LAN Emulation Over ATM on HP-UX

Nicolas Catania
Network Technology Department
HP Laboratories Bristol
HPL-96-38
March, 1996

ATM, LAN-Emulation, HP-UX

To aid the early adoption of ATM technology as a local area network the ATM Forum has defined the LAN Emulation over ATM specification which aims to present a service interface more compatible with that of previous LANs. This document describes and defines a set of services -LAN Emulation Services - together with the components used for their provision. The paper reports on an implementation of the ATM Forum's LAN Emulation specification in a HP-UX (UN*X) environment and describes the system's architecture, some of the techniques used in the implementation and some of the problems encountered.

Internal Accession Date Only

LAN Emulation over ATM on HP-UX

Nicolas Catania
Hewlett-Packard Laboratories, Filton Road, Bristol, UK

March 7, 1996

Abstract

To aid the early adoption of ATM technology as a local area network the ATM Forum has defined the LAN Emulation over ATM specification which aims to present a service interface more compatible with that of previous LANs. This document describes and defines a set of services —LAN Emulation Services— together with the components used for their provision. The paper reports on an implementation of the ATM Forum's LAN Emulation specification in a HP-UX (UN*X) environment and describes the system's architecture, some of the techniques used in the implementation and some of the problems encountered.

1 Introduction

To aid the early adoption of ATM technology as a local area network the ATM Forum has defined the LAN Emulation over ATM specification which aims to present a service interface more compatible with that of previous LANs. This document describes and defines a set of services —LAN Emulation Services— together with the components used for their provision. The paper reports on an implementation of the ATM Forum's LAN Emulation specification in a HP-UX (UN*X) environment and describes the system's architecture, some of the techniques used in the implementation and some of the problems encountered.

The first section consists of a brief overview of LAN Emulation over ATM. In the second section we shall discuss the set of design choices, that arise when implementing network protocol stacks in HP-UX. The main problem consists of deciding where the code should reside and execute, either in kernel and/or user space.

In the third section, we shall see how the LAN Emulation functions may be partitioned between the two execution spaces, together with an examination of various performance attributes such as throughput, response times, and bridging requirements.

An ATM test-bed based on low-cost EISA ATM adaptors and prototype ATM switches were used to develop the LAN Emulation software and analyze its performance. This identified that one component's performance —the *LAN Emulation Client*— was critical to the overall system operation. Its implementation in the kernel, and the avoidance of multiple data copies while protocol processing, has resulted in a significant increase in throughput. Several interesting aspects of the implementation will be explained in the fourth section.

Finally the paper will report and analyze the results of performance tests used to identify and isolate critical functions. Based on these observations a number of proposals are presented for further research.

2 LAN emulation overview

2.1 LAN emulation components

The *LAN emulation over ATM* (LE) is a service that emulates the *Medium Access Control* (MAC) functionality of traditional LANs. The MACs emulated are `Ethernet/IEEE 802.3` and `IEEE 802.5`. Thus, today's network protocols and applications using these MACs will be able to transparently connect to an ATM network. The whole system is functionally partitioned as follows:

- One or more *LAN Emulation Client* (LEC). The LEC sits in an ATM end-system (e.g. workstation) and provides:
  - A traditional MAC emulation to the upper layers, or some bridging capabilities, and
  - Address resolution and control functions.

  Each LEC belongs to an *Emulated LAN* (ELAN), and has the ability to set-up a direct *Virtual Circuit Connection* (VCC) with any other LEC on the same ELAN, and transmit data. LECs belonging to different ELANs must use some bridging or routing services to interconnect. A LEC joins a specific ELAN by successfully registering with the ELAN's associated *LAN Emulation Server* (LES) (see below).
- One *LAN Emulation Service* (LE Service). The LE Service could reside in an end-system or in a switch. It provides services to the LECs to help them emulate traditional MACs. The LE Service is composed of three servers:
  - The *LAN Emulation Server* (LES). There is one LES per ELAN supported by the ATM network. It implements the control coordination function for the Emulated LAN by registering and resolving MAC addresses to ATM addresses. Each new LEC should register with the LES to avoid address conflicts. If the LES cannot resolve a MAC to an ATM address, it can broadcast an LE_ARP request to all LAN Emulation Clients, similar to the way it is done in traditional LANs. The end-system with the requested MAC address replies to the sender, thus providing its ATM address.
  - The *Broadcast and Unknown Server* (BUS). Because of the connection oriented nature of ATM, a specific mechanism was designed to emulate traditional LAN broadcasts, that is data sent to an all 1's MAC address: the BUS. It handles broadcast, multi-cast and initial uni-cast frames. When a LEC sends data to the BUS, the latter serializes the packet and forwards it to every member of the ELAN. If a LEC does not know the destination LEC's ATM address, it broadcasts the frame, thus ensuring that at least the target ot the proxy receives it. As soon as the ATM destination address has been resolved, via the LES, the sender establishes

a direct VCC to the destination LEC and starts sending the data traffic over the VCC and stops forwarding data to the BUS. A very simple protocol is provided to avoid frame mis-ordering during path switching.

- One *LAN Emulation Configuration Server* (LECS). This manages the assignment of individual LECs to an Emulated LAN. A LEC desiring to join a specific ELAN, but not knowing its LES ATM address, sends a request to the LECS, providing a string that identifies the ELAN. The LECS searches its database, and replies providing the LES ATM address of the corresponding ELAN. This server is optional, and a LEC can have a preconfigured LES address in memory, and contact the LES directly to join its ELAN.

An interface called the *LAN Emulation User to Network Interface* (LUNI) and a well defined protocol, dictate how the LAN Emulation Clients and the LAN Emulation Service interact together.

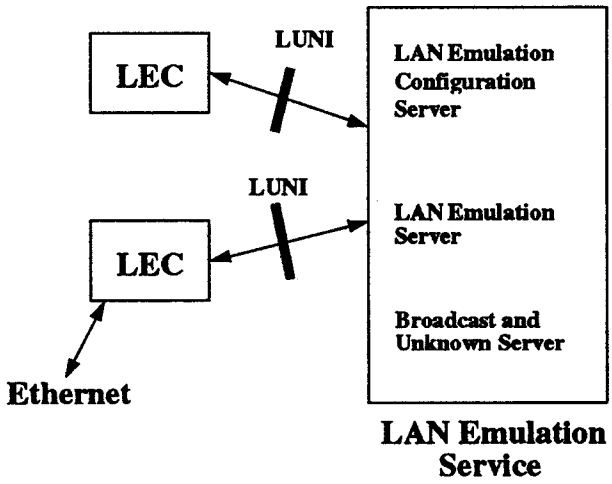

Figure 1: The LAN Emulation Components

2.2 LAN emulation layered architecture

A LAN Emulation Entity (i.e LEC, LES, BUS or LECS) interacts with the surrounding layers as follow:

1. The interface between the LAN Emulation Entity and the Higher Layers provides facilities for transmitting and receiving user data frames.

2. The interface between the LAN Emulation Entity and the ATM Adaptation Layer provides facilities for transmitting and receiving AAL5 frames.

3. The interface between the LAN Emulation Entity and the Connection Management Entity provides facilities to set-up or release VCCs.

4. The interface between the LAN Emulation Entity and the Layer Management Entity provides facilities to initialize and control the LAN Emulation Entity.

5. The interface between the LAN Emulation Client and the LAN Emulation Service (LUNI) includes facilities to handle broadcast traffic, address resolution and LE initialization.

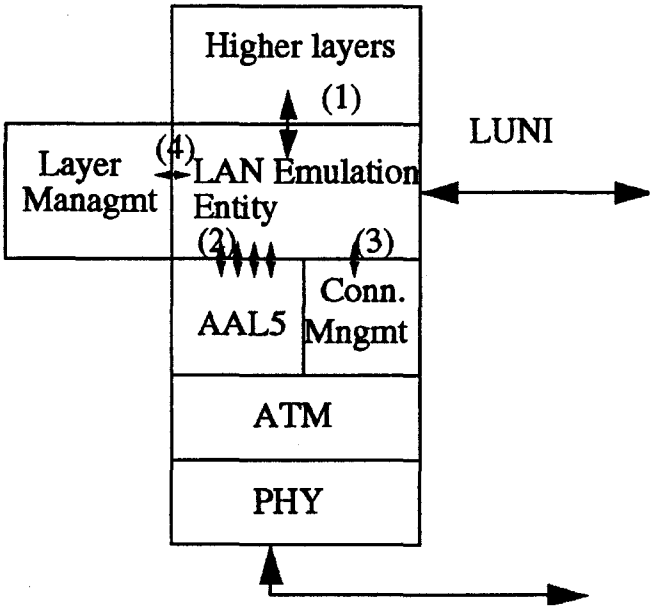

Figure 2: The layers

3 User-space vs. kernel implementation of networking protocols

The HP-UX environment is composed of the *kernel* (system) and *user-space*. The kernel provides four basic utilities; system startup, file system management, process control and communication (including networking) [2][3]. These services are accessible from *user-space* via system calls.

The user-space is where user processes execute. Each one is provided with a private address space and the kernel actions are invisible to an application. We shall examine the advantages and drawbacks of implementing a network protocol in each of these spaces [2].

3.1 User-space implementation

Although in an end-system the networking services usually reside in the kernel, some work has recently been carried out on implementing parts of the communication system as *daemons* running in user-space [4].

A big improvement with this approach is that it is easier to write and test networking code. The system calls available from user-space are generally well documented, and the compilation lighter. Additionally, if the program goes wrong, the whole machine does not crash, avoiding long boot times. Instead, an image of the process memory is dumped into a file that can be used for debugging (core dump).

There are, however, several drawbacks with this concept. First, HP-UX is not a micro-kernel and processes in user-space cannot control the scheduling policy in use. This means that another process intensively using the system could put a networking module into a *sleep* state for most of the time. This can affect responses to asynchronous network events. For the time being there is no easy means of modifying the system scheduler operation in order to guarantee a *real-time* response in user-space.

Another problem in running the networking module in user-space is that it would be exposed to all the vagaries of the system (e.g. daemon killed or not started, etc...).

Finally, because there is a clear demarcation between the kernel and the user-space, data moved from one space to another, must be copied word-by-word. This operation is mandatory anyway because we need to deliver data to the user at a certain point. But the choice in the layer on top of which we execute this copy is critical. For instance, if we transfer raw ATM cells from/to user-space, only 48 out of the 53 bytes moved are *user data*: this is a big overhead. It is clear that, when AAL5 re-assembly is done in the kernel, then upto 1516 bytes of user data maybe copied with only 8 extra bytes for the AAL5 trailer, which reduces significantly the overhead. Besides, as every layer provides some kind of checking for the validity of the data, by implementing only higher layers in user space, while keeping the lower ones in the kernel, we can avoid transferring corrupted data, and thus increase the efficiency of the system through early verification.

3.2 Kernel implementation

Contrary to what happens in user-space, code written in the kernel can mask interrupts in order to execute without being preempted by other parts of the kernel. Note that code segments executing at high priority, or even directly on the hardware interruption stack, must be kept as short as possible to avoid losing some interrupting events occurring at a lower level.

Furthermore, in the kernel, it is possible to choose how networking layers communicate with each other. In user space, shared memory between processes is the only viable solution for networking modules to exchange data. In the kernel, you can chose between an equivalent

operation, a *lose binding* consisting of enqueuing messages at the next layers entry point and posting interrupts or, because you can directly call any function in the kernel, use a *tight binding* that is a direct call to the next layer entry function. By doing this, you will ensure that the next layer is processing the data at the same interrupt level as you are, allowing you to have a fine control on the overall system response.

Finally, because moving data around the kernel was one of the early requirements in HP-UX, some optimized mechanisms have been developed to address this, namely `mbufs`. An `mbuf` is a small fixed-size chunk of memory that exists permanently in the kernel. `mbufs` not in use are arranged in a double linked chain: the *free list*. When the kernel needs one, it removes it from the free list, and fills it with data. When the `mbuf` is not needed any more, it is put back onto the free list. The fact that the memory for an `mbuf` (244 bytes) is permanently reserved, and that allocation and release are just pointer arithmetic, results in an highly efficient system operation.

The small size of the `mbuf` usually suits inter-process communication, but networking layers have different requirements in the size of data exchanged. For instance four ATM cells can be stacked in a single `mbuf`, but seven `mbufs` are needed to transfer an Ethernet frame (1516 bytes). To overcome this problem, and offer a more scalable solution, the concept of a `mbuf` has been extended to `clusters`. A layer that needs to move much more than 244 bytes allocates one or more memory clusters (multiples of 4096 bytes) and puts the cluster base address in the `mbuf`. A field in the `mbuf` data structure allows the receiver to detect that the data is stored somewhere else in memory. Clusterized `mbufs` take longer to allocate and release, but this is still faster than reserving many `mbufs`. clusters, being page aligned, facilitate the use of *Direct Memory Access* (DMA) from an external peripherial and page remapping.

4 LAN Emulation Design

In the light of the previous section, we can now define the final architecture of the system. The LEC appears to be a critical part of the system because it sits between the IP and AAL5 layers. To avoid implementing both IP and TCP stacks in user space, the LEC will reside in the kernel. This also enables, a workstation with an ATM and Ethernet interface to be used as a bridge. In that case, fast processing and forwarding is necessary and only an implementation at a 'low' enough level can provide this.

The LES is mainly a database, and is solicited only when a LEC registers with a particular ELAN and for address resolution (MAC to ATM). These aspects make an implementation in user space perfectly suitable. The same applies for the LECS, which is only called when a LEC needs to resolve an ELAN name to the LES's ATM address for this emulated LAN.

The physical realization of the BUS function is best implemented within an ATM switch through the establishment of VCC's. The control software that interacts with the switch control module can be done in user-space.

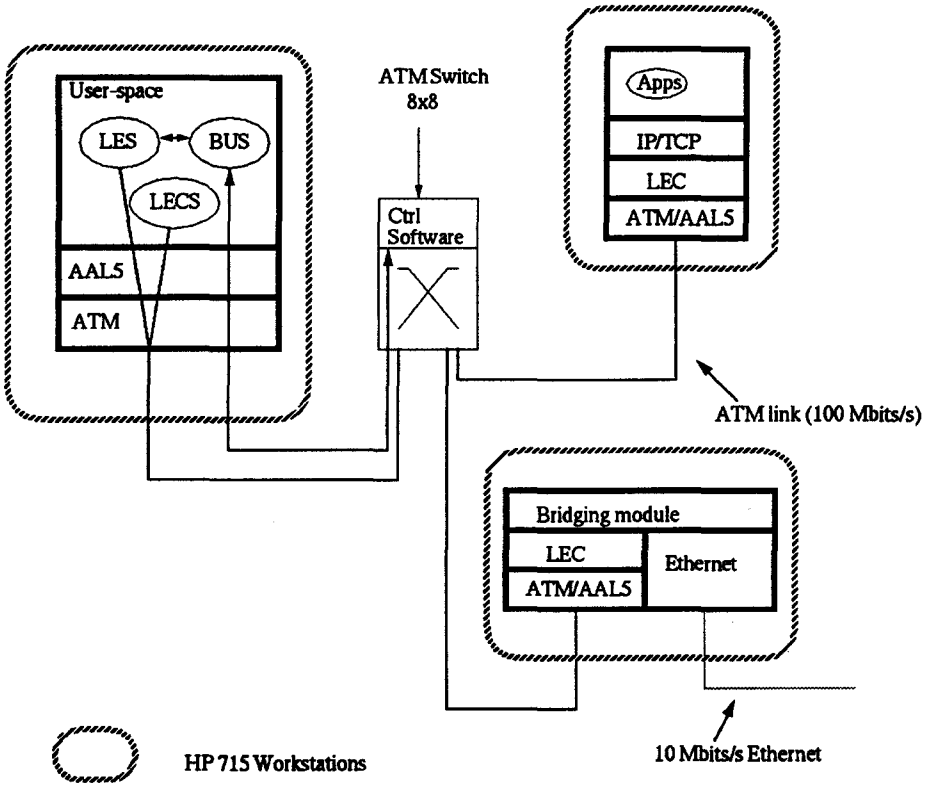

Figure 3: The system architecture

5 Implementation and performance

In this section, we would like to highlight some of the issues and problems encountered during this work.

5.1 The realisation

As shown in the previous section, the LAN Emulation Client is the more problematic part of the system and most of the work has been focused in this area. Moreover, the LAN Emulation Server is mandatory, in order to resolve MAC to ATM addresses. This is the reason why they represent most of the work done.

5.1.1 The LES

The LES has been implemented in user space because it is only occasionally solicitated. Here is a more detailed description of its operation.

A LEC that wishes to join a specific ELAN establishes a bidirectional VCC (Control Direct

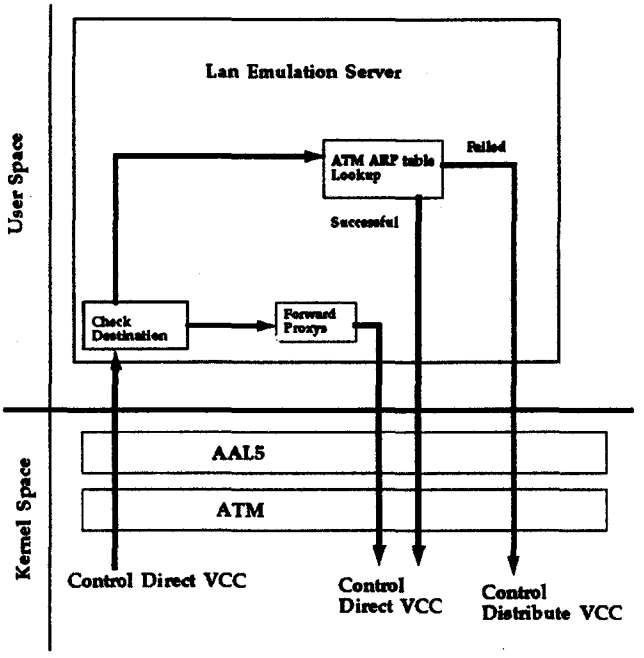

Figure 4: The LAN Emulation Server

VCC) with the corresponding LES. If the LES accepts the newcomer, it adds it to the point-to-multipoint VCC (Control Distribute VCC) of which the LES is the originator. This VCC is used to broadcast LE_ARP packets to all members of the ELAN (and not all the ATM end-systems of the network). When registering, a LEC provides a set of MAC addresses that it emulates and a corresponding ATM address. These are stored by the LES in an associative table.

When a LEC needs to resolve a MAC address to an ATM address, because it is not stored in its cache, it puts the Ethernet address in an LE_ARP request and sends it to the LES over its Control Direct VCC.

When the LES receives the packet:

- If the address is reachable via a bridge, it forwards the request to all LEC's registered as proxys. This comes from the fact that bridge topology may change over time. Thus the LES cannot be sure which bridge handles the traffic of the requested address. The selected proxy replies to the LES, thus providing its ATM address. The LES forwards the reply to every member of an ELAN over its Control Distribute VCC.
- If the address is local then the LES can resolve it according to its internal table. It replies directly to the requester by providing the associated ATM address.
- If the LES cannot resolve the request, it broadcasts it on its Control Distribute VCC.

The concerned LEC will reply back providing its ATM address. The LES in turn forwards the reply to every member over its Control Distribute VCC.

LECs can exploit LE_ARP responses they receive from the Control Distribute VCC even if they did not originated the request.

5.1.2 The LEC

LEC operation: First, concerning the LEC implementation, it has been found that writing a MAC layer that replaces the standard one is not a trivial task. Although the behavior of this new entity is well defined, in practice it is not that simple because not only IP uses the MAC services, but also ARP and the socket layer too. It has been necessary to investigate those in order to ensure a seamless integration.

Figure 5 represents the end-system block diagram. In the user space are applications using the networking services of the kernel via the socket layer. A small daemon in user-space controls the LEC in the kernel. The daemon is used for configuration, initialization and monitoring of the LEC. These actions are performed at the Link Level via a file descriptor representing the LEC device. Note that the LEC provides a Link Layer Access interface as specified by [7] and supports a range of actions common across the HP Link Layer modules.

Cells arriving on the ATM interface are re-assembled into AAL5 packets. Frames are then passed onto a **Lane_process_packet** function which is the core of the receiver. Based on the VCC the packet was received on, this function either:

- De-encapsulates the MAC frame and passes it to *LANC* if it is user data, or
- Updates the MAC to ATM ARP table if it was a LE_ARP response, or
- Passes it to the Control Layer if the packet indicates a configuration change.

We shall now describe the purpose of the LANC layer which is unique to HP-UX. This layer provides, to layers above, a homogeneous view of the lower layers and does some monitoring and statistical functionality that are used by network management applications. It routes received packets according to the protocol used. When LANC receives a packet, it looks at the protocol type field in the header and decides to either:

- Pass it to IP if it is a user-data packet, or
- Pass it to the traditional ARP module, if it is an ARP protocol, or packet
- Forward it to another interface, if it has been manually configured to do so.

If the packet is IP it passes it up to user-space following the "classical path", IP to TCP or UDP and then to a socket where it is delivered to the application.

The outgoing traffic usually proceeds through the UDP or the TCP module, depending on the type of socket opened, then through IP. Note that raw IP and raw Link Layer sockets are also supported. Then the packet reaches LANC which asks the *ARP module* (ARPCOMMON) to build the MAC header corresponding to the IP destination address. Two scenarios are possible:

1. An association between the requested IP destination address and a MAC address exists in the ARPCOMMON cache. In that case, the ARP module asks the MAC module to build the MAC header, providing the MAC destination address. Then ARPCOMMON returns the header to LANC which then appends it in front of the IP packet. LANC then schedules a `Hardware_request` for the MAC driver.
2. If there is no entry in the ARP table corresponding to the destination IP address, ARPCOMMON starts the classical resolution mechanism by sending an ARP request to every host on the network. When the address is resolved, a MAC header is built and the packet sent as explained above.

The same organization is used for resolving MAC addresses to ATM addresses. When the LEC receives a `Hardware_request`, it looks at the destination address.

1. If it is the all 1's then it encapsulates the packet with a LANE header and passes it to the BUS.
2. If not, it consults it's ATMARP table in the same manner as with classical ARP.
   (a) If an entry exists, this implies that there is a connection to the ATM destination address. The LEC sends the packet on this VCC.
   (b) If not, it forwards the MAC frame to the BUS, while sending to the LES an LE_ARP request. As soon as the LES provides a reply, the LEC establishes the VCC to the new destination and starts sending data over it.

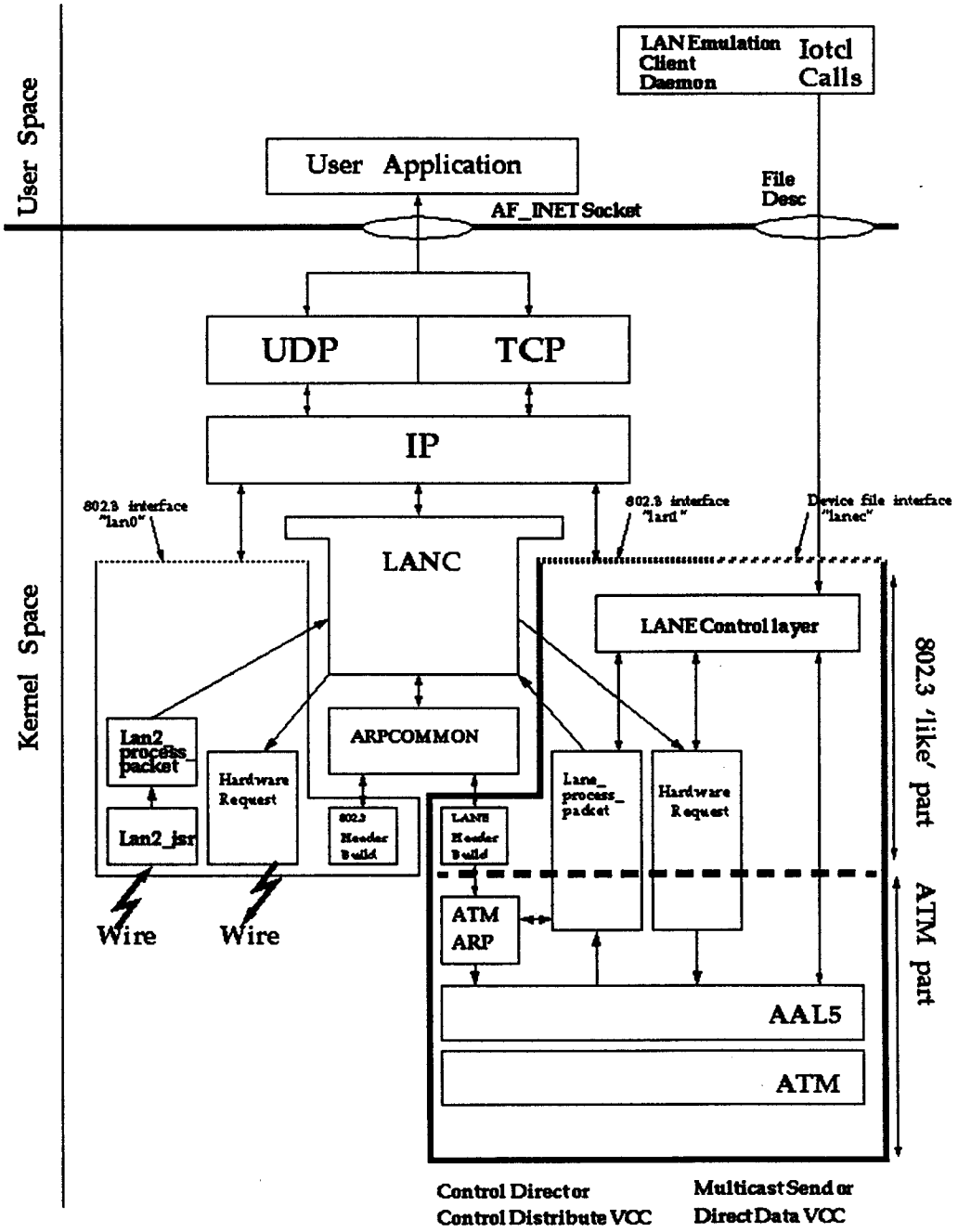

Figure 5: The LAN Emulation Server

Bridging: Because Ethernet frames are simply encapsulated on an emulated LAN, transferring data between the ATM interface and an Ethernet is not a problem and requires only a small amount of processing. In order to work properly in a bridged network, a spanning tree algorithm must be run on top of the two interfaces. Implementing this protocol is not a trivial task, although in a stable network with static configuration, a static look-up tables work fine (but maybe difficult to manage).

Performance Performance tests using Netperf [5] revealed a throughput of 45 Mbit/s. The main reason why we did not reach 100 Mbits/s is that the prototype ATM card does not have any DMA on board. Thus, central CPU must move data word-by-word between the ATM card and the central memory. Furthermore, in order to keep the cost of the card at a relatively low-level, all the AAL5 operations are done in software. This approach offers a great flexibility in the processing but slows the system a slightly.

5.2 Deviations

- A signalling environment compliant with the one presented by the ATM Forum. This could be a problem for inter-working with other similar realisations. Although great care has been taken defining the interfaces between ATM Connection Management and the different parts of the system in order to allow an easy replacement of the actual by a more standard one. Currently, the control of the switch to establish VCC's is done via a mechanism which does not fully conform to the one defined by the ATM Forum.
- The LECS is not mandatory for the overall system and as its operation is very simple this element has not been fully implemented. The LEC operation assumes an ATM network following the ATM Forum recommendations for signalling.
- The spanning tree algorithm. In order to have full bridging capacity, some specific software must be written. This task did not impact our results.

5.3 Future work

The following options offer possibilities for carring forward the work already done.

- Implementing support for Switched Virtual Circuits (SVC). This consists of fully implementing the UNI 3.1 specification from the ATM Forum.
- Implementing the LECS. Once SVCs are supported, having a configuration server makes sense, especially if the LAN emulation over ATM is deployed widely with many configurations supported.

- Implementing AAL5 in hardware and single memory copy. The very low cost ATM cards being used involve a lot of software processing. One possible evolution would be to enhance these with an AAL5 segmentation/re-assembly chip on the board. Furthermore, if a DMA is added, memory moves will be faster. Finally, if the ATM adapter card is memory mapped, then this enables single copies to be made, which increases the system throughput significantly.

6 Conclusion

During this work, we have implemented the major parts of the ATM forum recommandations (LAN emulation over ATM). Performance measurements showed us a big improvement when compared to the legacy LANs. Futhermore, ATM provides some inherent benefits; the connection oriented nature of the system provides a clear separation of the traffic sent between different end-system which gives more security in the data exchanged and reduces the influence that an end-system has on performance of others (dependent on the traffic it generates).

The scalability of the system does not seem to be problematic and relies mainly on the ability of the ATM switches to support concurrent point-to-multipoint connections and the limitations of the LES and BUS servers.

7 Bibliography

[1] *LAN Emulation over ATM Version 2*; ATM Forum Technical Committee, LAN Emulation Subworking Group. 5^{th} february 1996.

[2] *The Design and Implementation of the 4.3BSD UNIX Operating System*; S.J.Leffler, M.McKusick, M.Karels and J.Quaterman, Addison Wesley,1989.

[3] *The design of the Unix Operating System*; M.J.Bach, Prentice-Hall.

[4] *Experiences implementing a high-performance TCP in user-space*; Aled Edwards, Steve Muir. Proceedings SIGCOMM95.

[5] *Netperf: A Network Performance Benchmark*; Revision 1. Information Network Division, Hewlett-Packard Co., march 1993. The **netperf** package can be obtained via anonymous ftp from hpux.csc.liv.ac.uk in /hpux/Networking/Admin.

[6] *Device I/O*; HP 9000 Networking, Hewlett-Packard.

[7] *Link Layer Access Programmer's guide*; HP 9000 Networking, Hewlett-Packard.