Linda-LAN: A Controlled Parallel Processing Environment

George E. Cline and James D. Arthur

TR 92-37

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

July 29, 1992

Linda-LAN:
A Controlled Parallel Processing Environment

George E. Cline and James D. Arthur

Virginia Polytechnic Institute & State University
Department of Computer Science
Blacksburg, VA 24061

Abstract

An investigation is performed on a controlled parallel processing environment based upon the Linda paradigm, a conceptually simple programming and operational framework. The environment utilizes the unused computing resources available on a network of computers to provide a low-cost parallel processing solution. The environment efficiently and effectively manages system resources via a centralized control sub-system. A detailed overview of the environment executing on a local area network is presented along with analysis on its control sub-system. In addition, the control sub-system's execution characteristics of stability and scalability are substantiated.

I. Introduction

The computing problems of today are becoming increasingly more complex and time consuming. For example, mapping out the human genome requires a significant amount of computing power and time to analyze the approximately 100,000 genes in the human cell. In order to develop physical maps of each chromosome and to determine the sequences of various DNA chains, powerful parallel computers, vector processors and special-purpose computers are a necessity; however, the expense of such machines far exceeds the limits of many users. Alternative methods of achieving parallel performance at an economical price are desired. One such alternative results from the idle CPU cycles existing on local area networks. With the increase in the computing power of workstations and their declining costs, the unused computing power attached to a local area network (LAN) can effectively be transformed into a parallel processing environment. Exploiting such an environment requires a specification and operational framework that is portable, easy to use and efficient.

The focus of this research effort is to create a controlled parallel processing environment utilizing a conceptually simple parallel programming framework that operates on a local area network of low cost personal computers. The establishment of a controlled environment provides for effective and efficient management of utilized network resources. The employment of a conceptually simple parallel programming framework facilitates the utilization of such an environment and the application of low cost personal computers offers an economical solution to parallel processing. The intent of this paper is two-fold. First, the newly created environment and the sub-system which controls it are examined. Second, the desirable execution characteristics of scalability and stability are shown to exist within the control sub-system. We hypothesize that scalability and stability are achieved within the control sub-system by effectively and transparently controlling environment processors and the allocation of instantiated program processes to processors[1].

Scalability exists in two forms. The first form deals with the scalability of network processors and the second form deals with program processes. Network processors are added and removed arbitrarily during the life of the system. In order for the environment to remain useful to its users, the execution of the control sub-system must scale relative to the change in the number of processors. If relative scalability is not achieved within the control sub-system, users will not consider the environment to be an effective parallel processing alternative. The execution must also be able to scale with the change in the number

[1] The term effective refers to the efficient and appropriate selection of environment processors. Transparency refers to the hidden selection of a processor from an executing program so that the program is not required to be especially written, compiled, or linked for the environment, based on number of processors used.

of instantiated program processes. A dependency may exist between the number of program processes and program input. For instance, a parallel program to find occurrences of a string within a text file could be written to create a new search process for each line of data found within the text file. If this form of scalability is not attained, programmers may be burdened with the responsibility of controlling the number of utilized processes from within the program source code. The environment would not be regarded as being easy to use by its programmers.

The execution characteristic that is desirable for any processing environment is stability. If an environment can produce predictable patterns of program execution, the environment is judged to be stable. Only a stable environment provides the usability and dependability required by its users. Stability is especially desirable to this research endeavor in that it will help to identify those areas of the control sub-system which require enhancement and possibly provide information on future research. Without having stability within the control sub-system, stability within the environment cannot be achieved.

This paper begins by giving a brief background of the Linda paradigm, which was the chosen specification and operational framework for Linda-LAN[2]. A detailed overview of the current Linda-LAN environment is described, as well as its physical and logical topologies. In addition, an examination of the control sub-system is presented. Special attention is given to the control sub-system's allocation strategy of program processes to environment processors. Finally, the experimental results substantiating the stability and scalability of Linda-LAN's control sub-system are offered.

II. Background

Linda is an effective parallel computational framework which was specifically chosen for the Linda-LAN environment [CARRI89a, GELER85a and GELER85b]. There are a number of reasons for choosing the Linda paradigm. In particular, Linda

- establishes a coordination language which allows a parallel program to perform process creation, synchronization, inter-communication and the sharing of distributed data structures through a logically shared object memory, called Tuple Space[BERND89],

- simplifies the complexities of simultaneously executing parallel processes, by allowing the programmer to develop each executing process independently from the rest[CARRN90],

[2] Linda-LAN is a software-based framework implemented by the Linda research group at Virginia Polytechnic Institute and State University. The LAN is the only "specialized" hardware needed to support the parallel environment which executes C-Linda programs.

- is portable; implementations of Linda systems exist on several different machine architectures and network configurations[3].

In addition, by building upon the basic Linda paradigm, additional advantages for Linda-LAN's network environment are offered:

- A *true* parallel computational environment is established.
- *Idle CPU cycles* existing on local area networks are exploited.
- An architecturally *low cost* parallel environment is made accessible to a wide range of users.

Linda is a coordination language rather than a complete parallel programming language. A coordination language [CARRI90 and ZENIT90] provides the primitives to create processes as well as coordinate communication among processes. By virtue of being a coordination language, Linda primitives can be introduced into many base programming languages. The original implementation, using C as its base language, exploits a preprocessor approach which transforms Linda operations into C source code. Implementing the Linda primitives in this way is especially useful when parallel systems need to be developed in multiple languages. Linda has been embedded in a wide variety of other languages - C++, FORTRAN, various Lisps, PostScript, Joyce, Modula-2 and soon Ada [GELER90].

When one discusses the Linda paradigm, two characteristics are often touted: its ease of use and its portability. From a conceptual standpoint, parallel programming within the Linda framework is intentionally high level, which is exactly why it is so flexible and powerful. Not so surprising, however, this high-level approach is at the root of Linda's greatest criticism - its questionable performance [DAVID89]. Linda does exhibit acceptable performance on both shared and distributed memory parallel (MIMD) machines [BJORN88, BJORN89 and CARRI87]. In addition, Linda has demonstrated its applicability on local area network platforms, although performance suffers for applications with tightly coupled processes [LELER88 and WHITE88].

Because Linda does not rely on any specific type of architecture, Linda programs are easily ported to a wide variety of machines with little or no modification. Machines currently hosting Linda include workstations such as Sun, DEC, Apple Mac II and Commodore AMIGA 3000UX. Linda has also been ported to parallel machines such as the Sequent, S/Net and the Hypercube.

[3] Implementations include the Sequent and Encore multi-processor shared memory machines as well as VAX/VMS Ethernet networks.

The Linda approach supports process creation and inter-communication through a shared data/process repository called Tuple Space (TS). Linda provides operations to generate data tuples (`out`), to read data tuples (`rd`), and to remove them from TS (`in`). Tuple Space not only contains data tuples but also process tuples (created with the `eval` operation) which are often called "live tuples." These process tuples are instantiated and are eventually replaced by a data tuple when the instantiated process finishes executing. TS can also be used to share data structures among processes and synchronize the order of actions that processes perform.

The Tuple Space concept is highly desirable within a network environment. Each of the TS primitives, initiated from within a Linda program, corresponds directly with a series of communication messages interacting with Tuple Space. Tuple Space may reside on a dedicated network machine or be distributed across the network onto any number of network machines. Activated Linda process tuples may also reside on any number of network machines. Once again, although the Linda paradigm is conceptually sound, there are recognized performance bottlenecks in accessing Tuple Space on a network platform; however, these performance problems are being addressed and solved by current research[ARTHU91 and SCHUM91].

III. Linda-LAN Overview

The first objectives set forth within this research effort are to establish a controlled parallel processing environment within a conceptually simple specification and operational framework on a low-cost local area network. Linda-LAN is the environment which meets those objectives. Linda-LAN is a parallel programming environment based upon Linda, a conceptually simple paradigm supporting an easy to use coordination (parallel) language. Linda-LAN utilizes the multiple processors offered by the workstations of a local area network. The environment is primarily a software-based system and requires no specialized hardware. The system supports the distribution and execution of programs written in the C-Linda language. Linda-LAN establishes control of network resources and manages them via a control sub-system. Moreover, the idle cpu cycles existing on the LAN workstations are used by the environment, making this parallel processing system highly cost effective in solving the more complex and time consuming problems of today.

A. Physical Topology

The physical topology of the Linda-LAN environment is illustrated in Figure 1. The environment utilizes an Internet network of UNIX-based workstations. Although present implementations require a homogeneous environment of Commodore AMIGA 3000UX workstations, plans are being made to migrate into a heterogeneous environment. The Linda-LAN system is not limited to a local area network or to the physical Ethernet LAN as depicted in Figure 1. Machines may be connected to the environment over existing gateways or be interconnected via other physical network technology. This facility is provided through TCP/IP Internet Protocols. The current versions of Linda-LAN, however, do utilize a single physical Ethernet.

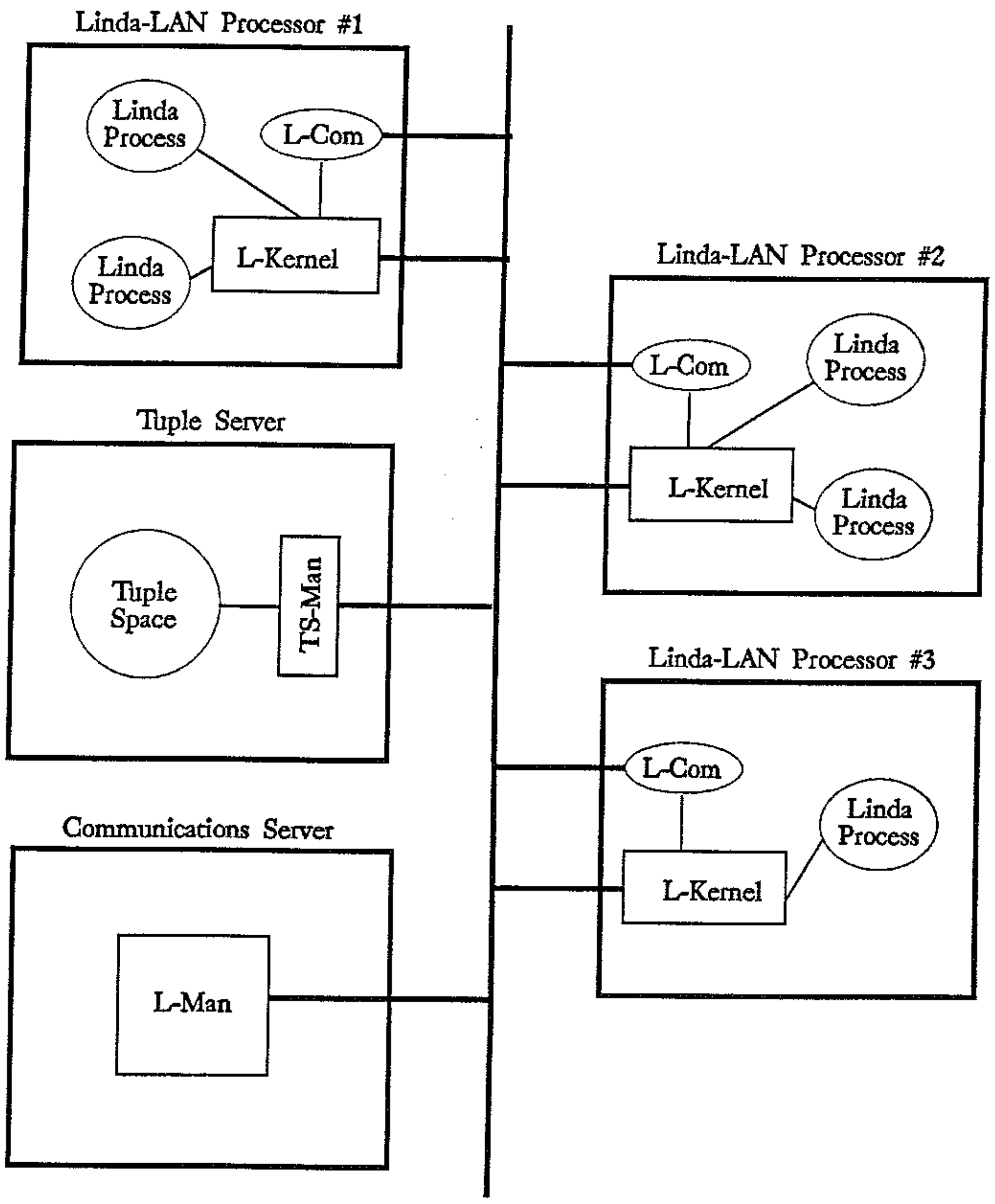

Figure 1. Physical Topology of Linda-LAN.

Linda-LAN is composed of multiple Linda-LAN Processors, Tuple Servers and the Communications Server. The attached workstations (nodes) within the environment can execute Linda program processes (active tuples), manage the shared Tuple Space, or manage the environment. As seen in Figure 1, the physical topology of the Linda-LAN environment allows for the distribution of multiple Linda processes initiated from within a single Linda program onto several Linda-LAN Processors (workstations) of a network. Linda-LAN Processors are able to support more than one executing Linda process; however, a maximum of one Linda program may execute at any given time on the system. This limitation was introduced into the system for the benefit of other network users. Current network utilization by non-Linda users is extremely high. Minor modification of the system is required to increase the number of executing Linda programs. Future research may be directed towards the scheduling and execution of multiple Linda programs.

The Linda program processes communicate with Tuple Space via a node resident process, called a Linda-LAN Kernel (L-Kernel). The L-Kernel packages each Tuple Space request and routes the request to a specified node containing another environment process, called a TS Manager (TS-Man), which manages and serializes all requests to Tuple Space. The TS Manager resides on a specified workstation of the environment, called the Tuple Server. Although a single Tuple Server is depicted in Figure 1, multiple Tuple Servers may exist. In fact, later versions of Linda-LAN have been implemented with multiple Tuple Servers. This flexibility offers enhanced performance to the system, by improving the access to Tuple Space. The communication demands of the system mainly occur between the workstations, which execute Linda processes, and the designated Tuple Server(s).

The communication between any two nodes of the system is point to point via the BSD *socket* communication protocols found within the System V Release 4 UNIX operating system. The point to point communications are made available by the system process, called the Linda-LAN Manager (L-Man), which executes on the last physical component of the Linda-LAN system called the Communications Server. Network address information on all participating workstations is maintained at the Communications Server. During system instantiation and program instantiation, the Communications Server relays needed network information to all executing system processes. Only one Communications Server exists per Linda-LAN system; however, multiple Linda-LAN systems may operate over common nodes of the network.

In addition to the network information, global system information is managed at the Communications Server. The global information is collected from system processes, called Linda-LAN Communication Managers (L-Coms), which reside on each participating workstation that executes a Linda program process. Each L-Com periodically provides L-Man with utilization information on the processor on which the L-Com resides. Due to the wide fluctuations in workstation activity during the execution of a program, the utilization information is a necessity in the mapping of Linda program processes to idle Linda-LAN Processors (workstations). One of the duties of the Linda-LAN Communications Server is to determine the Linda-LAN Processor to which a Linda program process is to be distributed. Without the utilization information, effective program execution would not be possible. In addition, if not for the Communications Server, the primary user of a workstation may find executing Linda program processes consuming unavailable cpu resources.

Since the global information maintained at the Communications Server is highly important, the loss of the Communications Server would result in the failure of the system and any executing program. An outage of any other node, except for the Tuple Server, may not result in the immediate failure of the system. The Tuple Server is also a key component in the robustness of the system. The breakdown in a Tuple Server would result in the loss of shared program data and cause the system to fail. Future single Tuple Server versions of Linda-LAN can offer some hope in maintaining the integrity of the system by following existing fault tolerance mechanisms provided by today's file server technology. Although greater performance benefits are seen with the multiple Tuple Server versions of Linda-LAN, the robustness offered by the single Tuple Server versions of Linda-LAN may be found to be more desirable. These fault tolerance issues are an area of research which require further investigation.

B. Logical Topology

Figure 2 represents the logical topology of the Linda-LAN environment. The environment is composed of a control sub-system and a data sub-system. The control sub-system is responsible for maintaining network information, system instantiation and termination, program instantiation and termination, data sub-system instantiation and termination, program scheduling, process to processor allocation, network monitoring, and executable code distribution. The data sub-system is responsible for handling all Tuple Space requests, process execution, process instantiation and termination, and all communication between program processes and Tuple Space.

Linda-LAN was specifically designed for the partitioning of the system into the control and data sub-systems. By dividing Linda-LAN, a more simplistic view of the environment has been established and our research efforts have become more focused. The management activities of the environment have been segregated into a separate entity. These activities can be effectively controlled and managed within the control sub-system. The data sub-system is left to concentrate on the efficient execution of a program. In addition, the separation has made the implementation of the system much smoother. Figure 3 represents the interaction between the control sub-system and the data-sub-system as viewed by the control sub-system. The remainder of this section concentrates on the control sub-system components and the advantages found within it. In addition, the control sub-system's process to processor allocation strategy is discussed. The data sub-system is not specifically addressed within this paper and is left for future publications to discuss.

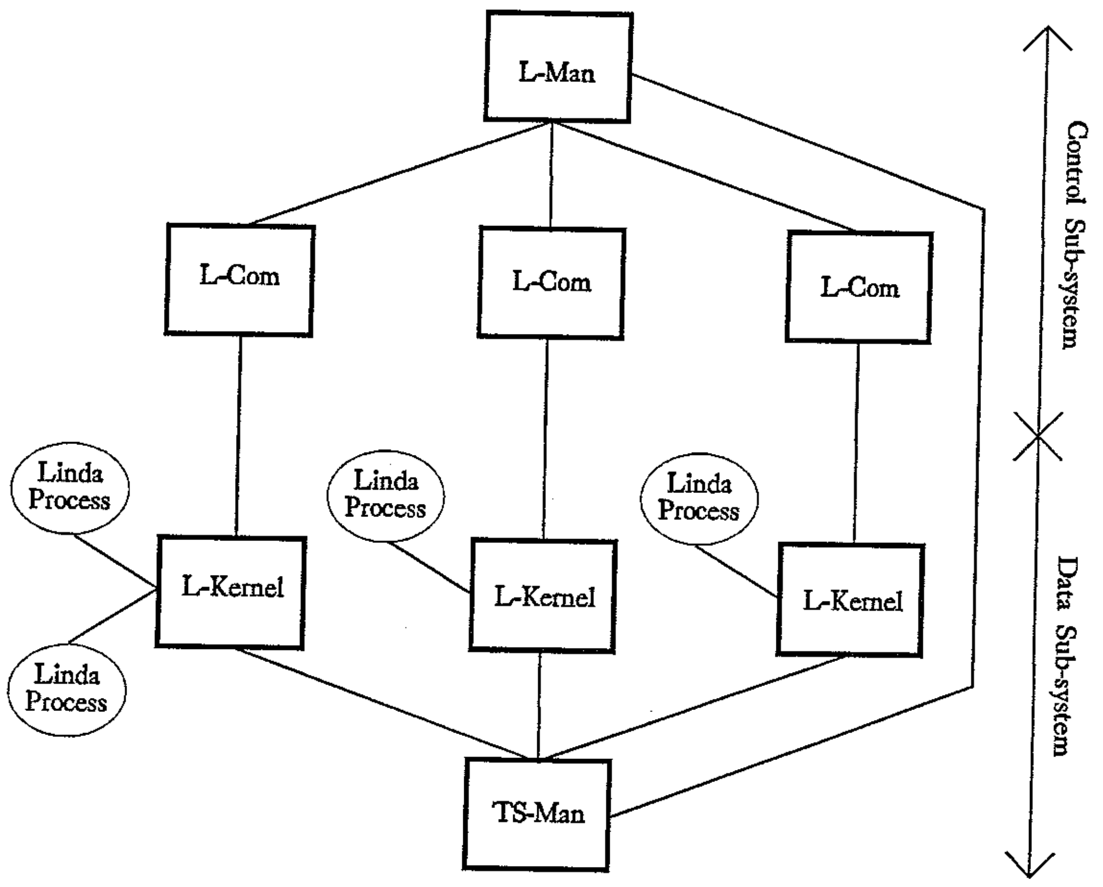

Figure 2. Logical topology of Linda-LAN.

C. Control Sub-System

The control sub-system of Linda-LAN consists of the Linda-LAN Manager (L-Man) and a multiple number of Linda-LAN Communication Managers (L-Coms). One L-Com exists per Linda-LAN Processor, while a single L-Man resides on the Communications Server. As seen in Figure 2, a hierarchy exists between the L-Coms and L-Man. The L-Man is in complete control of the system, while each L-Com is responsible for the workstation on which it resides. The components work together to perform program instantiation and termination, data sub-system instantiation and termination, program scheduling, process to processor allocation, and executable code distribution. In addition, the L-Man performs system instantiation and termination, program scheduling, and network monitoring, as well as keeping track of all network information.

System instantiation is performed at the Communications Server by a designated system administrator starting the Linda-LAN Manager. The responsibility then shifts to the L-Man to instantiate all L-Coms at the participating Linda-LAN Processors (workstations) listed within a system table. The data sub-system components are instantiated after a program has been scheduled for execution. By having the system initiated by an administrator, centralized control over the availability of the system is established. In addition, the environment provides the administrator with control over which machines participate in the system. The administrator is able to modify the system table of network information prior to the instantiation of the system. As each processor is added or removed from the system, the control sub-system scales appropriately by adding or removing a L-Com.

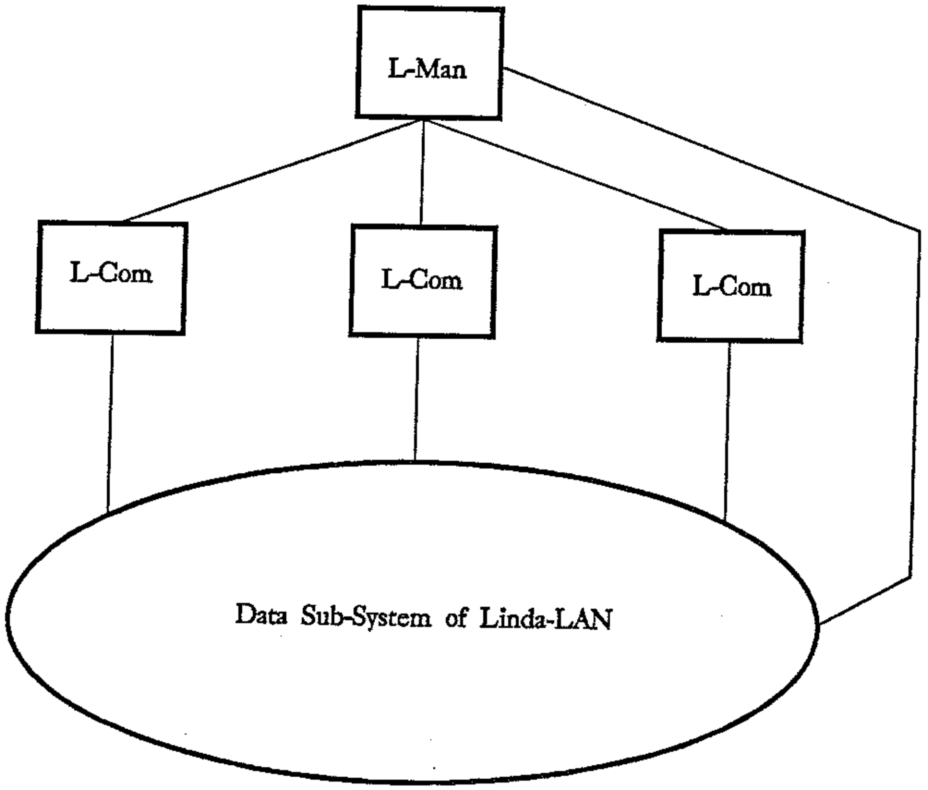

Figure 3. Linda-LAN as viewed by the control sub-system.

In order for a user to have a C-Linda program executed on the system, the user must follow a three step procedure. First, the user compiles and links the program at a participating Linda-LAN Processor. Second, the user submits a request to its local L-Com to distribute the executable code to all other Linda-LAN Processors. The request is first routed to the Linda-LAN Manager for approval. Once approved, the L-Man informs all other L-Coms to retrieve the executable code from the L-Com which originated the request. Finally, after the code has been distributed, the user must submit another request to its local L-Com for the program to be executed. This request is also passed on to the L-Man for approval. After being approved, the program is scheduled for execution and the L-Com, which originated the execution request, is informed by the L-Man to release the program into the system for execution.

There are numerous advantages offered by having the environment operate in this fashion. First, any Linda program process is able to execute on any participating Linda-LAN Processor. Second, once a program is compiled and linked for one Linda-LAN Processor, the same executable code is applicable to all other Linda-LAN Processors. There is no need for the program to be compiled or linked onto the other homogeneous machines. This procedure must change when Linda-LAN is migrated to a heterogeneous environment; however, the compilation and linking of the source program will only be required on a non-homogeneous subset of the Linda-LAN Processors. For example, if the system were composed of twenty machines utilizing three different machine architectures, the source program would only require compilation and linking on three machines which represent the three architectures. The remaining machines would copy the executable code from machines with an identical architecture. This new procedure is easily accomplished via the centralized control of network information found at the L-Man.

Another advantage of having the user perform the above procedure is the ability to add new Linda-LAN Processors to the environment without requiring the re-compilation, re-linking, or complete re-distribution of the program. Machines are merely added to the system tables found at the L-Man and the executable code is distributed to the new machine. Finally, it should be noted that Linda-LAN only requires a program to be re-compiled, re-linked, and re-distributed when the actual source code is changed. Programs can be repeatedly executed with changes in the number of instantiated Linda program processes, the program input, or the number of Linda-LAN processors utilized.

Before a Linda program is actually released into the system for execution, the data sub-system is instantiated by the control sub-system. The instantiation of the data sub-system is simple. First, the TS Manager is initiated by the L-Man at the designated Tuple Server listed within a system table. Once the TS-Man is instantiated, it returns network addressing information to the L-Man. The L-Man routes this information to all participating L-Coms. Each L-Com instantiates a local Linda-LAN Kernel, passing it

TS-Man's network addressing information. During each L-Kernel instantiation, a communications dialogue with the TS-Man is established. After a communications dialogue is established between the TS-Man and the L-Kernel on which the request for Linda program execution was initiated, the Linda program is released into the system.

Advantages are offered by having the data sub-system initiated at program instantiation. First, the data sub-system components are not required until a program has been executed. To have each workstation of the network manage unnecessary processes is wasteful. Second, the L-Coms are considered light-weight processes. Each L-Com utilizes a small amount of cpu processing and requires few system resources. The data sub-system components, on the other hand, require large amounts of system resources. Another advantage is found in the fact that the data sub-system components are not required to contain the code to manage themselves when there is no work to be done. Finally, the data sub-system components are alleviated from much of the communications setup. All network addressing information is supplied by the control sub-system. The data sub-system merely acquires the connections to the other needed components of the data sub-system and executing program processes.

Program termination is another function of the control sub-system. An executing program informs an L-Kernel of its desire to terminate. A request for termination may result from normal or abnormal termination. The L-Kernel routes the request to its local L-Com. The L-Com, in turn, sends the request on to the L-Man. The L-Man informs the TS-Man and all participating L-Coms of the program termination request. Each L-Com informs its local L-Kernel to terminate itself and all remaining Linda processes. The program termination is well controlled under both normal and abnormal termination. Data sub-system termination is also performed at program termination. The data sub-system components are no longer required and system resources can be freed.

To terminate the Linda-LAN system, the system administrator must issue a request to the Linda-LAN Manager. The L-Man aborts any active program via an abnormal program termination request and informs all L-Coms to terminate themselves. Once again, centralized control is made available to the system administrator.

Another managerial function of the system which is performed by the Linda-LAN Manager is the monitoring of the network. In order to determine the availability of a Linda-LAN Processor, the L-Man occasionally queries each L-Com for its workstation's cpu load utilization. This information is returned to the L-Man and is used for the allocation of program processes to environment processors. If an L-Com does not respond, it can possibly be removed from the table of participating Linda-LAN Processors. By removing the Linda-LAN Processor from the table, the environment assumes the machine is unavailable and continues processing requests. If a program is currently executing on the system, the current versions of Linda-LAN abort the program and the system. The main reason for operating in this fashion is to provide for some robustness within the system. Future research should be directed towards fault tolerance issues.

Although the control sub-system has many essential duties, the most critical task of the control sub-system is process to processor allocation. Without an effective mapping of Linda program processes to available processors, inconveniences to the primary users of the network workstations may result and valuable program execution time may be lost. The Linda-LAN Manager is responsible for the mapping of Linda program processes to Linda-LAN Processors. The L-Man is informed via a request from the TS Manager to select a Linda-LAN Processor to eval (spawn) a new Linda program process. In the current versions of Linda-LAN, the L-Man utilizes a round-robin approach in determining the next Linda-LAN Processor to execute a Linda program process. As long as a Linda-LAN Processor has not surpassed its cpu load utilization limit set at system instantiation, it may be selected as the next workstation to execute a Linda program process. If all Linda-LAN Processors have surpassed their cpu load utilization limit, the least recently selected workstation is chosen.

The main advantage of this mapping strategy is its simplicity and efficiency of selection. Although many different and sophisticated strategies are possible for the selection of Linda-LAN Processors, the current implementation does not require any additional information from a Linda-LAN Processor, except for the current cpu load utilization information. Other allocation methodologies may require knowledge on the termination of a Linda program process or on the expected execution time of a Linda program process or on the types of cpu processors available. Strategies, such as least-recently-used, most-recently-used, prioritized scheduling, and shortest-remaining-time scheduling can be implemented within the centralized Linda-LAN Manager; however, future research must be directed towards these strategies to determine their viability.

Once a Linda-LAN Processor has been selected, the L-Man informs the appropriate L-Com. The L-Com, in turn, informs its local L-Kernel of its selection. The L-Kernel initiates (spawns) another executable version of the Linda program onto the workstation. The Linda program establishes a connection with the local L-Kernel and requests from the TS Manager, via the L-Kernel, for the name of the Linda program process to execute and any applicable process parameters. After the necessary information has been obtained from TS, the Linda program process begins execution.

Overall, the Linda-LAN control sub-system centralizes the management activities of the system and gives control over the environment to a designated system administrator. The sub-system establishes monitoring capabilities for the entire system as well as providing for the collection of global system statistics. The control sub-system also relieves the data sub-system from performing any management activities. In addition, intelligent global scheduling of programs, processes, and processors are performed. Furthermore, the division of the system into a control sub-system and data sub-system has focused research efforts and has provided a more simplistic view of the system.

IV. Substantiating the Scalability and Stability of Linda-LAN

The efficiency and effectiveness of the Linda-LAN environment is influenced by both the execution of the control and data sub-systems. The results provided within this section are only applicable to the control sub-system of Linda-LAN. Results from the data sub-system will be addressed in future publications. While many goals are desired of Linda-LAN, the goals attributable to the control sub-system are for the provision of the execution characteristics of scalability and stability. Without achieving these desired goals, the Linda-LAN environment would be considered an ineffective and unusable solution for parallel processing. The execution of the system would not be able to respond with changes in the number of utilized environment processors nor with changes in the number of executed program processes. In addition, a repeatedly executed program would be unstable, undependable and unpredictable. Linda-LAN would not be deemed a viable low cost parallel processing solution which operates within a conceptually simple parallel programming framework.

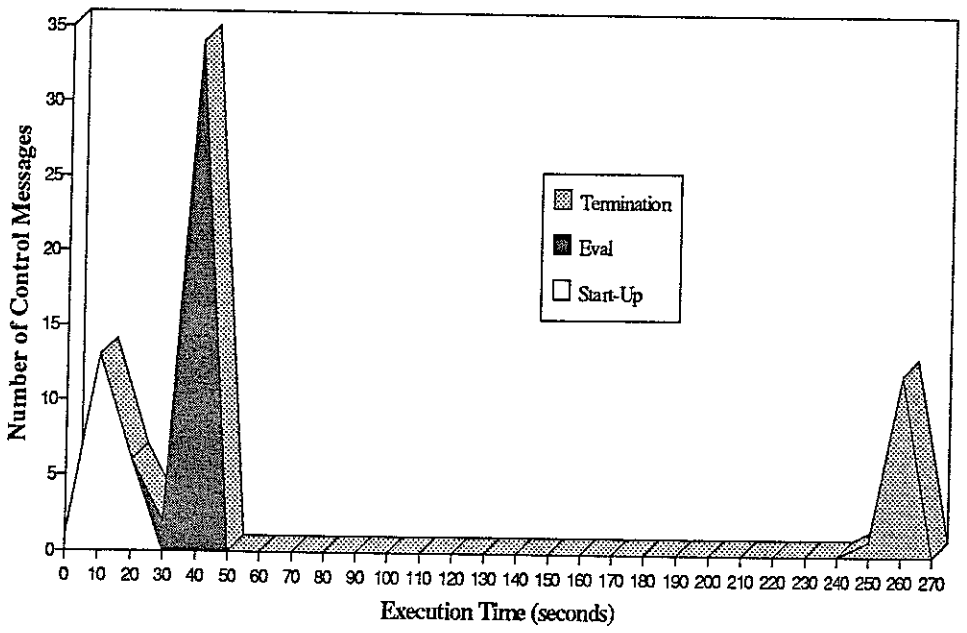

Figure 4. Program execution characteristics on the control sub-system of Linda-LAN.

We hypothesize that the desired goals of scalability and stability are attained within the control sub-system of Linda-LAN by effectively and transparently controlling the environment processors and the allocation of instantiated program processes to processors. Linda-LAN allows for the alteration in the number of utilized environment processors via system tables located at the Linda-LAN Manager. The modification of the system tables are transparent to the users of Linda-LAN. Furthermore, the Linda-LAN system allows a program to vary the number of processes utilized by the program based on its input data. Since the control sub-system manages the environment processors utilized and the mapping of the program processes to processors, proper scalability and stability should be achievable.

In order for the control sub-system of Linda-LAN to be considered stable, the repeated execution of a program across Linda-LAN must generate the same execution characteristics on the control sub-system. These execution characteristics can be seen in Figure 4. The execution characteristics are represented by the various control messages produced by an executing program. As seen in Figure 4, three control message types can occur during a program's execution[4]. Start-up messages refer to the control sub-system messages which occur after a user submits a request for program execution and before the program is actually released into the system. The eval messages refer to the control sub-system messages which occur during the instantiation of a new program process. Finally, the termination messages refer to the control sub-system messages which occur during program termination.

Figure 5 illustrates the multiple executions of the same C-Linda program across the Linda-LAN system under identical conditions[5]. As can be seen, the program repeatedly executed with relatively the same execution pattern across the control sub-system. Therefore, stability is shown to exist within the control sub-system. Although Figure 5 represents the execution of one C-Linda program, the results are indicative for all other C-Linda programs tested. Result parallel and agenda parallel programs as described by Carriero and Gelernter in How to Write Parallel Programs were tested. Predictable results for each type of program were achieved. The reasons for obtaining stability within the control sub-system are directly attributable to the centralized scheduling of program processes to processors. The data collected from the mapping of processes to processors showed identical distribution patterns by the Linda-LAN Manager component of the control sub-system. Exactly the same processors were selected for each instantiated process. The control sub-system behaved identically for each execution. The

[4] CPU load utilization messages are a fourth message type which can occur. Utilization messages are constantly issued at approximately the same time intervals by the L-Man during system execution. These messages have been ignored for simplicity.

[5] Truly identical conditions on the network and the utilized workstations can not be achieved; however, steps were taken to ensure idle network and workstation conditions existed. In addition, identical program input data was used so that the number of instantiated processes for each execution was exactly the same.

variability in Figure 5 is attributable to conditions outside of the control sub-system, such as the data sub-system, network communications and the process scheduling of the UNIX operating system.

Scalability within the control sub-system is also attainable by the Linda-LAN environment. Figure 6 illustrates the execution of the same C-Linda program as the number of environment processors are increased. Figure 7 illustrates the execution of the same C-Linda program as the number of program processes are increased. Both figures demonstrate the scalability of the control sub-system. As the number of environment processors or program processes increases, so do the number of control messages. Specifically, the number of start-up and termination messages increase linearly as the number of environment processors increases. This fact is shown in Figure 8. As the number of processors increase, the L-Man must inform each L-Com of the instantiation of a new program for execution. Since there exists one L-Com per Linda-LAN Processor, a linear increase in the number of start-up messages is expected. The same argument is true for the linear increase in the number of termination messages. Looking back at Figure 6, the execution time of the program shortens as the number of processors are added. The control sub-system is not adding undue overhead to the system as the number of processors are increased. As expected, the program execution time is also improving as the number of processors are increased.

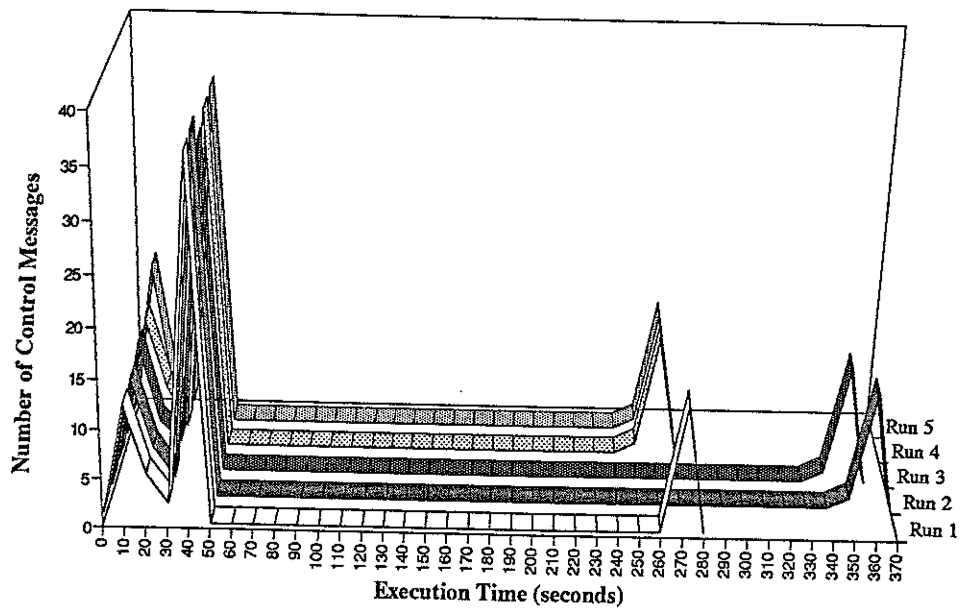

Figure 5. Execution patterns of a program repeatedly executed under identical conditions.

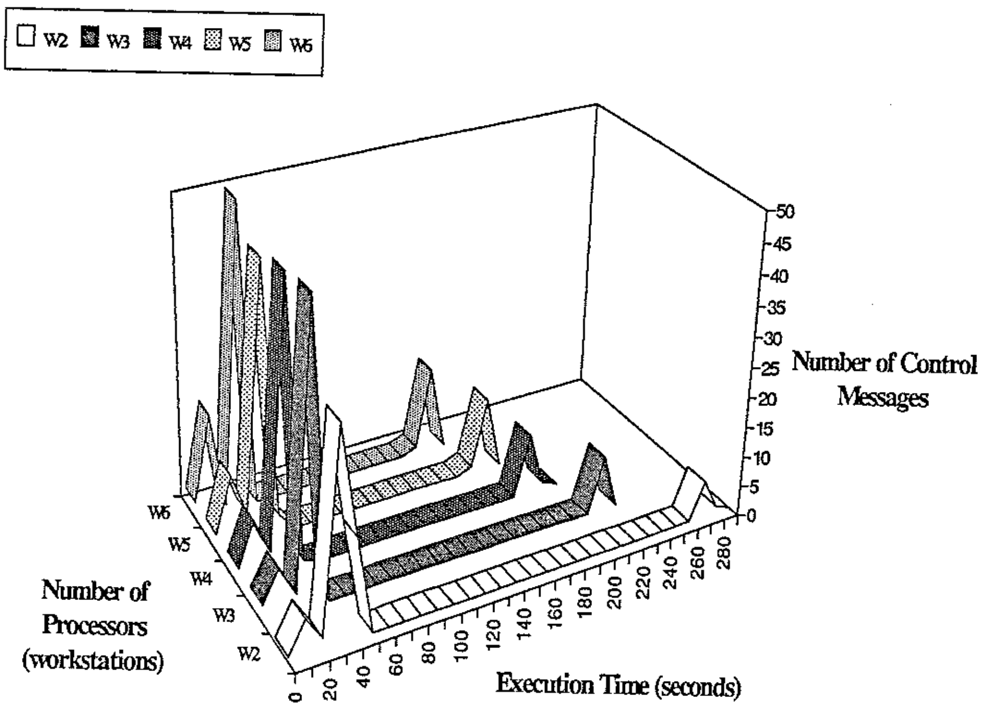

Figure 6. Repeated execution of a program as the number of environment processors are varied.

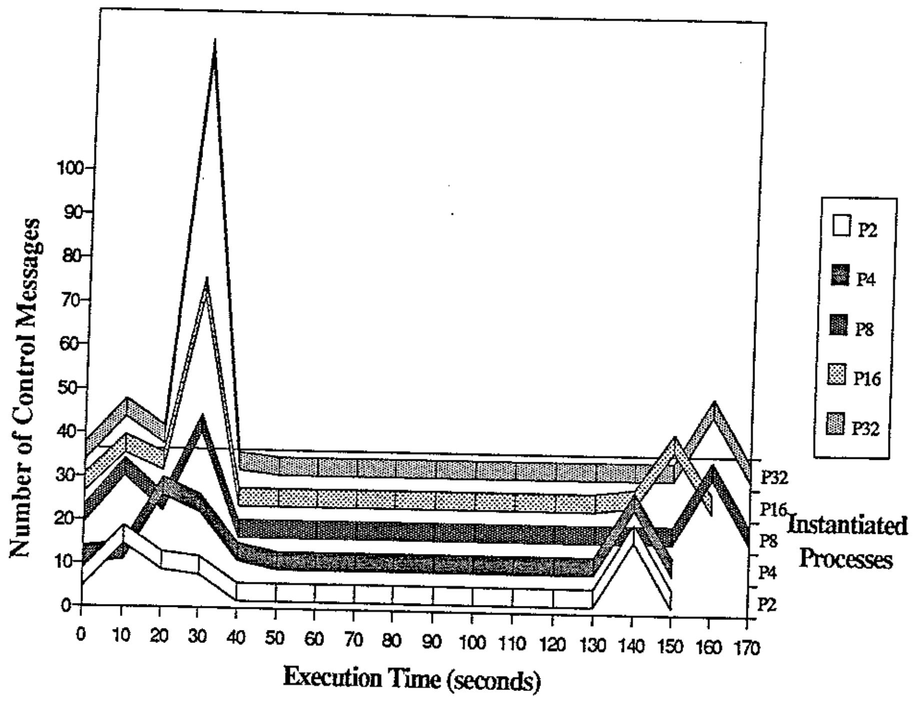

Figure 7. Repeated execution of a program as the number of instantiated processes are varied.

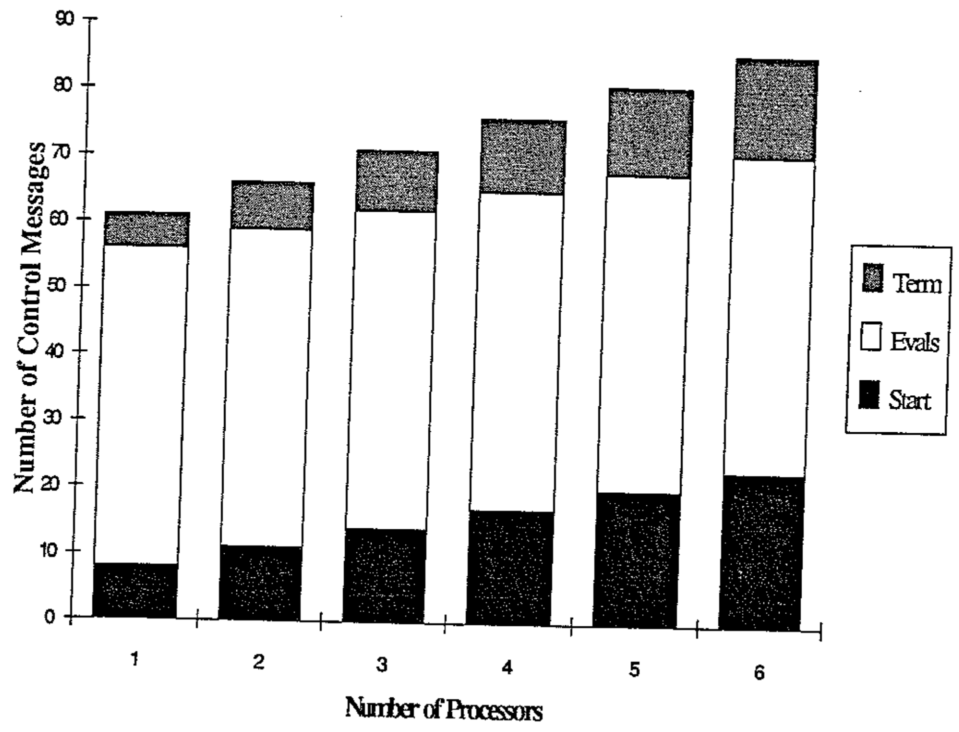

Figure 8. Scalability of control sub-system as the number of processors increases.

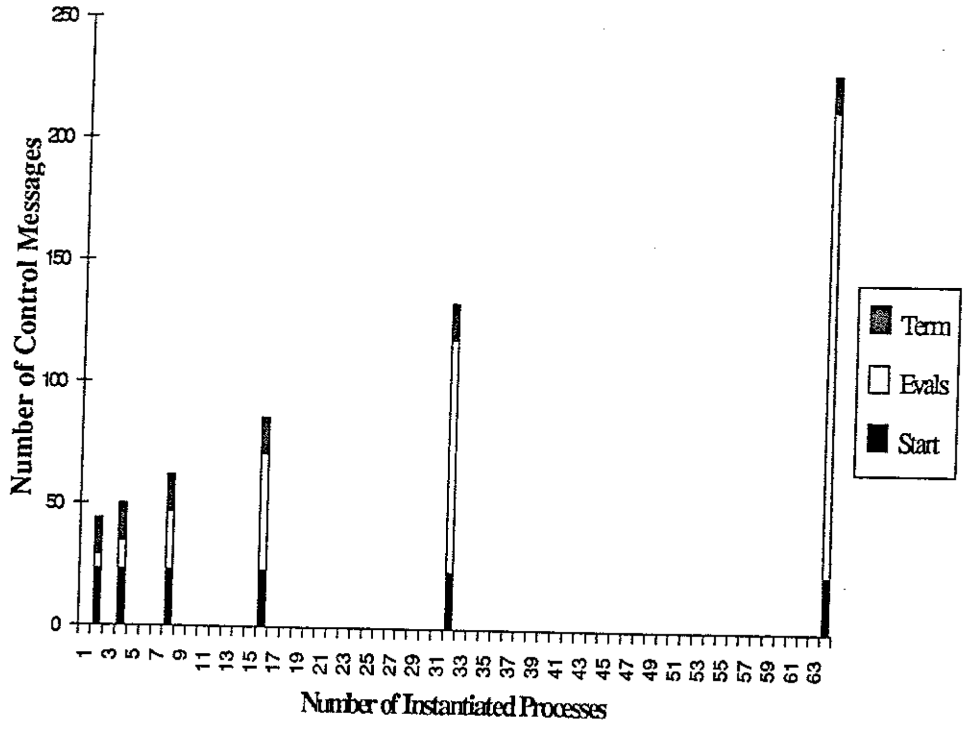

Figure 9. Scalability of control sub-system as the number of instantiated processes increases.

Figure 9 also demonstrates a linear increase in the number of control messages as the number of program processes increase. Once again, the centralized scheduling by the control sub-system is attributable to the scalability of the control sub-system as the number of instantiated processes increases. For each process instantiated in Figure 7, the number of control messages increases. Specifically, the eval messages which deal with process to processor allocation increase. Looking back at Figure 7, a different effect is seen on program execution time. As the number of instantiated processes increase, the program execution time lengthens. This fact has been linked to the overhead generated by the data sub-system's instantiation of a process on a Linda-LAN processor. The *forking* mechanism provided by the UNIX operating system and the retrieval of required start-up information from the Tuple Server are time consuming activities. On the other hand, the control sub-system's overhead in process instantiation is quite small in comparison. Although a linear relationship is seen in the scalability of the control sub-system as the number of program processes increase, the scalability of the environment in its entirety may not hold.

V. Concluding Remarks

With the increasing user demand for greater computational computing power, the Linda-LAN environment is a conceptually attractive configuration which provides

- a true parallel computational environment at an economical price,
- greater computing power than conventional uni-processor workstations,
- wider accessibility to a parallel environment,
- utilization of idle network cpu cycles, and
- centralized control of all management activities and system resources.

In addition, the Linda paradigm provides the system with a portable and easy to use operational framework. The Linda language gives the programmer the ability to design, develop and implement parallel programs in a comprehensible and structured manner. By partitioning the system into the control and data sub-systems, research efforts have become more focused and the implementation of the system has been simplified. In addition, a more simplistic view of the environment is created.

The control sub-system of Linda-LAN centralizes system management activities. Global scheduling of programs, processes, and processors is provided, as well as global system monitoring and the collection of system statistics. The control sub-system empowers a designated administrator with control over the environment. The control sub-system alleviates the data sub-system from performing any management activities. Furthermore, scalability and stability have been shown to exist for the control sub-system. A linear relationship exists between the number of control sub-system messages and the number of program processes instantiated, as well as the number of environment processors used. Overall, the current versions of the Linda-LAN system offer a viable parallel processing environment within a conceptually simple parallel programming framework that utilizes a local area network with low cost personal computers. In addition, the Linda-LAN system is well controlled and managed via a centralized control sub-system.

References

[ARTHU91]	J. D. Arthur, G. Cline and K. Landry, "Linda-LAN: A Distributed Parallel Processing Environment Based Upon The Linda Paradigm," *A Research Proposal*, Computer Science Department, Virginia PolyTechnic Institute and State University.
[BERND89]	D. Berndt, "C-Linda reference Manual (DRAFT) Beta Version 2.0," *Scientific Research Associates*, January 1989.
[BJORN88]	R. Bjornson, N. Carriero, D. Gelernter and J. Leichter, "Linda, the Portable Parallel," *Research Report YALE/DCS/RR-520*, January 1988.
[BJORN89]	R. Bjornson, N. Carriero, and D. Gelernter, "The Implementation and Performance of Hypercube Linda," *Research Report YALEU/DCS/RR-690*, March 1989.
[CARRI87]	N. Carriero, "Implementation of Tuple Space Machines," *Research Report YALEU/DCS/RR-567* (PhD thesis), December 1987.
[CARRI88]	N. Carriero and D. Gelernter, "Applications Experience with Linda," *Proc. ACM Symp. Parallel Programming*, July 1988.
[CARRI89a]	N. Carriero and D. Gelernter, "Coordination Languages and their Significance," *Yale Tech Report*, YALEU/DCS/RR-716, July 1989.
[CARRI89b]	N. Carriero and D. Gelernter, "Linda in Context," *Communications of the ACM*, Vol. 32, No. 4, April 1989.
[CARRI90]	N. Carriero and D. Gelernter, *How to Write Parallel Programs*. MIT Press, Cambridge, 1990.
[DAVID89]	C. Davidson, "Technical Correspondence on *Linda in Context*,", Communications of the ACM, Vol. 32, No. 10, October 89, pp.1249-1252.
[GELER85a]	D. Gelernter, "Generative Communication in Linda," ACM *Transactions on Programming Languages and Systems*, Vol. 7, No. 1, January 1985, Pages 80-112.

[GELER85b]	D. Gelernter, N. Carriero, S. Chandran and S. Chang, "Parallel Programming in Linda," *Proceedings of the 1985 International Conference on Parallel Processing*, August 1985, pp.255-263.
[GELER90]	D. Gelernter, "Ada-Linda: Motivation, Informal Description and Examples," *Yale Tech Report*.
[LELER88]	Wm. Leler, "PIX, the latest NeWS," *Cogent Technical Report*, Cogent Research, November 1988.
[LELER90]	Wm. Leler, "Linda Meets Unix," *IEEE Computer*, February 1990, pp.43 - 54.
[SCHUM91]	C. Schumann, K. Landry and J. D. Arthur, "Comparison of Unix Communication Facilities Used in Linda," *Proceedings of the 1991 Virginia Computer Users Conference*.
[WHITE88]	R. Whiteside and J. Leichter, "Using Linda for Supercomputing On a Local Area Network," in *Proc. Supercomputing '88*, November 1988.
[ZENIT90]	S. E. Zenith, "Linda Coordination Language; subsystem kernel architecture (on transputers)," *Research Report YALEU/DCS/RR-794*, May 1990.