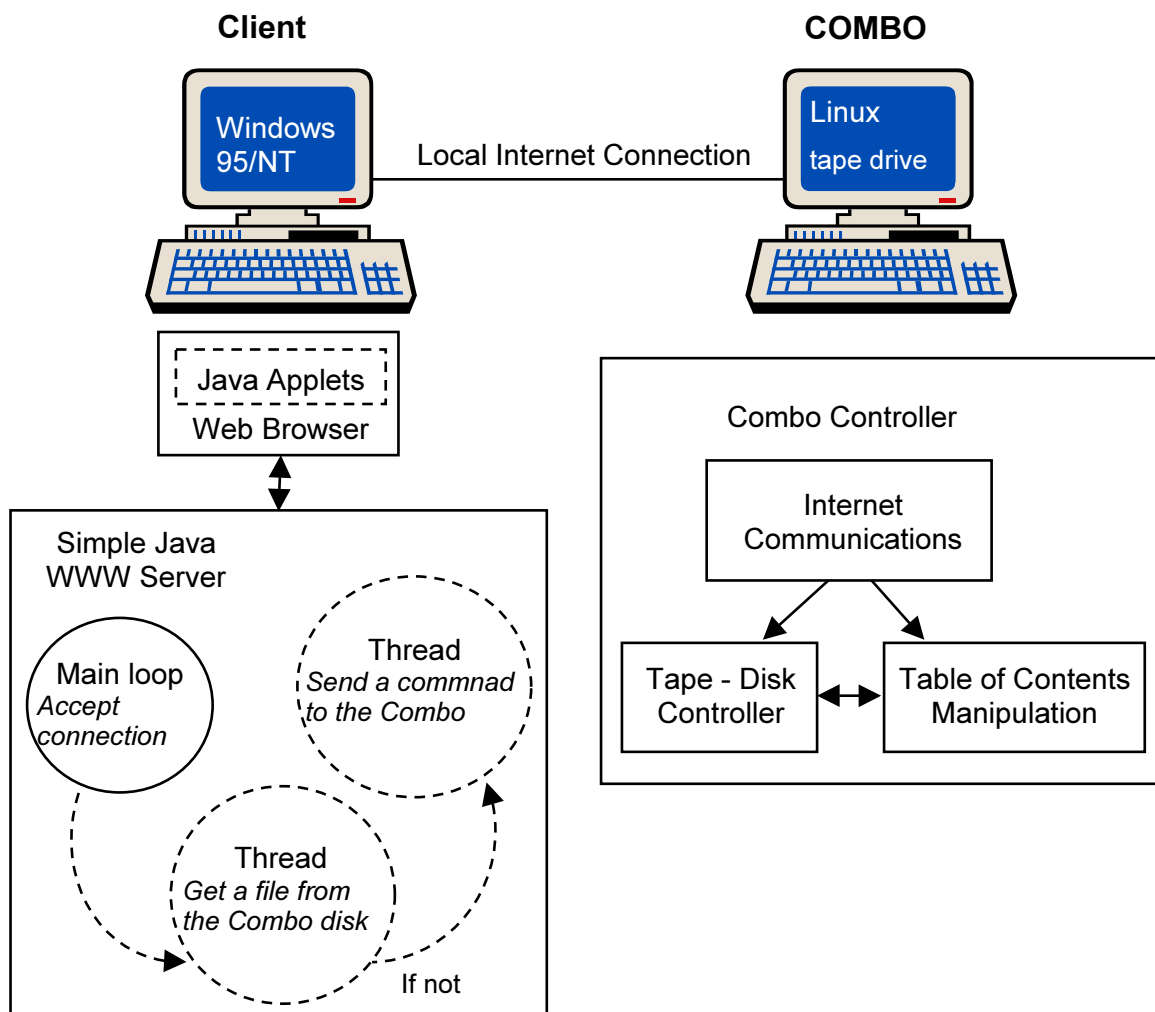


Figure 3 Server/Client application of REA.

4.1 A Graphic User Interface

A graphic user interface is a learner's door to the course material and it should be designed very carefully. It should be simple to use but with the clear overview of educational material and available functions. The interface could be implemented as ordinary hypertext pages, but usage of Java applets allows additional features like gathering of statistical information about the learning process and showing graphic map of the course. If the structure of the course is given only as a table of contents, the course is a hypertext book with additional multimedia materials on the document pages. The user interface can be improved by the graphic presentation of the educational material, which is more flexible and easier to remember. Mind patterns, which are similar to the human way of thinking, are graphic diagrams with different shapes and different types of connections, and therefore, they are suitable for representation of the relations between topics of the course. A simplified schematic example of the graphic user interface, which is built on the WWW browser, is shown in Figure 2. A window of the user interface is divided into four sub-windows. The left upper corner is used for showing a small graphic map of the whole course. This map is useful for finding out a temporary position of the learner in the structure of the course. The left lower window is a hypertext table of contents with links to subtopics. The table of contents defines the structure of the lesson,

which is defined by the teacher, and it could be upgraded over the network. The main window is the right lower one, which shows mind patterns. The mind patterns are a very flexible and efficient way to show the educational material. They use different shapes, sizes and colours, as well as different types of connections, to show the structure of the course and the relations between topics. The mind patterns don't show only the lesson prescribed by the teacher, but they show all available material included in the course, although the teacher does not require some topics. Multimedia technology enables more efficient presentation of the mind patterns, which can be additionally described by text, images or audio and video clips. The mind patterns are also links to subtopics and html document pages. They can be used to navigate through the course material. When an html document is reached by mind-patterns links or by links in the table of contents, this main window is used to show the multimedia data. If a link in an html document is followed, a new window for browsing is open to avoid losing of the learner's navigation through the course material. The last window of the user interface is the right upper toolbar with navigation arrows and action buttons for different functions like help, the interaction with a local mass storage device or the network interaction with the teacher's computer.

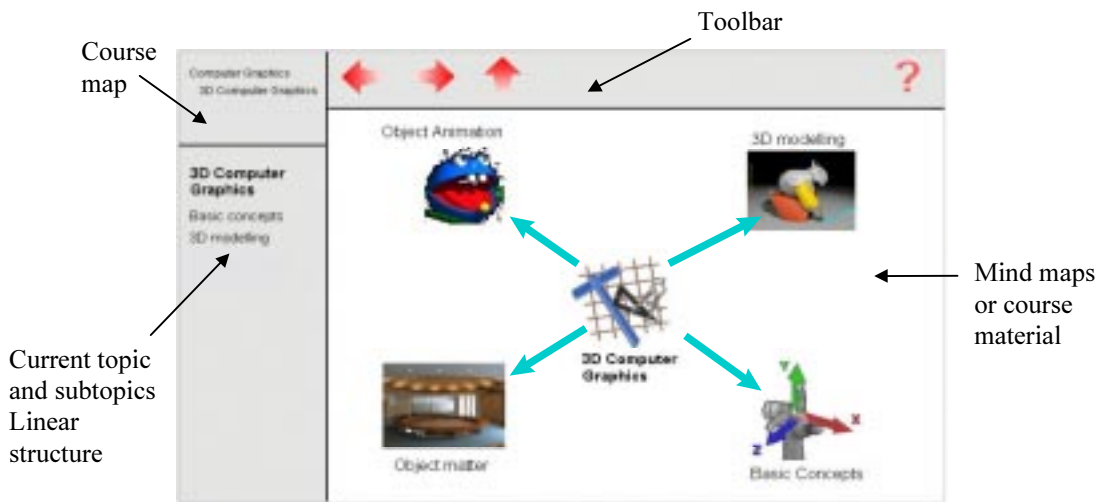


Figure 4 A graphic user interface with mind maps.

All course material is stored on a local mass device and it seems that the Internet is not needed, but this is not true. The network connection is used for additional browsing by the learner according to the links in the lesson as well as for obtaining the homework and instructions from the teacher and other modified data. For example, the structure of a lesson, which is defined by the teacher, can be obtained over the network if the teacher change the lesson. The learners can also send their homework back to the teacher, ask some questions, or they can communicate over the network between themselves. A real example of the user interface outlook can be seen in Figure 4.

5 Conclusion

The article presents a remote education application, where a local mass storage device is used to store all multimedia data locally. As a local mass storage device is used the COMBO [1,2], which combines a linear tape and a hard disk into one storage system. The educational system also supports the network connection, which is used for additional browsing of the WWW, for getting new modified data from the teacher, and for communication between the learners and the teacher. The application is not designed only for usage with a local mass storage device. It

can also be used with any other storage device or as a client-server application with all material stored on the server side. A simple WWW server is used as a connection between the education application (user interface) and the local mass storage device. The graphic user interface is based on a WWW browser and it implements mind patterns as multimedia supported graphic diagrams to present the course material and interconnections between related topics in a more efficient way. For the implementation of the user interface, simple WWW server, and the tape controller, the Java programming language has been used. Java environment is platform independent and suitable for network programming.

Acknowledgement

The work presented in this paper was supported in part by the European Community under the project AC018-SMASH. The authors would like to thank other partners on the project AC018-SMASH for their co-operation.

References

- [1] ACTS AC-018 SMASH project Deliverable #5, a0018/tud/it/ds/p/005/b1, 1996.
- [2] SMASH(ing) Home page, WWW: <http://www-it.et.tudelft.nl/pda/smash/>.
- [3] Tandberg MLR1 series, *Reference manual*, Tandberg Data ASA, August 1996.
- [4] Tandberg MLR1 series, *SCSI interface*, Tandberg Data ASA, March 1997.
- [5] R. O. Harger, "Teaching in a computer classroom with a hyperlinked, interactive book", *IEEE Trans. on Education*, vol. 39, no. 3, pp. 327-335, August 1996.
- [6] A. L. Sears, and S. E. Watkins, "A multimedia manual on the World Wide Web for telecommunications equipment", *IEEE Trans. on Education*, vol. 39, no. 3, pp. 342-347, August 1996.
- [7] S. A. Mengel, and W. J. Adams, "The need for a hypertext instructional design methodology", *IEEE Trans. on Education*, vol. 39, no. 3, pp. 375-380, August 1996.
- [8] D.A. Harris, and A. DiPaolo, "Advancing asynchronous distance education using high-speed networks", *IEEE Trans. on Education*, vol. 39, no. 3, pp. 444-449, August 1996.