

# Survey on Distributed Computing Networks - Networks of Workstations

By Bhavana Nagendra

---

**Abstract:** Distributed system is a programming infrastructure which allows the use of a collection of workstations as a single integrated system. Distributed Computing offers advantages for improving availability and reliability through replication, performance through parallelism and in addition flexibility, expansion and scalability of resources. This paper is a survey on the ongoing research in this area.

---

## Table of Contents

1. [Introduction](#)
2. [Distributed Computing Networks](#)
  1. [Changing Trends in Computing](#)
  2. [Distributed System Architecture](#)
  3. [Massively Parallel Processors \(MPPs\)](#)
  4. [Networks of Workstations \(NOWs\)](#)
    1. [Challenges for Networks of Workstations](#)
    2. [What is new about NOWs ?](#)
    3. [Opportunities for NOW](#)
3. [Message passing Issues and Distributed System Testing Software](#)
  1. [Communication Paradigms](#)
    1. [Remote Procedure Call](#)
    2. [Distributed Shared Memory](#)
  2. [Software Packages](#)
    1. [Parallel Virtual Machine \(PVM\)](#)
    2. [The p4 System](#)
    3. [Express](#)
    4. [Message Passing Interface \(MPI\)](#)
    5. [The Linda System](#)
4. [Advances in Networks of Workstations \(NOWs\)](#)
  1. [Low over-head communication](#)
  2. [GLUnix : A global layer Unix](#)

3. [xFS : Serverless network file service](#)
  5. [Massive Parallelism in WANs](#)
    1. [Concepts](#)
    2. [Managing Parallelism in WANs](#)
    3. [Implementation Issue](#)
  6. [Conclusion](#)
  7. [Appendix - Annotated bibliography](#)
- 

# 1. Distributed Computing Networks - Introduction

Distributed system is a programming infrastructure which allows the use of a collection of workstations as a single integrated system. The ultimate aim is to hide the hideousness of scattered resources across a number of hosts. A distributed system is composed of a number of autonomous processors, storage devices and databases which interactively co-operate in order to achieve a common goal. The communication network is used for information exchange, interaction and co-ordination among the various processes. Some systems are a library of routines intended at communication between hosts, while other systems link the various hosts tighter such that the application sees only one system. These two cases are called *loosely and tightly coupled* distributed systems respectively.

In the 1970s, the CPUs were not very fast. Data used to be accessed from the local disk which was quite slow but computation wasn't very fast either. The computation wasn't I/O intensive in nature and hence the slow I/O devices sufficed. With the advancement in Very Large Scale Integrated (VLSI) circuits, the CPUs of the present day computers are extremely fast, operating over 100MHz, performing many orders of MIPS (millions of instructions per second). This requires fast and efficient I/O devices. But the I/O speeds cannot match that of the CPU. This brought up a clear cut division in the usage of computers, depending upon the intensity of the task at hand. Jobs of modest size which required fast and predictable interactive performance are run on smaller systems like personal computers (PCs) and on workstations. Mainframes and minicomputers are preferred for jobs which required huge amounts of memory and disk space for demanding sequential and parallel applications.

Workstations and PCs give better performance due to the effect of volume manufacturing on computer price to performance ratios and are preferred. But sometimes task at hand may be bigger than will feasibly run on a workstation in which case a supercomputer has to be used. The Networks of Workstations (NOWs) caters to both the needs mentioned above and will be discussed in this survey paper.

The second section of this paper deals with the [Changing Trends in Computing](#), of [Massively Parallel Processors](#), challenges and opportunities of [Network of Workstations \(NOWs\)](#). The third section speaks of [Message Passing Issues and Distributed System Testing Software](#) currently used in the research of Distributed Systems. The recent [Advances in Networks of Workstations \(NOWs\)](#) is dealt with in the fourth section which also covers work done at the Berkeley University. The fifth section addresses the issues in [Managing Massive Parallelism in Wide area networks \(WANs\)](#). The [Appendix](#) contains an

annotated bibliography of the articles, web sites and books referred to in writing this survey paper.

[Back to Table of Contents](#)

---

## 2. Distributed Computing Networks

Distributed Computing Networks is a novel idea in networking which aims at using the computing power of the workstations connected by a network, when a large task is at hand which requires more computational power than what a single workstation can provide. Several distributed systems have a lot of similarities in that they are all implemented using either of the two communication technologies : *remote procedure call (RPC)* or *distributed shared memory* .

It is an emerging field in networking with a lot still left to be achieved. There is no general agreement as to the design of distributed systems, how they should be structured managed and organized, even though a number of loosely coupled distributed systems are available commercially.

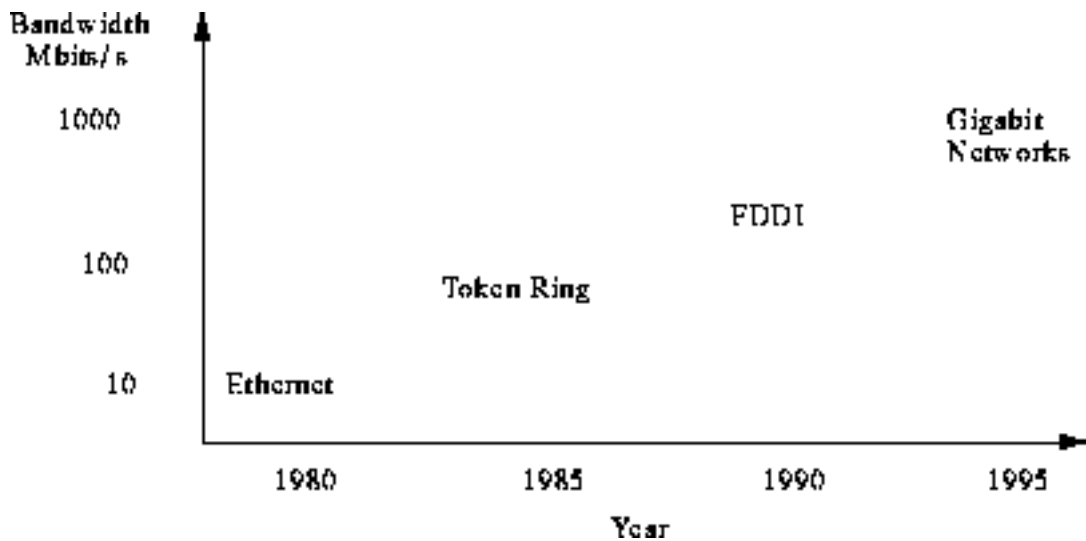
[Back to Table of Contents](#)

### 2.1 Changing Trends in Computing

Ongoing technological convergence of LANs and massively parallel processor interconnections will allow NOWs to replace the entire computer food chain. Since building computing systems out of small, mass produced computers is clearly attractive, the network that supports them has to be extremely fast. As the networking technology advanced the bandwidth supported increased. The following is a list of some of the networking technologies used :

(From PVM, Parallel Virtual Machines, by Al Geist et al)

- **Ethernet** is a popular local area packet switched network technology.
- **FDDI - Fiber Distributed data Interface** is a 100-Mbit/sec token-passing ring that uses optical fiber for transmission between stations and has dual counter-rotating rings to provide redundant data paths for reliability.
- **HiPPI - High-Performance Parallel Interface** is a copper-based data communications standard capable of transferring data at 800 Mbit/sec over 32 parallel lines or 1.6 Gbit/sec over 64 parallel lines. HiPPI is a point-to-point channel that does not support multidrop configurations.
- **SONET - Synchronous Optical Network** is a series of optical signals that are multiples of a basic signal rate of 51.84 Mbit/sec called OC-1 which is the Optical carrier level 1.
- **ATM** is the technique for transport, multiplexing and switching that provides a high degree of flexibility required by B-ISDN. This high speed network is a good choice of technology for connecting the distributed computing networks and the research is ongoing in various universities.



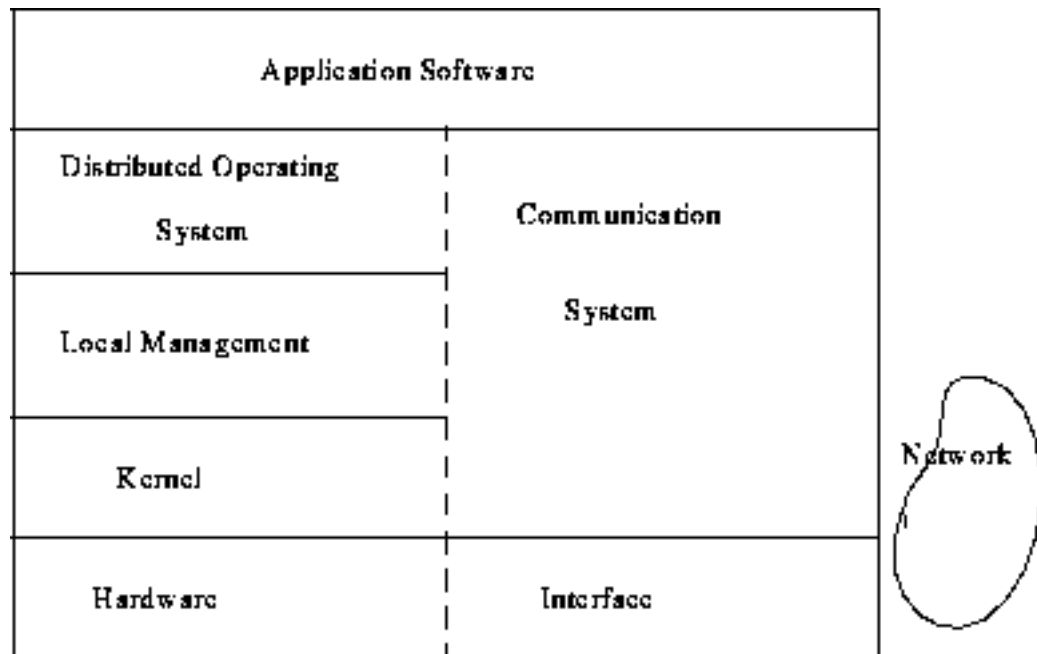
**Fig 1 : Networking Technologies**

(From PVM, Parallel Virtual Machines, by Al Geist et al)

[Back to Table of Contents](#)

## 2.2 Distributed System Architecture

The simplest structure of a distributed system is that of a collection of physically distributed computer nodes, fully interconnected by a communications network. Each node may be connected to several peripheral devices and can independently perform computations. The network is responsible for supporting information exchange. The software of the computer can be considered to be layered.



## Fig 2 : Layered software structure in a computer node

(From Network and Distributed Systems Management by Morris Sloman)

**Application Software** consists of the application functions decomposed into distributable processing units. In addition, the data may be distributed by partitioning or by replication. The communication and synchronization is by interprocess communication (IPC).

**Distributed Operating System (DOS)** provides local access and management of local and remote resources. Transperency calls for naming and access conventions for the services so that they can be accessed without any knowledge of the physical location. You ask for **a** printer and not **the** printer.

**Local management** provides the usual local resource management, with the conventional kernel and local IPC.

**Communication System** is responsible for delivering messages to/ from the computer node. There are several distributed operating systems but there is no general concensus or agreement to form a standard.

The Mach kernel is a specialized kernel of a distributed OS to support both loosely and tightly coupled multiprocessors. Other examples of kernels and DOS facilities include V-kernel - testbed for lightweight processes and IPC experiments, Amoeba - threads and objects with capabilities for protection, Chorus - threads, clusters and ports to provide UNIX-like distributed procesing and the Advanced Networks Systems Architecture (ANSA platform - an object oriented platform for open distributed processing).

[Back to Table of Contents](#)

## 2.3 Massively Parallel Processors (MPPs)

First step to harnessing the available resources in a distributed computing setup was taken when machines were constructed as a collection of workstations-class nodes which were connected by a dedicated, low latency network. The MPPs like T3D, Paragon and CM-5 exploit the latest technologies like a fast microprocessor; sophesticated cache; large, inexpensive Dynamic Random Access Memory (DRAM). Microprocessor performance has improved 50 to 100 percent per year. DRAM memory and disk capabilities have quadrapuled roughly every three years. Clearly this technology today is the leading drive in supercomputing, but the MPPs have had limited success. Pondering over the strengths and weaknesses of the MPPs gives a very good feel of what to expect from the Network of Workstations (NOWs), the constraints under which it has to work and the problems that have to be anticipated.

The MPPs had several weaknesses and are summed up here :

One important aspect of any technology is the speed with which it arrives to the market. This is specified in the literature as the engineering time lag. If too long a time is taken in building the sophisticated processor, technology would move on and this subtracts from the performance measure.

Due to low volume manufacturing, the huge systems are expensive when compared to a desktop computer like a workstation for interactive computation. Hence the cost/ performance ratio is high and this value can be interpreted as a degradation in performance. A possible solution would be to repackage the chips on the desktop motherboard, to improve system density and potentially reduce part costs. To

prove the point, several processors like Sparcstation-10s with one, two and four processors, SparcCenter-1000 and 2000 servers that contain upto 8 to 20 processors were compared. Also the comparison of 128-node MPP and either the Thinking machines CM-5 or the Meiko CS-2 with the cost-effective machines showed that these machines cost twice as much. This is because, the huge engineering effort of the latter machines have to be borne only by a small volume of machines.

A side effect of large scale connection of computers is the need to change the Operating Systems (OS) and other commodity software. There is a need to invest a lot in OS development to the same extent as microprocessor design and the applications which depend directly on the OS interface. There are two ways in which the OS mechanism is achieved

- custom message-passing kernels and applications had to be modified for each machine.
- full Unix on each node.

The latter mode aides in eliminating typical devices (local disk, serial port and ethernet) and has forced a split from the community OS development path.

Finally, (something which is not really recognized), the niche occupied is too narrow as it aims at high performance in certain applications where rewriting is feasible and tractable, but not versatile enough to provide fast interactive performance or high performance in day-to-day needs like file transfers.

However there are several advantages of the MPPs from which the technique of NOWs can borrow :

As a collection of workstation-class computers, it provides two key features. Current MPP systems provide a dedicated, high bandwidth network that scales with the number of processors and hence the communication performance is excellant. Though, the time spent in the processor preparing to send or receive a message called overhead is unavailable for computation as its the CPU time, the time spent in actual network hardware called latency can overlap with computation. Hence network performance is quite good as delays are reduced considerably.

The communication performance derives from several factors. The routing components are fast, as single chip switches are used with bee-line routing to decrease the delays. Another factor which abets the communication performance is the proximity of the network interface to the processor memory bus, rather than on the standard I/O bus. In MPPs, overheads of 1000 processor cycles are incurred. Using lean communication layers especially [Active messages](#), this overhead can be reduced by an order of magnitude. Although this is better than typical LAN overheads by an order of magnitude, it is still quite substantial when compared to CM-5.

The second, important achievement of MPPs is the Global system view it provides. As the name indicates, there is a single parallel program which runs on several nodes as a single entity, rather than as an arbitrary collection of processes. So the entire collection is controlled together and there is a sense of transparency among the processors as the files are uniformly accessible across all the nodes. An aspect of utmost importance is the fact that the processes are scheduled as a single unit, and the elements of a parallel program in reality run simultaneously.

Valuable lessons are learnt from the MPPs. It is not enough to exploit the commodity components, but we need to exploit full desktop building block including system and applications. This is because the the NOWs have to cater to a wide variety of users, those running computationally intensive jobs and also interactive users.

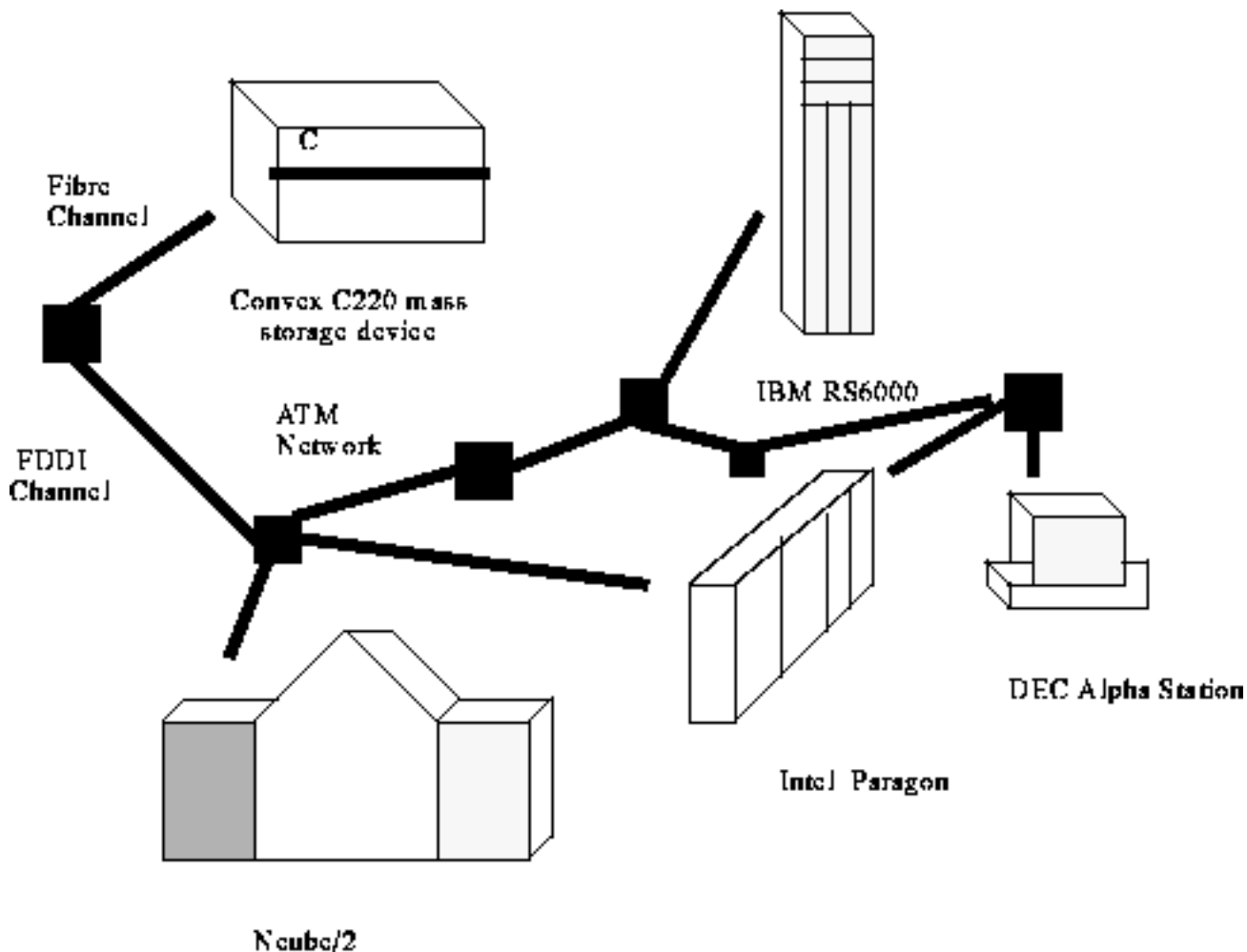
[Back to Table of Contents](#)

## 2.4. Networks of Workstations (NOWs)

Networks of Workstations (NOWs) are poised to become the primary computing infrastructure for scientific work and industry and may dramatically improve virtual memory and file system performance. NOW system offers more than a collection of workstations on a fast network. The file storage system is scalable, cheap and highly available due the fact that the network is highly transparent. This is one of the features dicussed in the MPPs, in the above section. The driving force for the development of NOWs is to use of the workstations for both interactive use and also for demanding jobs requiring a lot of computational power. The multiple CPUs can be used for parallel computing. As the OS view of the machines in the network has to be uniform, system software has to be given a face lift. This in turn helps in making progress in the traditional system functions like virtual memory and file systems as well as high performance parallel computing.

In order for the NOWs to be a success, effective communication is mandatory. This calls for efficient hardware and software. Apart from making use of the commodity components, we also need to exploit full desktop building block including system and applications. So, there has to be a global coordination of multiple workstation OSs. The key here is that the computers hooked to a network in a distributed system can vary from a desktop PC to a Cray Y-MP. The network file systems should be scalable and easily and quickly accessible.

NOW's can give excellant performance as far as a dedicated system goes. But can they run large applications with the performance of a dedicated machine and run small programs with the interactivity of a dedicated machine ?



**Fig 3 : High Performance Distributed Computing Network**

The units in a distributed environment could be as varied as a Paragon, DEC stations, Sun workstations, IBM RS6000, Ncube/2, Convex C3880, TMC CM-5 Thinking Machines, SGI workstations as shown in the fig 3. It is the network, the OS and the file system which together make the labyrinth of connections work.

## 2.4.1 Challenges for Networks of Workstations

The key issues involved in the successful performance of the NOWs is the ability to deliver the interactive performance of a dedicated workstation and at the same time provide the aggregate resources of the network for demanding sequential and parallel programs. The unused resources in the network have to be allocated to the jobs which need more computing power but this should not affect the local host. The resource allocation policy has to take care of allocating the DRAMs for memory intensive programs, disks for I/O bound programs and CPUs for CPU bound programs. The communication overhead has to be minimized and communication has to overlap with computation to the best possible extent.

[Back to Table of Contents](#)

## 2.4.2 What is new about NOWs ?

Parallel computing and the idea of harnessing the idle resources for a task needing more computing power than what a single workstation can provide, on a cluster of workstations is quite an old concept. The innovative feature of this new concept NOW or workstation cluster computing or hypercomputing, is the fact that the line between the processor and storage technologies like DEC Alpha, Cray 2S, Cray Y-MP, Convex C220 and system concepts like OS, disks, etc. is evaporating.

Due to the use of switched networks like ATM using low overhead communication protocols, communication over the LAN has increased tremendously and a number of processors can be used in parallel, since the bandwidth can be scaled.

The current workstations are extremely powerful. They provide about 1/3 the performance of a Cray C90 processor. In addition to processor performance, large disk capacity and memory is offered by the workstations which are idle. The key is the network hardware and software.

Even though processors are growing faster, the total performance will not be any better unless the I/O performance increases. NOWs offer a better alternative as large amounts of memory potentially exists in the network and this can be accessed more easily than local-disk storage. Also the disks of other workstations can be used very much like a RAID (Redundant array of inexpensive disks) The key technology again is the network.

Hence the advantages of NOW are not only for parallel computing but focus on making the system resources such as memory, disks and processors, available to the program in abundance. Hence the bottom line is that large systems have to be built out of high-volume hardware and software components in order to raise the level of work and utilization. So the high volume components should aim at being good building blocks for such large systems.

[Back to Table of Contents](#)

---

## 2.4.3 Opportunities for NOWs

The advantages of NOW are several, when implemented on a building-wide scale of hundreds of machines. The pool of resources in a NOW include memory, disks and processors.

Due to speedy network communication as a result of switched networks, which provide ample bandwidth, the NOW's aggregate DRAM can be used as a giant cache for disks. This wasn't practical in the past on Ethernet because that would consume a lot of the shared-media's bandwidth. Also even on an idle Ethernet, fetching data was only marginally faster than a local disk access.

By using the idle DRAM on a NOW, the number of disk accesses is dramatically reduced, alleviating the I/O bottleneck and improving user visible performance. There are two applications of this idea : virtual memory and file caching. The transfer of 8-Kbyte of data on a DEC AXP 3000/400 from both remote memory and disk is very expensive on Ethernet (6,250 micro-sec) but is approximately 15 times faster (400 micro-sec) over 155-Mbps ATM.

The concept of virtual memory which became impractical due to the performance gap between processor

and disk can be revitalized by the network RAM. The idea is the same, except that instead of paging from the disk, now paging is done across the network using the high-bandwidth, low latency networks and system software that can recognize when machines are idle. Simulations show that programs run 10 to 30 percent slower using network RAM than if the program fitted entirely in local DRAM. However, network RAM is 5 to 10 times faster than thrashing to disk.

File system performance can be improved by cooperatively managing the file caches on each client workstation. The benefits are twofold. More than one client uses files like executables and font files. On a cache miss, the system can fetch files from another client's memory, instead of going to the server's disk. Active clients can effectively increase the size of their cache by using memory of idle clients. The number of cache misses in cooperative caching is 8, with 42 workstations in a network, with a capacity of 16-Mbytes/workstation, and 128-Mbytes/server and with the client server is double at 16 misses. The read response time is quicker using cooperative caching at 1.6 milli-sec. (see reference [6])

Redundant arrays of workstation disks (RAID) use arrays of small disks and can be built in software rather than hardware, by redundantly writing data across an array of disks. RAID systems deliver higher bandwidth, capacity, and higher availability than a single large disk can possibly achieve. This is effective since the fast network needed for network RAM and cooperative file caching can also serve as the I/O backplane of a RAID system. Each workstation can have disk bandwidth only by the network link bandwidth. Since there is no central host, the availability is not subject to failure of the host. Any workstation can play host -- decentralized control of RAID. Hence the disadvantages of a hardware RAID like higher cost per byte of disk storage due to hardware needed to manage the RAID and connecting to a host which becomes a performance and availability bottleneck is overcome.

NOWs support a high degree of parallel computing in an everyday computing scenario. They provide processors with high sustained floating-point performance capability, networks with bandwidth that scales with the number of processors, parallel file I/O and low overhead communication and hence support parallel computing.

Sharing a single Ethernet limits the performance as the number of processors increases. An Ethernet utilization of 20% is considered as heavy load. If a cluster of 40 sparc stations are used, a physical bandwidth of 52.88 % is made use of. Hence the communication bandwidth limits the scalability of this approach. Using a high bandwidth networks using ATM say, drastically improves the performance of the transport phase, improving overall performance by an order of magnitude. But the bandwidth of the sequential file system is still a limitation. Adding a parallel file system reduces the execution time and replacing parallel virtual machine with a low overhead, low latency communication system further reduces the execution time by an order of magnitude, so that the NOWs can compete with the C-90 at a fraction of the cost. The floating point performance of a workstation is better than that of a Paragon, as the former's floating point performance exceeds that of a singler node in an MPP.

The following points have stemmed from the above discussion of the NOWs.

- make use of the DRAM available on the network and avoid going to the disk.
- try using all the disks on the network to speed up the remaining I/O, if higher speed is required for the application.
- parallelize the computational portion if high speed is required.

This layered approach seems better than completely rewriting the program as required by MPP users.

## 3. Message Passing Issues and Distributed System Testing Software

### 3.1 Communication Paradigms

Message passing is a communication technique in which communication between applications is in the form of messages. It is worthwhile to note that shared memory and message passing are not identical. Lauer and Needham have proved that shared memory and message passing were semantically equivalent. Also they showed that on a **single host**, shared memory and message passing could, in most cases, perform equally well. This has been a source of confusion that the two communication paradigms are equivalent. But the two paradigms are equal only within a single system.

#### 3.1.1 Remote Procedure Call (RPC)

Remote Procedure Call (RPC) is an important programming interface for distributed networks, which supports heterogeneous computing environments. RPC makes all communication across a network appear like simple procedure calls and so the network is hidden behind the call. Nelson and Birrell's version of RPC requires the existence of servers, and each of which support a set of procedures. A procedure call is made by the client when its needed to access a remote host. From the clients point of view its a procedure call to the server. The electronics of the remote call is taken care of by a combination of a message protocol and programming entry points called stubs. To make a call, the client calls a local stub procedure and it encapsulates its arguments and is passed onto the server. When the server receives the message it makes the actual procedure call and returns the value back to the local stub which inturn is returned to the application.

The process call can be made sequentially or parallelly. The parallel RPCs have the advantage of taking a much shorter time than its counterpart. In parallel RPCs the calls to the stub are made simultaneously and the stub responds to the request very quickly.

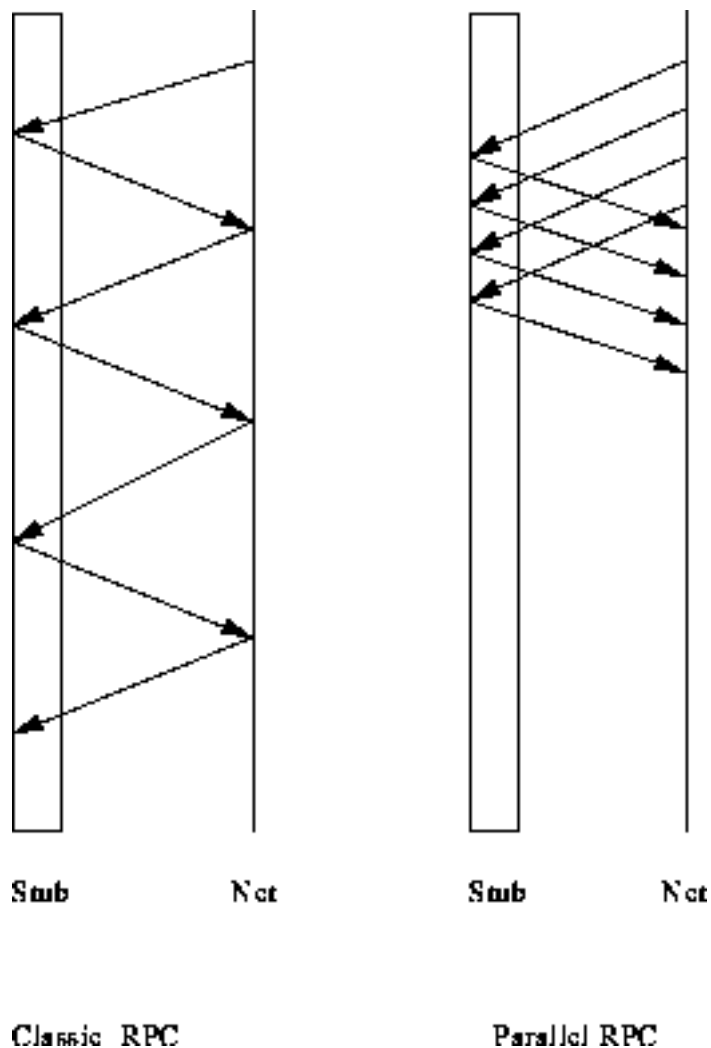


Fig 4 : Improving Performance with parallel RPC

## 3.1.2 Distributed Shared Memory

The concept of Distributed Shared Memory is that there exists a common address space which all the computers in the network share. A change in the data was seen by all the hosts on the network. Writing words in memory substituted the act of sending data across the network. Distributed memory was developed at the time when memories were not as fast as they are in the present day. But as memories became faster, the time required to send a message across the network became expensive and no longer feasible as the delays incurred meant copying, which is never efficient and should always be avoided in any programming paradigm.

Distributed virtual memory was used instead to cope up with the problem, wherein applications share selected portions of a large virtual memory maintained by the network and the broken up data can be retrieved selectively pretty much like virtual memory page retrieval.

The limitation of shared memory is that it is essentially homogeneous which mandates all the hosts to understand the same memory organization and common format of the data.

[Back to Table of Contents](#)

## 3.2 Software Packages

There are various software that have been developed for testing the distributed systems and for providing a combined view of the parallel programs.

### 3.2.1 Parallel Virtual Machine (PVM)

The Parallel Virtual Machine (PVM) software supports heterogeneous network computing by providing a unified framework for developing the parallel programs in an efficient and straightforward manner using the existing hardware. Since PVM handles all the message routing, data conversion, and task scheduling across the network of incompatible computers, it essentially views the distributed network as a single parallel virtual machine.

PVM model accommodates a wide variety of application program structures and is surprisingly simple. The tasks which the user writes access the PVM resources through a library of standard interface routines. The PVM message passing primitives are oriented towards heterogeneous operation, involving typed constructs for buffering and transmission. At any point in the concurrent application execution, the other existing tasks may exercise control.

Due to the all-pervading nature and because of its simplicity and completeness, the PVM has gained recognition as a high performance software programming and testing tool. PVM is a collaborative venture of Oak Ridge National Laboratory, the University of Tennessee, Emory University, and Carnegie Mellon University.

**Message passing library** : Under PVM, a collection of serial, parallel, and vector computers appears as one large distributed-memory computer. A per-user distributed environment must be setup before running PVM applications. A PVM daemon process runs on each of the participating machines and is used to exchange network configuration information. Applications, which can be written in Fortran or C, can be implemented by using the PVM message passing library, which is similar to libraries found on most of the distributed-memory parallel computers.

### 3.2.2 The p4 System

P4 is a library of macros and subroutines which supports both shared-memory model and distributed-memory model and was developed at Argonne National Laboratory for programming a variety of parallel machines. For the shared-memory model of parallel computation, p4 allows construction of monitors by providing a set of useful monitors as well as a set of primitives. For the distributed-memory model, it provides, typed send and receive operations and creates processes according to a text file describing group and process structure.

### 3.2.3 Express

Express is a toolkit which is different from the above described packages in the sense that it individually addresses various aspects of concurrent computation. The toolkit is developed and marketed commercially by ParaSoft Corporation, a company that was started by some members of the Caltech concurrent computation project. The bulk of the Express system is a set of libraries for communication,

I/O, and parallel graphics and computing is based on starting with a sequential version of an application and reaching a parallel version that is tuned for optimal operation, after following a certain life cycle.

### 3.2.4 Message Passing Interface (MPI)

The Message Passing Interface (MPI), unlike the previously described packages, is not intended to be complete and acts as a communication interface layer that would be built on facilities of the underlying hardware and is not a self contained software package. MPI is a community effort to try both syntax and semantics of a core of message-passing library routines that could be efficiently implemented on a wide range of MPPs.

### 3.2.5 The Linda System

Linda is a concurrent programming model that was proposed by the Yale University. Linda is based on an abstraction using which cooperating processes communicate. It is a set of programming language extensions for facilitating parallel programming. It provides a shared memory abstraction for process communication without requiring the underlying hardware to share physical memory.

[Table of Contents>](#)

---

## 4. Advances in Networks of Workstations (NOWs)

The tremendous demand for NOW systems have sparked off a lot of research interests. Many universities are working on this endeavour such as Brown University, University of California at Berkeley, Washington University to name a few. There are three very important features which have to be optimized in the NOW system for efficient functioning and they are processor overhead has to be minimal, a global operating system over the heterogenous systems constituting the NOWs and a decentralized network file system. These features have been taken care of by using high speed networks like ATM LANs and FDDI network interface and Active messages. GLUnix which is a global layer unix is used as a topmost layer on the existing OS of the heterogeneous computers in the network. In order to provide the abstraction of a single, serverless file system, xFS, serverless network file service has been adopted. Low-overhead, low latency communication orientation, quality of the interconnection itself and simplicity of achieving the objective are the interesting features of a successful venture at realizing the opportunities of NOW.

**Active messages** is an asynchronous communication mechanism intended to expose the full hardware flexibility and performance of a heterogenous interconnection networks. The network under active messages is viewed as a pipeline operating at a rate determined by the overhead and latency depending on the message length and the network depth. The header of the message contains the address of a user-level handler which is executed on message arrival with the message body as the argument. The idea is to hijack the message out of the network and into the computation ongoing the processor node, in an effort to reduce the overhead and latency. This is the basic idea of Active messages. (see reference [13] )

[Back to Table of Contents>](#)

---

## 4.1 Low overhead communication

The present day networks are becoming extremely fast with the advent of gigabit networks and high-bandwidth is the buzz word and is a very popular measure of performance of the network. In a distributed system setup, the network latency and processor overhead are two very important terms which need to be taken care of. As noted in section 2.2, processor overhead time of the CPU is unavailable for computation. So it has to be kept as minimal as possible. As shown in the NASA Ames/UCLA chemical tracer model called Gator, majority of the important messages are less than 200 bytes and hence quite sensitive to overheads. (see reference [6] in [Appendix](#))

On Sun Sparcstation-10s, the following data was noted with Ethernet and ATM.

It can be seen from the table that, even though the bandwidth increases by a factor of 8, due to the increase in the overhead and latency time, there is only a 20 percent increase in the overall performance. Hence its mandatory to decrease the processor overhead if the latest technologies have to be effectively used.

Network	Bandwidth	Overhead + Latency
Ethernet	9Mbps	456 micro-sec
ATM	78 Mbps	626 micro-sec

**Table 1 : Comparison between ATM and Ethernet**

The messages have to be passed between nodes without the involvement of the operating system. Hence the OS must map data and control access to the network interface into the user address space. The network interface keeps check on the incoming and outgoing message through a network process ID.

An initial prototype is a cluster of HP9000/735s using an experimental Medusa FDDI network interface that connects to the graphics bus and provides substantial storage in the network interface. Using Active messages provides a processor overhead of 8 micro-sec, with timeout and retry support.

[Back to Table of Contents>](#)

---

## 4.2 GLUnix : A global layer Unix

The idea of globally managing the network resources has been around, but never did reach the stage of being implemented for two reasons, incorporating the global services into the existing array of computers with varied Operating Systems wasn't seen as a simple task and also due to the aesthetics of resource sharing.

In order to implement the GLUnix, complete changes can be made to the existing OS to avoid salvaging or building artifacts, to meet the NOWs requirement and reimplementing the existing working software. This defeats the ideology on which the functioning of NOW is based upon. With respect to hardware, the premise under which we defined the NOWs in section 2.3 is that they are networked systems from existing hardware to increase the performance/ cost ratio. Same holds good for the software and so the existing OS of the host shouldn't be tampered with. The global unix layer has to be above the OS of the system and is OS independent, meaning that the GLUnix can run on DOS, Windows NT and the like. This efficient layering has been enabled by the software fault isolation. By modifying the code and placing a check at every memory store, the user and the implementation can be isolated.

Interactive users are peeved by the sharing of resources, as they fear that their resources might not be available to them and response time might be affected when a demanding job is being run. The supercomputer users perceive NOW suspiciously, fearing that the interactive users will have priority over their CPU demanding applications.

An important concept that comes to the rescue is process migration. When the interactive local user returns, the external processes of an idle machine are migrated. Currently if a demanding job is run on an idle machine all the memory pages and file cache contents are wiped out. Instead the memory contents can be saved, so that the state of the machine can be returned to the interactive user. With ATM bandwidth and a parallel file system, 64 Mbytes of DRAM can be restored in under 4 seconds.

Many parallel programs run as fast as their slowest program in addition to serving for interactive computing and this could be a compromise on the advantages of parallel computing. Thus the processes running on a no longer idle machine have to be migrated to another machine. It turns out that several machines on the network are usually idle and so there is usually a machine to which the evicted process can migrate.

In GLUnix unlike today's multiprocessors, if any node crashes new resources are being added and deleted into the network and even the OS software is upgraded. Hence in this scenario, it is impractical to use an MPP as a sole computing facility for a building. As far as the security issue is considered, the resource sharing will be only within a single administrative security domain.

[Back to Table of Contents>](#)

---

## 4.3 xFS : Serverless network file service

xFS provides an abstraction of a single file system shared among the users logged into a number of client workstations. The disk access is centralized and is effected by a request sent to the server and could result in a bottleneck with many users. At Berkeley, these problem are being addressed by building a completely serverless network file system called xFS. (see reference [6] in [Appendix](#) ). Data and control migrates between clients and this increases reliability of the network as any client can take over on the event of a failure and greatly simplifies load balancing. The client cache is managed as a huge cache for disk and client disk as a huge cache for robotic tape to reduce I/O bottleneck.

[Back to Table of Contents>](#)

---

# 5. Massive Parallelism in WANs

## 5.1 Concepts

In the above sections we considered the workstations in a building clustered in a LAN configuration. But if the workstations are in different locations, then it involves massive parallelism of networks. The machines on different LANs have to be connected together using a WAN or a Metropolitan Area Network (MAN). In cases wherein communication has to take place between LANs differences in the bandwidth will occur, nearly of two orders of magnitude. Depending on the needs of the application only low communication demands are feasible for massively parallel structures. If the dominating factors in the applications are comparisons, jumps and integer operations, these programs are better adapted to the architecture of RISC workstations with their relatively fast scalar integer units than to the supercomputer. Applications which don't require a lot of communication, support high degree of parallelism.

Problems from mathematical physics are concerned with local physics laws and differential equations with dominating operations being the floating point operations and the locality of the physics law is reflected in the algorithm by a high demand for communication. Hence massive parallelism with workstations in a WAN is possible but heavily depends on the application.

There exists a discrepancy between workstations on LAN and those on WANs as they are different in a few ways. Theoretically, the packages which assist in developing the parallel programs for LANs should also work for WANs. The bandwidth in a WAN is one to two orders of magnitude lower than that in LAN and latency is higher by one order of magnitude. In WANs, there could be traffic from connected LANs which results in higher variation in available bandwidth and latency. There could be delays in the commands which start and terminate processes remotely and timeouts could also be delayed by several minutes.

In wide area massive parallel networks, location transparencies of processes is undesirable as the programmer should know whether two processes reside in the same LAN or not and communication and synchronization is more difficult as it has to be achieved at the borders of a LAN and should be adapted to the topology of the WAN. Care has to be taken not to load a remotely logged in workstation as the local user might experience problems like looping programs, filling the file system with core dumps, consuming all the slots of the process table, using up temporary storage space, flooding networks or extensive swapping leading to more trouble on several machines connected to the WAN.

Since installation of the specific software used for running a program cannot be catered to on all the machines on the WAN, the programming style has to be highly portable. Also as there is less control over the remote machine, failures of the network or the workstation due to machines being powered off, has to be anticipated. Hence there is a unique challenge for controlling communication and process activities in a wide area domain which necessitate distinguishing an inter-LAN and an intra-LAN layer.

[Back to Table of Contents>](#)

---

## 5.2 Managing Parallelism in WANs

Managing parallelism in WANs is quite a difficult task and work has been done in this direction, at the university of Zurich (see reference [7]). An analysis needs to be done at the process level. In order to manage the activities of the cluster three types of processes are used : worker process which does all the computational work for the application, master and slave processes which take care of the coordination of the activities. A single master process takes care of all the top level system activities, provides an interface and starts slave and worker processes. The slaves coordinate larger configurations, since several hundred workers managed by a single master would produce a bottleneck. Therefore the master starts the slaves, preferably a small number in every LAN, and the slaves administrate several dozens of worker processes.

The communication topology on the inter-LAN layer is a tree with the master process at the root, the workers as leafs and the slaves in between. Communication on the interLAN can either go upstream (towards the root) or downstream (towards the leaves) and is under the wide area cluster software. Workers controlled by different slaves cannot communicate with each other and routing is required through the lowest common parent. This pictures the practical scenario whenein workers belonging to two different LANs have to be routed. Usually slaves are placed in machines with direct access so that data between workers in different LANs can be routed through the slave.

Before an application has to be run, all processes have to be started. When the master is running, it consults its host file to find out which processes have to start running when and where. Once a slave is operative in a remote machine, this process repeats until the tree of master, slaves and workers is operative.

After startup a process receives information as to where and how to establish communication with its parent, starts its own child processes and waits for status information. If there is no communication from the other end then those children are considered to be in error. The slave acting as an intermediary between its workers and the master waits for orders from the master, after sending the status information to the master, and if none are received, it times out.

This mechanism produces a downstream wave front from the root of the tree to the leaves. The master starts the slaves and they in turn start other slaves and workers until all the processes start running. The leaves send a summary of the performance based on a benchmark upstream and at every branch the information is composed and sent upstream. Error information is passed upstream to be displayed to the user. Having received the summary information, the master broadcasts the application data downstream based on which the leaves start their calculation. When the workers are done, they sent the data upstream and the slaves apply an application dependant function to the data and pass on the result further upstream. The final result is output by the master process.

But in larger configurations the probability of a host not reacting (due to it being powered off, for instance) is higher. Hence the number of workers is going to vary and the application algorithm has to cope up with the loss of worker process during the calculation. The slave keeps track of the workers it handed work out to and checks to see if a result was handed back and also looks out for new workstations. If a system crashes the respective work is distributed to other processes. Since when the worker crashes the state is lost the slave-worker cooperation is stateless. The worker must send all information back to the slave. At the inter-LAN layer no fault tolerance mechanism is used. This is

because the slave processes are running on servers or gateways with a connection to the WAN and are more stable than individual workstations and will not be turned off unexpectedly. Also the slaves produce only slight computational load and so there is no danger of overloading a server.

The inter-LAN programming layer is based on waves which traverse the tree. It provides tree structured communication at low bandwidth and high latency time. Communication between arbitrary processes is difficult to provide for processes in different LANs. The intra-LAN programming layer uses the workpool mechanism and can handle every communication topology. It provides a significantly higher bandwidth and smaller latency time.

It is mandatory to prevent disturbances to other users. All error conditions are presented to the user on the machine of the master process. All other processes running under the same user ID is terminated. Finally the process itself exits. If this shutdown is triggered in a slave, the communication connection to its children is lost. The shutdown of a non-leaf process lead to the termination of all the processes under it as an error message is obtained when the child processes try to communicate with that process. This shutdown mechanism is triggered on a number of occasions when some error conditions are incurred like "file system full" or "process table full" and signals like "segmentation fault".

The local user should not be disturbed by a compute intensive job. By specifying the niceness of the job, it is guaranteed that our job gets a time slice only when no other processes are running. This technique works only if the process consumes a small percentage of the CPU time, else a lot of time is wasted in swapping and paging.

[Back to Table of Contents>](#)

---

## 5.3 Implementation Issue

The above algorithm was implemented for 23 different workstation/ operating system combinations and installed on over 800 workstations in 31 LANs of the internet, situated on all five continents. The system is nicknamed LOLA -- loosely coupled large area and was used to develop a prototype for the comparison of genetic and protein sequence.

[Back to Table of Contents>](#)

---

## 6. Conclusion

The Networks of Workstations (NOWs) have carved a niche for themselves as a technological breakthrough in the concept of distributed computing and is here to stay and is the future of computing. The key factors in the design of these high ordered systems is a scalable, high-bandwidth, low-latency network and a low overhead network interface. Coupled with a global operating system layer, the speed of the network allows us to view the galaxy of resources on the network like processors, memories, and disks as a shared pool. This view provokes novel approaches to age-old system services like virtual memory, file caching, and disk striping. The most important utility of this technology is the opportunity for large scale computing within an everyday framework of interactive computing.

The challenge is to provide a congenial computing atmosphere for the interactive user and allowing the demanding tasks to use resources available in the network. ATM LANs are used to cope up with the networking demands of the NOWs. Hundreds of nodes on a distributed network can be supported due to the scalability of the ATM networks.

Though distributed computing networks, is still in the infant stages of its development, some research organizations have gone ahead and coupled several hundreds of workstations reaching across the LAN borders and successfully running a wide area network across continents. A parallel molecular sequence analysis was done in several minutes something which otherwise would have taken several days (see reference [16] [Appendix](#)). Another instance is the parallel computing in the Apollo workstation domain (see reference [17] [Appendix](#)).

Distributed computing networks research holds a lot of promise and there is a lot of prospect for researchers and expectations are high in this area, keeping the benefits in mind.

[Back to Table of Contents>](#)

---

## 7. Appendix - Annotated Bibliography

1. Craig Patridge, Gigabit networking, Addison-Wesley Publishing Company, 1993

This is an introductory book on Gigabit networks and has a chapter on Distributed Systems. It explains the Communication Paradigms namely the Remote Procedure Call (RPC) and the Distributed Shared Memory quite nicely.

2. Al Geist et al, PVM Parallel Virtual Machine - A Users' Guide and Tutorial for Networked Parallel Computing, The MIT Press, 1994

This book describes the PVM system for heterogeneous network computing and also explains how to develop programs using PVM. This book is meant to provide a fast entrance into the world of heterogeneous computing. The book is meant for both students and researchers.

3. Mukesh Singhal, Niranjana G. Shivaratri, Advanced Concepts in Operating Systems, McGraw Hill, Inc. 1994

This book has a chapter on Distributed Systems, the architectures, mutual exclusion, deadlock and other aspects of Operating Systems. Provides an introduction to the field of Distributed Systems.

4. William Stallings, Data and Computer Communications (Fourth Edition), Macmillan Publishing Company, 1994

Introduction to Networks, explains the OSI model, LAN configurations etc. Useful for an introduction and to brush up concepts.

5. Morris Sloman, Network and Distributed Systems Management, Addison-Wesley Publishing Company, 1994

This book has a chapter on Distributed Systems in which some applications, architecture, interfaces and communication primitives are described.

6. Thomas E. Anderson, David E. Culler and David A. Patterson, A Case for NOW (Networks of

Workstations), IEEE Micro Feb 1995, p 54 - 64

Gives an account of the prospects and some solutions to the problems faced by the NOW research endeavor. Also a case study done at the University of California is presented. A very informative paper and gives a complete picture of the problem.

7. Clemens H. Cap, Massive Parallelism with Workstation Clusters - Challenge or Nonsense ?, High Performance Computing and Networking, International Conference and Exhibition, Munich, Germany, April 1994

This paper speaks of wide area networking of heterogeneous computers. Several hundreds of them are connected together. Process control for managing massive networks is described.

8. Mengjou Lin and David H.C. Du, Distributed Network Computing over Local ATM Networks, IEEE Journal on Selected Areas in Communications Vol 13, No. 4, May 1995, p.733 - 748

Distributed Network Computing using ATM is discussed in this paper. The performance characteristics involving end-to-end communication is explained and four different kinds of application programming interfaces are compared.

9. Jonathan S. Turner, Issues in Distributed Control for ATM Networks, Tech Report WUCS-95-12, Washington University, St. Louis, 1995

This paper defines the problem of distributed control of networks and in particular ATM networks, and looks at some design issues. Lays a framework rather than propose an algorithm.

10. Juan Miguel del Rosario and Alok N. Choudhary, High-Performance I/O for Massively Parallel Computers - Problems and Prospects, Computer March 1994, p 59 - 68.

The fact that the effectiveness of the parallel computers is hampered by the I/O limitation is noted in this paper. The architectures of I/O in MPPs and the operating and file systems are explained. These are the aspects on which the functioning of a distributed network depends.

11. Arthur Chai and Sumit Ghosh, Modeling and Distributed Simulation of a Broadband-ISDN Network, Computer Sept 1993, p 37 - 51

The model of the B-ISDN network is touched upon and distribution simulation is performed on a loosely coupled parallel processor using as distributed algorithm and is described in detail.

12. Nanette J. Boden et al, Myrinet : A Gigabit-per-second Local Area Network, IEEE Micro Feb 1995, p 29 - 36

The Myrinet LAN employs the technology used for packet communication and switching using massively parallel processors. Myrinet demonstrates the highest performance per unit cost of any current LAN.

13. Thorsten von Eicken et al, Active Messages : a mechanism for Integrated Communication and Computation, Tech Report UCB/CSD 92/#675, March 1992, University of California, Berkeley.

Active messages, message passing architectures and hardware support is explained. Intended programming model for message driven architectures is enumerated.

14. Rezmi H. Arpaci, The Interaction of Parallel and Sequential Workloads on a Network of Workstations, University of California, Berkeley, 1994

This paper explores the possibility of the interactive and parallel jobs to be run in a NOW. Some issues which arise when combining the workloads is explained. A method for finding out the delay time for recruiting idle machines is looked into.

15. Amin Vahdat, Douglas Ghormley and Thomaas Anderson, Efficient, Portable, and Robust Extension of Operating System Functionality, Tech Report, University of California, Berkeley, 1994

Some operating system issues concerning the distributed systems are discussed and the global Unix is touched upon.

16. Volker Strumpfen, Coupling Hundreds of Workstations for Parallel Molecular Sequence Analysis, High Performance Computing and Networking, International Conference and Exhibition, Munich, Germany, April 1994

This is an example of hundreds of workstations connected as a WAN. Molecular sequence analysis was carried out in a few minutes time. If run sequentially it would have taken days.

17. Ferhan Perkergin, Parallel Computing Optimization in the Apollo Domain Network, IEEE Transactions on Software Engineering Vol 18, No 4, April 1992, p 296 - 303.

Performance of parallel computing in a network of Apollo workstations where the processes use Remote Procedure Call (RPC) mechanism for communication.

18. "The Berkeley NOW project", <http://now.cs.berkeley.edu/Case/case.html>

This web page gives the abstract, paper and slides and can be downloaded as a postscript file. Other related topic like GLUnix, xFS, Active messages etc can also be found.

19. "Computer Networks and distributed systems group", <http://media.ijs.si/networks.html>

This web page gives an account of the research carried on in this group.

---

[Other Reports on Recent Advances in Networking 1995](#)

[Back to Raj Jain's Home Page](#)

Last updated August 22, 1995