Journal of Computing and Information Technology - CIT 17, 2009, 4, 327–334
doi:10.2498/cit.1001319

UML Modeling of Network Topologies for Distributed Computer System

Vipin Saxena and Deepak Arora

Department of Computer Science, Babasaheb Bhimrao Ambedkar University (A Central University), Lucknow, India

Nowadays, distributed computer systems have become very popular approach of computing as it delivers high end performance at a low cost. In a distributed computing environment, autonomous computers are connected by means of a communication network, arranged in a geometrical shape called network topology. In the present paper a detailed study of network topologies is done for the distributed computer systems. A most popular Unified Modeling Language (UML) is used for modeling the different network topologies. A comparative study is done for 2D Mesh, Torus, and Hypercube network topologies and their performance is also evaluated after designing the UML Class, Sequence, and Activity diagrams for the same.

Keywords: network topology, class diagram, sequence diagram, activity diagram

1. Introduction

Distributed computing systems have become the essential aspect of growing information technology. The performance of any distributed system is certainly influenced by the technology, which we adopt in making network interconnections. Various researchers have done a lot of work on UML modeling, but limited research papers are available on UML modeling related to the distributed computer systems.

UML developed by Booch et al. [1] has proved itself as the most suitable visual presentation platform for modeling the real world problems. Conallen [2] has defined the extensions in UML for modeling the designing issues related to any web application architecture. Coulouris, G. et al. [3], has explored the major challenges regarding distributed systems through architectural and fundamental models; various technologies and case studies on Ethernet, wireless LAN and ATM are also discussed. Huang and Bode [4] compared the performance of ring and tree based network topologies for the distributed systems and also suggested a distributed management system for handling the failure. Issues related to the implementation of distributed systems are well explained by Milenkovic, M. [5]. OMG [6, 7] elaborate the latest UML specifications related to all real world problems, their modeling aspects and also the way of presentation for XML metadata specification in UML diagrams along with standard storage representations. Pllana and Fahringer [8] have suggested the way how one can do the UML modeling for high performance applications, which is an important reference in this regard. Further one more important reference by Pllana and Fahringer [9], describes the UML modeling of design issues related to the parallel and distributed applications. Another object-oriented distributed architecture system is recently defined by Saxena V. et al. [10] through UML modeling, with a case study of a multiplex system.

In the present paper, three important network topologies namely 2D Mesh, Torus and Hypercube are compared, which are commonly used for the concurrent processes execution in a distributed environment. In the paper, UML modeling is done for these three network topologies and the sequence diagram for process execution under distributed environment is also represented. The performance comparison is also studied by taking the variations in nodes for 2D Mesh, Torus and Hypercube topologies.

2. Background

2.1. Distributed Computer System

A distributed computer system consists of autonomous computers, connected via network. The computers involved in the distributed system can share both the localized as well as remote resources. However, it is quite expensive to access any remote resources in terms of CPU overhead and communication cost. Nowadays, a trend to develop distributed system is increasingly moving ahead due to its features like high performance computing, scalability, security and reliability.

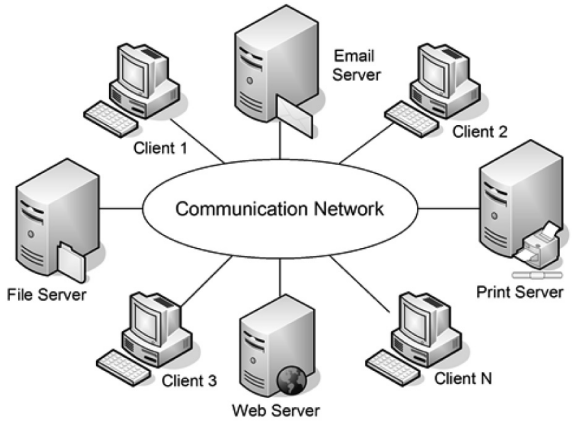

Figure 1. Distributed computer system.

Nowadays computation labs are adopting the concept of distributed computer systems, which is shown below in Figure 1. This figure shows the connection of 'N' autonomous clients to different servers connected through a communication network. This network can be based upon any kind of network topology as required, in terms of complexity, reliability and security.

2.2. Process Definition

Let us explain the definition of a process, which is the basic entity of execution in any distributed computing environment. The process is explained as a macro, subprogram, subroutine or block of code etc., which has an identification number called as process_id. A processing unit is defined as a Process_Execution_Controller (PEC), which accepts process_id through threading and executes the corresponding process after getting the required resources. The UML class diagram of process consists of a number of attributes, as shown in Figure 2(a). The instance of the process and multiple instances through the objects are also represented in Figure 2(b) and 2(c), respectively. This completes the process representation in a distributed environment. In the paper, each process is controlled by the separate threads, which is going to be executed under distributed environment. The parallel region is defined as a buffer, which is handled by the multiple threads, associated to the different processes. A stereotype parallel region UML class diagram is shown in Figure 2(d) and associated activities representation is given in Figure 2(e). In this figure the sub activities can be executed many times in concurrent fashion by the use of buffer, called code region.

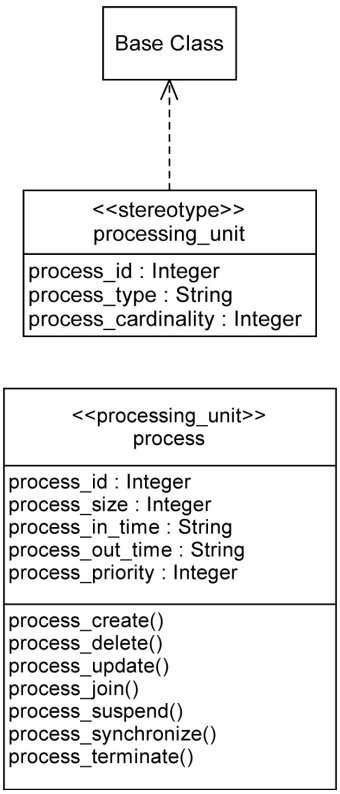

Figure 2a. UML class diagram of process.

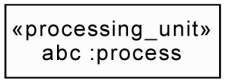

Figure 2b. Instance of process.

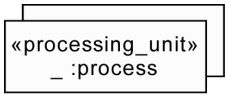

Figure 2c. Multiple instance of process.

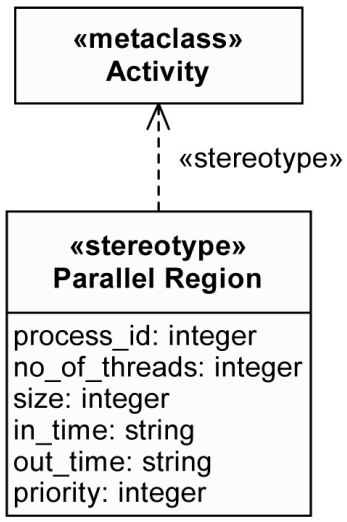

Figure 2d. UML class diagram of stereotype parallel region.

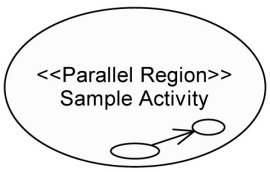

Figure 2e. Sample activity diagram.

2.3. Topology

The processes which are going to be executed, handled by the multiple threads, can use the parallel region in a concurrent manner. For the execution and communication purpose, these processes need to be arranged in a network topology. By the use of topology, the process can access the code region as per the availability of the resources, for faster execution.

In the present approach, different kinds of topologies are selected, which are widely used in setting the labs for distributed computations. Through these topologies, which are executed in concurrent fashion, one can save a lot of process execution time. The UML class diagrams for these three kinds of topologies are given in Figure 3(a), 3(b), and 3(c) for 2D Mesh, Torus, and Hypercube topology respectively. For the execution of process having unique process_id, the resources can be granted by the individual computer system or can be taken from different clients as per the availability of the network, which consists of autonomous computer systems arranged in the distributed computing environment.

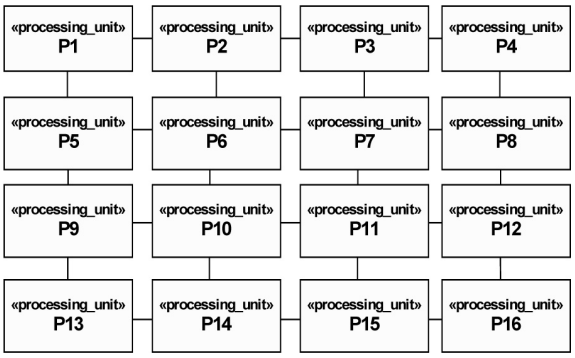

Figure 3a. UML class diagram of 2d mesh topology.

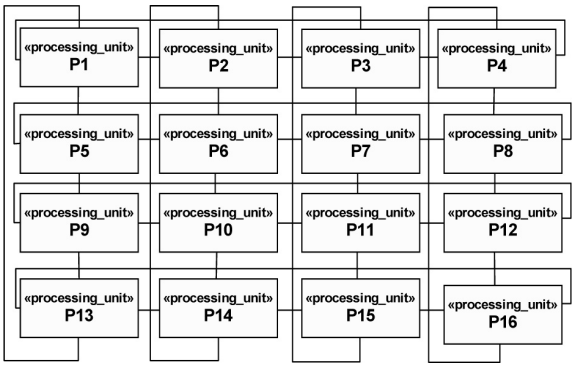

Figure 3b. UML class diagram of torus topology.

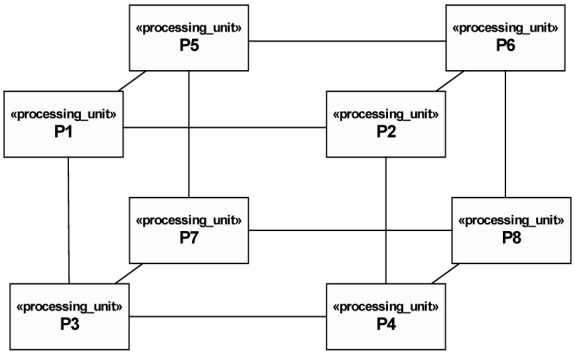

Figure 3c. UML class diagram of hypercube topology.

2.4. Communication Lines

The communication among the processes can be handled by the synchronization technique if the common resources are available at the client computer system. The communication can be of two types; one approach is point-to-point protocol method and another technique is broadcast protocol method. The communication lines can be connected as per the network demand and its topological specifications. UML class diagrams for broadcast and P2P signal are given in

Figures 4(a) and 4(b). Signal event for process communication via broadcast and P2P approach are given in Figures 4(c) and 4(d) respectively.

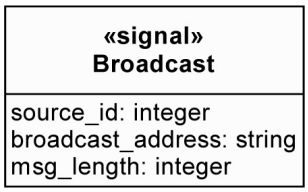

Figure 4a. UML class diagram of broadcast signal.

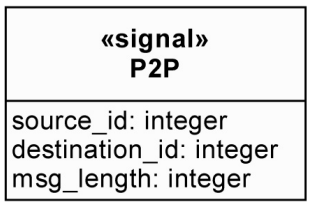

Figure 4b. UML class diagram of P2P signal.

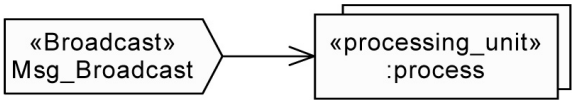

Figure 4c. Signal event for process communication via broadcast approach.

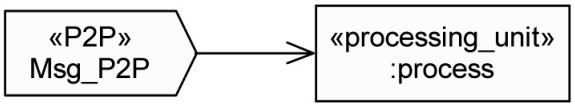

Figure 4d. Signal event for process communication via P2P approach.

3. UML Class Diagram for Distributed Computer System

In a distributed computing environment, messages get communicated to and from the nodes in the form of signals for the process synchronization.

In Figure 5, one can see that class Process_Execution_Controller (PEC) is directly interacting with the class Process, Communication_Line, Processor, Memory and D_Cache (Data Cache), I_Cache (Instruction Cache). Class Process is responsible for all the process-related tasks like process creation, deletion and synchronization etc.

Here, class I_Cache is responsible for caching the instructions whereas D_Cache is responsible for caching the data. The PEC would

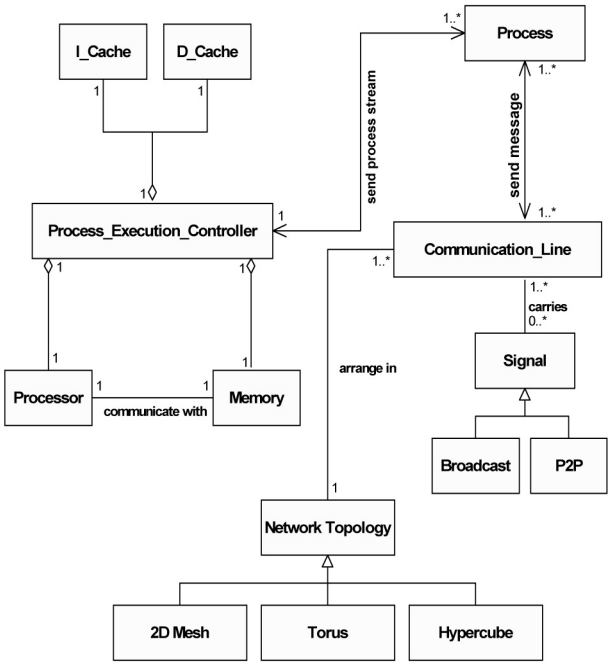

Figure 5. UML class diagrams of distributed computer system.

store/search the needed instruction or data during the process execution with the help of Processor and Memory. Here, PEC is fully responsible for any processes. Now, if one considers the communication lines, they must arrange in some topology, through which a message can refer to any process in the distributed environment. Also the process referred by message can exist on any of the nodes in the distributed environment.

So, the network topology adopted for the communication lines becomes a key point for measuring the performance of any distributed computer system. The authors have chosen three topologies, 2D Mesh Torus, and Hypercube for their purpose. The Broadcast and P2P are the classes which handle the broadcasting and point-to-point relay of communication signals over communication lines.

Also, one can see in Figure 5, class Process is directly communicating with the class Communication_Line for sending and receiving messages to or from other processes running concurrently in the distributed environment. Also the class Communication_Line will directly interact with the class Signal and Network_Topology, for getting the signal and topological specifications of communication network.

4. UML Activity Diagram for Distributed Computer System

Processes, which are going to be executed in a parallel fashion, must share a common parallel region. In Figure 6, it is shown that any subactivity belonging to any process can share the parallel region for executing in parallel along with other subactivities from SA_1, SA_2. . .. . .SA_n, which can belong to any process from P_1, P_2. . ...P_n. This will facilitate the process to share a common parallel region, to be executed with multiple instruction stream too. In Figure 6, one can see that it can be shown by using fork operation. When fork operation is applied on subactivity, it breaks apart into two different subactivities and starts executing parallely by sharing a common parallel region. After completing the most nested parallel subactivity, the joint operation can be performed. Through joining the result from each parallel subactivity, one can reach the final result of the parent subactivity. In this case, the subactivities executing in parallel can be applied on multiple data stream too.

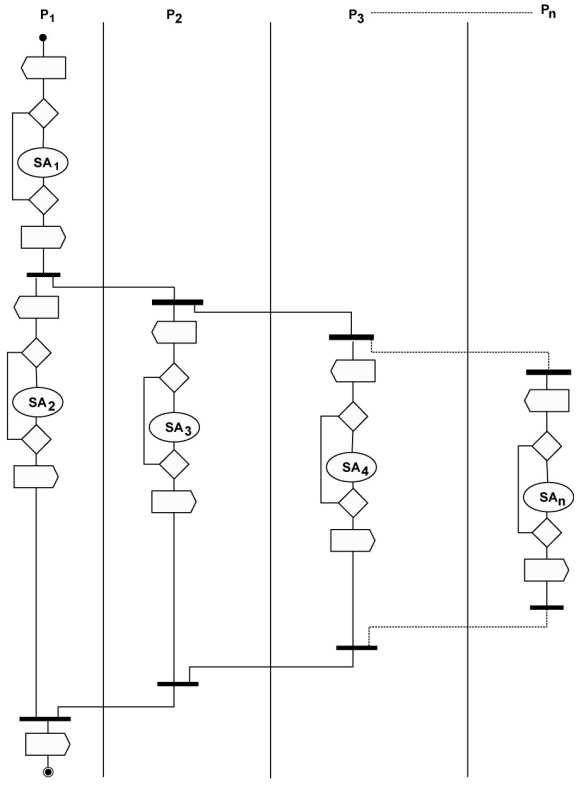

Figure 6. UML activity diagram for process under distributed environment.

5. UML Sequence Diagram for Process Execution in Distributed Computer System

Communication channel carries the messages in the form of signals among the process being executed concurrently in the distributed computing environment. In Figure 7 it is shown how the messages are passed among various classes like Process, Communication_Lines, Process_Execution_Controller, D_Cache, I_Cache, Processor and Memory, involved in process execution under distributed computer system. After receiving message from Communication_Line, the process stream, handeled by Process class, is submitted to the PEC for execution. When PEC receives complete process stream, it will start to cache the needed instructions and data separately into I_Cache and D_Cache respectively. Now PEC sends the message to the Processor to start decoding and after decoding the required operands and instructions are loaded into the memory taken from I_Cache and D_Cache with the help of PEC. Now PEC sends the message to the Processor and Processor starts the process execution. During the execution, Processor uses the Memory for saving intermediate results, data and instructions. After completing the execution, the final result would be saved in the Memory and Processor sends an execution completed message back to the PEC. Now PEC will also send back the executed process information to the Process.

6. Comparison of Space Complexities among Topologies

Topology	One to One	One to All	All to All
2D Mesh	$2(\sqrt{n}-1)$	$2(\sqrt{n}-1)$	$2x\lceil(\sqrt{n}/2)\rceil$
Torus	$\lfloor\sqrt{n}/2\rfloor$	$\lceil\sqrt{n}/2\rceil$	$2x\lceil\sqrt{n}/2\rceil$
Hypercube	$\text{Log}_2\ n$	$\text{Log}_2\ n$	$2\ \text{x}\ \text{Log}_2\ n$

Table I. Space complexities for 2D mesh, torus and hypercube.

Comparison among the performance of network topologies is done for 2d Mesh, Torus and Hy-

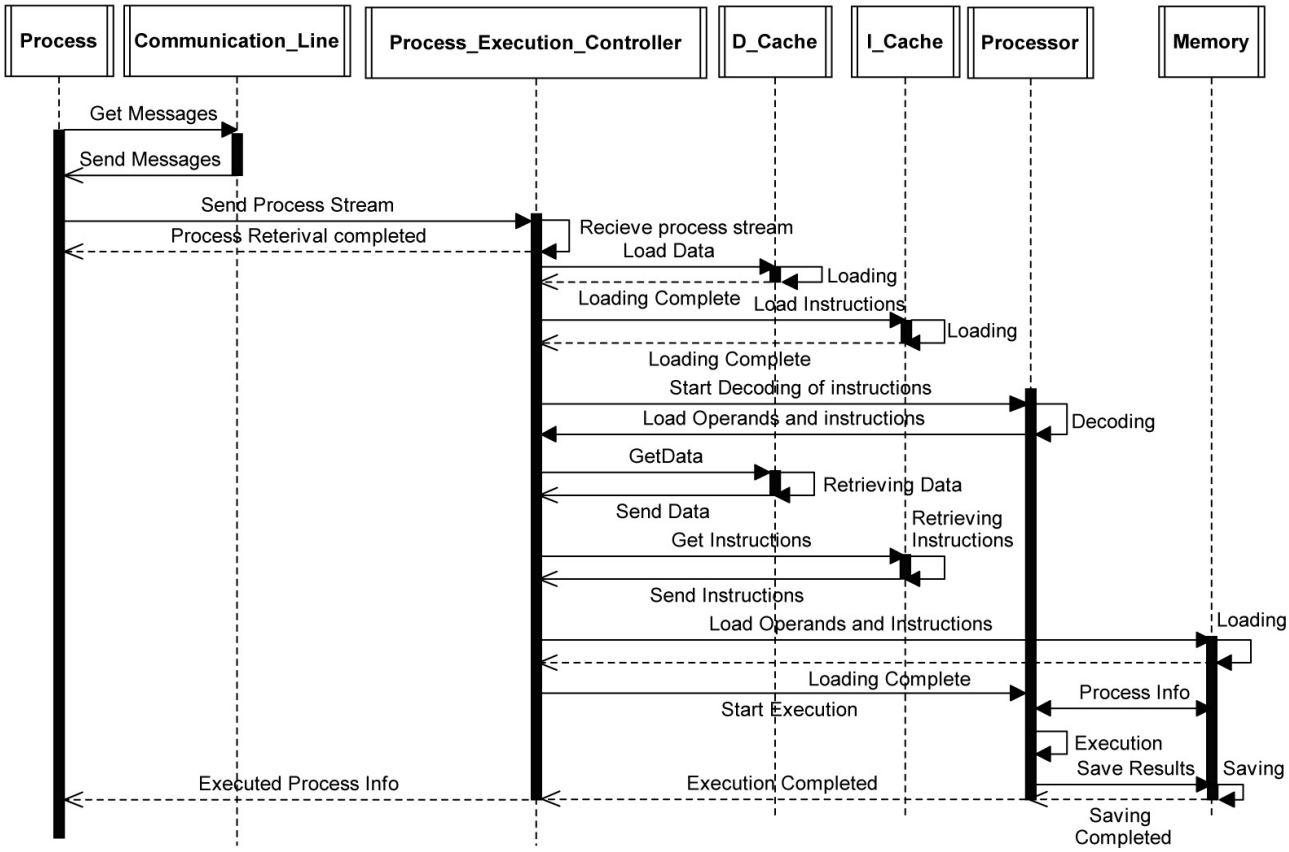

Figure 7. UML sequence diagram for process execution under distributed computer environment.

percube technologies. The communication is considered as one to one, one to all and all to all. Table I gives the measurements of space complexities for 2DMesh, Torus and Hypercube technologies. On the basis of this table, Table II, III and IV are designed, which shows the comparison of space complexities as number of nodes in the network system under distributed environment is increasing. As nodes are increasing, the performance of Torus topology comes best over the 2D Mesh and Hypercube network technologies since the space complexity is low in case of Torus topology for one to one communication. In case of one to all communication system, behaviour of Torus and Hypercube is the same whereas, in case of one to all, performance of 2D Mesh and Torus topologies is the same. This is also depicted in Figure 8, in which it is studied that, in general, performance of Torus topology is better in comparison with 2D Mesh and Hypercube technologies.

(MXN)	No. of Nodes	2D Mesh		
		One to One	One to All	All to All
2x2	4	2	2	2
2x3	6	3	3	4
2x4	8	4	4	4
2x5	10	4	4	4
2x6	12	5	5	4
2x7	14	5	5	4
2x8	16	6	6	4

Table II. Space complexities for 2D Mesh topology.

(MXN)	No. of Nodes	Torus		
		One to One	One to All	All to All
2x2	4	1	2	2
2x3	6	1	3	4
2x4	8	1	3	4
2x5	10	1	3	4
2x6	12	1	4	4
2x7	14	1	4	4
2x8	16	2	4	4

Table III. Space complexities for Torus topology.

(MXN)	No. of Nodes	Hypercube		
		One to One	One to All	All to All
2x2	4	2	2	4
2x3	6	3	3	5
2x4	8	3	3	6
2x5	10	3	3	7
2x6	12	4	4	7
2x7	14	4	4	8
2x8	16	4	4	8

Table IV. Space complexities for Hypercube topology.

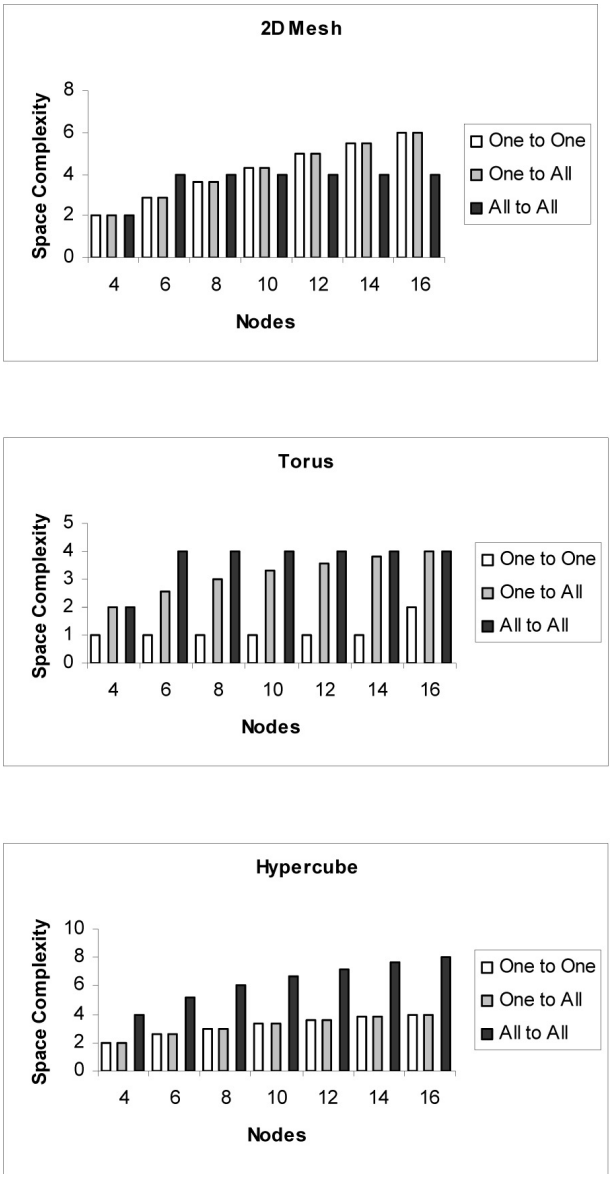

Figure 8. Comparison of space complexities.

7. Conclusion and Future Scope of Work

From the above work, it is concluded that UML is one of the important modeling languages used for the visual representation of research problem/software design. In this paper, performance of three kinds of topologies i.e. 2D Mesh, Torus and Hypercube is considered under the distributed environment and it is concluded that the Torus topology is the best kind of arrangement of computer system under distributed environment since the space complexity is less as the nodes are increasing in comparison with other topologies. Since the present work is only confined to the static interconnection of the network topologies, this research work can also be extended further, by considering the dynamic arrangement of interconnection of computer network.

References

[1] BOOCH, G., RUMBAUGH, J., JACOBSON, I., The Unified Modeling Language User Guide, Addison Wesley, Reading, MA, 1999.

[2] CONALLEN, J., Modeling Web Application Architectures with UML. *Communications of the ACM 42(10)*, (1999), pp. 63–70.

[3] COULOURIS, G., DOLLIMORE, J., KINDBERG, T., Distributed Systems, Concepts and Design, Third Edition, Addison Wesley, 2000.

[4] HUANG, M., BODE, B., A Performance Comparison of Tree and Ring Topologies in Distributed System. *Proceedings of the IEEE, Parallel and Distributed Processing Symposium*, (2005), Washington, D.C.

[5] MILENKOVIC, M., Operating Systems: Concepts and Design, Tata Mcgraw-Hill, 1997.

[6] OMG, Unified Modeling Language Specification. Available online via `http://www.omg.org` (2001).

[7] OMG, OMG XML Metadata Interchange (XMI) Specification. Available online via `http://www.omg.org` (2002).

[8] PLLANA, S., FAHRINGER T., On Customizing the UML for Modeling Performance-oriented Applications. *Proceedings of the <<UML>>2002,"Model Engineering Concepts and Tools"*, Springer-Verlag, (2002), Dresden, Germany.

[9] PLLANA, S., FAHRINGER T., UML-based Modeling of Performance-oriented Parallel and Distributed Applications. *Proceedings of the Winter Simulation Conference*, Vol. 1, Issue. 8–11, Dec. 2002, (2002), pp. 497–505.

[10] SAXENA, V., ARORA, D., AHMAD S., Object-oriented Distributed Architecture System through UML. In *Proceedings of the IEEE International Conference on Advances in Computer Vision and Information Technology (2007)*, Aurangabad (MS), India.

Received: August, 2008
Accepted: February, 2009

Contact addresses:
Dr. Vipin Saxena
Department of Computer Science
B.B. Ambedkar University (A Central University)
Vidya Vihar Rae Bareilly Road
Lucknow U.P. 226025, India
e-mail: vsax1@rediffmail.com

Deepak Arora
Department of Computer Science
B.B. Ambedkar University (A Central University)
Vidya Vihar Rae Bareilly Road
Lucknow U.P. 226025, India
e-mail: deepakarorainbox@gmail.com

VIPIN SAXENA is a Reader, Dept. of Computer Science, Babasaheb Bhimrao Ambedkar University, Lucknow, India. He got his M.Phil. Degree in computer application in 1991 & Ph.D. Degree in scientific computing from the University of Roorkee (renamed as Indian Institute of Technology, India) in 1997. He has more than 12 years teaching experience and 16 years research experience in the field of scientific computing & software engineering. Currently he is proposing software designs by the use of Unified Modeling Language for various research problems related to the software domains & advanced computer architecture. He has published more than 55 international and national publications.

DEEPAK ARORA is a Research Scholar, Dept. of Computer Science, Babasaheb Bhimrao Ambedkar University, Lucknow, India. He got his Master Degree in computer applications in 2003 and M. Phil. Degree in computer science in 2006. Currently he is actively engaged in the research work on distributed computing systems through the Unified Modeling Language. Ha has produced several outstanding publications on Distributed Computing Systems. Also, he is a member of 'The Indian Science Congress Association', Kolkata, India and 'Computer Society of India', Mumbai, India.