International Journal of the Physical Sciences Vol. 6(10), pp. 2348-2355, 18 May, 2011
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS11.483
ISSN 1992 - 1950 ©2011 Academic Journals

Full Length Research Paper

A novel cost-based framework for communication in computational grid using Anycast Routing

Abid Ali Minhas[1], Fazl-e-Hadi[1]* and Shakir Ullah Shah[2]

[1]Department of Graduate Studies and Applied Sciences, Bahria University, Islamabad, Pakistan
[2]Iqra University, Islamabad, Pakistan.

Accepted 22 April, 2011

Computational grid can perform the computationally extensive jobs by utilizing the wide spread processing capabilities of volunteer processors. In order to utilize the wide spread resources, failure options cannot be ignored. In this paper we give the detailed implementation of fault tolerance techniques, and also propose a modified forwarding mechanism which forwards the request to the next hop from which more receivers are available, the main contribution of this paper is the implementations of fault tolerant techniques using anycasting with modified forwarding mechanism and its analytical analysis for the computational grid. The study analyses the communication cost involved in the proposed scheme and its comparison with the previous techniques. The implementation of the computational grid is carried out in Alchemi toolkit which is based on Microsoft .Net framework.

Key words: Anycast, computational grid, communication cost.

INTRODUCTION

The Grid Technologies have greatly been applied to a number of different sectors especially for flexible and secure resources/power sharing/aggregation in coordinated manner. It can be used over the Internet and Intranet for the developing complex applications e.g. collaborative scientific simulation, distributed mission training, analysis of elementary particle physics etc. Grid computing fulfills the requirements of these applications such as high speed of parallel computing, storage capacity and network bandwidth. The basic idea behind the grid is not to buy additional resources but to use free cycles of existing underutilized grid resources. Grid can be classified, according to the types of shared recourses and their functionalities, into computational Grid, Data Grid, Storage Grid, Equipment Grid, knowledge Grid, Interaction Grid (Venugopal et al., 2006) etc.

Computational grid is a kind of distributed and parallel computing. It is composed of heterogeneous, geographically distributed resources connected by unreliable network media. It is used to solve the complex problems, in sophisticated and user friendly manner, which require large amounts of computing resources. A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities (Foster and Kesselman, 1999; Foster et al., 2001). It is used to increase the performance and reduce the cost of computer hardware and software in a variety of ways.

The components of a grid may be heterogeneous and dynamic in nature. Similarly packet loss is also common in a long range of geographically distributed network and heterogeneous in nature; so user assigned jobs are always prone to different type of failures, errors and faults (Tanimura et al., 2006). Fault tolerance is an important service of grid, which insures the delivery of a service despite the presence of fault. The issue of fault tolerance in grid computing is higher than traditional parallel computing (Nguyen-Tuong, 2000; Waheed et al., 2000; Medeiros et al., 2003), which include wide range of errors, failures and faults (Medeiros et al., 2003) and shows fragility of grid environment. So the fault tolerance

*Corresponding author. E-mail: fhadi76@yahoo.com.

becomes of very much important. Fault tolerance techniques can be categorized into reactive and proactive ones.

Retry: If a job fails its execution on a machine due to any type of failure, error or fault, it can be completed by other machine by submitting from the start. Submitting a failed job from start is called retry. This technique may not be suitable for jobs which require huge computational resources. Nazir and Khan (2006) proposed a proactive approach for scheduling jobs in computational grid. In this technique, history is maintained about the grid resources and jobs are scheduled according to their history.

Check pointing: is a common and an efficient technique to save the state of the computation on stable storage periodically (Roman, 2003; Krishnan, 2004). It is used to resume the job execution from the previous consistent stored state rather than from the beginning. It increases the application response time and improve the efficiency of a system. It helps in load balancing by migrating jobs from loaded machine to less loaded machines. Similarly it helps in fault resiliency by migrating a job from faulty to stable machine. It is mainly used for long running jobs to save the work to be recomputed from the beginning. The idea proposed in (Muhammad and Ansari, 2006) multicast technique was used, which multicasts address of executer machine in order to select backup machine. This technique causes the following problems:

1) Data loss or delivered out of order will increase unreliability.
2) Increases the network traffic delays.
3) Most multicast servers do not discriminate any clients.

Therefore it is easy to join a group and watch the data that is being sent to it, that is, Distributed Denial of Service {D (DOS)} attack is possible. The authors used multicast for backup selection of the executor machines by transferring a single packet to multiple recipients which causes the above mentioned problems.

Some authors used the Anycast for service adaptability like (Szymaniak et al., 2007). The idea of anycast for fault tolerance is available in (Imran et al., 2007) but the authors did not analyze the anycast communication against the multicast. In this study we have proposed in anycast backup selection mechanism with Receiver Based Forwarding (RBF). The study also analyzes the communication cost involved in multicast and anycast scenarios. The analysis has been done by creating a computational testbed using Alchemi middleware. The test bed results and mathematical proof shows that the anycast communication is far better for the selection of backup machines.

Moreover anycast provides not only better efficiency but also enhances the security. A test bed of grid has been developed for the analysis of communication cost. Alchemi executors were installed on different geographically distributed PCs. Alchemi Manager was installed on local powerful machine. They were connected through Local Area Network (LAN) having speed of 100 MB/s. The detail is given in Table 1.

The rest of the paper is organized as follows: First is a description of the related work, followed by that of the multicast and anycast techniques for backup selection. The mathematical model is then presented, after which the performance evaluation is demonstrated, and finally conclusion of the research with indication of directions for future works in this field.

RELATED WORK

The Grid Middleware plays a vital role in grid infrastructure and to be deployed on each machine to make it as a part of grid. There are various grid middleware which provides different functionalities. Globus Toolkit (Foster and Kesselman, 1997) and Alchemi (Luther et al., 2004) are famous example of middleware among open source middleware. Grid architecture is divided into different layers, each performing a specific functionality as depicted in Figure 1. The upper layer is application layer and user-centric. It is used for a variety of applications, engineering, science, business etc. Users interact with this layer via their browsers. The middleware layer brings different elements intelligently. The lower layer is hardware centric which is used to establish the connectivity among grid resources.

Architecture of Alchemi

Alchemi is .NET based desktop grid framework running on Windows-based environment and is very user friendly. It is used for flexible, platform independent, achieving high throughput, object oriented thread based applications (fine grained abstraction) and file based jobs (course grained abstraction). Thread is basic unit of grid application. Manager, Executor, owner and cross-platform manager are the component of Alchemi.

i) Manager: Execution is the responsibility of manager. Executors dedicatedly or non-dedicatedly register with manager. It creates threads for a job and distributes it for execution on available executors. It submits the result of executed threads to the owner.
ii) Executor: Executor executes the grid threads dedicatedly or non-dedicatedly. It receives threads from manager and executes it.
iii) Owner: Owners submit jobs to the Manager and inquire

Table 1. Nodes participating in the computational grid with their specification.

Nodes	Specification	Platform	No. of hops away from the Manager
Manager	P IV, 512 RAM, 3.2 GHz	Windows XP	--
Executer 1	P IV, 512 RAM, 1.73 GHz	Windows XP	3
Executer 2	P IV, 512 RAM, 2.0 GHz	Windows XP	4
Executer 3	P IV, 1 G RAM, 3.0 GHz	Windows XP	9
Executer 4	P IV, 1 G RAM, 3.0 GHz	Windows XP	4
Executer 5	P IV, 512 RAM, 3.0 GHz	Windows XP	8
Executer 6	P IV, 512 RAM, 3.0 GHz	Windows XP	10

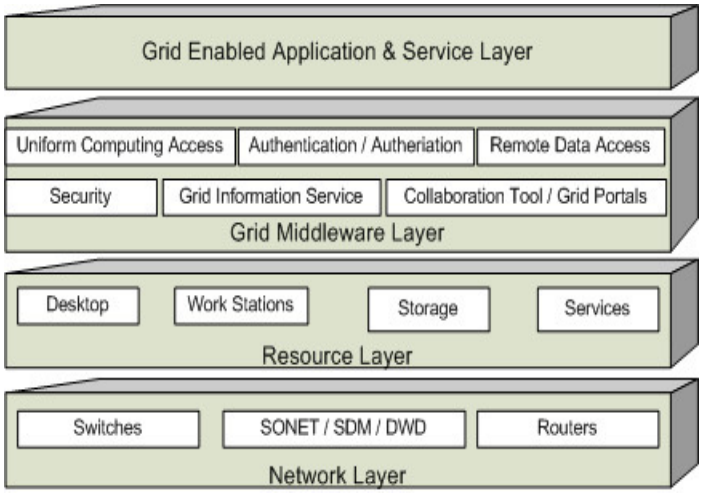

Figure 1. Grid architecture layers.

about the status of their jobs.

Alchemi supports .NET based distributed system which is a collection of independent and distributed processing components, that is, nodes connected via LAN/WAN as depicted in Figure 2. Nodes can share their resources among themselves through centralized authority, that is, Manager. The following steps describes the functionality of a grid application:

1) Owner (n) will make a request to Manager to solve a problem.
2) Manager will create threads and distribute to all available Executors.
3) Executors will send the result of a thread to Manager.
4) Manager will send the result of all threads to Owner (n).

PROGRAMMING MODEL

Alchemi support two types of parallel programming models: Job and Thread models (Luther et al., 2004). A brief description of these models is given below:

Job model: Alchemi supports highly level of abstraction in which the smallest unit of parallel programming is a process. It is called course-grained abstraction. In this approach of parallel programming, grid application deals with files like input, output and executable files (process). This model of parallel programming is very complex and is not flexible. This model helps develop legacy tasks using different languages like C, C++, Java. Cross platform Manager manages the legacy tasks using ASP.NET web services.

Thread model: The primary programming model of Alchemi is multi-threaded parallel programming which is more of low level and is called fine-grained abstraction. In this approach of parallel programming, the basic unit is Thread (independent unit of work). This approach is more powerful, easy to use and flexible. There are different reasons for the selection of Alchemi for creating

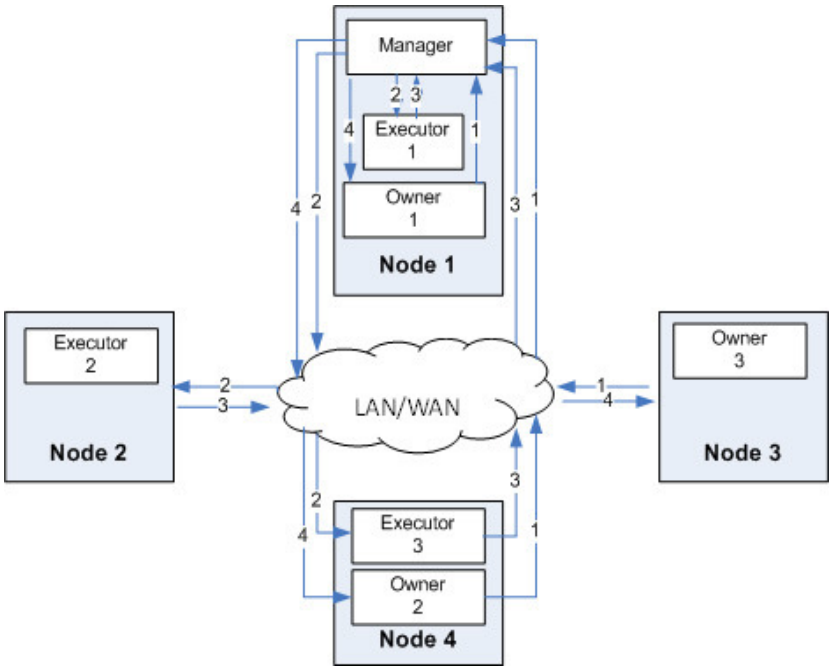

Figure 2. Block diagram of Alchemi.

computational grid. Some of these are given:

i) Globus toolkit is Linux-based and Alchemi is windows based. Most of the systems are windows based, so achieving better results, Alchemi was preferred.
ii) It is an open source middleware, so one can bring changes according to ones needs.

Fault tolerance in Alchemi

Due to heterogeneous and dynamic nature of resources in computational grid, faults, errors and failure become likely. Many middlewares have different level of fault tolerance but most of them are not fully fault tolerant. Fault tolerance in Alchemi requires much work to fulfill the challenges. Heart beating is the basic fault tolerance technique of Alchemi. Executor sends heartbeat signals to Manager periodically. When Manager receives these signals, it predicts that executor is alive and working. It is not necessary that executor is in good operational form.

MULTICAST AND ANYCAST

Figure 3 shows the backup machine selection using multicast technique. Alchemi manager distributes the executing threads to the executor 1, 2 and 3. In order to select a backup executor, the executing machines multicasts the 'backup request' to the available volunteers, that is, Backup 1, 2 and 3 Alchemi executors. The backup executors will reply back to sending machines. The executor will select a backup machine based upon the machine's configuration. This activity creates a lot of traffic which includes backup requests and redundant backup replies. The scenario is also vulnerable to D(DOS) attack in which a hacker will spoof the IP of requesting executor machine and will send its multiple backup machines in the network which will reply back to victim executor machine and will launch D(DOS) attack. As the network media is not reliable, huge traffic will also cause the media failure which will stop future communication of the computation grid.

Figure 4 shows the backup machine selection using anycast technique. Alchemi manager distributes the executing threads to the executor 1, 2 and 3. The Alchemi manager and executor will coordinate with a separate utility for finding the distance (number of hops) of the available backup machines. In order to select a backup executor, the executing machines will select a nearest backup machine e.g. 'Backup 1' will be a backup machine for 'Executor 1' rather than 'Backup 2 and 3' based upon the distance (number of hops) from 'Executor 1'. In anycast communication, the executor machines will coordinate with their relevant backup machines rather than create the redundant multicast requests and replies. The one to one anycast communication will minimize the chances of D (DOS) attack. As far as the communication

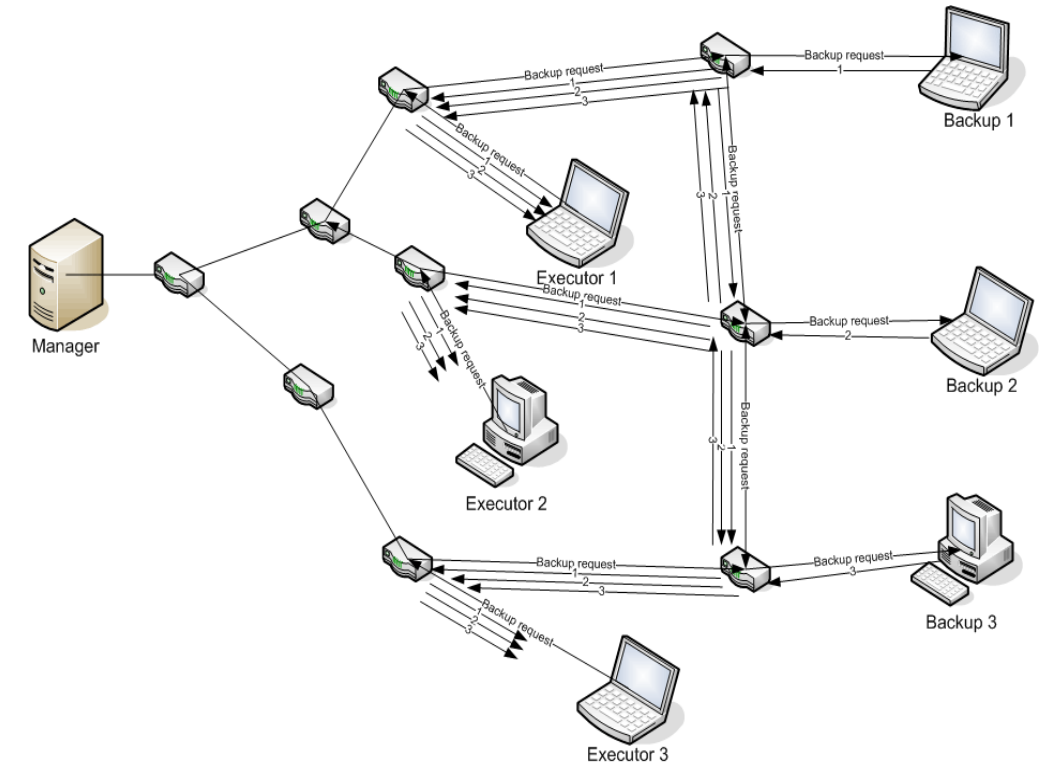

Figure 3. Backup selection using multicast.

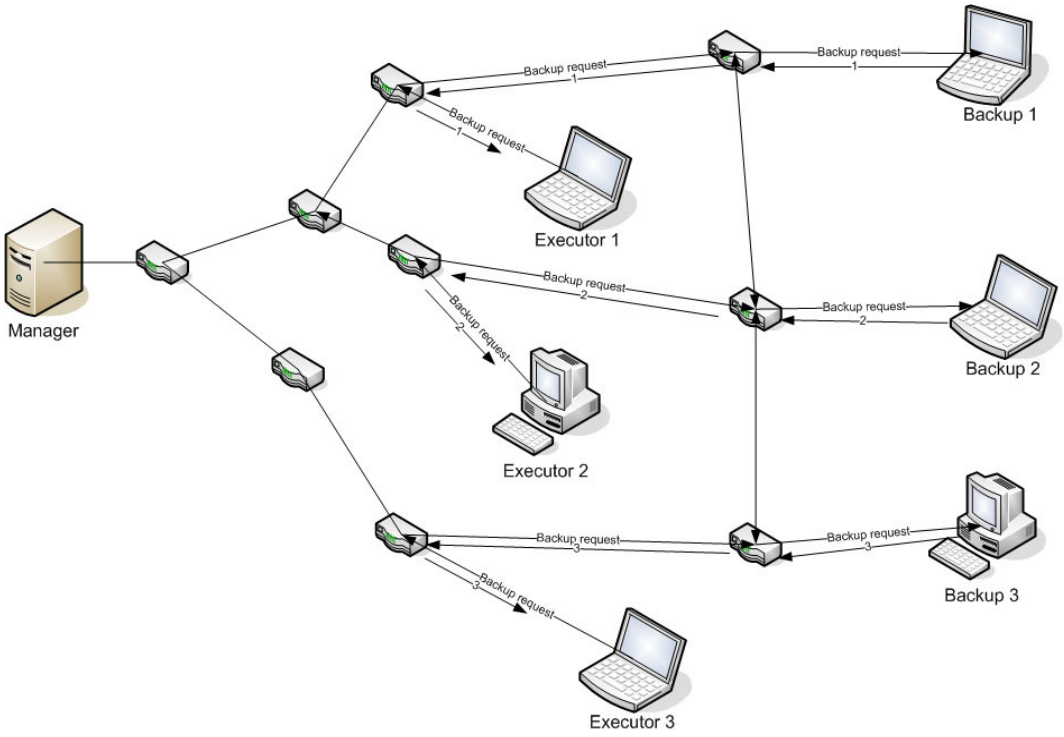

Figure 4. Backup selection using Anycast.

cost is concerned, it is substantially decreased in case of anycast communication. In order to maximize the probability of backup machine's availability due to unreliable network media, we are using our own proposed receiver based forwarding (RBF) (Fazle et al., 2007) mechanism while forwarding the anycast packet to the next hop. The RBF selects that next hop through which more anycast receivers are reachable.

Mathematical model

The generalized mathematical model depicting the communication cost for multicast, anycast and anycast with RBF is given below:

Let P = Set of all nodes in population
N= Number of nodes in P
n = Number of groups available in P
$g = \{g_1,g_2,g_3,....,g_i,....,g_n\}$: set of groups where i=1,2,3,..n; for any group g
$g= \{d_1,d_2,....,d_m\}$: set of available nodes in group g where d_1....to ...d_m are nodes in g_i
m=total number of nodes in g_i
t_i= Time required to reach a packed at d_i
D_i = Distance of a node d_i from source (Venugopal et al., 2006).

Multicasting

As all the members will receive the packet, so in this case:

Delay = Max(t_i)
or
$\text{Delay}= \text{Max}\{t_1,t_2,....t_m\} = t_{Max}$ (1)

Anycasting

In Anycasting, any one member among the groups will receive the packet.
Select i: D_i is min [send packet to di]

Case 1: select a node randomly then
$\text{Delay} = \text{Average}\ \{t_1, t_2, tm\} = t_{Avg}$ (2)

Case 2: select a node on FIFO rules or any other then

$\text{Delay} = \text{Min}\ \{t_1, t2\ ...tm\} = t_{Min}$ (3)

Comparison of Anycast and Multicast with respect to its delay

From Equations 1, 2 and 3, the following relationship between multicast and anycast is obtained:

$\text{Min}(t_1) <= \text{Average}(t_i) <= \text{Max}(t_i)$ (4)

Where equality holds only when

i) All nodes in g_i are at equiv distance from source but even in this case network traffic is substantially increased in case of multicast.
ii) A group may have only one node

It is clear from Equation 4 that anycast communication cost will always be less than the multicast communication.

Anycast with RBF

Receivers Based Forwarding (RBF) considers the number of anycast receivers available through a link as well as the path length to the nearest receiver through that link in deciding the next hop while forwarding an anycast packet (Fazle et al., 2007).

Condition: Select all shortest path with maximum number of receivers.

Let H= {h1, h2, h3,....L}: set of next hops which satisfy the stated condition
Let Ri = No. of receiver available at hi

Where hi € H and i =1, 2, 3.....L

Select i : Ri is Max [Send the packet to hi € H]

Probability of anycast without using RBF

The following three cases are possible while using anycast without RBF:

Average case:

$$\frac{\overline{R}}{\sum Ri} \quad (5)$$

Worst case:

$$\frac{MinR}{\sum Ri} \quad (6)$$

Best case:

$$\frac{MaxR}{\sum Ri} \quad (7)$$

A. Scenario

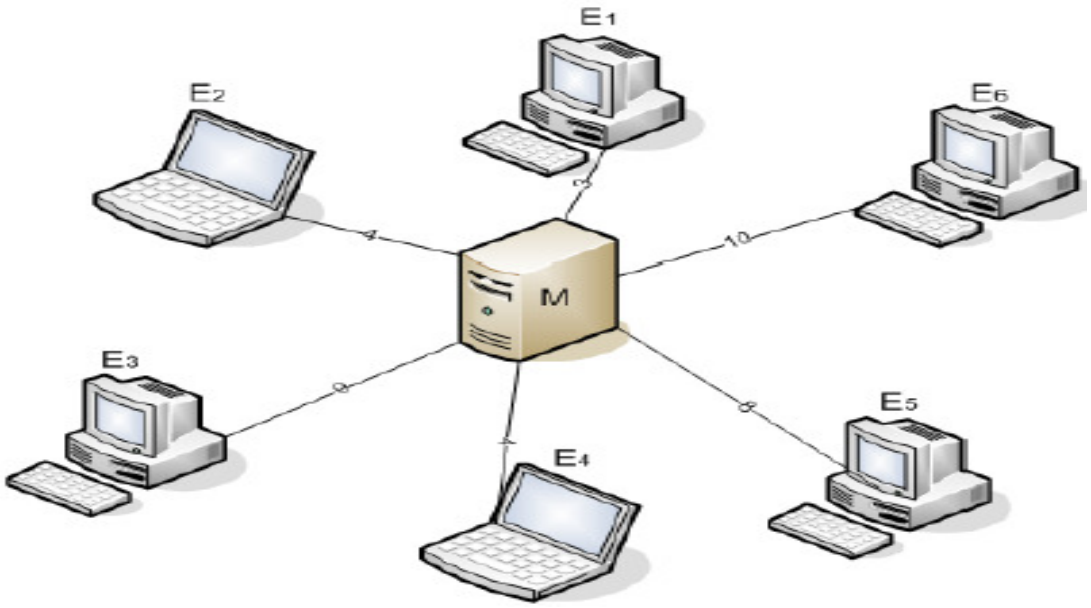

Figure 5. Scenario visualization.

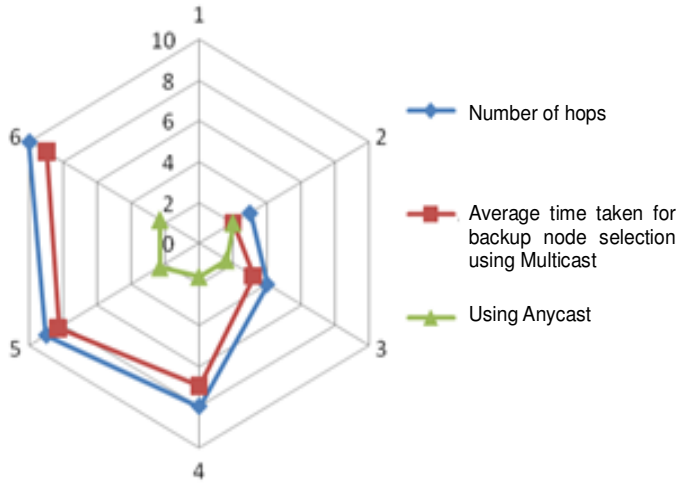

Figure 6. Cost analysis using Multicast and Anycast.

Probability of anycast with using RBF

Equation 8 shows the probability of anycast using RBF.

$$\frac{MaxR}{\sum Ri} \quad (8)$$

Equation 8 shows that it is always the best case of any casting without using RBF.

In Figure 5, 6 executors are connected with a single manager. The weight on edges shows the number of hops away from manager. The scenario is developed with central manager at Riphah International University, Islamabad, Pakistan (University), with the distributed executers at various universities in Pakistan and other parts of the world.

PERFORMACE EVALUATION

For performance evaluation, Microsoft .Net framework has been used with Alchemi open source toolkit (Lab, 2010). Alchemi packages include Alchemi manager and Alchemi executers with the backend database tools such as MySQL etc as explained earlier. As the Microsoft has a major market share so the study focuses the development of the computational grid using the window based toolkit to utilize the free cycles of widely spread window based machines. We have analyzed the communication cost involved by choosing the Pi calculation program (Lab) for the grid topology shown in Figure 6. The specifications are given in Table 1.

We have analyzed the communication cost involved for the backup node selection in case of failure while varying the number of hops of the executers. The study shows

that as in the multicast scenario the manager have to contact all the group members for their willingness and specification so larger communication cost is involved in term of delay, packet delivery ratio and reliability as compare to anycast. The reason for the difference in the cost is that in case of anycast it selects the one and the best option, if it fails, than it opts for the next best one.

CONCLUDING REMARKS AND FUTURE WORK

The study focused on the detailed implementation of fault tolerance techniques, the main contribution of this paper is the implementations of fault tolerant techniques using anycasting with modified forwarding mechanism and its analytical analysis for the computational grid against the multicast technique. The study analysis of the communication cost involved in terms of delay time and packet delivery ratio of the proposed scheme and its comparison with the previous techniques. It has been observed that using anycast technique for the backup selection is better than the other methods due to the less communication cost involved. Communication cost analysis using cross platform and mixed toolkits might be an interesting future work.

REFERENCES

Fazle H, Shah N, Syed A, Yasin M (2007). Adaptive Anycast: A New Anycast Protocol for Performance Improvement in Delay Tolerant Networks. Int. Conf. Integration Technol., ICIT '07.

Foster I, Kesselman C (1997). Globus: A Metacomputing Infrastructure Toolkit. Int. J. High Perform. Comput. Appl., 11(2): 115-128.

Foster I, Kesselman C (1999). The grid: Blueprint for a new computing infrastructure, Morgan Kaufman.

Foster I, Kesselman C, Tuecke S (2001). The Anatomy of the Grid: Enabling Scalable Virtual Organizations. Int. J. High Perform. Comput. Appl., 15(3): 200-222.

Imran M, Niaz IA, Haider S, Hussain N, Ansari MA (2007). Towards Optimal Fault Tolerant Scheduling in Computational Grid. Int. Conf. Emerg. Technol., ICET.

Krishnan S (2004). An architecture for checkpointing and migration of distributed components on the grid. Department of Computer Science, Indiana University. PhD.

Lab G (2010) Net Grid Computing Framework. URL: http://www.alchemi.net.

Luther A, Buyya R, Ranjan R, Venugopal S (2004). Alchemi: A .netbased grid computing framework and its integration into global grids. Tech. Rep. Australia.

Medeiros R, Cirne W, Brasileiro F, Sauve J (2003). Faults in Grids: Why are they so bad and What can be done about it? Proceedings of the 4th Int. Workshop on Grid Computing. IEEE Comput. Soc., pp. 18-24.

Muhammad A, Ansari MA (2006). Distributed Fault Management for Computational Grids. Fifth Int. Conference on Grid and Cooperative Computing.

Nazir B, Khan T (2006). Fault Tolerant Job Scheduling in Computational Grid. Int. Conference on Emerging Technologies ICET '06.

Nguyen-Tuong A (2000). Integrating fault-tolerance techniques in grid applications. University of Virginia Charlottesville, VA, USA. PhD.

Roman E (2003). A survey of checkpoint/restart implementations, Lawrence Berkeley National Laboratory, LBNL-54942: 1-9.

Szymaniak M, Pierre G, Simons-Nikolova M, Steen M (2007). Enabling service adaptability with versatile anycast, J. Con. Comp.: Practice Experience, 19(13): 1837-1863.

Tanimura Y, Ikegami T, Nakada H, Tanaka Y, Sekiguchi S (2006). Implementation of Fault-Tolerant GridRPC Applications. J. Grid Comput., 4: 145-157.

University RI. URL: http://riphah.edu.pk/.

Venugopal S, Buyya R, Ramamohanarao K (2006). A taxonomy of Data Grids for distributed data sharing, management, and processing. ACM Comput. Surv., 38(1).

Waheed A, Smith W, George J, Yan J (2000). An Infrastructure for Monitoring and Management in Computational Grids. Selected Papers from the 5th Int. Workshop on Languages, Compilers, and Run-Time Systems for Scalable Computers, Springer-Verlag, pp. 235-245.