ISSN:
Print - 2277 - 0593
Online - 2315 - 7461

Journal of Natural Science, Engineering and Technology

REPLICATION-BASED COST SCHEDULING STRATEGY FOR FAULT TOLERANCE IN DISTRIBUTED KNOWLEDGE MANAGEMENT SYSTEMS

O. O. BAMGBOYE[1*] , O. FOLORUNSO[2], A.T. AKINWALE[2] AND G.A. ADEBAYO[2]

[1] Computer Science Department, Moshood Abiola Polytechnic, Ojere, Abeokuta, Ogun State
[2] Department of Computer Science, Federal University of Agriculture, Abeokuta Ogun State.

***Corresponding author:** seunbamgboye13@gmail.com **Tel:** +234(0)8037284536

ABSTRACT

Distributed Knowledge Management Systems (DKMS) often depends on the Semantic Web Peer-to-Peer (SW-P2P) model. The reason for this is based on its support for autonomy of knowledge node, ease of accessibility and scalability. The susceptibility to failure experienced during knowledge retrieval has been a concern for the SW-P2P. This paper presents a fault tolerance system in order to resolve the problem of the DKM. The architecture of this design consists of five components namely; Replication Manager (RM), Fault Detector (FD), Fault Notifier (FN), Recovery Mechanism (RMe) and Global Control Monitor (GCM). This design adopted dynamic replication strategy and group constitution procedure to guarantee knowledge availability on knowledge nodes. The dynamic replication strategy was used to create and delete replicas based on the changes in the DKMS environment. The group constitution procedure suggested the efficiency of fault recovery process in terms of the best available replica among knowledge service group. The fault tolerance system execution cycle was performed on a set of Virtual Machines (VM) using the VMware Workstation version 7.0.1, while Java programming language was used to implement the group and ungroup replicas. Sample data of varying magnitude in ranges of 225Kilobytes to 512Kilobytes and 450Kilobytes to 512Megabytes were tested at different time intervals on both the grouped and ungrouped replicas at a threshold between 0.85 and 0.9 of knowledge retrieval. The results showed a reduction in the average response time of the grouped replicas which was measured to be 34 milliseconds and 68.2 milliseconds against ungrouped replica that was estimated as 53 milliseconds and 107.2 milliseconds respectively. The effect of this reduction in response time was that the grouped replica was faster than the approach of ungroup replica. In addition, the group replica occupied less memory space because it does not need to store replicas on the active knowledge peer when recovering from failure. This result showed that the system guarantees the fault tolerance of each knowledge node in a DKMS.

Key words: Computer Network, Fault Tolerance, Knowledge Management, Recovery Systems, Replication Algorithm, Peer-to-Peer

INTRODUCTION

With the growth of the computer networks, especially the internet, fast dissemination and access to data and knowledge are becoming an integral part of our daily life (Zhe *et al.*, 2012). As a result of increased com-

plexities and globalization, companies have quickly transformed its hierarchical structure to flexible business networks, heavily depending on inter-organization and intra-organizational communication (Cuel *et al*., 2005). Knowledge Management (KM) is increasingly viewed as a core requirement in order to compete in the modern social and economic environment (Senge 1990; Nonaka and Takeuchi 1995; Davenport and prusak, 1998). Researchers and practitioners agree that those intellectual assets that are embedded in working practices, social relationships, and technological artifacts constitute the only source of value that can sustain long term differentiation, quality of services, innovation, and adaptability (Stewart, 2002). Nonetheless, due to a debatable success of current KM implementations, it is still unclear how such matter should be managed in highly complex, distributed and heterogeneous settings. The general knowledge management types are the centralized and distributed knowledge management systems (Schmidt *et al.,* 2009).

Centralized KMS is aimed at creating large, homogeneous knowledge repositories, in which corporate knowledge is made explicit, unified, represented and organized according to a unique conceptual schema.

Distributed Knowledge Management (DKM) is defined as management of autonomous groups that create local knowledge and exchange it across groups (Bonifacio *et al*., 2002). Distributed KMS are commonly described as socio- technical KMS aiming to actively engage users in knowledge acquisition and dissemination processes (push and pull approach). DKMS responds to two basic principles – autonomy and coordination (Cuel, 2003). Autonomy designate the possibility each organizational unit to have the opportunity to conceptualize its local knowledge through maps, ontologies, contexts, etc. This could also be achieved with tags and folksonomies. The second principle – coordination, reflects the mechanism of projecting what other units know into its own interpretation schema. Thus, DKM suggests autonomous management of locally produced knowledge and coordination among different units without centrally defined view.

DKMS are often based on Peer-to-Peer (P2P) collaboration and rely on bottom-up knowledge management. P2P systems are distributed systems composed of independent nodes that run software with equivalent functionality without centralized control or hierarchical organization (Ehrig *et al*., 2003). The general interest to P2P architectures is due to their ability to function, scale and self-organize, in the presence of a highly transient population of nodes, networks, and computer failures and without the need of central server (Androutsellis and Spinellis, 2004). According to the literature (Ehrig *et al*., 2003; Androutsellis and Spinellis, 2004; Bonifacio *et al.,* 2004; Buchegger and Datta, 2009), the P2P is well suited for DKMS because of their scalability, acceleration of communication processes and reduced collaboration costs, lack of centralized control, privacy of the user data, increased access to resources, opportunity to maintain own knowledge structure and conservation of original knowledge context.

The P2P approach helps to resolve a number of limitations such as; costs of design, implementation and maintenance of centralized server; simplifying system

complexity, reducing the barriers for participation, integrating the shared knowledge work space with personal knowledge work spaces (Maier, 2007). The most common applications of P2P remain the file-sharing and communication which are the main issues in DKMS.

Managing knowledge in a distributed environment is crucial and the main idea is to allow Community-of –Practice (CoP) members to continuously engage in knowledge exchange activities even in the presence of fault. This can be achieved by estimating the cost of generating knowledge replicas that can enhance knowledge availability.

The rest of the paper is organised as follows: section 2 presents related works on fault tolerance using replication techniques. Section 3 contains the overall architecture and design details of FTA-DKM. The implementation of the system is discussed in section 4 and the conclusion and future works are presented in section 5.

RELATED WORKS

Much work has been done on fault tolerance using replication in distributed systems and several algorithms have been developed (Abbadi *et al.,* 1985; Agrawal and Abbadi., 1990; Davidson *et al.,* 1997). Fault tolerance in distributed systems is based on two fundamental classes of replication techniques: primary-backup replication (Guerrouri and Shiper, 1997) and active replication (state machine approach) (Schneider, 1990). Table 1 presents a summary of related works on fault tolerance in distributed systems.

Table 1: Summary of Related Works.

	Author(s)	Title	Approach	Results
1	Rangarajan et al., (1995)	A fault-tolerant algorithm for replicated data management.	Proposed quorum consensus approach that groups data sites into subgroups	Provides balance between low message overhead and high data availability.
2	Mansouri and Dastghaibyfard (2012)	A dynamic replica management strategy in Data Grid	Stores replica in sites where a parameter file has been accessed most instead of storing file in many sites	Provide less job execution time in comparison with other strategies especially when the grid site have small storage size
3	Park et al., (2003)	Grid replication strategy based on internet hierarchy	Develop sites into regions whereby the network bandwidth is kept lower between the regions rather than within the region	Decreases data access time by maximizing network-level locality and avoiding network congestion.
4	Andronikou et al., (2012)	Dynamic QoS- aware data replication in Grid environments based on data importance	Proposed a set of interoperable new data replication strategy that takes into account the infrastructural constraints as well as importance of data.	The system is scalable and support strategies that are easily Implemented on grid environment to provide fast execution.
5	Saadat, and Rahmani, (2012)	PDDRA: A Pre-fetching Based Dynamic Data Replication Algorithm in Data Grid.	It predicts the future requirements of grid sites and pre-replicates the before it is requested	Improves job execution time, network usage, number of replications and, percentage of storage requirements

Source: Reviewed Literature

REPLICATION STRAEGY FOR FAULT TOLERANCE SYSTEM IN DKMS

The proposed fault tolerance system is based on the Combined Cost Scheduling Strategy and Replication technique for fault tolerance. Data replication is becoming a popular strategy in many fields such as cloud storage, data grid, and P2P systems. By replicating files to other servers/nodes, network traffic and file access time can be reduced, and increase data availability to react to natural and man-made disasters (Zhe et al., 2012). In this paper, the fault tolerance technique adopts dynamic replication strategy for the knowledge services across multiple nodes on the distributed knowledge management environment. The dynamic replication strategy creates and deletes replicas based on the changes on the distributed knowledge management system. The cost of creating a replica is based on the node that is frequently accessed by failed node during fault recovery. The replicas that are not frequently used are deleted over a period of time. Dynamic replication strategy may be implemented either in a centralized or in a distributed approach (Najme et al., 2013). It implements a replication algorithm as indicated in figure 1. The algorithm is based on principle of group structure (Andrew et al., 2007) that suggests that two or more replicated knowledge services can be formed into either of the flat or hierarchical group structure depending on some indicators that is being specified in a knowledge exchange activity. In case of failure of any k-peer, it is expected that any of the active replicated knowledge services will resolve the failure without seriously disturbing the rest of the system. This new approach of group structure will build a reliable fault tolerance system. Some of the previous research works on fault tolerance (Liang, 1999; fang et al., 2007) pointed out that a fault tolerance system should contain four fundamental functionalities that includes; replication management, fault management, logging and recovery mechanism and client fault *transparency).*

```
Procedure Replication_State{
        IF (KG == Time Critical true) Then / / KG is the Knowledge Group
          PM1 ← KD {          / / KD is the Knowledge Descriptor
                    Determine : replication_ style();
                    Min _number _ of_ replica();
                        consistency_ style();
                     Membership_ sytle();
                          }
                Locate shortest node proximity();
                PM2     Hierarchical group; / / PM2 is the Priority Manager
                elseIf ←
                KG == Safety Critical;
                Compare Initial faulty knowledge object with Non-faulty replica;
                PM2     Flat group;
                endIf ←
        Store or keep log;
End Replication_State( )
```

Figure 1: Algorithm for Knowledge Replication

THE FAULT TOLERANCE ARCHITECTURE FOR DKMS

Fault tolerance structure is often implemented through replication of process or objects in a real time distributed system (Arvind *et al.,* 2011). The specification for this structure is the determination of replication style and the number of replica of knowledge objects on a distributed knowledge management system which is a necessary condition for the design of fault tolerance structure. In other to support the fundamental fault tolerance functionalities, the Fault Tolerant System for Distributed Knowledge Management (FTA-DKM) introduced four components:

1. Replication Manager (RM)
2. Fault Detector (FD)
3. Fault Notifier (FN)
4. Logging/Recovery mechanism

A new component called the Global Control Monitor (GCM) was also introduced in other to build an efficient fault tolerance system.

The Replication Manager performs the replication of knowledge services using certain set of desired properties such as replication style, minimal replication degree, number of replica, etc. It also performs the group constitution which is initiated through the Global Control Monitor interface. The specification for a particular group structure is achieved based on the priority levels (which can be the time specificity of transaction, safety of knowledge property or a combination of both) as being clearly specified by a knowledge seeker (a device capable of initiating a knowledge request per time) in a knowledge transfer operation. The Fault Detector performs monitoring operation on each of the knowledge node in other to detect the occurrence of failure and the fault notifier is responsible for the fault notification.

The logging mechanism captures and logs request for the recovery mechanism. The group constitution feature of the proposed fault tolerance architecture identifies two types of groups; these are the hierarchical and flat group structures respectively. The hierarchical group structure implement the principle of the best available replica (Nnebe *et al.,* 2012) while the flat group structure implement the principle of the most available replica among replicated knowledge services. The flat group structure estimates the property of the replica that is involved during fault recovery by confirming the size and context of the requested knowledge on the knowledge provider device before the transfer is done. These properties are later compared with all available replicas to determine the best fit replica for recovery in a safety level priority specification. The Hierarchical group structure on the other hand measures the signal strength of the devices in a wireless network that holds each replica. It uses the proximities of replicas to determine the most suitable replica in a time critical knowledge exchange process.

The fault tolerance life cycle for the DKMS is divided into three phases namely: initialization, run-time/recovery and client fault transparency phase.

Initialization Phase

The Knowledge Seeker (K-Seeker) uses the group constitution function of the GCM service to constitute the service group of replica as shown in figure 2. The GCM registers the group replica (as either flat or hierarchical group structures) to be formed to the replication manager. It then requests the RM to

create the service group using the active replica. The RM automatically performs the complete group constitution procedure: request that knowledge factory (a component that specifies knowledge representation format) on N_1 and N_2 deploy their members based on the fault tolerance properties; activates fault detectors to monitor the knowledge services; subscribes fault notification from fault notifier for the service group; chooses the active and primary replica based on group level priority specification and backup other members. Finally, RM publishes the address of the knowledge replica to the centralized user's Knowledge Base (KB) for K-seeker that intends to access the service.

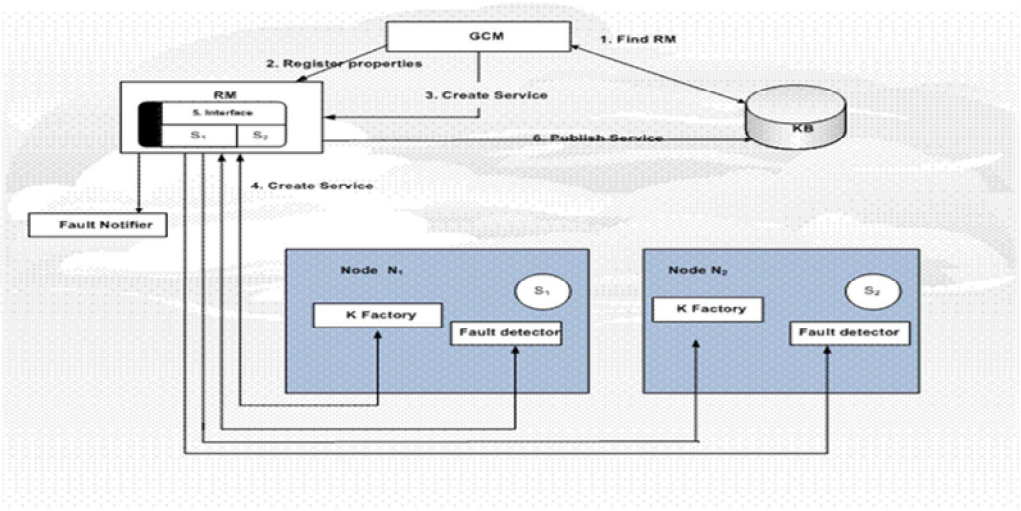

Figure 2: Showing the Initialization of Fault Tolerance System and Group Replica Constitution

Runtime Phase / Recovery Phase

The Replication Manager requests fault detectors D_1 and D_2 monitor the replicas S_1 and S_2 respectively. The fault detector D_1 sends fault report to the fault notifier whenever it detects a service failure on the active member S_1 as shown in figure 3 below. The fault notifier therefore, notifies the RM to start the recovering process that includes the following steps:

i. RM promotes S_2 as the new active replica based on the group constitution policy that is being initiated (that is either as hierarchical or flat group structure) through the GCM for fault rectification.

ii. It instantiates S_3 on N_3 for the case of hierarchical (in time critical transaction), otherwise it picks on the best available replica (in transaction where the safety of the knowledge property must be guarantee using the flat group structure) as warm backup.

iii. The RM informs the recovery mechanism to recover the state of S_2 and;

iv. Finally, RM updates the GCM of the service to reflect the new active replica S_2 and the new secondary replica respectively.

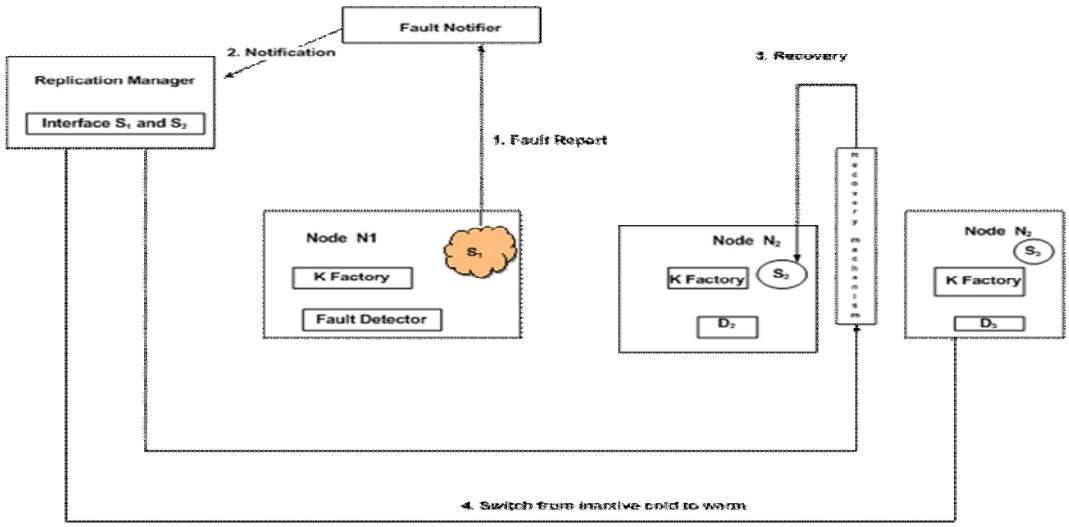

Figure 3: Fault Recovery Process

Fault Transparency Phase

The fault recovery process is achieved anytime the k-seeker device is capable of redirecting request to another replica in case the primary k-provider failed. This is often achieved when the k-seeker create acquaintance with another service device or knowledge node that owns a replica of the previously failed service to form a new k-peer. As shown in figure 4, the K-factory on the K-seeker sends a request to the primary K-peer service S_1. The K-factory receives a service fault exception if the primary member S_1 is down. The K-factory then searches for the next available replica using the specified priority level (that is considers either the best or most available replica). The new peer on node N_3 returns result if S_2 is able to serve this request. Thus, the K -seeker is unaware of the primary failure

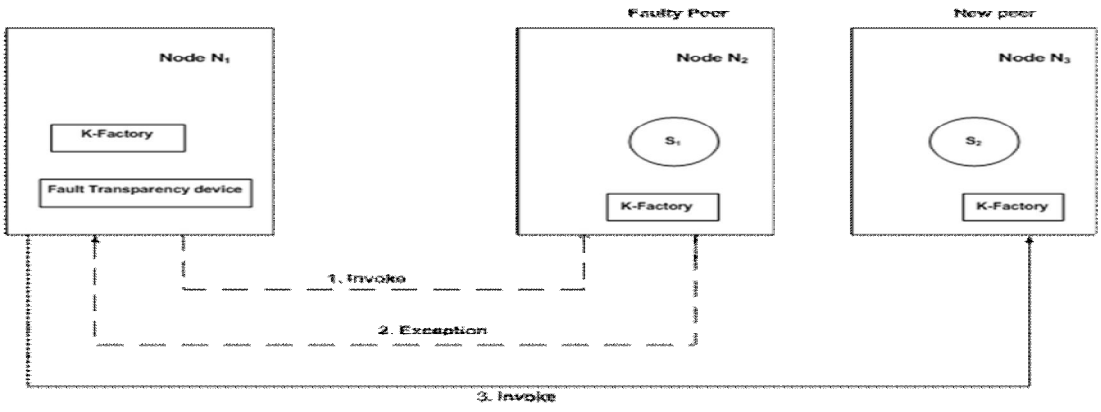

Figure 4: Failure Masking through Service invocation

COST SCHEDULING STRATEGY FOR FAULT TOLERANCE

It is often expected that there can exist a queue of knowledge request during the process of redirecting the request to other replica in a failed transaction, this queue of request are managed by the Cost scheduling Strategy. The main rationale behind the combined cost scheduling strategy is to maintain the cost and consistency in managing the queue of knowledge requests in the fault tolerance domain of the distributed knowledge management environment. This will also maintain the knowledge availability present at each node or host.

The proposed Combined cost Scheduling Strategy (CSS) at first determines the Best Node (BN). The Best Node is defined as a region that has most of the requested Knowledge (from size point of view). In other words, let KZ be the total size of requested knowledge available in node n, therefore the Relative Cost R_n;

$$R_n = \gamma \times \Sigma |KZ| \quad (1)$$

where γ is the constant reflecting the degree of parallelism. CSS computes R_n for each node and selects the best node i.e. node with largest R_n value. Then the combined cost for each host within the BN is computed and the request assigned to the host with minimum combined cost (MCC). Therefore, CSS does not search all resources to find the best one that has the lowest cost i.e.

$$MCC = \text{Min } CC_{i,j} \quad (2)$$

where n is the number of host within BN
Let $J_i = \{F_1, F_2, \ldots. F_m\}$ be the m required knowledge for knowledge transaction i
Let $UKF_{i,j}$ = Σ size of unavailable knowledge request for transaction i in node j

$$\text{Let Load j} = \frac{Q_j}{S_j}$$

where Q_j and s_j are the number of transactions waiting in queue and computing capacity of node j respectively. Therefore, the sum of these along with the actual weight (w_1, w_2) for each factor yields the combined cost of executing knowledge transaction i in node j as follows:

$$CC_{i,j} = (w_1 \times UKF_{i,j}) + (w_2 \times \text{Load j}) \quad (3)$$

SYSTEM IMPLEMENTATION

This section describes a series of experiments conducted based on the architecture and algorithm that were described in section three. The various parameters that serve as input to the system include the datasets and the induced fault that was generated during the experiment. A faulty knowledge peer is the one that has been disconnected from the node or peer during knowledge exchange. The variation in data size is relatively uniform throughout the experiment. Each of the knowledge peer also have the ability to transit into a faulty state at any given time. The fault tolerance property is estimated at a well defined threshold during the experiment.

IMPLEMENTATION ENVIRONMENT AND TOOLS

The design is implemented on a VMware environment. The VMware provides high availability and well secured framework that can allow different operating systems to interface with each other. Each VMware was configured on operating systems which were hosted by a node. A node consists of six individual operating systems that is running on different virtual machines. A node is configured on a range of hardware specifications such as 100GB-700GB of Hard disk space, 1GB-6GB of RAM and various capacities of

intel processor. A node is connected to another node through a high speed router. A total of three nodes were considered for the purpose of fault injection during the experiment and each of the VMware can create acquaintance with other VMware on the network to form a peer. Each peer is being coordinated and managed through the Global Control Monitor. The Global Control Monitor was designed using java programming language on a Java Netbean development environment. Methods were also developed in java for capturing, replicating and grouping knowledge services.

DESCRIPTION OF THE KNOWLEDGE SITES

The knowledge sites are implemented on the VMware workstation, with each workstation being part of a node in other to form a DKMS. The node is configured on a computer system with large amount of memory and high disk space. Each workstation is installed with different versions of windows operating system in other to verify the interoperability of the various platforms. Two nodes were configured for the experimentation with each node containing three virtual machines. A total of eight knowledge sites were created for the purpose of this experiment.

The use of synthetic data was also considered for the experimentation of this work. The hardware configuration for these knowledge sites is indicated in table 2

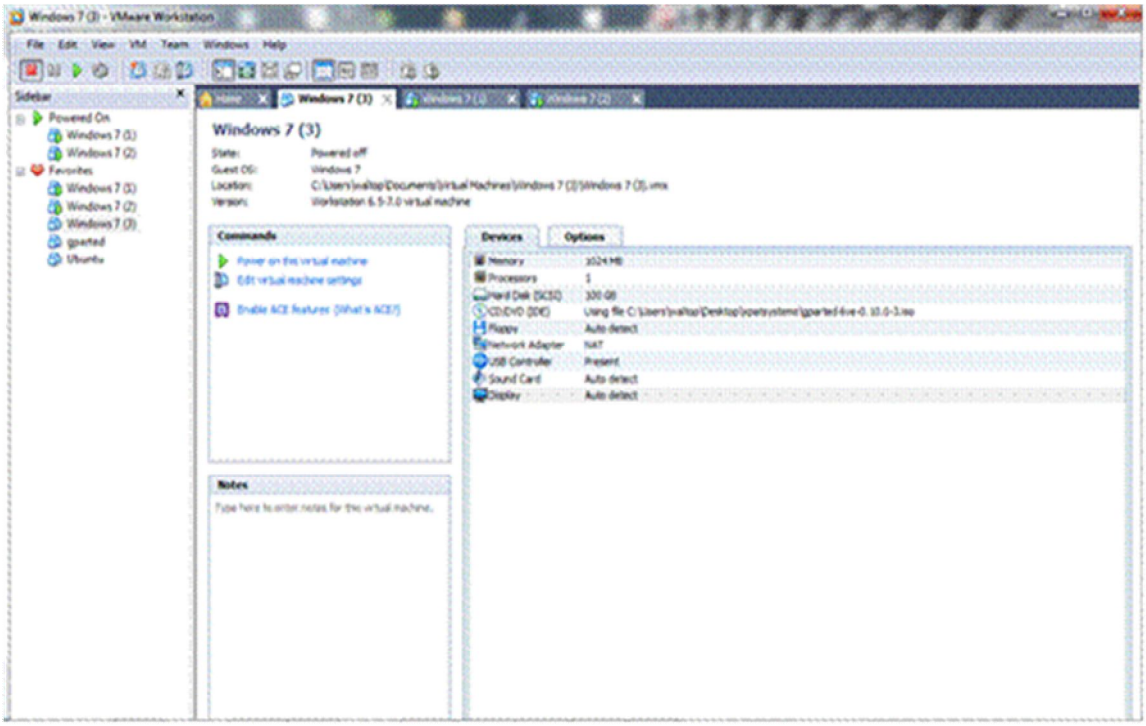

Figure 5: The Virtual knowledge Sites on VMware environment

Table 2: **Hardware Specification for site 1(Node 1)**

Knowledge site	Components	Size/Type
Node 1	Hard Disk	700 GB
	Memory	6GB
	Processors	quad Core(s)
	Operating System	Windows 8
Node 2	Hard Disk	500 GB
	Memory	4GB
	Processors	2 Core(s)
	Operating System	Windows 7 Home Premium
Node 3	Hard Disk	100 GB
	Memory	1 GB
	Processors	1
	Operating System	Windows 7 Ultimate

DISCUSSION OF RESULTS

In other to properly analyze the strength of the fault tolerance system, a series of experiment were conducted in a distributed environment by directly injecting fault into the system. The experiment was conducted in other to be able to measure the efficiency of group replica of knowledge services based on the proximity of the knowledge site to an active node or knowledge seeker when initiating a knowledge transaction. It also attempts to verify the property of each of the active knowledge service replica in a fault tolerance system.

The experiments consist of two nodes N_1 and N_2 with each node accommodating three virtual machines respectively. Each node is connected over a high speed router. Each virtual machine is capable of initiating an acquaintance with other machine to form a knowledge peer (K-peer) and it can choose to dissociate and form a peer with another anytime. Each K-peer is made to engage in autonomous management of knowledge services which may be later required by another peer or node.

The data source for the experimentation and performance analysis is from http://www.dmoz.org.The website is a collection of internet links organized in a hierarchy (Marc et al, 2003). It contains 190,000,000 documents of text characters. The contents of documents were distributed randomly among the k-peers. The distribution of documents within the peer-to-peer network is varied at regular interval in other to influence the size of knowledge (data) that is exchanged or shared by a peer. The experiments are conducted on two rounds of varying data sizes. The range of data size in the first experiment is between 225 Kilobytes to 512Kilobytes. The second round of the experiment was conducted on data size that ranges from 450Kilobytes to 512 Megabytes. A set of k-peers are selected at random, and knowledge exchange activities are disconnected at interval of 20minutes and 50minutes respectively in other to measure the response times of the system.

A series of experiments were performed to

measure the average response time of each of the group replica as defined by this work. The sizes of data were varied in different times of experiment and the fault detection as performed by the fault detector was conducted between 0.85 to 0.9 thresholds during knowledge retrieval. Each experiment was performed at numerous times on each of the group replica in other to collect a reasonable set of experimental results.

The response time for knowledge service failure is defined as the time span from when a primary or active knowledge service crashes to when a warm replica of knowledge service resumes the primary responsibility.. The response time of each of the group replica is considered to be the elapsed time between the faults is detected and when recovery is completed. It was measured in milliseconds (ms). The figures 6 and 7 shows that the combination of the group structures (hybrid) is more efficient in terms of response time compared to the other two types of group replicas (that is flat and hierarchical). The response time in fault recovery is also shown to be directly proportional to the knowledge size in a knowledge transaction. The ungroup structure requires high response time in fault recovery compared to the other group structures as identified by this work. In general, the average response time of the grouped replicas was measured to be 34 milliseconds and 68.2 milliseconds against ungrouped replica that was estimated at 53 milliseconds and 107.2 milliseconds respectively.

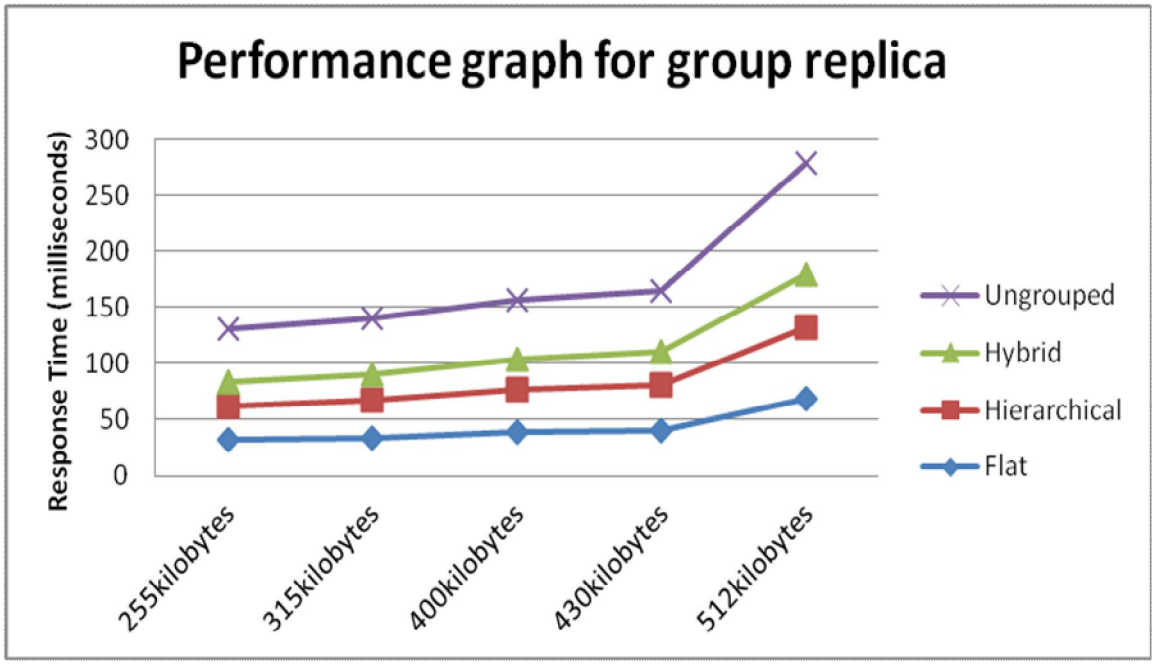

Figure 6: Fault-tolerance performance for group structure on first round of experiment

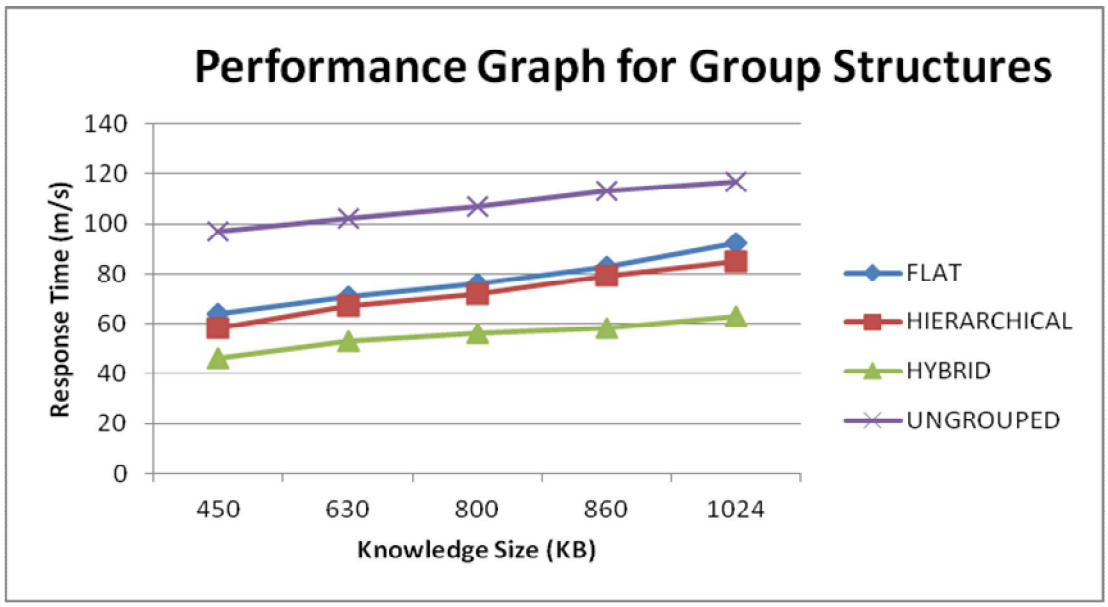

Figure 7: Fault-tolerance performance for group structure on second round of experiment

CONCLUSION AND FUTURE WORKS

In this paper, we developed a novel approach of fault tolerance that adopts knowledge service replication technique in a Distributed Knowledge Management System to enhance knowledge availability and fault tolerance. The fault tolerance architecture adopts 3-phase architecture and the knowledge replication algorithm to resolve the connectivity problem that can be associated with knowledge transfer operation in a knowledge transaction. It also implements a group constitution procedure in other to improve the performance and effectiveness of the system. This approach to fault tolerance is efficient in terms of low response time in failure recovery and maintains the availability of knowledge. With the replication scheme, it is clear that the system can equally guarantee a reliable and highly scalable knowledge exchange environment.

In the future, the approach can be extended by introducing the principle of the agent-based technology the architecture in other to improve the efficiency of the system. Furthermore, this paper provides a theoretical background for researchers to explore the potentials of Distributed Knowledge Management System particularly on the aspect of maintaining knowledge availability and replica consistency.

REFERENCES

Abbadi, A.E., Skeen, D., Gristian, F. 1985. An Efficient Fault Tolerant Protocol for Replicated Data Management. *Proc. Symp. Principles Database Syst*. 215–228.

Accorsi, F. L. & Costa, J. P. 2008. Peer-to-Peer Systems Consubstantiating the Basic Concept. *The Electronic Journal of Knowledge Management* Vol. 6, Issue 1, pp 1 – 12.

Agrawal, D., Abbadi, A.E. 1990. Exploiting logical structures in replicated databases. *Inform. Process. Lett*. 33 (5) 250–260.

Akoumianakis, D. 2008. Distributed Knowledge Management in Virtual Organizations: the 'Social' Experience Factory. *The Electronic Journal of Knowledge Management* Vol. 6 Issue 1 pp. 13 – 32.

Andronikou, V., Mamouras, K., Tserpes, K., Kyriazis, D., Varvarigou, T. 2012. Dynamic QoS- aware data replication in Grid environments based on data importance. *Future Generation Computer Systems* 28 (3):544–53.

Andrew, T., Maarten, S. 2007. Distributed Systems-Principles and paradigm. Pearson Prentice Hall, Upper saddle River, New York.

Androutsellis, S., Spinellis D. 2004. Survey of P2P Content distribution technologies. *ACM Computing Surveys*, Vol. 36 Issue 4, pp 335-357.

Arvind, K., Rama S., Yadav, R, Anjali J. 2011. Fault Tolerance in Real Time Distributed System. *International Journal on Computer Science and Engineering*. Vol.3 No.2, pp 933-939.

Bonifacio, M., Bouque, P., Busetta P., Danieli, A., Don, A., Mameli, G. & Nori, M. 2004. KEEx: A Peer-to-Peer Tool for Distributed Knowledge Management. *Proceedings I-KNOW '04*. http//:www.bg-openaire.eu *(Accessed May 16, 2012)*

Bonifacio, M., Giunchiglia, F., Zaihrayeu, I. 2005. Peer-to-Peer Knowledge Management. *Proceedings I-KNOW '05*. http:// www.*eprints.biblio.unitn.it/771/1/042.pdf (Accessed Oct.13, 2012)*

Buchegger, S., Datta, A., 2009. A Case for P2P Infrastructure for Social Networks Opportunities and Challenges. *In Proceedings of WONS 2009*, Utah, USA, February 2-4.

Cuel, R. 2003. A New Methodology for Distributed Knowledge Management Analysis. *Proceedings of I-KNOW '03 Industry meet Science*, Graz - Austria.

Cuel, R., Bouquet, P., Boniface, M. A. 2005. Distributed Approach to Knowledge Management, the Concept of Knowledge Nodes, and their Implications. in Schwartz D.G. Ed., Encyclopedia of Knowledge Management.

Davenport, T., Prusak, L. 1998. Working Knowledge – How Organizations Manage What They Know. Harvard Business School Press, Boston, MA.

Davidson, S.B., Garcia-Molina H. & Skeen, D. 1997. Consistency in partitioned networks, *ACM Comput. Surveys* 17 (3) 341–370.

DMOZ. Data source for DKMS < http:// www.dmoz.org> accessed on October 4, 2013.

Ehrig, M., Tempich, C., Staab, S., Harmelen, F., Siebes, R., Sabou, M., Broekstra,J., Stucken S. H. 2003. SWAP: Ontology-based Knowledge Management with Peerto- Peer. London.

Fang, C., Liang, D., Lin, F. 2007. Fault tolerant Web Services.*Journal of Systems Architecture*, Vol. 53, Issue 1, January 2007, Pages 21–38.

Guerraoui, R. & Schiper, A.(1997). Software-Based Replication for Fault Tolerance. *IEEE Computer*, (30)4:68-74, Cited on Pages 328,629.

Mansouri, N., Dastghaibyfard, G.H. 2012. A dynamic replica management strategy in Data Grid. *Journal of Network and Computer Applications* 35(4):1297–303.

Maier, R., 2007. Knowledge management systems. 3rd ed., XIV, 720 p. 125 illus.

Marc, E., Christoph, S., Steffen, S., Julien, T., Christoph, T. 2003. Towards Evaluation of Peer-to-Peer-based Distributed Information Management Systems. *American Association for Artificial Intelligence*

Najme, M., Gholam, H. D., Ehsan, M. 2013. Combination of data replication and scheduling algorithm for improving data availability in Data Grids. *Journal of Network and Computer Applications* pp 711–722.

Nonaka, I., Takeuchi, H. 1995. The Knowledge Creation Company. New York, OXFORD University Press.

Nnebe, S. U., Onoh, G.N., Ohaneme, C.O., Nwankwo V.I. 2012. Empirical Analysis of Signal Strength-Distance Variations in IEEE 802.11b in WLAN. *International Journal of Engineering and Innovative Technology (IJEIT*), Vol. 2, Issue 6, pp.470-472.

Park, S.M., Kim, J.H., Go, Y.B. & Yoon, W.S. 2003. Dynamic Grid replication strategy based on internet hierarchy, *in international workshop on grid and cooperative computing.* 1001:1324–1331.

Rangarajan, S., Setia, S., Triapthi, S.K. 1995. A fault-tolerant algorithm for replicated data management. *IEEE Trans. Parallel Distrib. Syst*. 6 (12) 1271–1282.

Saadat, N., Rahmani, A.M. 2012. PDDRA: A Pre-fetching Based Dynamic Data Replication Algorithm in Data Grid. *Future Generation Computer Systems* (28) 666–681.

Sashi, K., Thanamani, A.S. 2010. A new dynamic replication algorithm for European data grid. in: *Proceedings of the Third Annual ACM Bangalore Conference,* p. 17.

Schneider, F.B. 1990. Implementing fault-tolerant services using the state machine approach. a tutorial, *ACM Comput. Surveys* 22 (4) 299–319.

Schmidt, A., Hinkelmann, K., Ley, T., Lindstaedt, S., Maier, R.. & Riss, U. 2009. Conceptual Foundations for a Service-oriented Knowledge and Learning Architecture. in Schaffert et al, (eds) Networked Knowledge-Networked Media,

Senge, P. 1990. The fifth discipline: the art and practice of the learning organization. New York, NY: Doubleday.

Wei, L., FuMin Y., Gang, T., LiPing P. and Xiao, Q. 2007. TERCOS: A Novel Technique for Exploiting redundancies in Fault-Tolerant and Real-Time Distributed Systems. *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications* (RTCSA).

Zhe, W., Tao, L., Naixue, X., Yi P. 2012. A novel dynamic network data replica-

tion scheme based on historical access record and proactive deletion. *Journal of Supercomputing*, Vol. 62, Issue 1, pp. 227-250.

(Manuscript received: 5th September, 2014; accepted: 20th April, 2015).