Int. J. Communications, Network and System Sciences, 2015, 8, 471-482
Published Online December 2015 in SciRes. http://www.scirp.org/journal/ijcns
http://dx.doi.org/10.4236/ijcns.2015.812042

Fault Tolerance Mechanisms in Distributed Systems

Arif Sari, Murat Akkaya

Department of Management Information Systems, Girne American University, Kyrenia, Cyprus
Email: arifsari@gau.edu.tr, muratakkaya@gau.edu.tr

Received 13 October 2015; accepted 14 December 2015; published 17 December 2015

Abstract

The use of technology has increased vastly and today computer systems are interconnected via different communication medium. The use of distributed systems in our day to day activities has solely improved with data distributions. This is because distributed systems enable nodes to organise and allow their resources to be used among the connected systems or devices that make people to be integrated with geographically distributed computing facilities. The distributed systems may lead to lack of service availability due to multiple system failures on multiple failure points. This article highlights the different fault tolerance mechanism in distributed systems used to prevent multiple system failures on multiple failure points by considering replication, high redundancy and high availability of the distributed services.

Keywords

Fault Tolerance, Distributed System, Replication, Redundancy, High Availability

1. Introduction

A faulty system creates a human/economic loss, air and rail traffic control, telecommunication loss, etc. The need for a reliable fault tolerance mechanism reduces these risks to a minimum. In distributed systems, faults or failures are limited or part. A part failure in distributed systems is not equally critical because the entire system would not be offline or brought down, for example a system having more than one processing cores (CPU), and if one of the cores fails the system would not stop functioning as though that's the one physical core in the system. Hence, the other cores would continue to function and process data normally. Nevertheless, in a non-distributed system when one of its components stops functioning, it would lead to a malfunction of the entire system or program and the corresponding processes would stop.

Fault tolerance is the dynamic method that's used to keep the interconnected systems together, sustain reliability, and availability in distributed systems. The hardware and software redundancy methods are the known techniques of fault tolerance in distributed system. The hardware methods ensure the addition of some hardware components such as CPUs, communication links, memory, and I/O devices while in the software fault tolerance method, specific programs are included to deal with faults. Efficient fault tolerance mechanism helps in detecting of faults and if possible recovers from it.

There are various definitions to what fault tolerance is. In dealing with fault tolerance, replication is typically used for general fault tolerance method to protect against system failure [1] [2]. Sebepou *et al*. highlighted three major forms of replication mechanism which are [1] [2]:

The State Machine;

Process Pairs;

Roll Back Recovery.

1) State Machine

In this mechanism, the process state of a computer system is replicated on autonomous computer system at the same time, all replica nodes process data in analogous or matching way and also there's coordination in their process among the replica nodes and all the inputs are sent to all replica at the same time [2] [3]. An active replica is an example of state machine [3] [4].

2) Process Pairs

The process pairs functions like a master (primary)/slave (secondary) link in replication coordination. The primary workstation acts in place of a master to transmit its corresponding input to the secondary node. Both nodes maintain a good communication link [3]-[5].

3) Roll Back Recovery (Check-Point-Based)

This mechanism collects check point momentarily and transfers these checkpoint states to a stable storage device or backup nodes. This enables a roll back recovery to be done successfully when or during recovery process. The checkpoint is been reconstructed prior to the recent state [3]-[6].

2. Distributed System

Distributed system are systems that don't share memory or clock, in distributed systems nodes connect and relay information by exchanging the information over a communication medium. The different computer in distributed system have their own memory and OS, local resources are owned by the node using the resources. While the resources that is been accessed over the network or communication medium is known to be remote resources [5]-[7]. **Figure 1** shows the communication network between systems in the distributed environment.

In distributed system, pool of rules are executed to synchronise the actions of various or different processes on a communication network, thereby forming a distinct set of related tasks [6]-[9]. The independent system or computers access resources remotely or locally in the distributed system communication environment, these

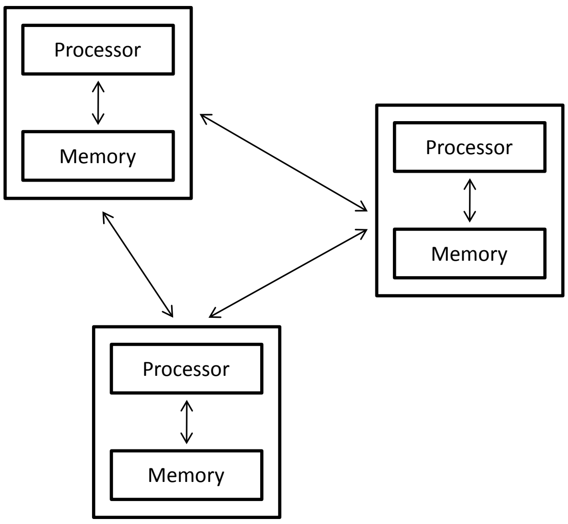

Figure 1. Distributed system.

resources are put together to form a single intelligible system. The user in the distributed environment is not aware of the multiple interconnected system that ensures the task is carried out accurately. In distributed system, no single system is required or carries the load of the entire system in processing a task [8] [9].

3. Distributed System Architecture

The architecture of distributed system is built on existing OS and network software [8]. Distributed system encompasses the collection of self-sufficient computers that are linked via a computer network and distribution middleware. The distribution middleware in distributed system, enables the corresponding computers to manage and share the resources of the corresponding system, thus making the computer users to see the system as a single combined computing infrastructure [9] [10]. Middleware is the link that joins distributed applications across different geographical locations, different computing hardware, network technologies, operating systems, and programming languages. The middleware delivers standard services such as naming, concurrency control, event distribution, security, authorization etc. **Figure 2** shows the distributed system architecture, with the middleware offering its services to the connected systems in the distributed environment [10] [11].

In distributed system, the structure can be fully connected networks or partially connected networks [12]-[15]. As shown in **Figure 3**, a full connected network, is a network where each node is connected together. The disadvantage of this network is that when a new computer added, it physically increase the number of nodes connected to nodes, because the network connects node to node. Because of the increase in nodes, the number of file descriptors and difficulty for each node to communicate are increased heavily. File Descriptors is an intellectual indicator used to access a file or other input/output resource, such as a pipe or network connection [15]-[17]. Hence, the ability for the networked systems to continue functioning well is limited to the connected nodes ability of open the file descriptors and also the capability to manage new connections. The fully linked network systems are reliable because the message sent from one node to another node goes through one link, and when a node fails to function or a link fails, other nodes in the network can still communicate with other nodes.

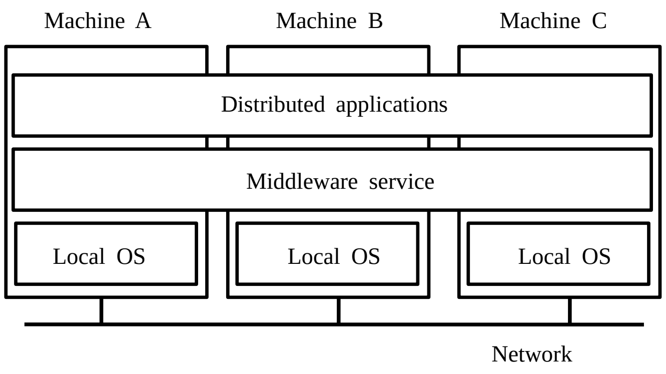

Figure 2. A simple architecture of a distributed system.

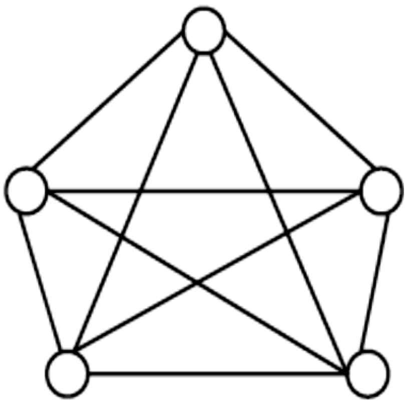

Figure 3. Fully connected network.

In the partially connected network, some node have direct links while others don’t. Some models of partially connected networks are star structured networks, multi-access bus net work, ring structured network, and tree structured network. In **Figures 4**-7 illustrates the corresponding networks. The disadvantages in these network

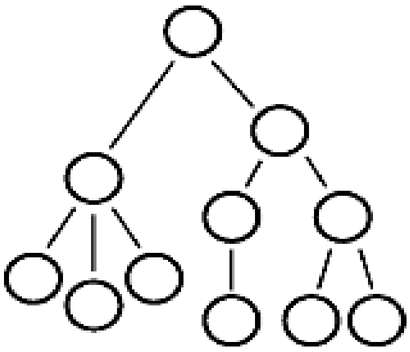

Figure 4. Tree structured network.

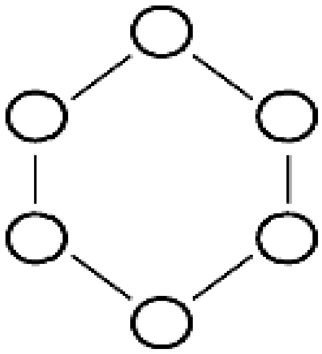

Figure 5. Ring structured network.

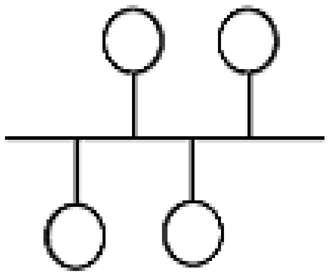

Figure 6. Multi-access bus network.

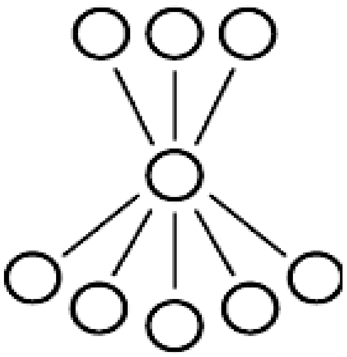

Figure 7. Star structured network.

in are: in the Star designed network, when the main node fails to function the entire networked system stops to function they collapse. In multi-access bus network, nodes are connected to each other through a communication link "a bus". If the bus link connecting the nodes fails to function, all other nodes can't connect to each other, and the performance of the network drops as more nodes or computers are added to the system or heavy traffic occurs in the system. In the ring network, nodes are connected at least to two other nodes in the network creating a path for signals to be exchanged between the connected nodes. As new nodes are added to the network, the transmission delay becomes longer. If a node fail every other node in the network can be inaccessible. In the tree structured network, this is like a net work with hierarchy, each node in the network have a fixed number nodes that is attached to it in the sub level of the tree. In this network messages that are transmitted from the parent to the child nodes goes through one link.

For a distributed system to perform and function according to build, it must have the following characteristics; Fault Tolerant, Scalability, Predictable Performance, Openness, Security, and Transparency.

4. Fault Tolerance Systems

Fault tolerance system is a vital issue in distributed computing; it keeps the system in a working condition in subject to failure. The most important point of it is to keep the system functioning even if any of its part goes off or faulty [18]-[20].

For a system to be fault tolerant, it is related to dependable systems. Dependability covers some useful requirements in the fault tolerance system these requirements include: Availability, Reliability, Safety, and Maintainability.

Availability: This is when a system is in a ready state, and is ready to deliver its functions to its corresponding users. Highly available systems works at a given instant in time.

Reliability: This is the ability for a computer system run continuously without a failure. Unlike availability, reliability is defined in a time interval instead of an instant in time. A highly reliably system, works constantly in a long period of time without interruption.

Safety: This is when a system fails to carry out its corresponding processes correctly and its operations are incorrect, but no shattering event happens.

Maintainability: A highly maintainability system can also show a great measurement of accessibility, especially if the corresponding failures can be noticed and fixed mechanically.

As we have seen, fault tolerance system is a system which has the capacity of or to keep running correctly and proper execution of its programs and continues functioning in the event of a partial failure [21] [22]. Although sometimes the performance of the system is affected due to the failure that occurred. Some of the fault is narrowed down to Hardware or Software Failure (Node Failure) or Unauthorised Access (Machine Error). Errors caused by fault tolerance events are separated into categories namely; performance, omission, timing, crash, and fail-stop [22]-[24].

Performance: this is when the hardware or software components cannot meet the demands of the user.

Omission: is when components cannot implement the actions of a number of distinctive commands.

Timing: this is when components cannot implement the actions of a command at the right time.

Crash: certain components crash with no response and cannot be repaired.

Fail-stop: is when the software identifies errors, it ends the process or action, this is the easiest to handle, sometimes its simplicity deprives it from handling real situations.

In addition to the error timing, three situations or form can be distinguished: 1) Permanent error; these causes damage to software components and resulting to permanent error or damage to the program, preventing it from running or functioning. In this case a restart of the program is done, an example is when a program crashes. 2) Temporary error; this only result to a brief damage to the software component, the damage gets resolved after some time and the corresponding software continues to work or function normally. 3) Periodic errors; these are errors that occurs occasionally. For example when there's a software conflict between two software when run at the same time. In dealing with this type of error, one of the programs or software is exited to resolve the conflict.

Most computers if not all have some fault tolerance technique such as micro diagnosis [25] [26], parity checking [27]-[29], watchdog timers [30]-[34], etc. an incompletely fault tolerant system have inbuilt resources to cause a reduction in its specified computing capability and reduce to a smaller or lower system by removing some programs that have been used previously or by reducing the rate at which specified processes are executed.

The reduction is due to the decrease or slowdown in the operational hardware configuration or it may be some design faults in the hardware.

5. Basic Concept of Fault Tolerance Systems

Fault tolerance mechanism can be divided into three stages; Hardware, Software, and System Fault [34].

Hardware Fault Tolerance: This involves the provision of supplementary backup hardware such as; CPU, Memory, Hard disks, Power Supply Units, etc. hardware fault tolerance can only deliver support for the hardware by providing the basic hardware backup system, it can't stop or detect error, accidental interfering with programs, program errors, etc. In hardware fault tolerance, computer systems that resolves fault occurring from hardware component automatically are built. This technique often partition the node into units that performance as a fault control area, each module is backed up with a defensive redundancy, the reason is that if one of the modules fails, the others can act or take up its function. There are two approach to hardware fault recovery namely; Fault Masking and Dynamic Recovery [35]-[37].

Fault Masking: This is an important redundancy method that fully covers faults within a set of redundant units or components. Other identical units carry out or implement the same tasks, and their outputs were noted to have removed errors created by a defective module. Commonly used fault masking module it the Triple Modular Redundancy (TMR). The TMR triplicate the circuitry and selected [38] [39]. The selected electrical system can also be triplicated so that the selected circuitry failures can be corrected by the same process. The selected system in the TMR needs more hardware, this enables computations to continue without been interrupted when a fault is detected or occurs, tolerating the operating system to be used [40] [41].

Dynamic Recovery: In dynamic recovery, special mechanism is essential to discover faults in the units, perform a switch on a faulty module, puts in a spare, and carryout some software actions necessary to restore and continue computation such as; rollback, initialization, retry, and restart. This requires special hardware and software to make this work in single computer, but in a multicomputer situation, the function is carried out by other processors [42]-[45].

Software Fault Tolerance: This is a special software designed to tolerate errors that would originate from a software or programming errors. The software fault tolerance utilize the static and dynamic redundancy methods similar to those used for hardware fault [46]. N-version programming approach uses the static redundancy like an independently program that does the same function creating out that are selected at special checkpoint. Another approach is the Design Diversity which this adds both hardware and software fault tolerance by deploying a fault tolerant system using diverse hardware and software in the redundant channels. In the Design diversity, every channel is intended to carry out the same function and a mechanism is in check to see if any of the channels changes from others. The aim of the Design Diversity is to tolerate faults from both hardware and software. This approach is very expensive, its use mainly is in the aircraft control applications.

Note: Software Fault Tolerance also consists of checkpoints storage and rollback recovery. Checkpoints are like a safe state or snapshot of the entire system in a working state. This is done regularly. The snapshot holds all the required information to restart the program from the checkpoint. The usefulness of the software fault tolerance is to create an application that would store checkpoints regularly for targeted systems.

System Fault Tolerance: This is a complete system that stores not just checkpoints, it detects error in application, it stores memory block, program checkpoint automatically. When a fault or an error occurs, the system provides a correcting mechanism thereby correcting the error. **Table 1** shows the comparison of three fault tolerance mechanism.

Table 1. Comparison of fault tolerance mechanism.

Mechanism	Hardware Fault Tolerance	Software Fault Tolerance	System Fault Tolerance
Major technique	Hardware backup	Checkpoint storage Rollback recovery	Architecture with error detecting & correcting
Design complexity	Low	Medium	High
Time/cost expenditure	Low	Medium	High
Fault-tolerance Level	Low	Medium	High

6. Fault Tolerance Mechanism in Distributed Systems

6.1. Replication Based Fault Tolerance Technique

The replication based fault tolerance technique is one of the most popular method. This technique actually replicate the data on different other system. In the replication techniques, a request can be sent to one replica system in the midst of the other replica system. In this way if a particular or more than one node fails to function, it will not cause the whole system to stop functioning as shown in **Figure 8**. Replication adds redundancy in a system.

There are different phase in the replication protocol which are client contact, server coordination, execution, agreement, coordination and client response. Major issues in replication based techniques are consistency, degree of replica, replica on demand etc.

Consistency: This is a vital issue in replication technique. Several copies of the same entity create problem of consistency because of update that can be done by any of the user. The consistency of data is ensured by some criteria such as linearizability [47], sequential consistency and casual consistency [48] etc. sequential and linearizability consistency ensures strong consistency unlike casual consistency which defines a weak consistency criterion. For example a primary backup replication technique guarantee consistency by linerarizability, likewise active replication technique.

Degree or Number of Replica: The replication techniques utilises some protocols in replication of data or an object, such protocol are: Primary backup replication [49], voting [50] and primary-per partition replication [51]. In the degree of replication, to attain a high level of consistency, large number of replicas is needed. If the number of replica is low or less it would affect the scalability, performance and multiple fault tolerance capability. To solve the issue of less number of replica, in [51] adaptive replicas creation algorithm was proposed.

6.2. Process Level Redundancy Technique

This fault tolerance technique is often used for faults that disappears without anything been done to remedy the situation, this kind of fault is known as transient faults. Transient faults occurs when there's a temporary malfunction in any of the system component or sometimes by environmental interference. The problem with transient faults is that they are hard to handle and diagnose but they are less severe in nature. In handling of transient fault, software based fault tolerance technique such as Process-Level Redundancy (PLR) is used because hardware based fault tolerance technique is more expensive to deploy. As shown in **Figure 9**, the PLR compares processes to ensure correct execution and also it creates a set of redundant processes apiece application process. Redundancy at the process level enables the OS to schedule easily processes across all accessible hardware resources.

The PLR provides improved performance over existing software transient fault tolerance techniques with a 16.9% overhead for detection of fault [53]. PLR uses a software-centric approach which causes a shift in focus from guaranteeing hardware execution correctly to ensuring a correct software execution.

Check Pointing and Roll Back: This is a popular technique which in the first part "check point" stores the current state of the system and this is done occasionally. The check point information is stored in a stable storage device for easy roll back when there's a node failure. Information that is stored or checked includes environment, process state, value of the registers etc. these information are very useful if a complete recovery needs to be done [50] [51]. **Figure 10** illustrates the check pointing techniques. The two most known type or roll back

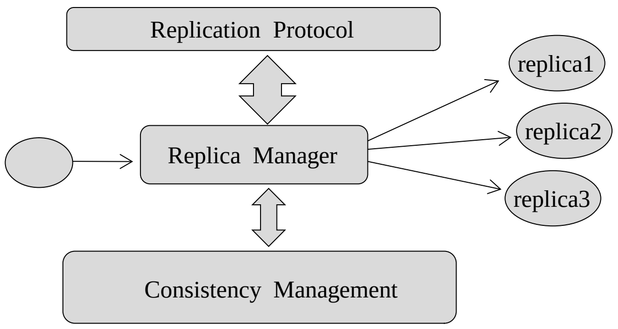

Figure 8. Replication based technique in distributed system.

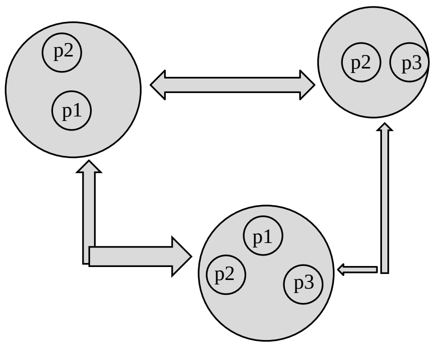

Figure 9. Process redundancy.

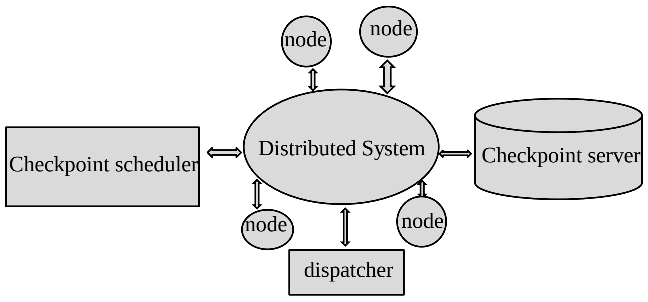

Figure 10. Check pointing technique.

recovery are the checkpoint and log based roll back recovery technique. Each of the type of rollback recovery technique uses different mechanism; the checkpoint based uses the checkpoints states that it has stored in a stable storage device, while the log based rollback recovery techniques combines both check pointing and logging of events [51].

In recovery form system failures, there are two type of check point technique that is used; coordinated and uncoordinated checkpoint techniques, these techniques are related with message logging [34].

Coordinated Check Point: In this technique, check are coordinated to save a consistent state because the coordinated checkpoint are consistent set of checkpoint. If the checkpoints are not consistent a full and complete rollback of the system can't be done [52]. In a situation where there's frequent failure, coordinated check point technique can't be used. The recovery time can be set to a higher value or lower value, when set to a lower value, it improves performance of the technique because it only select the recovery to last correct state of the system instead from the very first state or checkpoint.

Uncoordinated Check Point: This technique combines the message logging to ensure that the rollback state is correct. The uncoordinated check point technique executed checkpoints independently as well as recovery. There are three type of message logging protocols: optimistic, pessimistic and casual. In the optimistic protocol ensures all messages are logged. The pessimistic protocol makes sure that all the message that is received by a process are logged appropriately and stored in a stable and reliable storage media before it is forwarded into the system. While the causal protocol just log the message information of a process in all processes that are causally dependent [53].

6.3. Fusion Based Technique

Replication is the most widely used method or technique in fault tolerance. The main downside is the multiple of backups that it incurs. Because the backups increase as faults increase and the cost of management is very expensive, the fusion based technique solves that problem. Fusion based technique stands as an alternative because it requires fewer backup machines compared to the replication based technique. As shown in **Figure 11**, the backup machines are fused corresponding to the given set of machines [53] [54]. The fusion based technique has a very high overhead during recovery process and it's acceptable in low probability of fault in a system.

From **Table 2**, it is clear that all methods having capability to handle multiple faults. In all methods

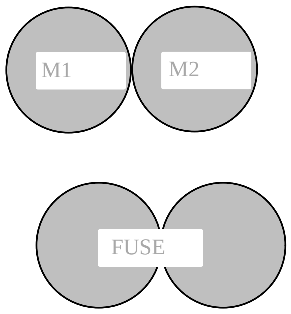

Figure 11. Fusion process technique.

Table 2. shows compares the different fault tolerance technique or mechanism in distributed system.

Major Factors	Replication Based Technique	Checking Point/Roll Back Technique	Fusion Based Technique	Process Level Redundancy Technique
Working	Redirected to replica	State saved on stables to rage used for recovery	Back up machine	A set of redundant process
Consistency	Some criterion; linearizability.	Avoiding orphan messages	Among backup machines	Not a major issues
Multiple Faults Handling	Depend upon number of replica.	Depend upon Check pointing scheduling	Depends upon number of back machine	Depends upon set of redundant process
Performance	Decreases as number of replica increases.	Decrease with frequency and size of checkpoint	Decrease in case of faults as recovery cost is high	Decrease as faults appears disappear
N-Faults	N replicas ensure n-1 faults	Uncoordinated Pessimistic and N level disk less used for N-1 Faults	In order handle Extra N faults N backups machine are required	Scaling the number of process and Majority voting
Multiple Failure Detector	Reliable, Accurate, Adaptive	Reliable, Accurate, Adaptive	Reliable, Accurate, Adaptive	Reliable, Accurate, Adaptive

performance can be improved by focusing on or addressing the serious aspects involved. In all the techniques involved, there is strong need for reliable, accurate and pure adaptive multiple failure detector mechanism [53], [54].

7. Conclusion

Fault tolerance is a major part of distributed system, because it ensures the continuity and functionality of a system at a point where there's a fault or failure. This research showed the different type of fault tolerance technique in distributed system such as the Fusion Based Technique, Check Pointing and Roll Back Technique, and Replication Based Fault Tolerance Technique. Each mechanism is advantageous over the other and costly in deployment. In this paper we highlight the levels of fault tolerance such as the hardware fault tolerance which ensures that additional backup hardware such as memory block, CPU, etc., software fault tolerance system comprises of checkpoints storage and rollback recovery mechanisms, and the system fault tolerance is a complete system that does both software and hardware fault tolerance, to ensure availability of the system during failure, error or fault. Future research would be conducted on comparing the various data security mechanisms and their performance metrics.

References

[1] Sebepou, Z. and Magoutis, K. (2011) CEC: Continuous Eventual Checkpointing for Data Stream Processing Operators. *Proceedings of IEEE/IFIP* 41*st International Conference on Dependable Systems and Networks*, 145-156. http://dx.doi.org/10.1109/dsn.2011.5958214

[2] Sari, A. and Necat, B. (2012) Impact of RTS Mechanism on TORA and AODV Protocol's Performance in Mobile Ad Hoc Networks. *International Journal of Science and Advanced Technology*, **2**, 188-191.

[3] Chen, W.H. and Tsai, J.C. (2014) Fault-Tolerance Implementation in Typical Distributed Stream Processing Systems.

[4] Sari, A. and Necat, B. (2012) Securing Mobile Ad Hoc Networks against Jamming Attacks through Unified Security Mechanism. *International Journal of Ad Hoc*, *Sensor & Ubiquitous Computing*, **3**, 79-94. http://dx.doi.org/10.5121/ijasuc.2012.3306

[5] Balazinska, M., Balakrishnan, H., Madden, S. and Stonebraker, M. (2008) Fault-Tolerance in the Borealis Distributed Stream Processing System. *ACM Transactions on Database Systems*, **33**, 1-44. http://dx.doi.org/10.1145/1331904.1331907

[6] Sari, A. (2014) Security Approaches in IEEE 802.11 MANET—Performance Evaluation of USM and RAS. *International Journal of Communications*, *Network*, *and System Sciences*, **7**, 365-372. http://dx.doi.org/10.4236/ijcns.2014.79038

[7] Elnozahy, E.N.M., Alvisi, L., Wang, Y.M. and Johnson, D.B. (2002) A Survey of Rollback-Recovery Protocols in Message-Passing Systems. *ACM Computing Surveys*, **34**, 375-408. http://dx.doi.org/10.1145/568522.568525

[8] Sari, A. (2014) Security Issues in RFID Middleware Systems: A Case of Network Layer Attacks: Proposed EPC Implementation for Network Layer Attacks. *Transactions on Networks & Communications*, **2**, 1-6. http://dx.doi.org/10.14738/tnc.25.431

[9] Andrew, S. (1995) Tanenbaum Distributed Operating Systems. Prentice Hall, Upper Saddle River.

[10] Sari, A. (2015) Lightweight Robust Forwarding Scheme for Multi-Hop Wireless Networks. *International Journal of Communications*, *Network and System Sciences*, **8**, 19-28. http://dx.doi.org/10.4236/ijcns.2015.83003

[11] Coulouris, G., Dollimore, J. and Kindberg, T. (2001) Distributed Systems: Concepts and Design, 4th Edition, Pearson Education Ltd., New York.

[12] Carter, W.C. and Bouricius, W.G. (1971) A Survey of Fault-Tolerant Computer Architecture and Its Evaluation. *Computer*, **4**, 9-16.

[13] Short, R.A. (1968) The Attainment of Reliable Digital Systems through the Use of Redundancy—A Survey. *IEEE Computer Group News*, **2**, 2-17.

[14] Sari, A. (2015) Two-Tier Hierarchical Cluster Based Topology in Wireless Sensor Networks for Contention Based Protocol Suite. *International Journal of Communications*, *Network and System Sciences*, **8**, 29-42. http://dx.doi.org/10.4236/ijcns.2015.83004

[15] Cooper, A.E. and Chow, W.T. (1976) Development of On-Board Space Computer Systems. *IBM Journal of Research and Development*, **20**, 5-19. http://dx.doi.org/10.1147/rd.201.0005

[16] Tanenbaum, A. and Van Steen, M. (2007) Distributed Systems: Principles and Paradigms. 2nd Edition, Pearson Prentice Hall, Upper Saddle River.

[17] Koren, I. and Krishna, C.M. (2007) Fault-Tolerance Systems. Elsevier Inc., San Francisco.

[18] Sari, A. and Onursal, O. (2013) Role of Information Security in E-Business Operations. *International Journal of Information Technology and Business Management*, **3**, 90-93.

[19] Avizienis, A., Kopetz, H. and Laprie, J.C. (1987) Dependable Computing and Fault-Tolerant Systems, Volume 1: The Evolution of Fault-Tolerant Computing. Springer-Verlag, Vienna, 193-213.

[20] Sari, A. and Çağlar, E. (2015) Performance Simulation of Gossip Relay Protocol in Multi-Hop Wireless Networks. *Social and Applied Sciences Journal*, *Girne American University*, **7**, 145-148.

[21] Harper, R., Lala, J. and Deyst, J. (1988) Fault-Tolerant Parallel Processor Architectural Overview. *Proceedings of the* 18*st International Symposium on Fault-Tolerant Computing*, Tokyo, 27-30 June 1988.

[22] Sari, A. and Rahnama, B. (2013) Addressing Security Challenges in WiMAX Environment. In: *Proceedings of the* 6*th International Conference on Security of Information and Networks*, ACM Press, New York, 454-456. http://dx.doi.org/10.1145/2523514.2523586

[23] Briere, D. and Traverse, P. (1993) AIRBUS A320/A330/A340 Electrical Flight Controls: A Family of Fault-Tolerant Systems. *Proceedings of the* 23*rd International Symposium on Fault-Tolerant Computing*, Toulouse, 22-24 June 1993.

[24] Charron-Bost, B., Pedone, F. and Schiper, A. (2010) Replication: Theory and Practice. *Lecture Notes in Computer Science*, **5959**.

[25] Sari, A. (2015) Security Issues in Mobile Wireless Ad Hoc Networks: A Comparative Survey of Methods and Techniques to Provide Security in Wireless Ad Hoc Networks. New Threats and Countermeasures in Digital Crime and Cyber Terrorism, IGI Global, Hershey, 66-94.

[26] Sari, A. and Rahnama, B. (2013) Simulation of 802.11 Physical Layer Attacks in MANET. *Proceedings of the Fifth International Conference on Computational Intelligence*, *Communication Systems and Networks* (*CICSyN*), Madrid, 5-7 June 2013, 334-337. http://dx.doi.org/10.1109/cicsyn.2013.79

[27] Tanenbaum, A.S. and van Steen, M. (2002) Distributed Systems: Principles and Paradigms. Pearson Education Asia.

[28] Sari, A., Rahnama, B. and Caglar, E. (2014) Ultra-Fast Lithium Cell Charging for Mission Critical Applications. *Transactions on Machine Learning and Artificial Intelligence*, **2**, 11-18. http://dx.doi.org/10.14738/tmlai.25.430

[29] Ebnenasir, A. (2005) Software Fault-Tolerance. Computer Science and Engineering Department, Michigan State University, East Lansing. http://www.cse.msu.edu/~cse870/Lectures/SS2005 /ft1.pdf

[30] Birman, K. (2005) Reliable Distributed Systems: Technologies, Web Services and Applications. Springer-Verlag, Berlin.

[31] Obasuyi, G. and Sari, A. (2015) Security Challenges of Virtualization Hypervisors in Virtualized Hardware Environment. *International Journal of Communications, Network and System Sciences*, **8**, 260-273. http://dx.doi.org/10.4236/ijcns.2015.87026

[32] Avizienis, A. (1975) Architecture of Fault-Tolerant Computing Systems. *Proceedings of the* 1975 *International Symposium on Fault-Tolerant Computing*, Paris, 18-20 June 1975, 3-16.

[33] Sari, A. (2015) A Review of Anomaly Detection Systems in Cloud Networks and Survey of Cloud Security Measures in Cloud Storage Applications. *Journal of Information Security*, **6**, 142-154. http://dx.doi.org/10.4236/jis.2015.62015

[34] Short, R.A. (1968) The Attainment of Reliable Digital Systems through the Use of Redundancy—A Survey. *IEEE Computer Group News*, **2**, 2-17.

[35] Sari, A. (2014) Influence of ICT Applications on Learning Process in Higher Education. *Procedia—Social and Behavioral Sciences*, **116**, 4939-4945. http://dx.doi.org/10.1016/j.sbspro.2014.01.1053

[36] Huang, M. and Bode, B. (2005) A Performance Comparison of Tree and Ring Topologies in Distributed Systems. *Proceedings of the* 19*th IEEE International Parallel and Distributed Processing Symposium*, Denver, 4-8 April 2005, 258.1. http://dx.doi.org/10.1109/IPDPS.2005.57

[37] Huang, M. (2004) A Performance Comparison of Tree and Ring Topologies in Distributed System. Master's Thesis. www.osti.gov

[38] Minar, N. (2001) Distributed Systems Topologies: Part 1. http://openp2p.com

[39] Wiesmann, M., Pedone, F., Schiper, A., Kemme, B. and Alonso, G. (2000) Understanding Replication in Databases and Distributed Systems. Research Supported by EPFL-ETHZ DRAGON Project and OFES.

[40] Herlihy, M. and Wing, J. (1990) Linearizability: A Correctness Condition for Concurrent Objects. *ACM Transactions on Programming Languages and Systems*, **12**, 463-492. http://dx.doi.org/10.1145/78969.78972

[41] Ahamad, M., Hutto, P.W., Neiger, G., Burns, J.E. and Kohli, P. (1994) Causal Memory: Definitions, Implementations and Programming. TR GIT-CC-93/55, Georgia Institute of Technology, Atlanta.

[42] Rahnama, B., Sari, A. and Makvandi, R. (2013) Countering PCIe Gen. 3 Data Transfer Rate Imperfection Using Serial Data Interconnect. *Proceedings of the* 2013 *International Conference on Technological Advances in Electrical, Electronics and Computer Engineering* (*TAEECE*), Konya, 9-11 May 2013, 579-582. http://dx.doi.org/10.1109/TAEECE.2013.6557339

[43] Budhiraja, N., Marzullo, K., Schneider, F.B. and Toueg, S. (1993) The Primary-Backup Approach. In: Mullender, S., Ed., *Distributed Systems*, ACM Press, New York, 199-216.

[44] Gifford, D.K. (1979) Weighted Voting for Replicated Data. *Proceedings of the Seventh ACM Symposium on Operating Systems Principles*, Pacific Grove, 10-12 December 1979, 150-162. http://dx.doi.org/10.1145/800215.806583

[45] Osrael, J., Froihofer, L., Goeschka, K.M., Beyer, S., Galdamez, P. and Munoz, F. (2006) A System Architecture for Enhanced Availability of Tightly Coupled Distributed Systems. *Proceedings of the First International Conference on Availability, Reliability and Security*, Vienna, 20-22 April 2006.

[46] Cao, H.H. and Zhu, J.M. (2008) An Adaptive Replicas Creation Algorithm with Fault Tolerance in the Distributed Storage Network. *Proceedings of the Second International Symposium on Intelligent Information Technology Application*, Shanghai, 20-22 December 2008, 738-741.

[47] Shye, A., Blomstedt, J., Moseley, T., Reddi, V. and Connors, D.A. (2008) PLR: A Software Approach to Transient Fault Tolerance for Multicore Architectures. *IEEE Transactions on Dependable and Secure Computing*, **6**, 135-148. http://dx.doi.org/10.1109/TDSC.2008.62

[48] Agarwal, V. (2004) Fault Tolerance in Distributed Systems. Indian Institute of Technology, Kanpur. www.cse.iitk.ac.in/report-repository

[49] Jung, H., Shin, D., Kim, H. and Lee, H.Y. (2005) Design and Implementation of Multiple Fault Tolerant MPI over Myrinet (M3). In: *Proceedings of the* 2005 *ACM/IEEE Conference on Supercomputing*, IEEE Computer Society, Washington DC, 32. http://dx.doi.org/10.1109/SC.2005.22

[50] Elnozahy, M., Alvisi, L., Wang, Y.M. and Johnson, D.B. (1996) A Survey of Rollback-Recovery Protocols in Message Passing Systems. Technical Report CMU-CS-96-81, School of Computer Science, Carnegie Mellon University, Pittsburgh.

[51] Alvisi, L. and Marzullo, K. (1995) Message Logging: Pessimistic, Optimistic, and Causal. *Proceedings of the* 15*th International Conference on Distributed Computing*, *Systems* (*ICDCS* 1995), Vancouver, 30 May-2 Jun 1995, 229-236. http://dx.doi.org/10.1109/ICDCS.1995.500024

[52] Garg, V.K. (2010) Implementing Fault-Tolerant Services Using Fused State Machines. Technical Report ECE-PDS-2010-001, Parallel and Distributed Systems Laboratory, ECE Department, University of Texas, Austin.

[53] Xiong, N., Cao, M., He, J. and Shu, L. (2009) A Survey on Fault Tolerance in Distributed Network Systems. *Proceedings of the* 2009 *International Conference on Computational Science and Engineering*, Vancouver, 29-31 August 2009, 1065-1070. http://dx.doi.org/10.1109/CSE.2009.497

[54] Tian, D., Wu, K. and Li, X. (2008) A Novel Adaptive Failure Detector for Distributed Systems. *Proceedings of the* 2008 *International Conference on Networking*, *Architecture*, *and Storage*, Chongqing, 12-14 June 2008, 215-221. http://dx.doi.org/10.1109/NAS.2008.37