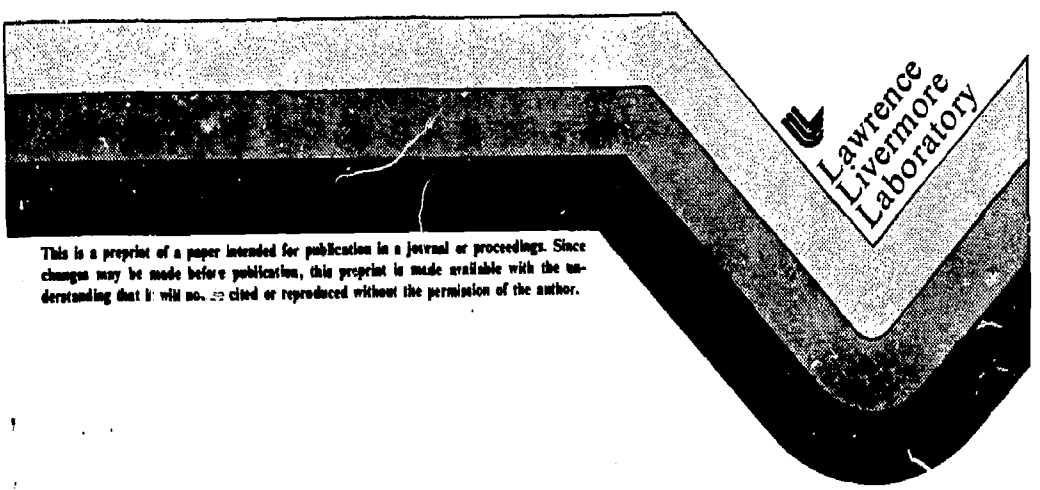
UCRL- 88359
PREPRINT

Conf-821201--6

LINCS: LIVERMORE'S NETWORK ARCHITECTURE

John G. Fletcher

This paper was prepared for submission to the
1982 Fall DECUS U.S. Symposium, Anaheim,
California, December 6-10, 1982



LINCS: LIVERMORE'S NETWORK ARCHITECTURE

John G. Fletcher
Lawrence Livermore National Laboratory
Livermore, California

ABSTRACT

Octopus, a local computing network that has been evolving at the Lawrence Livermore National Laboratory for over fifteen years, is currently undergoing a major revision. The primary purpose of the revision is to consolidate and redefine the variety of conventions and formats, which have grown up over the years, into a single standard family of protocols, the Livermore Interactive Network Communication Standard (LINCS). This standard treats the entire network as a single distributed operating system such that access to a computing resource is obtained in a single way, whether that resource is local (on the same computer as the accessing process) or remote (on another computer). LINCS encom- passes not only communication but also such issues as the relationship of customer to server processes and the structure, naming, and protection of resources. The discussion includes: an overview of the Livermore user community and computing hardware, the functions and structure of each of the seven layers of LINCS protocol, the reasons why we have designed our own protocols and why we are dissatisfied by the directions that current protocol standards are taking.

UCRL--88359

DE83 004141

OCTOPUS

The Lawrence Livermore National Laboratory (LLNL) is operated by the University of California under contract from the Department of Energy. It is engaged in research and development, chiefly relating to nuclear explosives and energy resources. The physicists, chemists, engineers, and other professionals at the Laboratory make extensive use of computers, primarily to simulate physical processes. Such simulations greatly reduce (but do not eliminate) the need for actual experimentation and testing, with consequent reductions in cost and hazard. Many of the simulations stretch the limits of the largest computing machines that are currently available, and still more complex simulations await the development of even more powerful machines.

Most of LLNL's computing facilities are organized into a single local-area network called Octopus, which currently includes the following:

- computers: four Cray-1, three CDC 7600, numerous DEC PDP-10, VAX, PDP-11, LSI-11, SEL 32, TI 980A, Modcomp II, etc.;
- storage: Braegen Automated Tape Library (2 trillion bits), CDC 38500 (1 trillion bits), and numerous disks;
- i/o: four FR-80 microfilm recorders, two Honeywell nonimpact printers (each 18000 lines/min), numerous smaller printers, card readers, and tapes, over 1000 television monitors, and over 1600 interactive keyboard terminals (both hard and soft copy);
- communication: a Network Systems Corporation (NSC) Hyperchannel (40 megabits/sec) and numerous serial channels (up to 224 kilobits/sec).

All this has evolved from a beginning in the mid-1960's with one CDC 6600 and twelve Model 33 Teletypes. As Octopus has expanded and incorporated new technology over the years, every effort has been made to maintain a consistent and orderly design, yet it is probably inevitable that the end result of such extensive growth involves certain nonuniformities and unnecessary complexities.

Therefore, we are now engaged in a major effort to review the architecture of our network, identify ways in which various activities can be carried out in a more uniform and consistent manner, and appropriately alter our design. The result of this effort is LINCS -- the Livermore Interactive Network Communication Standard -- a hierarchy of communication protocols and other conventions for

*This work was performed under the auspices of the U. S. Department of Energy by Lawrence Livermore National Laboratory under Contract No. W-7405-ENG-48.

our network. We have named the various layers of LINCS protocols in conformity with the OSI Model (1), and their functionality follows the outline in section 6.2 of that document. However, we do not always follow the more detailed specifications in later sections of the document, which in our view do not entirely adhere to the principles of section 6.2.

ALTERNATIVE PROTOCOLS

It might be wondered why we have chosen to design our own protocols rather than adopt standard protocols or the protocols of some vendor. The reason is that we find unacceptable deficiencies in those protocols, such as the following:

- The protocol is not yet defined. This is the case for most standard protocols.
- The protocol is unimplemented. This is even more the case for standard protocols.
- The protocol is subject to unpredictable change. This is the case for vendors' protocols and is a source of particular difficulty when more than one vendor is represented within a network.
- The protocol provides limited facilities. For example, many protocols provide only for terminal interaction and file movement, rather than for the general situation indicated by the next point.
- The protocol does not provide for general interprocess communication.
- The protocol treats local (within the same computer) vs. remote (among computers) communication and access differently. This is in part a shortcoming of vendors' operating systems, but it is also a shortcoming of protocols, which often attach too much significance to the system or host on which a process resides. The OSI Model in fact is named as applying to the interconnection of "open" systems, not of processes, while LINCS treats a network as a single distributed operating system.
- The protocol is poorly modularized, often forcing the implementation of complicated features when only simple ones are needed. Proposed so-called virtual terminal protocols are particular offenders in this regard.
- The protocol is gratuitously complex, that is, has complexity well in excess of what is needed to do the job. X.25 (levels 2 and 3) is only one of many examples.
- The protocol involves extensive "handshakes" at many levels, often requiring the exchange of a dozen or more control packets to effect the transfer of a single packet of data.
- The protocol is inefficient in time and space. We have compared LINCS implementations with implementations of alternatives, and factors of ten in favor of LINCS are not uncommon.
- The protocol is incomplete; that is, it provides so many options that it really does not define a uniform standard. This is a typical result of the design-by-committee political process within standards' organizations.
- The protocol embodies technologically obsolescent concepts. One such obsolescent concept is that interprocess communication should be modeled after process-to-terminal communication.
- The protocol has inadequate security features.

Of course, whenever a system not adhering to LINCS is attached to Octopus, a gateway module must be provided to translate between the differing protocols involved. In particular, we will provide gateways between LINCS and interantional standard protocols, as the latter are defined. A gateway usually cannot compensate for deficiencies in a protocol; so the services offered through a gateway may be limited.

LINCS HIERARCHY

We now review the LINCS protocols, from the physical (first and lowest) to the application (seventh and highest) layers.

Physical Layer

LINCS views the physical layer as encompassing whatever is within the communication channels that interconnect the nodes (computers) that make up the network. This is the only layer for which we adhere entirely to vendors' and/or standard protocols; that is, the physical protocol is defined by the channel hardware. An important notion in regard to the physical layer is recursive encapsulation: LINCS regards a channel as embodying just the physical layer, even though examination of its inner workings might reveal multi-layer complexity, perhaps even an entire network.

Link Layer

The main purpose of the link layer is to provide assurance and flow control (AFC) across a channel. By assurance is meant that data is delivered in proper sequence and without omission, alteration, or duplication. Clearly the link protocol depends on the nature of the underlying physical protocol and will be more or less complex depending on the inherent quality and reliability of the physical layer; so the appropriate link protocol varies from channel to channel.

Although we will support X.25, level 2, on serial channels connecting central Octopus to users' minicomputers running vendors' systems, we prefer a protocol of our own design called ALP -- A Link Protocol.[2] ALP is simple and efficient, yet effective. The entire protocol is embodied in about 50 lines of Algol; contrast this with X.25, level 2, which is defined in 16 pages of documen-

tation, yet never exhibits an algorithm. ALP is also used with the NSC Hyperchannel, which (like other multipoint communication media such as Ethernet) is a channel, not (as is sometimes said) a network.

Network Layer

The primary purpose of the network layer is to route packets from node to node in zero or more hops from origin to destination. As with higher layers, there are great benefits in an entire network using a single network protocol. The LINCS protocol is called DeltaGram. It is a best effort datagram protocol that routes each packet independently; that is, it does not set up virtual circuits, which generally entail a higher overhead. Addressing at this layer is to the port on a process, not just to the node or "host". This modularizes the entire routing and addressing activity into a single layer, eliminating the need to implement such activity repeatedly in layer after layer. It also tends to eliminate inappropriate preoccupation with which node a process happens to be in. However, addresses do have a hierarchical structure that reflects, at least in part, the network topology. This is necessary in order to keep network routing tables and algorithms of manageable size.

The origin generates data packets of some size. If a packet is too large for some part of the network through which it passes, then a network module will fragment the packet into smaller packets. To make this easier, a packet is assigned a sequence number corresponding to the ordinal position in the data stream of its first bit; so an appropriate numbering for the packet fragments is obvious. This same sequence number is also used by the transport (next higher) layer. This sharing of the sequence number field (and other fields) in the packet heading provides efficiency. Some may feel that such sharing violates the principles of proper layering, but in fact it does not; it is only a question of what information is passed across interlayer boundaries. Fragmented packets need not be put back together by the network layer.

Transport Layer

Above the network layer, one has the abstraction that ports on processes communicate with streams of bits, all knowledge of the structuring into nodes connected by channels being concealed. Each of the next three layers (transport, session, and presentation) logically consists of one independently executing module per end of association, where an association is a pair of communicating ports.

The transport layer provides AFC across the association from end to end (as opposed to the link layer, which provides AFC over each channel). The LINCS transport protocol, Delta-t, is timer- based; that is, it uses timers to decide when it can discard the connection records for an association.[3] This avoids the overhead of opening and closing "handshake" packets.

Session Layer

The session layer is quite trivial. Its chief function is to delimit the data stream in each direction on an association into logical units: messages and monologs. To do this it uses special marker flags in packet headings. A message expresses a complete thought, at the end of which the sender intends that the receiver should "wake up" and take action. A monolog is a sequence of related messages; at the end of a monolog the communicating processes can discard context or state relating to it.

Presentation Layer

The presentation layer defines the format of the data contained in messages. LINCS recognizes three formats:

- The "raw" data protocol defines just a stream of bits, not to be interpreted by the recipient. For example, a file server receiving data to be written into a file regards it as just a stream of bits; records and other structures within a file are, at least for the present, regarded by LINCS as an application layer issue.

- The virtual terminal protocol defines a stream of 8-bit characters adhering to rules based on the ANSI documents X3.4 (ASCII), X3.41 (code extension), and X3.64 (additional controls). This protocol is used between those processes that directly control interactive terminals and those processes, called command interpreters, that interface between the human-oriented virtual terminal protocol and the computer-oriented protocol discussed next.

- The control protocol is used for most interprocess communication of requests for service and replies about service performed. In contrast with the virtual terminal protocol, information is expressed in forms likely to be suitable for computer processing; for example, integers are expressed as binary bits, not as decimal ASCII digits. We see no reason to burden computer processes with translating to and from human-oriented forms of expression when communicating among themselves.

Application Layer

The application layer, unlike the lower layers, is concerned with more than just communication. It defines standard views of several issues that are of concern to many processes, including the following:

- It defines the relationship between servers (processes that provide services to other processes) and customers (processes seeking services), including such matters as whether the server performs requests serially or in parallel, when replies are generated, and how the customer can cause replies to be directed to a third party.

- It defines a standard model for resources, such as files, directories, i/o devices, clocks, accounts, processes, etc., including in particular a model of heading or state information.

- It defines standard functions to operate on resources. For example, "create", "write", "read", and "destroy" are functions of wide applicability.

- It defines a universal means for effecting the movement of bulk data (files to be stored, output to be printed, etc.) among processes.

- It defines a standard scheme for naming and proving right of access to resources. The scheme makes use of capabilities, where a capability is a coded record, generated by the server and given to the customer when a resource is created and then returned to the server whenever the resource is accessed. A capability is a computer-oriented name for a resource. Human-oriented names are provided using resources called directories. A directory is a list of entries, each entry associating a mnemonic character string with a capability. A human user generally names a resource with a chain name, a sequence of mnemonics which, starting with a given root directory, leads from one directory to another until the capability to the desired resource is reached.

- It defines standard basic services, such as file service, directory service, output service, etc.

The consequence is that a user of LINCS can read just one basic manual and then know how to do a wide variety of common computing activities, such as accessing files, without having to relearn them for each computer used.(4)

Multilayer Issues

The preceding discussion has by no means covered all issues. In particular there are certain matters that of necessity have to be dealt with within several layers. One such issue is security and protection. In an environment such as Octopus, where information of varying sensitivity exists, it is necessary to label most computational objects -- resources, processes, packets, channels, nodes, capabilities, and (human) users -- as to their levels of sensitivity, authority, and/or protection offered. Then certain relationships among these levels must be enforced. For example, the network or link layers must see to it that packets are not transmitted over channels that do not offer the appropriate level of protection, while it is servers' application layers that must prevent access to resources more sensitive than the authority of the customer. The network layer should also verify that all packets are properly identified with their origin addresses.

CONCLUSION

We are just now entering into the full swing of the conversion to LINCS; so it is not yet possible to comment on our success. It undoubtedly will be less than we might hope, but it seems certain that the greater uniformity of LINCS will bring about considerable improvements in the ease with which new facilities can be added to the network and the ease with which users will be able to learn about the system. We are also quite sure that the generality of LINCS has not been bought at the price of significant degradation in performance. In fact, the primary cause of delay in completing our design has arisen from the need to take care that all aspects of it admit of efficient implementation. Among the computers for which we are currently implementing LINCS are DEC VAX and LSI-11.

REFERENCES

(1) Computer Networks 5, 2, pp. 81-118 (April, 1981).

(2) J. G. Fletcher, "Serial Link Protocol Design: A Critique of the X.25 Standard, Level 2", UCRL-83604, Lawrence Livermore National Laboratory (1979).

(3) J. G. Fletcher and R. W. Watson, "Mechanisms for a Reliable Timer-Based Protocol", Computer Networks 2, pp. 271-290 (1978).

(4) LINCS manuals, available from the author.

ACKNOWLEDGMENTS

LINCS is the product of the combined efforts of many persons in the Computation Department of LLNL.

DISCLAIMER