

Ir. H.A.M. Luijff and Ir. P.L. Overbeek
TNO Physics and Electronics Laboratory (FEL-TNO)
P.O.Box 96864, 2509 JG The Hague, The Netherlands

ABSTRACT

The FEL-TNO Uniform Open Systems Model is based upon the ISO/OSI Basic Reference Model and integrates operating systems, (OSI) networks, equipment and media into one single uniform model. Usage of the model stimulates the development of operating system and network independent applications and puts efforts like POSIX, OSF, X/OPEN in a wider perspective. Examples of the usage of the model are presented in the paper.

INTRODUCTION

In this paper the FEL-TNO Uniform Open Systems model will be presented, a model that describes the way applications interact with operating systems, networks, equipment and media. The model described in this paper has earlier been presented at the ECODU-47 and VIM-50 conferences¹. The conceptual model is based upon extensions to the well known ISO/OSI basic reference model. It will be demonstrated that there is no basic difference between ISO/OSI protocol layers and operating system layers.

The strength of the new model is that now a single uniform model can be used to describe both applications running on a single system as well as applications running in a large distributed environment. The model gives means to evaluate market initiatives and directions like SAA, POSIX, X/OPEN, IPSE's, X-Windows as well as disk clusters, file servers, hierarchical data storage, etcetera.

Using the model, we have designed applications in an operating system and network independent way in our own multi-vendor network environment consisting of PC's, workstations, DEC VAX systems, CDC systems and a (mini)supercomputer on one hand and a mixture of operating systems as MS-DOS, XENIX, NOS/VE, VAX VMS and several implementations of the 'UNIX standard' on the other.

The usage of the model will be demonstrated. The examples will be the processes involved in a file copy in a VAX VMS CI-cluster environment and an application which uses several different computing elements in a multi-vendor network.

BACKGROUND AND HISTORY

The TNO Physics and Electronics Laboratory (FEL-TNO) is part of the Netherlands Organization for Applied Scientific Research TNO. The principal contract authorities of the FEL-TNO are the Netherlands Ministry of Defense and the Netherlands Armed Forces. The laboratory also does research for civil organizations and is research studies partner in ESPRIT and IEPG programs. Main research areas are: Operations Research (simulation, evaluation, analysis and decision support), System Development and Information Technology (C²I, computer graphics, image processing, transputers, AI, trainers, security), Radar and Communication, and Physics and Acoustics (laser, LIDAR, remote sensing, infra-red, sonar).

About 4 years ago, while working on studies on performance and on security of local area networks^{2,3}, both authors of this paper got the idea that it would be feasible to model operating systems in the same way as has been done with open networks using the ISO/OSI model.

As time went by, we found more and more evidence from market directions in the computer industry and from marketing slogans that many vendors were and are looking into the area of network wide applications and/or the area of portable applications.

It became more and more apparent to us that our, still mental, model fitted well and that the model might give a solid theoretical basis behind developments as SAA, POSIX, X-OPEN, Network Computing System (NCS) and so on.

Using the model, we have been successful in developing applications in our own multi-vendor network that otherwise could not have existed in such a user-friendly way.

THE ISO/OSI MODEL

It is assumed that you all know the layers of the ISO Open Systems Interconnection (OSI) Basic Reference Model⁴ as shown in figure 1.

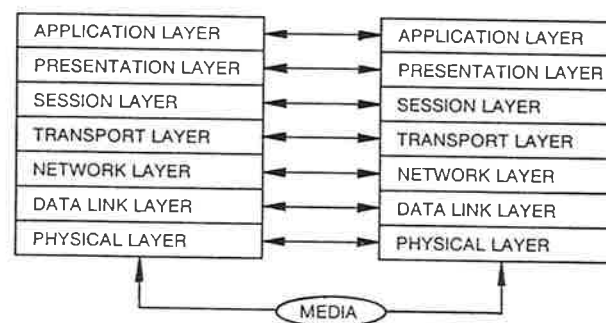


Figure 1: The ISO/OSI model

The completion of one or more standards for each layer leads to plug-and-play layers to interconnect open systems. The main purpose of the model is to allow easy and reliable communication between applications on distinct systems. The ISO/OSI model has proven to be successful in interconnecting applications and

systems with different architectures.

What is not included in the OSI model are the ideas and needs of the outside world. That what users call 'applications'. This is what the application layer is interfacing with.

In the rest of this paper we will refer to 'applications' when we want to refer to application interfaces provided by (open) systems as well as applications which can be seen as standard delivered operating system utilities. When we refer in this paper to 'user applications', we will refer to applications written by the user to interact with the outside world making use of both applications (-utilities) and application interfaces.

Starting at the top of the ISO/OSI model, the application layer allows an application developer to concentrate on his/her user application writing job. The application-service user can neglect everything from the lower layers. This top-down principle is well known from the information technology design theory. From that theory it is also known that the top-down approach only works with well defined interfaces to lower layers, which should provide all the services needed. Keep this in mind, we will come back to this later.

At the bottom end of the OSI model, an environment can be found which is part of the real-life world, in which one finds an infrastructure capable of 'storing', transporting and 'retrieving' data. Note that we here describe the functional aspects, rather than the narrow OSI functions of connecting one system to another. From a technical point of view the OSI model seems to be structured bottom-up: starting with the physical layer where the connecting transport medium can be found, signal descriptions and protocols capable of transporting data packets are added on top of that. The next layers include alternate path selection, error recovery and so on. All this deals with the real bits and bytes, rather than the high level abstraction of data and control information.

OPERATING SYSTEM MODELS

Hardly any global models concerning operating systems can be found in the literature. The only model we found is a model of a distributed system architecture using only four layers: application, distributed operating system services, OS-kernel and hardware/firmware⁵. Nearly every vendor uses his own structure in the development of operating systems. Until several years ago this worked well. However, when new features had to be brought into the operating systems, the operating system structures turned out to be less solid and straightforward than they should have been.

Examples that clearly demonstrate this are the way networking and security features have been 'layered onto' the 'old' operating systems.

It is well-known that many security breaches originate from this 'layering onto' operating systems not designed for security. The only proper way of including security into operating systems is to include the security features into the kernel of the operating system, controlling logical streams which cannot be overridden.

Another example is that most problems that users meet in using networking interfaces, stem from the structural and philosophical differences in the way, for instance, the file system is addressed, either locally or via the network.

During the last couple of years, users, application developers and system administrators in multi-vendor networks, feel the need for a uniform look at operating systems. This need emerges in the drives for common application

interfaces and common applications (e.g. CAIS, PCTE+, IEEE POSIX, X/OPEN, OSF), for the same touch and feel on systems of different vendors, operating systems and/or capabilities (IBM's Systems Application Architecture (SAA), X-windows, OSF/MOTIF, RPC, DECWindows to name some) as well as the need for standard networking.

This to create the transparent computing environment, the micro to supercomputer growth path, network fusion, DEC's Network Application Support (NAS) and IBM's standard application structure (SAA) and so on, for developers, programmers, administrators and, of course, the users.

At lower levels even standards for common binary interfaces are discussed (Application Binary Interfaces or ABI) which allow application porting between systems with the same type of processor.

The reasons behind these directions and developments are that software development as well as user training cost a lot. Creating portable software means that one can move the application from one piece of hardware to a more powerful one, without impact on either the application code or the user interface.

Modelling of operating systems can lead to a better understanding of the operating system's structure and helps operating system developers to avoid the problem of 'glueing' new features to the system in an unstructured way. It also helps to avoid new functions being added to the operating system which duplicate functions provided by other pieces of code. This well-known 'unstructured design method' still seems to be in use...

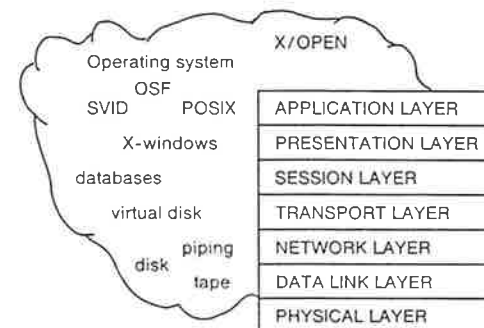


Figure 2: The OSI model and the Operating System cloud

THE FEL-TNO UNIFORM OPEN SYSTEMS MODEL.

Did you hear a call above for structured design? What about a top-down design in a layered manner? If so, which layers? Well, let us take a close look again at the well accepted ISO/OSI basic reference model⁴.

When we studied the ISO/OSI model, we found in the text of the standard that ISO/OSI explicitly precludes any involvement with operating systems: "OSI is concerned with the exchange of information between open systems (and not the internal functioning of each individual system)" as well as "Any other aspects of systems which are not related to interconnection are outside the scope of

OSI⁶. ISO/OSI seems to regard operating systems, disks and media as a cloud which interacts with OSI in a way not to be discussed (figure 2). Nevertheless, OSI defines some standard applications not only to be used for networking. Also, the OSI management services interact with the real (operating system) world.

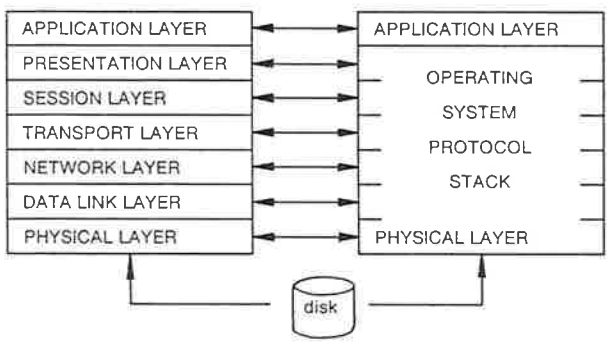


Figure 3: Shared disk in the Operating Systems model.

Of course, OSI concentrates on interconnection of (open) systems in the sense of datacommunication. But we asked ourselves the question what is the big difference between a coax cable connecting two so called open systems and, for instance, a disk shared by two tightly coupled systems exchanging data between two applications? Why are these systems not considered to be open? What about disk servers and disk cluster systems? What about a magnetic tape moved between two systems? And what about the parallel tasking on multiple CPU's synchronizing via a common bus or a semaphore memory? Our conclusion was that there does not exist a principal difference. Then the following step was taken: we modelled an operating system protocol stack in the same way as the OSI protocol/layer stack. In figure 3 it is shown how disk sharing between two applications takes place in the six operating system layers below the application layer, called the OS protocol stack for short.

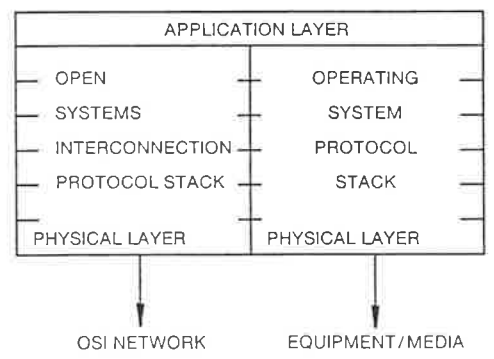


Figure 4: OSI and Operating System stacks.

In the OS-stack the application interfaces can be recognized at the top, and going down, layers providing for instance encryption, database services, disk striping, device drivers and intelligent controllers, media and equipment.

Now, another step will be taken. Why should a distinction be made between the operating system services delivered at the border between the application and the presentation layers and the network services delivered by ISO/OSI as shown in figure 4 ? As layers with the same kind of functionality in both stacks can be recognized, it is obvious that we joined both stacks into one new: the OSI-stack, where OSI can be seen as an acronym of Operating Systems and Integrated (figure 5).

We called this conceptual model the "FEL-TNO Uniform Open Systems model". Uniform because it integrates operating systems, open networks and all kinds of information containers (media) in one single model.

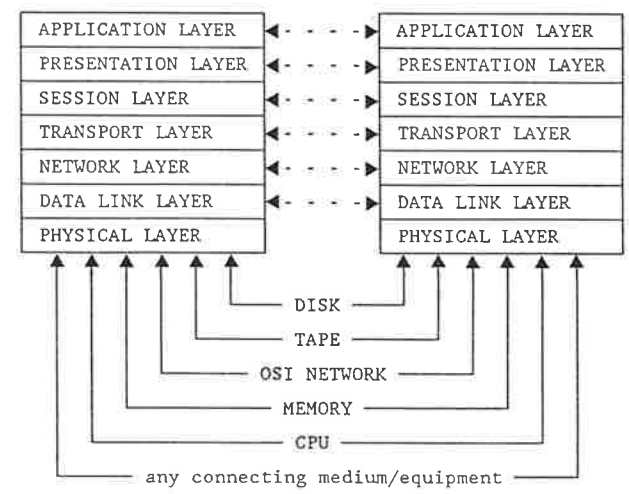


Figure 5: The FEL-TNO Uniform Open Systems model

WHAT IS THE BIG DEAL ABOUT THIS MODEL?

Well, nothing really new you say. It is so obvious that you understood it from the very beginning. And thinking a little bit longer about it, you will say: "I am using that model in my everyday business, although I was not aware of a model behind it!" Exactly, that is the big deal!

Let us return to something said earlier. Looking at the model from the top, users will use applications and protocols only for the benefits of making use of applications in a network of open systems. The same users will only be interested in the services provided by the standard applications in the application layer and the capabilities which can be delivered by the (open) system components in the network. We recognize here the top-down approach.

The previous conclusion points out the most important achievement of the model: application developers do not need to know anymore about the underlying layers. When this currently is not the case for most users, it will soon be. Just think about the upcoming popular (X-)window applications which hide the underlying ULTRIX, VAX VMS or whatever operating system from the user. The user will be able to concentrate solely on developing and using applications without the need of knowing all kinds of system commands.

If you agree with us that the user application interface is the most important part of the whole OSI model, do you understand why POSIX, X/OPEN and others neglected the ISO/OSI achievements? And why on the other hand ISO/OSI likes to preclude operating system interaction (read: operating system interfaces)? There is a great need of the outside world for an operating system and network independent set of user application interfaces. We hope that it will be clear to you that the FEL-TNO Uniform Open Systems model can help structuring these needs and developments.

THE DESCRIPTION OF THE LAYERS

For those who like to know a little bit more about the precise description of the layers and what kind of current defacto or international standards, products and program modules can be identified to belong to a layer, a description of these layers is given below. As said before, we regard the application layer as the most important one. Therefore, we start with the description of that layer.

The Application layer (layer 7)

In the ISO/OSI model, the application layer delivers services only to the applications that make use of networking facilities. In our model, this has been extended to include all kinds of information facilities. Only through this layer, operating system and networking services will be available. First, services are provided through a system call interface. There is no distinction between 'local' and 'network' in this interface as this is a lower layer task. Secondly, standard applications are offered as operating and network system commands, functions and utilities (e.g. FTAM, file access and management, E-mail). Once again, these applications work both in the local and in the network environment.

Examples of products and standards at this layer are the OSI standard network applications (MMS (MAP), VTP, JTM, FTAM, MHS). De-facto standards as FTP, NFS and SMTP, all belonging to the TCP/IP suite, can be thought of being placed here too.

Furthermore, some standardization efforts take place in providing common operating system applications and application interfaces:

- 1 the Unix System V Interface Definition (SVID) which delivers an operating system interface and hardly any networking. The IEEE POSIX 1003.1 standard is more or less a subset of SVID delivering an interface for graphics libraries, tools and data management to the Unix kernel. Networking layers interface directly to the Unix kernel and are not covered by POSIX yet. In the same arena, OSF and X/OPEN are active to standardize Unix operating system interfaces.
- 2 IBM's Systems Application Architecture (SAA). IBM has split SAA into several components:

- Common User Access or CUA: description of workstation functions (menu's, windows and function keys) and the uniform presentation of the SAA system to the user.
 - Applications Programming Interface or API: allows application developers to work in a system and network independent way. Remote file and processing access is supported. API is part of IBM's Common Programming Interface, which is not completely system type independent. CPI also provides database access and network protocol accesses.
 - SAA applications: distributed database, distributed file system, SAA office, SAA network management, decision support and SAA application design.
 - Common communications support or CCS: defines protocols which allow applications to communicate over IBM and non-IBM networks. Examples are e.g. SAA OSI, SAA Data Link Controls (SDLC) and SAA Network (X.25).
- 3 DECWindows, X-windows, OSF/MOTIF deliver only a uniform user presentation interface (libraries as XToolkit and DEC/X-libraries) to applications. These windowing views are becoming very popular as it relieves the application users from the need to know much about operating systems and networking details.
 - 4 Graphical and database interface standards and their specific language bindings (PHIGS(+), GKS, SQL).
 - 5 ISO's drive for a standard Operating System Control and Response Language (OSCRL)⁸. This standardization effort drive seems to have died in the violence of the X/OPEN, OSF and Unix International drives for operating system and interface standardization.

The Presentation layer (layer 6)

The main purpose of the presentation layer is to provide services to the application layer, which allow to present (transform) the application specific data structures into pieces of information being used by the system/network and vice-versa. At the same time, the presentation layer will provide security services (e.g. confidentiality)⁹. Examples are, for instance, the transformation of EBCDIC into ASCII, the support of man-machine interfaces (windows), library files and record types, page formatting (PostScript) and data format conversion (e.g. ODIF to PostScript). Lower parts of the X-Windows Xlib library can partly be found here as well as in the session layer.

The Session layer (layer 5)

The session layer provides services to control the start of processes (applications) as well as services to synchronize and guard the proper execution of requests by the presentation layer. Resources will be reserved (e.g. a tape unit, CPU, memory) or locked (record in a database) and connections with other services (file server) will be made. Shutting down at the session layer is more complicated as files have to be closed, accounting information needs to be collected and reserved resources have to be returned. It is even more complicated when one of the peer connections of the session layer breaks. Decisions about timing-out, sending terminating signals to the application and so on must be made. For the presentation layer, the session layer guarantees a synchronized, error free exchange of information with the equivalent peer presentation layer. Notice that sessions can be nested. The access to an external database is a session. This within a session to connect to another system within a session between login-logout.

The Transport layer (layer 4)

The transport layer provides the translation between the logical data structures of the higher layers and the medium oriented layers (1-3) below. The transport layer has knowledge of all hardware components and their properties. At the same time, the transport layer has knowledge of the logical data structures and the rules. Transport provides reliable and synchronized data to the session layer as well as security services as for instance access control⁹.

The transport layer relieves the session layer of sorrows about the underlying hardware. The higher layers do not even need to know which hardware components are available.

For instance with hierarchical file systems, it is irrelevant for the higher layers how many remote (served) disks, local disks or memory mapped disks are available and which one is currently being used.

Transport might be able to support disk striping, to stage file data from secondary to primary storage and to handle disk overflow situations.

For printer services, transport selects the first printer in the network that becomes free and has the appropriate forms and other property settings.

For serving a CPU schedule request, transport could schedule the most appropriate processing unit, which either might be a special database processor, a vector processor, parallel execution on multiple processors or an n-dimensional hypercube.

Example: A particular request for a file operation will be translated into an action on a resolved file path (a specific file in the file tree structure). Then the file distribution across the disks, which might be distributed across the network, is examined. For instance disk striping and disk pack overflow are handled here. As a result, the appropriate disk, which is either a solid state, optical or magnetical disk, or a piece of memory, in case of a virtual disk, is selected for the requested file access.

Standards at the transport layer include, for instance, the UNIX STREAMS Transport Level Interface (TLI).

The Network layer (layer 3)

The network layer relieves the transport and higher layers of the need to know the exact location of the devices and the routing to reach such a location. Access control (by peer) service and connection confidentiality takes place at this layer⁹.

This layer has, for instance, functions to translate virtual disk names into real disk addressing information. Network might find out that a file is located on a served disk. Network will then send a request to its peer network layer at the file server system, which in turn provides the requested service.

Another example is the virtual tape capability which some systems offer. After requesting a tape, the system completely copies the tape to the disk subsystem and unloads the tape. For writing, the process is reversed, the tape is written at the moment that the application sends the signal 'unload/return tape'.

A well-known example is virtual memory. The application thinks in terms of a linear memory, while the network layer maps the memory in a paged physical memory, which has a limited capacity. Thus the linear application memory can be found scattered across cache, real memory, virtual disk and physical disk.

The network layer provides the route of units of information through the system, network, equipment and media. The old name of this layer before the OSI model was standardized: 'routing layer', describes even more accurately the functions of this layer in our model.

The units of information are independent of both the device and the higher level

information structure. For example, the data units might either be a single character, a record or a part of a file being sent either to a magnetical, optical or solid state disk, a tape or a floppy. Only the route(s) to the device is(are) important to this layer.

The Data Link layer (layer 2)

The data-link layer solves all device dependent problems for the network and higher layers. In this layer, one finds all device drivers for networking, tape, disk, memory, printer and other equipment and media. The data link layer is responsible for a correct and error free/corrected read/write operation per information unit. This information unit is device dependent (frame, packet, sector, tape block, card). Sometimes these device drivers are completely encapsulated in hardware and firmware. The SCSI controllers, which both serve disk and tapes, are an example of this development.

Standards which appear at this layer are for instance the IEEE802.x standards and the system interface part of the Small Computer Systems Interface (SCSI).

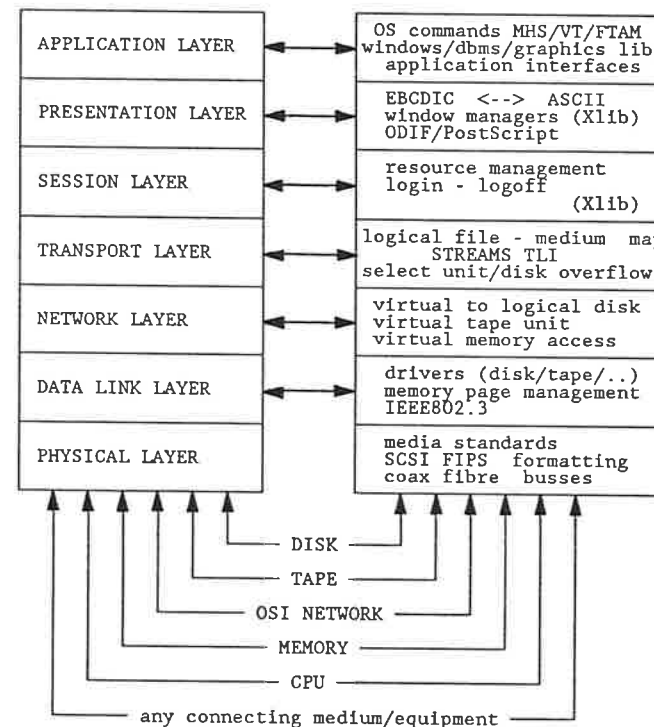


Figure 6: The layers of the FEL-TNO Uniform OS model

The Physical layer (layer 1)

The physical layer writes and reads information to/from a medium in an electrical, mechanical, magnetical, optical or other way. The physical components process an information (bit) stream at the top end. At the bottom end, the information is put in electrical, magnetical or signals otherwise detectable. Beside passive components (read-many, write-once), active components can be found which allow multiple reads and (re)writes. A CPU for instance can move data between input registers and output registers, while modifying the information stream.

At this layer we find both the media characteristics (e.g. disk size) and the physical characteristics (e.g. formatting, defective sector map, how to access the media). Internationally accepted standards at this layer are: FIPS-, SCSI- and other channel interfaces, coax and fiber-optic cables and media standards (sizes, formatting) for tape, floppy and disk.

All this information on the specific layers is brought together in figure 6. Most of the OSI protocols are not shown in the layers to keep the figure readable. The similar approach to devices in comparison with the OSI network will be clear.

EXAMPLES

Example 1: Remote disk in a VAXcluster environment.

The DEC VAXcluster features are a good example to demonstrate the model. A VAX system, called CLIENT can mount and access a local disk at another VAX system (named REMOTE) via a CI-cluster connection. Files located at the remotely mounted disk are accessed by VAX CLIENT in the same way as files on disks which are directly connected to CLIENT.

When a user enters the following command in his session on VAX CLIENT: COPY local-file CLIENT\$DUAN:remote-file , the following processes are started (see figure 7):

- 1 The COPY command verifies the existence of the local-file by asking the file status to the file system. This is done by trying to open the local file via a \$OPEN call. As a result of this \$OPEN call, the VAX VMS system determines at the transport layer (layer 4) that the file belongs to a locally connected disk. The file system tables are accessed to determine whether the file exists and whether the file permits (UIC's and ACL's) grant permission.
- 2 The COPY command tries to find out the status of the to-file by opening the to-file via a \$OPEN call. The transport layer determines that the to-file is located at a disk connected to a remote system (step 2a). Thus transport opens the peer-to-peer connection with the remote system (step 2b) using an ancillary control process.
Note that when the two systems were booted, the CI-cluster drivers were started, which established the 'mount' tables, reserved buffers and maintain checks for system health.
The transport peer-to-peer connection provides the information flow from the peer connection (on REMOTE) with respect to the mounted disk status and services the remote mounted disk accesses.
- 3 After checking the validity of the transfer, COPY starts reading of the local-file using QIO's. The local-file data might come from any local device: (optical) disk or tape.
In the same way, data streaming into the system via network connections

(e.g. DECNET or TCP/IP's FTP) can be thought of providing data, if we ignore for a moment the peer-to-peer processes which are also present for that data transfer. When these type of connections are modelled too into the same model, the CLIENT system receiving the data can be regarded as a relay/routing processor.

- 4 COPY copies the local-file data to the to-file using QIO's. The data file format may be converted in case of a COPY to a different file format.
- 5 The write QIO's result in data being provided to the CI-cluster. As the data is not intended to go to the cluster disks, but to the connected VAX REMOTE, the driver CNDRIVER is selected by the class driver. This CNDRIVER will co-operate with another CNDRIVER at system REMOTE. NETACP will control this process at the transport layer.
- 6 The data blocks which arrive at system REMOTE are transported to memory buffers.
- 7 As a memory buffers is filled, it will be emptied to the local disk at system REMOTE by the I/O control processes running at that system.

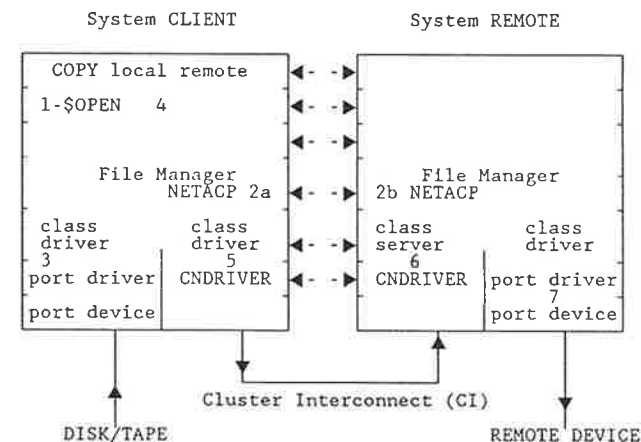


Figure 7: COPY via Cluster Interconnect (CI)

It will be clear that there doesn't exist a principal difference on VAX systems between local and remote files. Only some relay/router hops have to be taken. In the same way as with the OSI layers, it is possible to exchange the lower layers (driver and equipment) with other protocols and media as for instance DECnet in the form of Local Area VAXcluster (LAVC), Remote Management Services (RMS) or FDDI in the future. This allows loosely coupled VAX systems to act as file server.

Talking about the transport layers for remote file access, the transport layer at the controlling system might negotiate in the peer-to-peer communication that, for a COPY command with both the 'from' and 'to'-files located at disks at the remote system, the actual copy completely takes place at the server system under (transport) control of the controlling system without using the network or CI-busses for transferring data.

Example 2: A data processing application

At the TNO Physics and Electronics Laboratory, PC's and minicomputers are used for controlling experiments as well as for collecting data from these experiments. In many cases, the amount of processing involved is too much for this type of computers.

In one case, the following application had to be developed:

- collect measurement data at a PC (or minicomputer)
- move the data to the mainframe
- save the data on tape
- add the data to a data library
- extract information, including the new data, from the data library
- make an analysis of the selected data on a supercomputer (numeric intensive work) and produce an file with graphical information
- postprocess the resulting graphical data on an image processing station connected to a VAX.

A block diagram of these steps is shown in figure 8.



Figure 8: A data processing application.

In a traditional design, one would start looking for a way to transfer data from the PC to the mainframe. Then the user would have to login, either from his PC file-transfer session or in a separate session and start a job to save the data on tape and to add the data to the data library. The next step would be to transfer the data to the numeric intensive computer (NIC), in our case a (mini)-supercomputer. At that system, the user will start the computation manually. Finally, the resulting graphical information will be moved to the VAX for the final graphical postprocessing.

With our new conceptual model in mind, we would not start the design bottom-up as presented above. We start to look at the application, which is: produce results from the collected data. The next step is to identify the hops or connections between the subapplications. These steps, or systems to be passed, will be regarded as systems providing the function of a router or gateway: incoming data is transmitted, after processing, to another system or application. Only then, we will start thinking about how to fill in these functions at these routing systems using standard communication software means. It is obvious that the design should strive for the least possible user interaction, that is none, at the intermediate steps. It turned out that a very valuable feature of operating systems is the possibility to do epilog or 'catch logout' (in UNIX terms: trap signal 0) processing. During epilog processing one is able to detect that a new file became available and one might react upon that in an appropriate manner. Thus it will be easy to do the simple tasks described above automatically.

Even an automatic display of the resulting data can be organized by activating a daemon on a UNIX or ULTRIX workstation which starts graphical processing as soon as a certain file has been received via for instance DECnet, NFS or FTP from another system in the network.

FUTURE DEVELOPMENTS.

Although the model currently is only a conceptual model, we have demonstrated to you that it is very valuable when one wants to design a network wide application in a multi-vendor environment.

We will refine our model when time allows us to do so. Especially, the FEL-TNO security risk analysis model for systems in a local area network will be extended^{2,3}. Security risk analysis of a local or even a wide area network, where both network and operating systems threats and risks are addressed by using one single model, looks very promising.

CONCLUSION.

At first glance, the steps we took to explain the model to you seemed to be very obvious. But when you realize what the potentials are of this FEL-TNO Uniform Open Systems model, we are almost certain that you cannot wait to start developing your own new network-wide applications.

We do not expect that all vendors will adopt our ideas tomorrow. Nevertheless, we hope that we have identified the standardization efforts taking place at the hardware side, the bottom end of our model. At the top end, the developments by OSF, X/OPEN, POSIX and Unix International might be much more valuable to us all in case standardization would take place in such a way that there is no principal difference between networking and other (connecting) media.

And last but not least, we hope to have given you a valuable background for understanding and evaluating the forthcoming changes in technology which lead to real transparent network-wide user applications running on several processors.

REFERENCES.

- 1 The FEL-TNO Uniform Open Systems model
Ir. P.L. Overbeek and Ir. H.A.M. Luijff
Proceedings of the ECODU-47/VIM-50 conferences
Wolfsburg, Germany (4/89) and Orlando, USA (5/89), pp 1.112-134
- 2 Beveiligings- en capaciteitsaspecten van Local Area Networks
FEL-TNO Internal Report 1986-17. (in Dutch)
Author: P.L.Overbeek, 1986
- 3 Security Aspects of Local Area Networks
Ir. P.L. Overbeek
Proceedings of the ECODU-42 conference
European Control Data Users, Aachen, 9/1986, pp 1.26-36.
- 4 Open Systems Interconnection, Basic Reference Model
International Standard ISO 7498.
- 5 Distributed Systems - Architecture and Implementation
R.W.Watson, Univ. of California Lawrence Livermore Laboratory
Chapter 2 in: Lecture notes in Computer Science 105, edited by G.Goos and J.Hartmans
Springer Verlag, Berlin, 1981.

- 6 Open Systems Interconnection, Basic Reference Model
International Standard ISO 7498, paragraph 4.2.
- 7 Open Systems Interconnection, Application Layer Structure
International Standard ISO 9545.1.
- 8 Draft Operating System Command and Response Language (OSCR),
ISO/TC97/SC21/N965
- 9 Open Systems Interconnection, Security Architecture
International Standard ISO 7498/2.

Proceedings

1989 DECUS Europe Symposium

The Hague, Holland

September 18th to 22nd, 1989

'Towards an Open World'
