INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN: 2278 – 0211 (Online)

Designing Issues For Distributed Computing System: An Empirical View

Dr. S.K Gandhi,
Research Guide
Department of Computer Science & Engineering,
AISECT University, Bhopal (M.P), India
Pawan Kumar Thakur
Ph. D Research Scholar, Department of Computer Science & Engineering,
AISECT University, Bhopal (M.P), India

Abstract:

Distributed system is a collection of multiple processors interconnected by communication networks. Designing a distributed system is more difficult than designing a centralized operating system for a number of reasons. To design a centralized computing system the operating system has access to complete and accurate information about the environment in which it is functioning. In a distributed system the resources are physically separated there is no common clock among the multiple processors delivery of messages is delayed and messages could even be lost. The purpose of this paper is to study the various designing issues of distributed computing and provide an overview for how to implement these issues. The paper also represents the models used by communication network to interconnect the multiple processors and provide a comparative study of different model to prove which model is suitable for distributed system.

Keywords: *Distributed System, Transparency, Reliability, security, Heterogeneous, scalability, Mini-computer, workstation, pool-processor, hybrid*

1.Introduction

Distributed system is a collection of independent computers that appears to its users as a single coherent system. –

1.1.Tanenbaum

Distributed system is not an operating system nor is it an application. But it is an integrated set of services and tools that can be installed as a coherent environment on top of existing operating systems and serve as a platform for building and running distributed applications. The primary goal of distributed system is vendor independence [2]. It runs on many different kinds of computers, operating systems and networks produced by different vendors. It can be used with any network hardware and transport software, including TCP/IP, X.25, as well as other similar products. The heterogeneous nature of the system is transparent to the applications programmers, making their job of writing distributed applications much simpler [1].

This works represents the various designing issues of distributed system and different models of interconnected network. In section 2 represents the background and limitation of previous technology, section 3 provide the objective of study, section 4 represents the designing issues, section 5 focus the different model of distributed, section 6 represent comparison and discussion on these model and find out that which model is best suitable for distributing computing, section 7 represent the observation and finding.

2.Background And Limitations Of Technology

The previous operating system used heterogeneous multicomputer systems. Although managing the underlying hardware is an important issue for an operating system, the distinction from traditional operating systems comes from the fact local services are made available to remote clients[1]. The previous system fall in the following drawback:

- Explicit data and network programming of applications on each device.
- Multiple types of heterogeneity of data stores.
- Poor support for maintaining global consistency of data stores.
- Poor Middleware support.
- Difficult peer-to-peer interaction (no data serving capabilities).
- Poor or no support for dis -connectivity, location independence, group
- Collaboration, atomic transaction.

3.Objective Of The Study

This works study the various designing issues of distributed system and put the principles to deal and implement these issues. After that we will study the various model of distributed system. Our compare and discussion section compare the entire models of distributed system and find the suitable model for distributed computing. Finally, observation of these issues and answers the finding.

4.Designing Issues

A distributed operating system must be designed to provide all the advantages of a distributed system to its users. That is, the users should be able to view a distributed system as a virtual centralized system that is flexible, efficient, reliable, secure, and easy to use. To meet this challenge, the designers of a distributed operating system must deal with several design issues. Some of the key design issues are described below.

4.1.Transparency

In a distributed system the resources are physically separated there is no common clock among the multiple processors, delivery of messages is delayed, and messages could even be lost [3]. Achieving complete transparency is a difficult task and requires that several different aspects of transparency be supported by the distributed operating system. The eight forms of transparency identified by the International Standards Organization's Reference Model for Open Distributed Processing [4].

- Access transparency.
- Location transparency.
- Replication transparency.
- Failure transparency.
- Migration transparency.
- Concurrency transparency.
- Performance transparency.
- Scaling transparency.

4.2.Reliability

Distributed system, which manages multiple resources, must be designed properly to increase the system's reliability by taking full advantage of this characteristic feature of a distributed system. For higher reliability, the fault-handling mechanisms of a distributed operating system must be designed properly to avoid faults, to tolerate faults, and to detect and recover from faults. Commonly used methods for dealing with these issues are fault avoidance and fault tolerance.

4.3.Flexibility

Another important issue in the design of distributed operating systems is flexibility. Flexibility is the most important feature for open distributed systems. The design of distributed operating system should be flexible due to the following reasons:

- Ease of modification
- Ease of enhancement

4.4.Performance

The overall performance should be better than or at least equal to that of running the same application on a single-processor system. Some design principles considered useful for better performance are as follows:

- Batch if possible.
- Cache whenever possible.
- Minimize copying of data.
- Minimize network traffic.
- Take advantage of fine-grain parallelism for multiprocessing.

4.5.Scalability

A distributed operating system should be designed to easily cope with the growth of nodes and users in the system. That is, such growth should not cause serious disruption of service or significant loss of performance to users. Some guiding principles for designing scalable distributed systems are as follows:

- Avoid centralized entities.
- Avoid centralized algorithms.

- Perform most operations on client workstations.

4.6.Heterogeneity

A heterogeneous distributed system consists of interconnected sets of dissimilar hardware or software systems. Because of the diversity, designing heterogeneous distributed systems is far more difficult than designing homogeneous distributed systems, in which each system is based on the same, or closely related, hardware and software. However, as a consequence of large scale, heterogeneity is often inevitable in distributed systems [2]. Furthermore, often heterogeneity is preferred by many users because heterogeneous distributed systems provide the flexibility to their users of different computer platforms for different applications.

4.7.Security

In order that the users can trust the system and rely on it, the various resources of a computer system must be protected against destruction and unauthorized access. Enforcing security in a distributed system is more difficult than in a centralized system because of the lack of a single point of control and the use of insecure networks for data communication. Therefore, as compared to a centralized system, enforcement of security in a distributed system has the following additional requirements [12]:

- It should be possible for the sender of a message to know that the message was received by the intended receiver.
- It should be possible for the receiver of a message to know that the message was sent by the genuine sender.
- It should be possible for both the sender and receiver of a message to be guaranteed that the contents of the message were not changed while it was in transfer.

5.Model Of Distributed System

There are various models are used for building distributed computing systems. These models can be broadly classified into five categories-minicomputer, workstation, workstation-workstations server processor-pool, and hybrid. They are briefly described below:

5.1.Mini-computer Model

A distributed computing system based on this model consists of minicomputers. They may be large supercomputers as well interconnected by a communication network.

Each minicomputer usually has multiple users simultaneous logged on to it as shown in Fig.1. For this, several interactive terminals are connected to each minicomputer each user is logged on to one specific minicomputer, with remote access to other Minicomputers [2].

The network allows a user to access remote resources that are available at some machine other than the one on to which the user is currently logged. The early ARPANET is an example of a distributed computing system based mini-computer model.

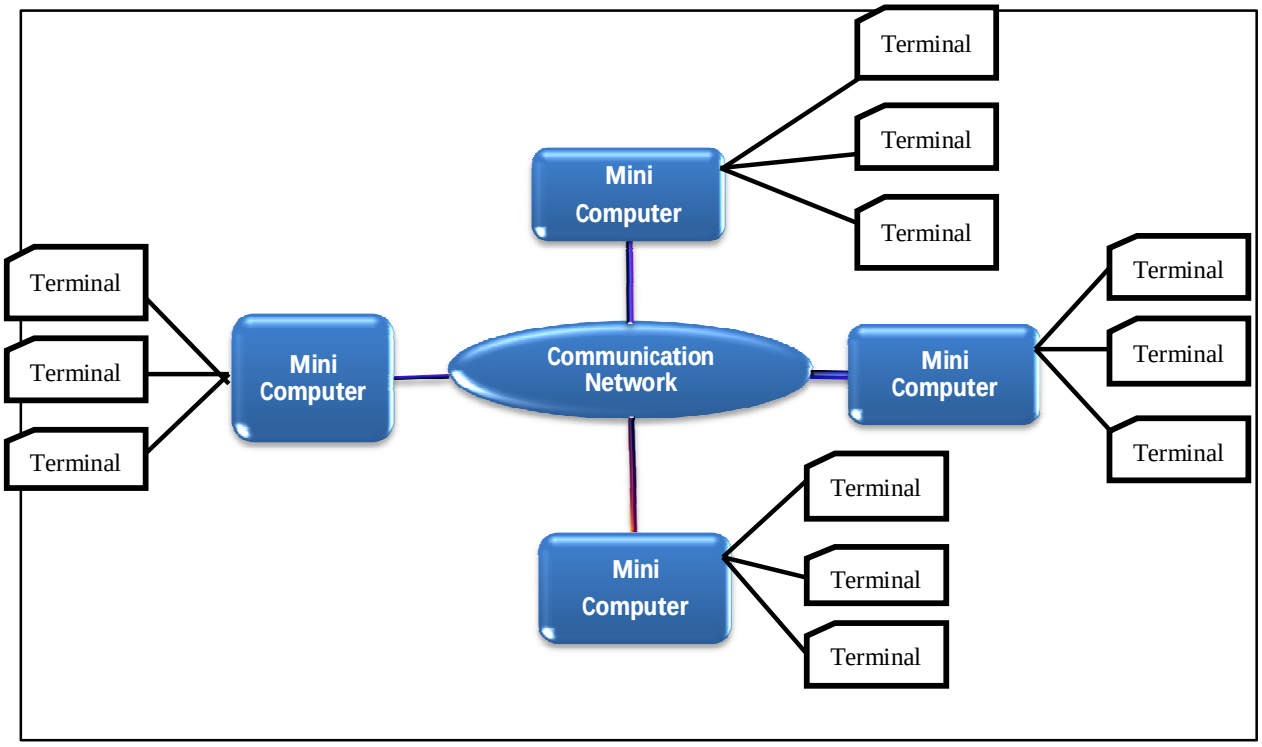

Figure 1: Mini Computer Model for Distributed System

5.2.Workstation Model

The distributed computing system based on the workstation model consists of several workstations interconnected by a communication network as shown in Fig 2. A company's office or a university department may have several workstations scattered throughout a building or campus, each workstation equipped with its own disk and serving as a single-user computer [2].

The Sprite system and an experimental system developed at Xerox PARC [5] are two examples of distributed computing systems based on the workstation model. This model is not so simple to implement as it might appear at first sight because several issues must be resolved. These issues are [2] as follows:

- How does the system find an idle workstation?
- How is a process transferred from one workstation to get it executed on another workstation?
- What happens to a remote process if a user logs onto a workstation that was idle until now and was being used to execute a process of another workstation?

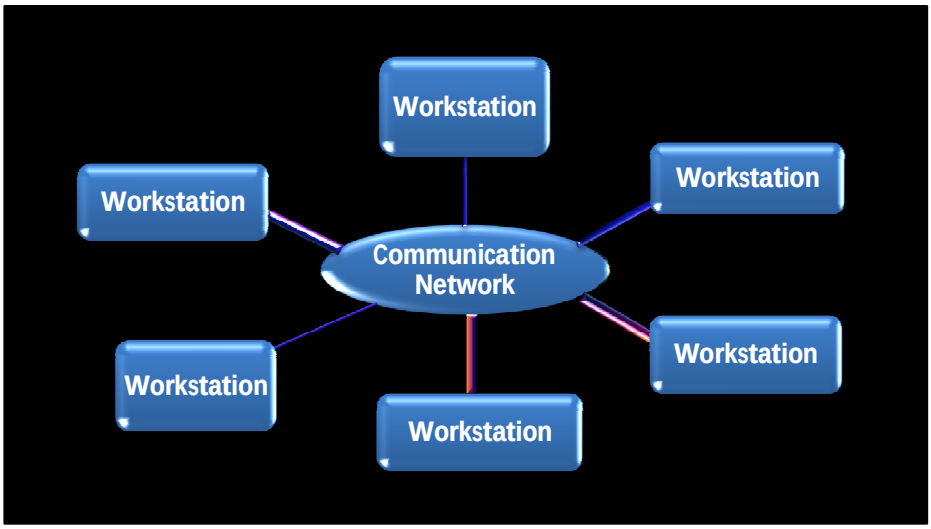

Figure 2: Workstation Model for Distributed System

5.3.Workstation-Server Model

A distributed computing system based on the workstation server model consists of a few minicomputers and several workstations most of which are diskless but a few of which may be disk full interconnected by a communication network as shown in Fig. 3.

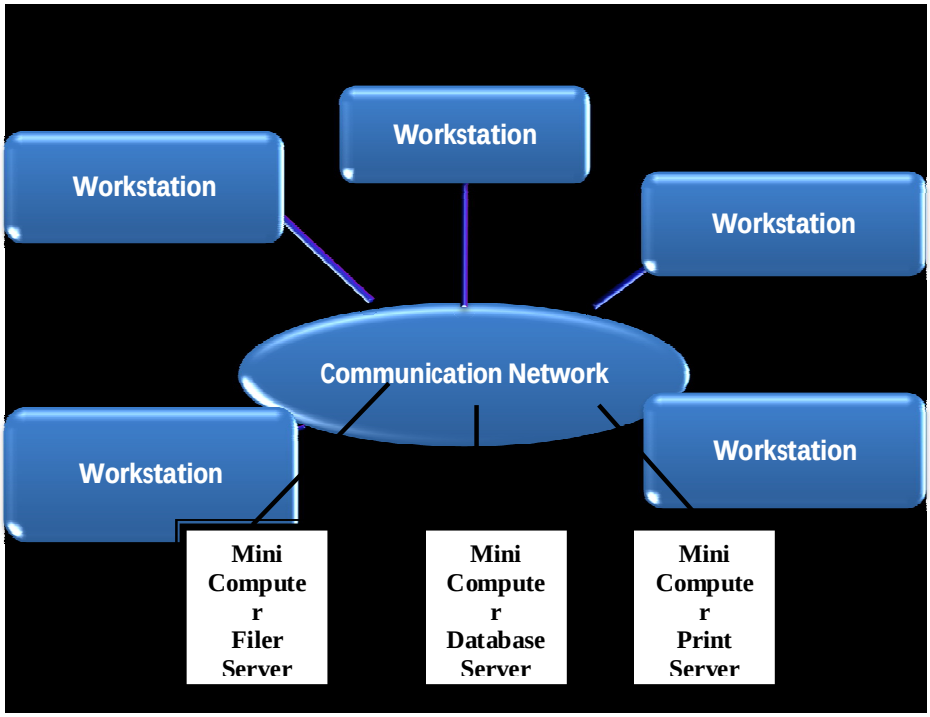

Figure 3: Workstation Server Model for Distributed System

In this model, a user logs onto a workstation called his or her home workstation. Normal computation activities required by the user's processes are performed at the user's home workstation, but requests for services provided by special servers (such as a file server or a database server) are sent to a server providing that type of service that performs the user's requested activity and returns the result of request processing to the user's workstation. Therefore, in this model, the user's processes need not be migrated to the server machines for getting the work done by those machines. The V-System [5] is an example of a distributed computing system that is based on the workstation-server model.

As compared to the workstation model, the workstation-server model has several advantages:

- In general, it is cheaper to use a few minicomputers with fast disks that are accessed over the network than a large number of disk full workstations where each workstation having a small and slow disk.
- Diskless workstations are also very useful for system maintenance point of view. Backup and hardware maintenance are easier to perform with a few large disks than with many small disks scattered all over a building or campus.

- In the workstation-server model all the files are managed by the file servers. The users have the flexibility to use any workstation and access the files in the same manner irrespective of which workstation the user is currently logged on. This is not true with the workstation model where each workstation has its local file system. The mechanisms to access local and remote files are different [13].
- In the workstation-server model, the request-response protocol described above is mainly used to access the services of the server machines. Therefore, unlike the workstation model, this model does not need a process migration facility, which is difficult to implement.
- A user has guaranteed response time because workstations are not used for executing remote processes. However, the model does not utilize the processing capability of idle workstations [14].

5.4.Processor-Pool Model

The pool of processors consists of a large number of microcomputers and minicomputers attached to the network as shown in Fig.1.4. Each processor in the pool has its own memory to load and run a system program or an application program of the distributed computing system [3]. The processor-pool model is based on the observation that most of the time a user does not need any computing power but once in a while he or she may need a very large amount of computing power for a short time (e.g., when recompiling a program consisting of a large number of files after changing a basic shared declaration).

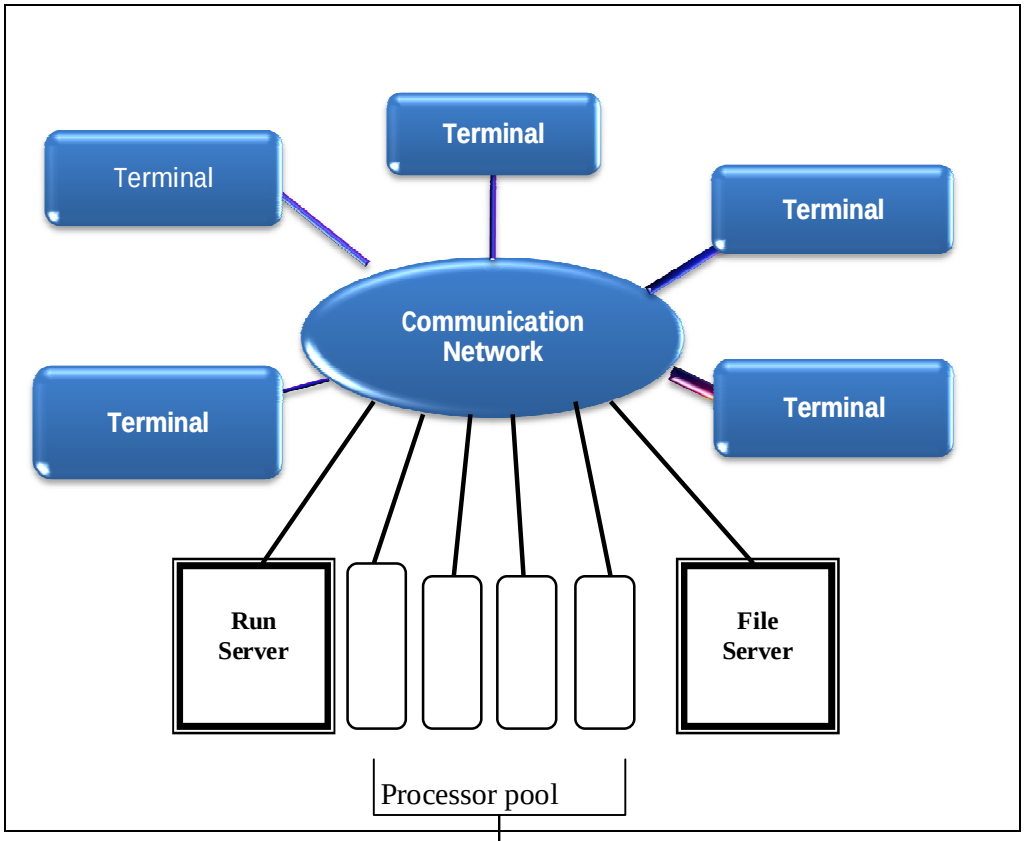

Figure 4: Processor-Pool Model

Amoeba [8,9], Plan 9 , and the Cambridge Distributed Computing System [10] are examples of distributed computing systems based on the processor-pool model.

5.5. Hybrid Model

To combine the advantages of both the workstation-server and processor-pool models, a hybrid model may be used to build a distributed computing system.

7.Comparison And Discussion

In Minicomputer model it is found that if one of the mini-computer fail several interactive terminals are connected to each minicomputer will not works. This is one of the major drawbacks of minicomputer model.

In workstation model, it has been often found that in such an environment, at any one time, a significant proportion of the workstations are idle that is not being used, resulting in the waste of large amounts of CPU time. Therefore, the idea of the workstation model

is to interconnect all these workstations by a high-speed LAN so that idle workstations may be used to process jobs of users who are logged onto other workstations and do not have sufficient processing power at their own workstations to get their jobs processed efficiently[11].

As compared to the workstation model, the workstation-server model has several advantages:

- In general, it is cheaper to use a few minicomputers with fast disks that are accessed over the network than a large number of disk full workstations where each workstation having a small and slow disk.
- Diskless workstations are also very useful for system maintenance point of view. Backup and hardware maintenance are easier to perform with a few large disks than with many small disks scattered all over a building or campus.
- In the workstation-server model all the files are managed by the file servers. The users have the flexibility to use any workstation and access the files in the same manner irrespective of which workstation the user is currently logged on [13]. This is not true with the workstation model where each workstation has its local file system. The mechanisms to access local and remote files are different.
- In the workstation-server model, the request-response protocol described above is mainly used to access the services of the server machines. Therefore, unlike the workstation model, this model does not need a process migration facility, which is difficult to implement.

As compared to the workstation-server model the processor-pool model is better in following way:

- The processor-pool model allows better utilization of the available processing power of a distributed computing system. This is because in the processor-pool model the entire processing power of the system is available for use by the currently logged-on users [12].
- This is not true for the workstation-server model in which several workstations may be idle at a particular time but they cannot be used for processing the jobs of other users.

- The processor pool model provides greater flexibility than the workstation-server model in the sense that the system's services can be easily expanded without the need to install any more computers; the processors in the pool [14] can be allocated to act as extra servers to carry any additional load arising from an increased user population or to provide new services.

It is found that if we mixtures the above model and design a new model it is suitable for distributed system. For example the hybrid model is based on the workstation server model but with the addition of pool of processors is better for distributed computing environment in following way:

- The processors in the pool can be allocated dynamically for computations that are too large for workstations or that require several computers concurrently for efficient execution and gives a guaranteed response to interactive jobs by allowing them to be processed on local workstation of the users.
- In addition to efficient execution of computation-intensive jobs, the hybrid model gives guaranteed response to interactive jobs by allowing them to be processed on local workstations of the users.

8.Observation And Finding

8.1.Observation 1

The aiming at full distribution transparency may be too much:

- Users may be located in different continents distribution is apparent and not something you want to hide.
- Completely hiding failures of networks and nodes is (theoretically and practically) impossible. We cannot distinguish a slow computer from a failing one. We can never be sure that a server actually performed an operation before a crash.
- Full transparency will cost performance, exposing distribution of the system. Keeping Web caches exactly up-to-date with the master copy.

8.2.Observation 2

Implementing openness general a distributed system provides only mechanisms:

- Allow (dynamic) setting of caching policies, preferably per cache item.
- Support different levels of trust for mobile code.
- Provide adjustable QoS parameters per data stream.
- Offer different encryption algorithms.

Implementing openness requires support for different policies specified by applications and users:

- What level of consistency do we require for client cached data?
- Which operations do we allow downloaded code to perform?
- What level of secrecy do we require for communication?

8.3.Observation 3

In distributed system applying scaling techniques is easy except for one thing:

- Having multiple copies (cached or replicated), leads to inconsistencies: modifying one copy makes that copy different from the rest.
- Always keeping copies consistent and in a general way requires global synchronization on each modification.
- Global synchronization precludes large-scale solutions.

8.4.Observation 4

If we can tolerate inconsistencies we may reduce the need for global synchronization.

8.5.Observation 5

Tolerating inconsistencies is application dependent.

9.Conclusion

This works bring to a close that distributed computing systems are much more complex and difficult to build than the traditional centralized systems. The complexity and the difficulty of building, the installation and use of distributed computing systems are rapidly increasing. But a distributed operating system has better transparency and fault tolerance capability and provides the image of a virtual uni-processor to the users. The main issues involved in the design of a distributed operating system are transparency, reliability, flexibility, performance, scalability, heterogeneity, security, and emulation of existing operating systems. Computers with ordinary power are suitable for ordinary data processing jobs, whereas high-performance computers are more suitable for complex mathematical computations. A distributed computing system may have a pool of different types of computers, in which case the most appropriate one can be selected for processing a user's job depending on the nature of the job. So a distributed computing system that is based on the mix model interactive jobs can be processed at a user's own workstation and the processors in the pool may be used to process non-interactive, computation-intensive jobs.

10.Reference

1. Tanenbaum A.S, “Distributed Operating System, Pearson Education”, 2007.
2. Sinha P.K, Distributed Operating Systems Concepts and Design, Prentice-Hall of India private Limited, 2008.
3. Coulouris, G., Dollimore, J., Kindberg, T.,“Distributed Systems Concepts and Design”, Pearson Education, 2003.
4. Basic Reference Model of Open Distributed Processing, Part 1: Overview and Guide to use IEC JTC1/SC212/WG7 CD10746-1, International Standards Organization, 1992.
5. Ousterhout, J. K., Cherenson, A. R., Douglis, F., Nelson, M. N., and Welch, B. B., "The Sprite Network Operating System," IEEE Computer, Vol. 21, No. 2, pp. 23-36, 1988.
6. Shoch, J. R, and Hupp, J. A., "The Worm Programs: Early Experiences with a Distributed Computation," Communications of the ACM, Vol. 25, No. 3, pp. 172-180 ,1982.
7. Cheriton, D. R., "The V Distributed System," Communications of the ACM, Vol 31, No. 3, pp. 314-333,1988.
8. Mullender, S. J. (Ed.), Distributed Systems, 2nd ed., Addison-Wesley, Reading, MA ,1993.
9. Mullender, S. J., and Tanenbaum, A. S., "Protection and Resource Control in Distributed Operating Systems," Computer Networks, Vol. 8, pp. 421-432, 1984.
10. Needham, R. M., and Herbert, A. J., The Cambridge Distributed Computing System, Addison-Wesley, Reading, MA, 1982.
11. Bibliography containing references on Distributed Computing can be found at: ftp:ftp.cs. umanitoba.ca/pub/bibliographies/Distributed/Osser.html
12. Bibliographies containing references on Distributed Systems can be found at:ftp:ftp.cs. umanitoba.ca/pub/bibliographies/Distributed/Dcs-1.0.html,ftp:ftp.csumanitoba.ca/pub /bibliographies /Distributed/dist.sys. l.html
13. List of publications of the Stanford Distributed Systems Group (DSG) can be found at: http: www-dsg.stanford.edu/Publications.html
14. List of publications of the Distributed Systems Research Group (DSRG) at Oregon Graduate Institute can be found at: http:www.cse.ogi.edu/DSRG/osrg/osrg.html#Current Paper