M.I.T. Laboratory for Computer Science
Computer Systems Research Division

September 22, 1976
Request for Comments No. 124

Revised CSR Proposal to ARPA/ONR

from David D. Clark

The original version of the continuation proposal to ARPA for calendar 1977, which was distributed as RFC 117, has been extensively revised and expanded as a result of discussion with ARPA, sufficiently so that it seems desirable to distribute the revised version. I would suggest that those in the group who are interested in our new research direction read this version, since it contains much more concrete statements describing what we expect to do.

6. DISTRIBUTED COMPUTER SYSTEMS

The Computer Systems Research Division of the M.I.T. Laboratory for Computer Science proposes to continue the research, started as a study last year, in the area of distributed computer systems.

6.1 Introduction

The term "distributed computing" is appearing with increasing frequency in current technical material, often referring to such common computer organizations and operating system strategies as multiprocessing, array processing, time-sharing with remote terminals, and so forth. This broad usage is generating considerable confusion as to what distributed computing really is. To some, it seems the wave of the future, capable of solving all current problems in large scale computing and information processing. To others the term distributed computing is dismissed as a new catchword for old techniques that do not work well. While we do not believe that distributed computing is a panacea, we do believe that there are significant and fundamental functional advantages to be derived in certain applications from the use of distributed computing, advantages whose potential can be exploited through a well organized program of research. It is this line of research that this proposal addresses.

6.2 Justification for Research

The following are the most important functional advantages of distributed computing that we have so far identified:

I) Reduced vulnerability to attack, sabotage, and failure.

There are many circumstances in which a single centralized computer facility can not be sufficiently protected or reliable to provide required levels of availability. In such a circumstance, a distributed system with some degree of physical separation appears to provide an alternative architecture that can be highly robust in the face of failure of individual computing nodes. Not only does distribution allow continued service in the face of isolated failures, but it also prevents the compromise of one single site from revealing the totality of the information stored in the system. This latter point may make a distributed information system much more acceptable for storing certain civilian information as well as military information.

Another kind of failure common in shared, centralized systems is the administrative communication failure, in which the computer system is intentionally shut down at a time when some individual needs its service. First hand experience with such administrative trouble is probably one of the primary forces leading to decentralization decisions.

II) Ability to grow.

A principal potential of distributed systems is that they can be capable of growing in the same unbounded way that telephone networks can and do grow. The interesting aspect to this growth is not so much increasing computational capacity, but increasing information capacity and the accessibility of that information. A properly designed distributed system should be capable of growing with a cost that is linearly related to the size of the computation performed, and with no increase in the complexity of the organizational algorithms. In contrast, while a centralized realization of a computation can often be moved to a larger machine, the cost in complexity as well as dollars is often increased by a factor much larger than the absolute increase in the size of the facility.

III) Rapid response through parallel computation.

Parallelism is not the same thing as distributed computing, and we are not proposing to study the general topic of parallel structuring of algorithms; much work has already been done by other workers in this area. However, it must be noted that parallel computation may provide a fundamental solution to certain requirements of speed, an especially interesting point to us when the limitation on speed is not raw calculation, which can be tackled with a more complex central processor, but rather a limitation on the ability to access and manipulate data.

IV) Increased function for information utilization.

By increased function, we do not mean the ability to perform new sorts of computations not performable on a centralized site. We feel that any computation performable on a distributed system is in principle capable of being performed on a centralized machine. Rather, what we mean here is increased ability of the user to access the information on which his computations are to be performed. Information stored in a single central site may be difficult and unwieldy to use if the eventual producers and consumers of this information are geographically distributed. Conversely, information maintained on separate, isolated machines is very difficult to use as an aggregate. We anticipate an increasing need for the fusion and integration of related information which is generated from many individual and geographically distributed sources. While this point is somewhat more subjective than the previous, we believe that this increased ability to make use of information is, by itself, a compelling reason for the study of distributed systems.

V) Economic Advantage.

While the economic arguments in favor of distributed systems are not functional ones, as the previous arguments have been, they may be quite compelling in the long run. We observe in practice that the cost of the computing hardware grows directly as the product of its processing capacity P in MIPS (million instructions per second) and its storage capacity M in MBYTES (million bytes). Accordingly, assuming low communication costs, a large system consisting of N processors with N memory units, each of capacity P and M is considerably cheaper than a single unit with the same overall capacity (NP)*(NM), since N*(PM) < (NP)*(NM).

The preceding items seem to us to represent a clear enough justification of the advantages of distributed computing that research in this area is justified in order to realize these benefits. Even in the absence of these positive benefits, there is a very compelling reason for investing the effort now to understand the implications of distributed computing, following from a reverse application of the previous argument of economics. The apparent economic advantage of dedicating a small computer to each small task is causing organizations to decide at a low administrative level to purchase local, dedicated computers, rather than participating with other organizations in using a shared computer. Since each application is intended to run in isolation, the software cost is not influenced by this decision to use a separate machine, and local organizations achieve a significant reduction in hardware cost. By itself, each such decision seems quite reasonable, but insufficient attention is paid to the inevitable future need for these systems to communicate with other locally justified small computers. While for any single special case, it is possible to string communication lines and invent protocols to connect the machines together, a need is certain to develop for a greater degree of coherence across such operations. It is likely that "patching on" of coherence across systems that are not prepared for it will be expensive and impractical. Many computer users are just now involved in unsuccessfully trying to "patch on" security in systems that were not prepared for it -- the issues seem quite parallel. For this reason, it is appropriate to study how the machines and their information should be structured so that eventual unification as part of a distributed system is feasible.

6.3 <u>Relevance to ARPA and the Department of Defense</u>

The five advantages of distributed systems presented in the previous section are all directly applicable to problems facing the DoD now or in the near future. Looking at the broad spectrum of DoD computing, we see these advantages as being relevant in a variety of ways. For example, if the DoD is to have an advanced command and control system on which users can depend in a crisis situation, several problems must be solved which may only be approachable by means of the sort of distributed system we envision. In order to provide the rapid response needed in a crisis situation, computation by multiple processors may be required. To obtain the level of availability which will allow end users to depend on the system, the reliability obtainable through distribution will almost certainly be required. To enable the system to adapt to changing demands, the growth potential of a distributed system will be needed.

Less demanding computing applications of the DoD will not require the increased functionality of distributed systems, but even in these sorts of applications, it is clear that the advantages of administrative convenience and autonomy and lower hardware costs obtainable from decentralization are already a force throughout the Defense establishment, and decentralization can be detected in some part of almost every new procurement of computer-based systems. The need for coherence among these independently acquired systems will soon be one of the dominant problems of computer use facing the Defense Department. We believe that it is crucial for ARPA to fund research that will enable it to contribute to the solution to this problem of coherence in the right time frame.

While a policy decision could be made to discourage this trend toward decentralization in those cases which do not absolutely demand it, we feel that this would be inappropriate. The benefits of distributed systems are widely applicable, since advantages such as increased availability are appreciated in almost every application, and contribute materially to the eventual acceptance of a system by its end users. The ultimate failure of users to accept a system is a serious problem now being faced by creators of new computer applications. The potential of distributed systems to help solve this problem is a strong benefit which ARPA can derive from this proposed research.

The previous paragraphs indicate why we believe that research in distributed systems is justified within the DoD in order to enable new applications and avoid chaos. The next section will discuss how we propose to carry out this research in order to achieve these goals.

6.4 Strategy for Research

At a high level, our research goals can be very simply stated: we wish to discover and develop a structure for a system of interconnected computers such that the positive goals outlined in the previous section are realized. We emphasize our goal at this high level because, in our opinion, much of the current research in distributed computing has, even at this high level, missed the point, in that it has not addressed itself to realizing any of the fundamentally new capabilities that will depend on distributed computing. Superficially, justifications of higher reliability and increased speed (of parallel operation) are often used, but these justifications usually turn out, on more careful analysis, to be based on the economic advantages of modern mini- and micro-computers, rather than on any intrinsic limitations of the functional capability of traditional system organizations. More strongly, current research has missed the point by failing to build systems whose intended applications are those that will benefit from the advantages listed above. For example, a distributed computing system designed to deliver as its end product a fairly traditional time-shared programming environment does not provide a vehicle to demonstrate the potential for unlimited growth. Such a system could provide increased reliability, but only as increased availability of the computing resource, not as increased reliability of the distributed information stored in the system.

Thus we assert that legitimate research in distributed system can only be carried out in the context of particular sorts of applications that can benefit most strongly from the structural advantages of a distributed realization. The two most promising examples we have considered are a distributed message passing system and a distributed data base management system.

The technical problem that seems most central in dealing with either of the above kinds of applications is the naming, accessing, and protection of the objects with which the distributed system deals. We expect that this topic will represent the major focus of our research for the coming year. The following discussion expands on these ideas.

Several ideas come together under the topic of naming of objects. First is the concept of object-oriented addressing. Our experience with Multics has convinced us of the great benefit of a segmented address space, in which the user may refer to a number of segments, information containers, that are separately named. The Multics realization of segments, however, is deficient in two respects. First, segments in Multics are rather expensive, so that the user cannot afford to have a very large number, each perhaps rather small. Rather, he tends to use large segments with a great deal in each one. Second, the names used by the hardware to refer to segments do not have sufficient scope in Multics: they exist for the duration of a process, while they should be unique for all time. The importance to a distributed computing system of providing a programming environment in which these deficiencies are corrected is that these inexpensive segments with universal names, often called "small objects", appear to be the entities that will form the basis of inter-system transfer in the distributed system, and it is crucial, if the system is to provide support for the distributed computation, that the elements being transfered are entities understood at the system level.

The second issue is the question of locating and resolving names generated at distant sites. In a traditional centralized computer system all objects are under central control and are consequently known. In a distributed system, objects are necessarily generated in an independent way, hence there exists no central location where they are all known. This important distinction has far reaching consequences in the design and organization of a distributed system. The problems here range from that of finding a particular record of a data base stored on some foreign machine to locating the mailbox of the user whose home machine is unknown.

Third, there is the problem of how objects shall be represented in the programming environment of the user. In Multics, a segment from the file system is made accessible to a user program by a simple mapping between two name spaces, rather than by the copying of this file into the user's address space. This form of direct access appears to simplify the construction of programs, but it is not a natural representation in the case that the object being referenced is in fact on a different site. The particular mode of access developed on Multics was designed to facilitate the direct sharing of information between users. It appears that while the pattern of access itself is very desirable, the ability to share objects directly is not exploited to a significant extent. To the extent that this is true, as we are now attempting to determine, it may be possible to devise a means of representing objects in the programming environment of the user that maintains the same programming advantage as the direct access file system but is more natural in the case where some of the objects being referenced are stored in another node of a distributed system.

Fourth, objects in a distributed computer system must be protected against inappropriate reference or modification. There are several aspects to this problem. A user, properly identified on one machine, may need to have access to information stored on a different system. How is the user's identity and access rights to be passed from one machine to the other? Another problem has to do with users who are not permitted access to data itself, but are permitted to perform certain manipulations of the data, for example to produce statistical summaries. The necessity of imposing this sort of restriction on the user may strongly constrain where certain computations may be performed in the distributed system. This sort of constraint must fit into the computing patterns of the system in a natural way.

Last, there is a protection issue of a different kind. In addition to worrying about unauthorized access to a data base, we must worry about two authorized users who render a data base inconsistent by updating it simultaneously. This problem is cured in a centralized system by various sorts of locking strategies; however most locking strategies do not extend well to a distributed case, in which delays or failures may occur in the locking protocol. We have had some success in attacking this problem, with the discovery of a process synchronization mechanism, called eventcounts, that seems to work well in the distributed environment. Much more work is required, however, to demonstrate its applicability.

6.5 Uniqueness of this Approach

Although there are many research projects underway claiming the banner of "distributed systems," most of these projects are addressing other issues, such as hardware architecture, process synchronization, or high computation rate, rather than the coherent naming issues raised in the previous section. Two recent or current ARPA-supported research projects do have some objectives similar to the work proposed here: the RSEXEC system, and the National Software Works.

The RSEXEC system was an ARPANET experiment in using files from a distance, on homogeneous machines (PDP-10's running TENEX) whose operating systems could be modified slightly if necessary. The National Software Works (NSW) is an ongoing project to create a program development system using facilities from several ARPANET hosts, and is based on a concept of a network-wide cataloging system. Both of these systems deal with relatively large objects (files) and use naming techniques for inter-system file reference that could not (economically) be used for more fine-grained information structures. In both of those systems, the intended applications do not call for more fine-grained structures, but future, different applications will cal for larger numbers of inter-object references, and more refined addressing techniques need to be invented. Another way of describing the same distinction is to note that both RSEXEC and NSW operate by copying entire files; no finer-grained operations exist, and there is no concept of sending a process to work on the data at its original site. The work described in this proposal is intended to explore exactly such opportinities.

Another distinction separating NSW and RSEXEC from the proposed work is that neither of the earlier systems had explicit goals of lower vulnerability to failures and unhampered growth, and fail in several respects to display these features.

Finally, NSW in particular was designed under the constraint that it should fit into previously existing inhomogeneous systems and the ARPANET: for its application that constraint is acceptable, but it means that techniques requiring either higher bandwidth or significant changes to the underlying systems could not be considered. The proposed work considers these two constraints to be substantially relaxed, with the probable result that much more rapid time scales of interaction should be feasible.

In summary, the previous and ongoing research projects have goals and constraints that differ significantly from the proposed work.

6.6 Experimental Programming Environment

The studies proposed in section 6.4 are conceptual in nature, and while a proposal for an actual system implementation may evolve at a later date, no immediate system development is appropriate. It is very important, however, that we have available to us as a tool a programming environment in which we can evaluate concepts and experiment with new ideas in distributed computing. One of the particular strengths of our research group has been that we combine a strong expertise in developing new and theoretically interesting programming structures with a willingness to construct practical systems to demonstrate the validity of our ideas. It is clear, looking back on our Multics experience, that our ability to implement our ideas as we proposed them was instrumental in shaping these ideas to their final viable form.

We propose to use as an experimental test bed for our research the LCS Local Network, implementation of which was started in calendar 1976. The network will provide a valuable facility through which we can experiment with the naming and referencing of objects in a practical environment. During 1976, as we proposed, we will complete a network design based on a buffered version of the Ethernet, originally developed at Xerox PARC. We hope that initial operation of a few PDP-11 hosts will take place by the end of the current year. In our evaluation, which led to the decision to use the Ethernet technology for this local network, we determined that we could discover no significant distinctions between the Ethernet technology and the ring network developed at the University of California, Irvine, except for one issue related to acknowledgement of certain sorts of broadcast messages. We believe that we can devise an interface to our network which is independent of whether the ultimate technology is a ring or an Ethernet. Depending on the availability and suitability of ARPA developed interfaces to a ring network, we will implement part of our network as a ring, in order to evaluate for ARPA the comparative merits of the two architectures.

The LCS local net, in addition to providing an experimental medium for studying issues and problems of distributed computing systems, will serve as a tool for making more effective use of ARPA provided resources in order to carry out research missions of the Laboratory. Since all available ARPANET ports are now used, and all available terminal ports on the TIP are now occupied or committed, the existance of the local network, equipped with suitable concentrators and gateways, will eliminate present and future problems related to the necessary interconnection of resources provided in support of ARPA research projects.

While the development of the network itself is clearly not a research project, we feel that the installation of this sort of network in the context of LCS will make a significant contribution to several ARPA goals. First, while ARPA has considerable experience in the area of file transfer and message passing, it has been based largely on the ARPANET, which is almost 100 times slower than this local net. This increased speed should, even in the case of interconnecting pre-existing systems, lead to techniques for system interaction which are more intimate and sophisticated than in the ARPANET. We expect that the file systems of machines connected together with the local net should be able to resolve a reference by a program to a file on another machine in a manner invisible to the program, and without intervention by the user of the program. An understanding of the actual advantages which follow from this increased speed should be generally valuable. Second, the LCS network will serve as a means by which CSR can participate in the ongoing ARPA research in internetworking. As a result of our initial protocol design efforts, we believe that we can define a large common element between our work and the ARPA internetwork protocols, where this common element represents the fundamental mechanism required for internetworking. Working with other relevant ARPA researchers, we propose to define this fundamental protocol. Finally, the LCS local network can serve as an example for imitation by the many other organizations within ARPA and the DoD that have encountered a need for computer interconnection, but have no model of how to proceed.

6.7 Qualifications of the Division

The Computer Systems Research Division of the MIT Laboratory for Computer Science has as its major interest the discovery of pragmatic ways for systematically engineering useful computer systems. It brings to this research a perhaps unique combination of strong expertise in the development of theoretically interesting system structures, along with an interest in practical demonstration and evaluation of its ideas through actual implementation in real systems. In the past, this research thrust has led to the development of time sharing through the vehicle of the Compatible Time Sharing System, and the development of the information sharing computer utility, through the vehicle of Multics. More recently, the Division has been working on protection and security in information sharing systems, again using Multics as a vehicle. The development and continued experimentation with and evaluation of Multics has given the group great strength in the area of understanding issues of object naming, accessing, and protection, which we feel are crucial in the successful understanding of distributed systems. The Division is also well acquainted with current research in the area of distributed systems, through participation in various projects, including the National Software Works, and through a detailed study of ongoing research performed last Spring, some of the conclusions of which are presented in this proposal.

6.8 Proposed Work

We expect that successful completion of the research proposed here will require continued effort for a period of two or three years. As detailed below, we expect the first year to be spent understanding how the concept of object-oriented addressing can be integrated into a machine architecture. Successful completion of this task will give us a much better idea of the constraints that must be imposed on object names in order that they be realizable. Research in subsequent years can then address such issues as resolving these names in a distributed environment. The result of this research should be a detailed description of a prototype distributed computing system. The system we develop will explicitly address itself to the solution of the problems discussed in section 6.1. Specifically, we envision an interconnected collection of machines, with each machine providing an independent file manager, but allowing access by programs to files in other machines with the convenience and utility now associated with advanced centralized operating systems. A typical application which could be constructed on this distributed system would be an integrated file manager, in which records of files could be linked together across machines using hardware-oriented names (addresses) in the same manner that records can be interlinked on centralized operating systems today. For example, the functionality that our system is to provide would be suitable as the storage management substrate of a typical data management system, but in a distributed rather than centralized form. These interconnected machines will operate independently of one another, providing necessary reliability, and as new machines are connected, their file systems will be accessible as part of the existing file structure, without the programs or the user having to take explicit steps to learn about the structure of the file system on that new machine. This particular architecture will provide a successful solution to the constraints of reliability and rearrangeability present in advanced applications such as command and control, and will help to organize functions such as logistics in a natural and efficient way.

While the primary output of this project will be technical reports and theses which will collectively provide the system description specified above, there will also be the results of various experimental implementations which will of necessity be performed to validate the design which we specify. These implementations, which will be performed using the computers at the laboratory and the local net discussed in section 6.6, will serve as an additional means of communicating our results.

The results of this research can probably best be exploited by ARPA in the following way. Following the completion of our system description, an implementation of the specified machine should be undertaken, as a viability proof. This is not a minor project, since it will also require the construction of an operating system for the machine. An additional part of this viability proof is the implementation of one or more applications, such as a data management system, in order to demonstrate the features of the design. With advancing technology for creation of database management systems, implementation of applications to run on this machine should not be an overwhelming task. It is not clear whether this construction phase would best be performed by our group, perhaps in conjunction with some manufacturer, or could best be done by some other agency.

As described, the system specification resulting from our research will provide a distributed file management environment in which an applications builder can construct a spohisticated package for the management and accessing of data in a straightforward manner. There is a related problem, which is the integration after the fact, of preexisting data files which were not initially constructed using a uniform structure for the data representation. We do not discuss this problem in the current proposal, because we feel that the most fruitful approach to the problem will not be apparent until we have a much better grasp of certain issues related to naming and protection of objects. However, we can indicate the extent to which we feel the proposed research can contribute to the solution of this problem. First, a primary component of the structure of a complex data organization is the means by which related items of data are tied together. In other words, what naming scheme is used to link one data record to another? In almost all such structures existing today, the naming strategy is specifically devised for the particular application, so that names of records are meaningless outside the domain of the particular programs constituting the application. This very severe limitation is directly addressed by our research, in that it is our intention that all such interconnections between data items should be made using the standard object naming scheme of the computer, rather than an application specific naming scheme. Another way in which our research may contribute to the solution of this problem is that as part of the protection of data items, it will be necessary to explore how to transmit a request for a data manipulation from one machine to another, so that the transformation can be performed in the protected environment in which the data is stored. To the extent that this mechanism is required, it will also provide a means by which a system can be requested to transform data, the format which is unknown to the requestor. Beyond these rather preliminary speculations, we are as yet uncertain as to how the problem of high-level representation and transformation of data ought to be handled. We believe that this problem, although very relevant to our research, is separable, and we note that it can be studied in the context of a centralized system as well as a distributed one.

The following specific activities are proposed for the calendar year 1977:

1) Begin the specification of a suitable distributed system architecture.

First, we will produce a specification of a system which supports small objects for the storage of information, as discussed in section 6.4. We intend to begin by studying the intrinsic structure of various existing data management programs, to understand exactly what attributes small objects must have in order to meet the demands of these applications. For example, can aggregates of small objects be considered as a unit for purposes of storage allocation and protection, and if so, are the same aggregates appropriate for both of these functions? We are currently attempting to understand the need for direct sharing of stored information, as a first step in this study. We must also understand the currently available strategies which can be used to implement the mapping of object names in an acceptably efficient manner.

Second, we will identify constraints which must be imposed on distributed systems if they are to provide the functional advantages discribed in section 6.1. Initially, we expect the most success to come from addressing the issues of reliability and growth potential.

Finally, we propose to determine any additional benefits of distributed computing which should be included as goals of the architecture which we design. While we feel that the list of advantages presented in 6.1 directly addresses ARPA needs in a compelling way, we hope to identify other potential advantages.

2) Complete the implementation of our experimental programming environment, the LCS local network. We expect that the hardware design and low-level protocol development for the net should be completed by the end of the current year, but effort will be required to place the network in operation. To the maximum extent possible, we intend to make use of technology already developed by ARPA for this network; in particular, for the gateway which we must provide between the ARPANET and the LCS local net we plan to use the gateway machine which has been developed by ARPA. As discussed in section 6.6, we will perform two explicit experiments using this network. First, to the extent that ARPA developed technology for ring networks is applicable, we will perform an experimental comparison between the ring network technology and the Ethernet technology which we intend to employ initially, for the purpose of determining any possible operational advantages of one over the other. Second, we will attempt to develop a basic inter-network protocol, embodying the commonality between TCP and the rather similar protocols which we have developed for our own local net, in order to better understand the fundamental

mechanisms required for inter-networking. This effort is intended to be viewed as part of the inter-networking research currently being carried out by ARPA. If possible, we would like to broaden the range of networks available for our studies in inter-networking by use of ARPA packet radio technology. If ARPA can provide two packet radio transceivers, we will connect the Multics machine to the LCS Network using packet radio, thereby creating a multiple network consisting of the ARPANET, the LCS local net, and packet radio. This experiment with the interconnection of subnetworks using radically differing design should enhance our contribution to the ARPA research in inter-networking.

3) Implement a version of the "Transmission Control Program" (TCP) for Multics, so that Multics can communicate with other systems which have implenmented a TCP. This implementation will enhance our ability to communicate with other sites on the ARPA Network, and will allow Multics to participate in inter-networking activities.

6.9 National Software Works Participation.

We anticipate that by December 1976, most of the significant work required to make Multics a tool-bearing host, participating in the National Software Works will be complete, and that it will be possible to edit NSW files, translate them, and use the Multics GCOS simulator, remotely through the facilities of NSW. In addition, the NSW interface for Multics has been developed in a way that many other Multics commands should be either directly usable, or usable with only minor changes. For this reason, we propose to reduce our level of activity on this project to that needed to tie up loose ends, and to provide technology transfer support. We are actively looking for an organization that is interested in taking over support of the special NSW packages created for Multics.