The research performed under this grant is concerned with distributed languages and algorithms. Work during this past year can be divided into three areas.

A. Distributed Algorithms

We continue to be concerned with low level algorithms which, for example, might be used in a distributed operating system to support resource allocation, manage distributed data or enhance reliability. One such algorithm which we have developed is a protocol for updating multiple copy databases where serial consistency is not required. In this case the database is a dictionary which supports the insertion and deletion of entries. We have improved upon some earlier work in this area [FM82] by showing how communication costs can be reduced and the algorithm tailored to the topology of the network. A paper describing the algorithm and a proof of its correctness will be presented at a forthcoming conference [WB84]. We have developed several extensions to this model. One, which we refer to as the multiple insertion case, allows a given dictionary entry to be inserted more than once. In the original formulation of the problem this is prevented by tagging each entry with a unique number, thus forcing each entry to be unique. This may not be appropriate in applications where the creation of truly identical entries is unavoidable, the result of the delays involved in propagating information through a network. A second extension is a technique for reducing the size of data structures when the network becomes very large.

Some additional work has been done on the distributed stable storage algorithm described in last year's Summary of Research Report ('82-'83). The algorithm is a technique for reliably storing data in a broadcast network by replicating it at a number of nodes. The Markov model which was developed to study this algorithm has been refined and a program has been written to display the results. The mean time to data loss is obtained as a function of the degree of replication and the probability of individual node and communication failures. A revision of the original report has been produced and is currently being reviewed for publication [Be83].

A major new research direction in this area is the study of distributed algorithms involved in implementing a distributed file system. Our interest in these algorithms is prompted by the recent award by the National Science Foundation of a Coordinated Experimental Research Grant to the Computer Science Department. (Stony Brook was ranked first in the nation this year and received a large award.) The proposed research involves building a system in which a relational database plays a central role. As one of the principal investigators on this grant my concern has been with the distributed nature of the system and is an
outgrowth of AFOSR supported research over the past few years. Our work in this area over the past year has been concerned with developing distributed algorithms to support naming of relations and concurrency control for transactional access to relations in the network. In the naming area we have developed a technique to extend the UNIX file structure so that relations distributed across the network can be accessed in a transparent way from any site. The goal is similar to that of the LOCUS system [WPEKT83] but the approach we have taken involves no modifications to UNIX. The stress has been on developing an algorithm which imposes no additional overhead if the relation to be accessed is stored locally. A technique for dealing with name collisions is an important aspect of the work.

In the concurrency control area we have developed a multiversion optimistic concurrency control algorithm. Intentions lists, which must be integrated into any transaction system capable of coping with failures and aborts, serve double duty by providing multiple versions of a relation for use by executing transactions. By choosing the correct version it is possible for each transaction to see a consistent view of the database without resorting to locking and thus validation of read only transactions is eliminated. We claim this as a major advantage of the approach (and in particular, an improvement over standard optimistic concurrency control [KR81]) since in many applications read only transactions are considerably more numerous than transactions which update the database. The latter transactions must be validated in the normal way. An important part of the work is a technique which allows a transaction to extract the appropriate version simply. We are currently in the process of extending the algorithm to function in a distributed environment.

B. Distributed Languages

The work in this area has entered an implementation phase. A report describing the proposed language structures has been written [AB3] and submitted for publication. The emphasis here is on interprocess communication and, in particular, multicast. In traditional approaches to interprocess communication a message is addressed to a process (or to a port attached to a process). Instead, we use a name based addressing approach in which a message is addressed to a name which is visible to a subset of the processes in the system. These processes constitute a multicast group, all of whom may receive a copy of any message addressed to the name. We intend to implement our ideas in the context of Modula-2 and a preprocessor for this purpose is currently being designed. The preprocessor will perform type checking related to interprocess communication and then covert programs using name based addressing into standard Modula-2 programs which call upon library modules to support the new features. Another aspect of this work is the development of multicast protocols within UNIX 4.2 to support the language constructs. This is described in the next section.

Since the language work is directed towards supporting low level distributed algorithms, timeout plays a significant role. As a separate project we have studied the semantics of timeout to develop a formal verification technique for
message passing programs in which a sender or receiver of a message may
timeout if message transmission is delayed beyond a specified interval. An
important issue here is the notion of predicate transfer between the communicat-
ing processes. The predicate describes the information which can be deduced by
the process which has timed out about the state of its correspondant. A report
on this work has been submitted for publication [Be84].

C. Multicasting in a Network

Two projects are in progress in this area. The first is in support of the work
in distributed languages and involves developing a multicast protocol to be
embedded in UNIX 4.2. A protocol has been designed which integrates with the
4.2 socket structure. Multicast sockets operate in datagram mode with the
difference that a set of processes (the multicast group) may essentially be con-
ected to the same socket and thus receive copies of each message sent to the
socket. The multicast protocol (MP) is implemented using the internet protocol
(IP) in a manner analogous to the implementation of TCP. Multicast addresses
on the ethernet are associated on a one-to-one basis with names in the distributed
language. All nodes in the net which support processes in the multicast group
respond to the address corresponding to the associated name. A version of the
protocol has been designed for use on a single ethernet and is currently being
implemented.

In extending the protocol for use in an internet environment two approaches
can be taken. The simplest is one in which the sending node unicasts a copy of
the message to be sent to the multicast group to some representative node on
each net in the internet containing a process in the multicast group. Each
representative then multicasts the message to local members of the multicast
group (i.e., members on its net) using the above scheme. This is analogous to
directed broadcast as described in [Bo82] and suffers from the inefficiency that,
particularly in large internets with large multicast groups, duplication of unicast
messages may result. This follows from the fact that distinct copies must be sent
to each net even though they may follow essentially the same route. This can be
avoided by using a more elaborate scheme, which we call extended multicast, in
which a tree is dynamically maintained in the internet using a distributed algo-
rithm and serves as a routing structure for delivering the message to be multicast
to each net. This work has been described in a recent conference presentation
[FWB84]. Another aspect of the work involves the use of the dictionary algo-
rithm described above to keep track of membership in the multicast group.

No patents have been requested on this research.

References

[AB83] M. Ahamad and A. Bernstein, "The Application of Name Based Address-
ing to Low Level Distributed Algorithms", Tech. Report #83/050, Dep't of
Computer Science, State Univ. of New York, Stony Brook, NY, Sept 1983.


