configuration of nodes. The topology used in the Sigma-DSP3 is a simple one directional linear array.

There are three versions of the system which accommodate 32, 44 and 60 channels respectively and are intended for use with existing 24 channel, 32 channel and 48 channel digital tape machines. The processing power of the Sigma-DSP3 ranges from 1 Gflop in its 32 channel *version to 1.64 Gflops in its 60 channel version.*

The software environment is hierarchical (Figure 3). The top layer is Sun/Unix with c-compilers, c++ "front", and DSP32C assembler and simulator. This highest layer is used for software development only and is not part of the final system. The next layer is the Vertex operating system running on the RTH 68030. This layer handles Ethernet communications, booting, loading, configuring, and starting and regulating application procedures in the nodes. Vertex is multi-tasking and features a real-time clock, semaphores, mailboxes and queues for interprocess regulation. The lowest software level consists of the mixing and processing code and a custom operating system kernel of packet communication and topology switching routines used in inter-node communications. In the case of the Sigma-DSP3, all code running on the nodes is written in assembler for maximum speed.

III. Control

The intended control of the Sigma-DSP3 is the set of mixing faders and EQ and dynamics knobs on a scanned analog mixing console, or the movement of a cursor on a high resolution graphics screen of a computer virtual console. The method used is to send four byte messages over a TCP/IP Ethernet stream socket. The Sigma-DSP3 has an Excellan Ethernet VME card and an Ethernet transceiver. The physical connection to the system is a BNC "thinnet" cable to the transceiver. The Sigma-DSP3 has a unique Internet address and creates the server side of a TCP/IP socket during initialization after power-on. Clients on Ethernet need only create their end of the socket and connect to the already existing Sigma-DSP3 socket. Usually there are local operating system functions to accomplish these actions. For example, in Unix the functions are socket() which returns a socket descriptor, and connect() which establishes the software connection using supplied information about the server Sigma-DSP3. Once the socket is established the sending of control messages to the Sigma-DSP3 is exactly analogous to writing data to a file. The function call is identical but uses the socket descriptor in place of a file descriptor. Error checking, buffering and requests for resend are handled invisibly as part of TCP/IP. The Ethernet bandwidth is ten megabits per second so is more than adequate for the stream of fader and knob control information. The system expects four byte messages which communicate channel number, function type and parameter value.

A process running on the real-time host reads incoming messages, translates them into *information for the nodes and writes this information directly into the local memory of the* target node or group of nodes. An arbitrary number of bytes can be written and, in some cases, a semaphore is set in order to prevent the node processor from reading part of a previous data set and part of a new set.

IV. Determinism Via Serial Input Counter

An important technique for making any application more efficient is to make it deterministic. Then there is no need for case testing and the use of a bus can be precisely planned for maximum efficiency. For mixing in particular, the send of each channel packet can be made to occur at a precise time when the bus can be guaranteed to be free, and the arrival of packets at a mix node can be made to occur in pre-determined order so no checking of their source is necessary and mixing scalers can be read from a simple incrementing pointer.

The easiest way to achieve determinism is for all nodes to have an absolute time reference so actions can be scheduled globally to permit precise ordering of events. The DSP3 has no system clock accessible by all nodes. Therefore, one clock taken from the multi-track digital tape machine is used to to control all audio input and output, and each node keeps a local synchronized count based on the arrival or departure of samples on the serial port. Nodes in the system have two general functions. Channel nodes do equalization and dynamics processing, and mix nodes scale and accumulate samples sent from the channel nodes. The basic processing and mix unit is the packet which consists of 70 audio samples, i.e., 1.58ms duration. Channel nodes count samples by counting serial interrupts. Channel nodes always have three packets within their workspace. Packet N-2 was received from the serial port two packet "periods" ago and was loaded into the output FIFO in the previous packet period. In the present packet period packet N-2 waits to be sent to the mix nodes. Packet N-1 is the packet accumulated from serial interrupts during the previous packet segment and is now processed during the non-interrupt periods of the present packet period. Packet N is being accumulated