IJCSMS International Journal of Computer Science and Management Studies, Vol. 11, Issue 02, Aug 2011
ISSN (Online): 2231-5268
www.ijcsms.com

Distributed Computing: An Emerging Paradigm for New Technologies

Swati Gupta[1], Kuntal Saroha[2] , Samiksha Mehta[3]

[1]Lecturer, RIMT, Chidana
Swati.mangla.555@gmail.com

[2]Research Scholar, IIIT Gwaliar
sarohakuntal@gmail.com

[3]Asst. Professor, HIT, Asouda
ersam_mehta@yahoo.co.in

ABSTRACT

Terms such as 'Distributed Computing' have gained a lot of attention, as they are used to describe emerging paradigms for the management of information and computing resources. Distributed computing is that type of computing that uses geographically and administratively disparate resources. In distributed computing, individual users can access computers and data transparently, without having to consider location, operating system, account administration, and other details. In distributed computing, the details are abstracted, and the resources are virtualized. In this paper , we presented this technology , its characteristics and also its architecture.

Keywords: *Distributed computing, grid computing, distributed architecture, network models*

I INTRODUCTION

The word *distributed* in terms such as "distributed system", "distributed programming", and "distributed algorithm" originally referred to computer networks where individual computers were physically distributed within some geographical area. The terms are nowadays used in a much wider sense, even referring to autonomous processes that run on the same physical computer and interact with each other by message passing. While there is no single definition of a distributed system, the following defining properties are commonly used: There are several autonomous computational entities, each of which has its own local memory.

- The entities communicate with each other by message passing.

In this paper, the computational entities are called *computers* or *nodes*.

A distributed system may have a common goal, such as solving a large computational problem. Alternatively, each computer may have its own user with individual needs, and the purpose of the distributed system is to coordinate the use of shared resources or provide communication services to the users.

Other typical properties of distributed systems include the following:

- The system has to tolerate failures in individual computers.
- The structure of the system (network topology, network latency, number of computers) is not known in advance, the system may consist of different kinds of computers and network links, and the system may change during the execution of a distributed program.
- Each computer has only a limited, incomplete view of the system. Each computer may know only one part of the input

Distributed computing is a field of computer science that studies distributed systems. A distributed system consists of multiple autonomous computers that communicate through a computer network. The computers interact with each other in order to achieve a common goal. A computer program that runs in a distributed system is called a distributed program, and distributed programming is the process of writing such programs.

Distributed computing also refers to the use of distributed systems to solve computational problems. In distributed computing, a problem is divided into many tasks, each of which is solved by one or more computers.

Distributed vs Grid Computing

There are actually two similar trends moving in tandem--distributed computing and grid computing. Depending on how you look at the market, the two either overlap, or distributed computing is a subset of grid computing. Grid Computing got its name because it strives for an ideal scenario in which the CPU cycles and storage of millions of systems across a worldwide network function as a flexible, readily accessible pool that could be harnessed by anyone who needs it, similar to the way power companies and their users share the electrical grid.

Sun defines a computational grid as "a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to computational capabilities." Grid computing can encompass desktop PCs, but more often than not its focus is on more powerful workstations, servers, and even mainframes and supercomputers working on problems involving huge datasets that can run for days. And grid computing leans more to dedicated systems, than systems primarily used for other tasks.

Large-scale distributed computing of the variety we are covering usually refers to a similar concept, but is more geared to pooling the resources of hundreds or thousands of networked end-user PCs, which individually are more limited in their memory and processing power, and whose primary purpose is not distributed computing, but rather serving their user. As we mentioned above, there are various levels and types of distributed computing architectures, and both Grid and distributed computing don't have to be implemented on a massive scale. They can be limited to CPUs among a group of users, a department, several departments inside a corporate firewall, or a few trusted partners across the firewall.

II DISTRIBUTED SYSTEM ARCHITECTURE

Distributed systems are built up on top of existing networking and operating systems software. A distributed system comprises a collection of autonomous computers, linked through a computer network and distribution middleware. To become autonomous there exist a clear master/slave association Between two computers in the network. The middleware enables computers to coordinate their activities and to share the resources of the system, so that users perceive the system as a single, integrated computing facility.

Thus, middleware is the bridge that connects distributed applications across dissimilar physical locations, with dissimilar hardware platforms, network technologies, operating systems, and programming languages. The middleware software is being developed following agreed standards and protocols. It provides standard services such as naming, persistence, concurrency control to ensures that accurate results for concurrent processes are produced and obtains the results as fast as possible, event distribution, authorization to specify access rights to resources, security etc. The middleware service extends over multiple machines. Figure 1 shows a simple architecture of a distributed system

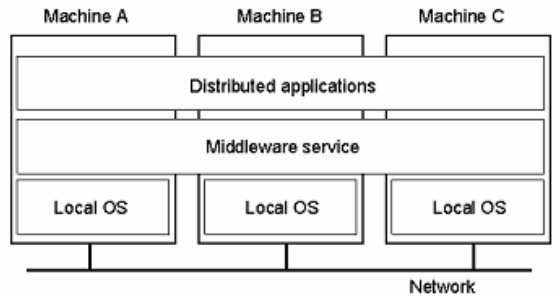


Figure 1 : Distributed System

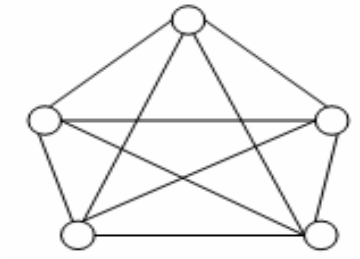
The distributed system can be viewed as defined by the physical components or as defined from user or computation point of view. The first is known as the physical view and the second as the logical view. Physically a distributed system consists of a set of nodes (computers) linked together by a communication network. The nodes in the network are loosely coupled and do not share their memory. The nodes in the system communicate by passing messages over the communication network. Communication protocols are used for sending messages from one node to another. The logical model is the view that an application has of the system. It contains a set of concurrent processes and communication channels between them. The core network is treated as fully connected. Processes communicate by sending messages to each other. A system is synchronous if during a proper execution, it all the time performs the intended operation in a known fixed time, otherwise it is asynchronous. In synchronous system the failure can be noticed by a lack of response from the system. Therefore, timeout based techniques are used for failure discovery.

A distributed system can be constructed by means of fully connected networks or partially connected networks . A fully connected network (figure 2) is a network in which each of the nodes is connected to each other. The problem with such a system is that adding new nodes to the system results in the increase of number of nodes connected to the node. Due to this the number of file descriptors and complexity for each node to implement the connections are increased heavily. Thus, the scalability (capability of a system to continue to function well when the system is changed in size or volume) of such systems is limited by each node's capacity to open file descriptors and the ability to handle the new connections. The communication cost - the message delay of sending a message from the source to the destination- is low because a message sent from one computer to another one only goes through one link.

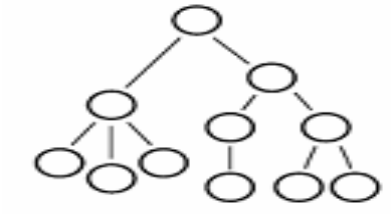
Fully connected systems are reliable because when a few computers or links fail, the rest of the computers can still communicate with others. In a partially connected network, direct links exist between some, but not all, pairs of computers. A few of the partially connected network models are star structured networks, multi-access bus networks; ring structured networks, and tree-structured networks (figure 2). Some of the traditional distributed systems such as client/server paradigm use a star as the network topology. The problem with such a system is that when the central node fails, the entire system will be collapsed. In a multi-access bus network, a set of clients are connected via a shared communications line, called a bus. The bus link becomes the bottleneck and if it fails, all the nodes in the system cannot connect to each other.

Another disadvantage is that performance degrades as additional computers are added or on heavy traffic. In a ring network each node connects to exactly two other nodes, forming a single continuous pathway for signals through each node. As new nodes are added, the diameter of the system grows as the number of nodes in the system, resulting in a longer message transmission delay. A node failure or cable break might isolate every node attached to the ring. In a tree-structured network (hierarchical network), the nodes are connected as a tree. Each node in the network having a specific fixed number, of nodes associated to it at the next lower level in the hierarchy. The scalability of the tree-structured network is

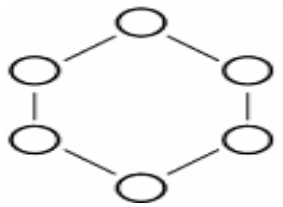
better than that of the fully connected network, since new node can be added as the child node of the leaf nodes or the interior nodes. On the other hand, in such systems, only messages transmitted between a parent node and its child node go though one link, other messages transmitted between two nodes have to go through one or more intermediate nodes.



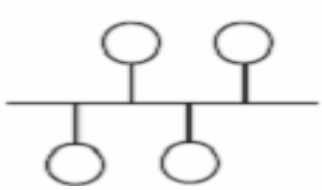
Fully Connected Network



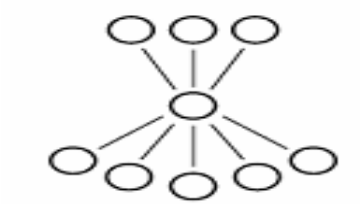
Tree Structured Network



Ring Structured Network



Multi-access Bus Network



Star Structured Network

Figure 2 : Network Models

III CHARACTERISTICS OF A DISTRIBUTED SYSTEM

A distributed system must possess the following characteristics to deliver utmost performance for the users :

1. **Fault-Tolerant:** Distributed systems consist of a large number of hardware and software modules that are bound to fail in the long run. Such component failures can escort to service unavailability. Hence, the systems should be able to recover from component failures without performing erroneous actions. The goal of fault tolerance is to avoid failures in the system even in the presence of faults to provide uninterrupted service. A system is said to be fault tolerant if it can mask the presence of faults. The aim of any fault tolerant system is to increase its reliability or availability. The reliability of a system is defined as the probability that the system survives till that time. A reliable system prevents loss of information even in the event of component failures. Availability is the fraction of time for which a system is available for use. Usually

fault tolerance is achieved by providing redundancy.

Redundancy is defined as those parts of the system that are not needed for its correct functioning. It is of three types – hardware, software and time. Hardware redundancy is achieved by adding extra hardware components to system which take over the role of failed components in case some faults occur in them. Software redundancy includes extra instructions and code included for managing the extra hardware components, and using them correctly for uninterrupted service, in case of some component failure. In time redundancy the same instruction is executed many times. This is used to handle temporary faults in the system .

2. Scalable: A distributed system can operate correctly even as some aspect of the system is scaled to a larger size. Scale has three components: the number of users and other entities that are part of the system, the distance between the farthest nodes in the system, and the number of organizations that exert administrative control over pieces of the system. The three elements of scale affect distributed systems in many ways. Among the affected components are naming, authentication for verifying someone's identity, authorization, communication, the use of remote resources, and the mechanisms by which users observe the system. Three techniques are employed to manage scale: replication, distribution, and caching . Replication creates multiple copies of resources. Its use in naming, authentication, and file services reduces the load on individual servers and improves the reliability and availability of the services as a whole. The two important issues of replication are the placement of the replicas and the mechanisms by which they are kept consistent. The placement of replicas in a distributed system depends on the purpose for replicating the resource. If a service is being replicated to reduce the network delays when the service is accessed, the replicas are sprinkled across the system. If the majority of users are local, and if the service is being replicated to improve its availability or to spread the load across multiple servers, then replicas may be placed near one another. If a change is made to the object, the change should be noticeable to everyone in the system. For example, the system sends the updates to any replica, and that replica forwards the update to the others as they become available. If inconsistent updates are received by different replicas in different orders, timestamps (the date/time at which the update was generated) are used to differentiate the copies.

Distribution, another mechanism for managing scale in distributed systems, allows the information maintained by a distributed service to be extended across several servers. Distributing data across multiple servers reduces the size of the database that must be maintained by each server, dropping the time needed to search the database. Distribution also spreads the load across the servers reducing the number of requests that are handled by each. If requests can be distributed to servers in proportion to their power, the load on servers can be effectively managed. Network traffic can be reduced if data are assigned to servers close to the location from which they are most frequently used. In tree structured system, if cached copies are available from subordinate servers, the upper levels can be avoided.

Caching is another important technique for building scalable systems. Caching decreases the load on servers and the network. Cached data can be accessed faster than if a new request is made. The difference between replication and caching is that cached data is a short-term data. Instead of propagating updates on cached data, consistency is maintained by nullifying cached data when consistency cannot be guaranteed. Caching is usually performed by the client, reducing frequent requests to network services. Caching can also occur on the servers executing those services. Reading a file from the memory cached copy on the file server is faster than reading it from the client's local disk.

3. **Predictable Performance**: Various performance metrics such as response time

(elapsed time between the end of an inquiry or demand on a computer system and the beginning of a response), throughput (the rate at which a network sends or receives data), system utilization, network capacity etc. are employed to assess the performance. Predictable performance is the ability to provide desired responsiveness in a timely manner.

4. **Openness:** The attribute 'openness' ensures that a subsystem is continually open to interaction with other systems. Web services are software systems designed to support interoperable machine-to-machine interaction over a network. These protocols allow distributed systems to be extended and scaled. An open system that scales has benefit over a completely closed and self-reliant system. A distributed system independent from heterogeneity of the underlying environment such as hardware and software platforms achieves the property of openness. Therefore, every service is equally accessible to every client (local or remote) in the system. The implementation, installation and debugging of new services should not be very complex in a system possessing openness characteristic.

5. **Security:** Distributed systems should allow communication between programs/users/ resources on different computers by enforcing necessary security arrangements. The security features are mainly intended to provide confidentiality, integrity and availability. Confidentiality (privacy) is protection against disclosure to unauthorised person. Violation of confidentiality range from the discomforting to the catastrophic. Integrity provides protection against alteration and corruption. Availability keeps the resource accessible. Many incidents of hacking compromise the integrity of databases and other resources. "Denial of service" attacks are attacks against availability. Other important security concerns are access control and non repudiation. Maintaining access control facilitates the users to access only those resources and services to which they are entitled. It also ensures that users are not denied resources that they legitimately can expect to access. Non repudiation provides protection against denial by one of the entities involved in a communication. The security mechanisms put into practice should guarantee appropriate use of resources by different users in the system.

6. **Transparency:** Distributed systems should be perceived by users and application developers as a whole rather than as a collection of cooperating components. The locations of the computer systems involved in the operations, concurrent operations, data replication, resource discovery from multiple sites, failures, system recovery etc. are hidden from users. Transparency hides the distributed nature of the system from its users and shows the user that the system is appearing and performing as a normal centralized system. The transparency can be employed in different ways in a distributed system (Figure 3) .

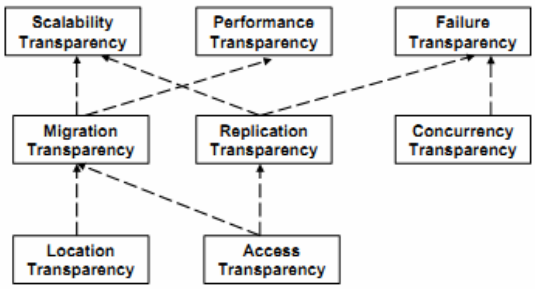


Figure 3 : Transparency in Distributed Systems

Access transparency facilitates the users of a distributed system to access local and remote resources using identical operations. (e.g. navigation in the web).

Location transparency describes names used to identify network resources (e.g. IP address) independent of both the user's location and the resource location. In other words, location transparency facilitates a user to access resources from anywhere on the network without knowing where the resource is located. A file could be on the user's own PC, or thousands of miles away on other servers.

Concurrency transparency enables several processes to operate concurrently using shared information objects without interference between them (e.g.: Automatic Teller Machine network). The users will not notice the existence of other users in the system (even if they access the same resources).

Replication transparency enables the system to make additional copies of files and other resources for the purpose of performance and/or reliability, without the users noticing. If a resource is replicated among several locations, it should appear to the user as a single resource (e.g. Mirroring - Mirror sites are usually used to offer multiple sources of the same information as a way of providing reliable access to large downloads).

Failure transparency enables the applications to complete their task despite failures occurring in certain components of the system. For example, if a server fails, but users are automatically redirected to another server and the user never notices the failure, the system is said to show high failure transparency. Failure transparency is one of the most difficult types of transparency to accomplish since it is hard to determine whether a server has actually failed, or whether it is simply responding very slowly. Moreover, it is generally unfeasible to achieve full failure transparency in a distributed system since networks are unreliable.

Migration transparency facilitates the resources to move from one location to another without having their names changed. (e.g.: Web Pages). Users should not be aware of whether a resource or computing entity possesses the ability to move to a different physical or logical location.

Performance transparency ensures the load variation should not lead to performance degradation. This could be achieved by automatic reconfiguration as response to changes of the load. (e.g.: load distribution)

Scalability transparency allows the system to remain efficient even with a significant increase in the number of users and resources connected (e.g. World-Wide-Web, distributed database)

IV WORLD WIDE WEB– A MASSIVE DISTRIBUTED SYSTEM

The Internet - a massive network of networks, connects millions of computers together worldwide, forming a network in which any computer can communicate with any other computer provided that they are both connected to the Internet. The World Wide Web (WWW), or simply Web, is a way of accessing information over the medium of the Internet. WWW consists of billions of web pages, spread across thousands and thousands of servers all over the world. It is an information-sharing model that is built on top of the Internet. The most well-known example of a distributed system is the collection of web servers. Hypertext is a document containing words that bond to other documents in the Web. These words are known as links and are selectable by the user. A single hypertext document can hold links to many documents.

The backbone of WWW are its files, called pages or Web pages, containing information and links to resources - both text and multimedia - throughout the Internet. Internet protocols are sets of rules that allow for inter-machine communication on the Internet. HTTP (HyperText Transfer Protocol) transmits hypertext over networks. This is the protocol of the Web. Simple Mail Transport Protocol or SMTP distributes e-mail messages and attached files to one or more electronic mailboxes. VoIP (Voice over Internet Protocol) allows delivery of voice communications over IP networks, for example, phone calls. A web server accepts HTTP requests from clients, and serving them HTTP responses along with optional data contents such as web pages.

The operation of the web relies primarily on hypertext as its means of information retrieval.

Web pages can be created by user activity. Creating hypertext for the Web is accomplished by creating documents with a language called hypertext markup language, or HTML. With HTML, tags are placed within the text to achieve document formatting, visual features such as font size, italics and bold, and the creation of hypertext links.

Servers implementing the HTTP protocol jointly provide the distributed database of hypertext and multimedia documents. The clients access the web through the browser software installed on their system. The URL (uniform resource locator) indicates the internet address of a file stored on a host computer, or server, connected to the internet.

URLs are translated into numeric addresses using the domain name system (DNS). The DNS is a worldwide system of servers that stores location pointers to web sites. The numeric address, called the IP (Internet Protocol) address, is actually the "real" URL. Once the translation is made by the DNS, the browser can contact the web server and ask for a specific file located on its site. Web browsers use the URL to retrieve the file from the server. Then the file is downloaded to the user's computer, or client, and displayed on the monitor connected to the machine. Due to this correlation between clients and servers, the web is a client-server network. The web is used by millions of people every day for different purposes including email, reading news, downloading music, online shopping or simply accessing information about anything. In fact, the web symbolizes a massive distributed system that materializes as a single resource to the user accessible at the click of a button. In order for the web to be accessible to anyone, some agreed-upon standards must be pursued in the formation and delivery of its content. An organization leading the efforts to standardize the web is the World Wide Web (W3C) Consortium.

Web Information Retrieval

Web information retrieval is the process of searching the world's largest and linked document collection – the World Wide Web, for information most relevant to a user's query. The various challenges of information retrieval on the web are:

(i) data is distributed - data spans over many computers, of a variety of platforms,
(ii) data is volatile - computers and files are added and removed frequently and unpredictably,
(iii) volume of data is very huge - growth continues exponentially,
(iv) data quality is inconsistent - data may be false, error-ridden, invalid, outdated, ambiguous and multiplicity of sources introduces inconsistency and
(v) heterogeneous data - multiple media types and media formats and multiple languages and alphabets. As a result, it would be physically unfeasible for an individual to sift through and examine all these pages to find the required information. Usually, in order to search for information on the internet a software tool called Search Engine is used. When a user enters a query into a search engine from their browser software, their input is processed and used to search the database for occurrences of particular keywords. A variety of search engines such as Google, Yahoo! Search, are available to make the web retrieval process very faster. Two main architectures used for web searching are centralized and distributed search.

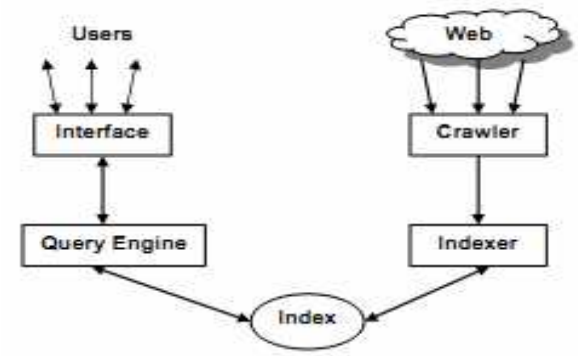


Figure 4 : Search Engine (Centralized Architecture)

Centralized Architecture: The aim of centralized approach is to index sizeable portion of Web, independently of topic and domain. The centralized architecture based search engine has main three parts: a crawler, an indexer, and query handler. The crawler (spider or robot) retrieves web pages, compress and store into a page repository. This process is called crawling (sometimes known as robot spidering, gathering or harvesting).

Some of the most well known crawlers include Googlebot (from Google) MSNBot (from MSN) and Slurp (from Yahoo!). Crawlers are directed by a crawler control module that gives the URLs to visit next. The indexer processes the web pages collected by the crawler and builds an index, which is the main data structure used by the search engine and represents the crawled web pages. The inverted index contains for each word a sorted list of couples such as docID and position in the document. The query engine processes the user queries and returns matching results using the index. The results are returned to the user in an order determined by a ranking algorithm. Each search engine may have a different ranking algorithm, which parses the pages in the engine's database to determine relevant responses to search queries. Some search engines keep a local cache copy of many popular pages indexed in their database, to allow for faster access and in case the destination server is temporarily inaccessible.

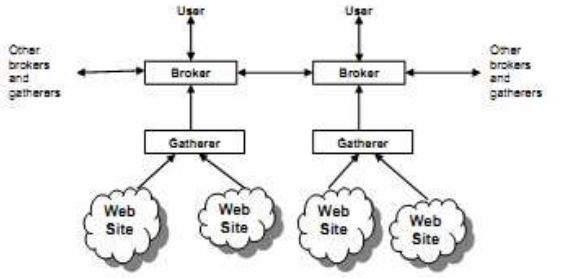


Figure 5: Search Engine – Distributed Architecture

Distributed architecture: Searching is a coordinated effort of many information gatherers and brokers. Gatherer extracts information (called summaries) from the documents stored on one or more web servers. It can handle documents in many formats: HTML, PDF, Postscript, etc. Broker obtains summaries from gatherers, stores them locally, and indexes the data. It can search the data; fetch data from other brokers and makes data available for user queries and to other brokers. The advantages of distributed architecture are the gatherer running on a server reduces the external traffic on that server and evading of gatherer sending information to multiple brokers reduces work repetition.

V CONCLUSION

In this chapter an overview of distributed systems are presented. The architecture, various characteristics are discussed.further to that world wide web and web information retrieval is also discussed in this paper. The future of distributed computing is still quite uncertain since it is one of many new types of computing. The technology has truly shown its worth as a useful tool for various complex applications.

VI REFERENCES

[1] The Object Management Group. "Common Object Request Broker: Architecture and Specification." OMG Document Number 91.12.1 (1991).

[2] [Parrington, Graham D. "Reliable Distributed Programming in C++: The Arjuna Approach." USENIX 1990 C++ Conference Proceedings (1991).

[3] Black, A., N. Hutchinson, E. Jul, H. Levy, and L. Carter. "Distribution and Abstract Types in Emerald." IEEE Transactions on Software Engineering SE-13, no. 1, (January 1987).

[4] Dasgupta, P., R. J. Leblanc, and E. Spafford. "The Clouds Project: Designing and Implementing a Fault Tolerant Distributed Operating System." Georgia Insti-tute of Technology Technical Report GIT-ICS-85/29. (1985).

[5] Microsoft Corporation. Object Linking and Embed-ding Programmers Reference. version 1. Microsoft Press, 1992.

[6] Linton, Mark. "A Taste of Fresco." Tutorial given at the 8th Annual X Technical Conference (January 1994).

[7] Jaayeri, M., C. Ghezzi, D. Hoffman, D. Middleton, and M. Smotherman. "CSP/80: A Language for Communicating Sequential Processes." Proceedings: Distributed Computing CompCon (Fall 1980).

[8] Cook, Robert. "MOD- A Language for Distributed Processing." Proceedings of the 1st International Conference on Distributed Computing Systems (October 1979).

[9] Birrell, A. D. and B. J. Nelson. "Implementing Remote Procedure Calls." ACM Transactions on Com-puter Systems 2 (1978).

[10] Hutchinson, N. C., L. L. Peterson, M. B. Abott, and S. O'Malley. "RPC in the x-Kernel: Evaluating New Design Techniques." Proceedings of the Twelfth Symposium on Operating Systems Principles 23, no. 5 (1989).

[11] Zahn, L., T. Dineen, P. Leach, E. Martin, N. Mish-kin, J. Pato, and G. Wyant. Network Computing Archi-tecture. Prentice Hall, 1990.