Challenges and New Avenues in Existing Replication Techniques

Furat F. Altukhaim and Almetwally M. Mostafa
Information Systems, King Saud University, Riyadh, Saudi Arabia

Keywords: Load Balance, Replication, Replication Challenges, Active Replication, Primary-Backup Replication, Chain Replication, Mencius, Egalitarian Paxos, Object Ownership Distribution.

Abstract: Over recent years, the curve of the importance of data replication has risen steeply owing to the fact that databases are increasingly deployed over clusters of different workstations over time. A variety of replication techniques have been introduced to the distributed systems field which, in this paper, are classified based on whether they have an unbalanced load between servers or not (classic and modern). Replication techniques from both categories can be enhanced by avoiding some of the challenges that are illustrated in detail in this paper. Moreover, this paper analyses replication techniques in each category by exploring their strengths and weaknesses as well as providing possible novel solutions that can diminish or eliminate these challenges and introduces a brief description of the Dynamic Object Ownership Distribution Protocol that aims at increasing throughput by increasing the rate of performing transactions locally in addition to viewing a promising preliminary results of its performance.

1 INTRODUCTION

We have witnessed a noticeable increase in attention paid to the variety of replication techniques since the world has now become a small village which makes the need for data to be deployed over the whole globe. Systems use redundant data through the utilization of replication techniques. Replication can be defined as creating two or more copies of a data object with the intention of providing high availability, consistency, and fault-tolerance. These replicated data objects are stored at various servers to allow accessibility by clients in cases where a server is pulled away either by obligation or as an option (Özsu and Valduriez, 2011, Charron-Bost et al., 2010, Mostafa and Youssef, 2014b).

Replication techniques can be categorized into 1) classic replication techniques which include Active Replication when it is used by Paxos, the Primary-Backup Replication which uses Passive Replication, and Chain Replication which is derived from the Primary-Backup Replication; and 2) modern replication techniques which include Mencius, Egalitarian Paxos, Object Ownership Distribution Replication, and others. In this paper, when we mention Paxos, we mean the replication process in Paxos that uses Active Replication. Active Replication (State Machine Replication), is a technique that has been massively used to implement critical systems such as data stores and coordination services and in Internet scale infrastructures such as Yahoo, Google, and MSN. It works by making all the servers in the system execute the same set of operations in the same order which requires that the process hosted by the servers to be deterministic (Charron-Bost et al., 2010, Schneider and Zhou, 2005, Sousa and Bessani, 2012, Dettoni et al., 2013). In passive replication, there is one server, the primary server (sequencer), which acts as a single organizer for other servers in the system because it executes operations and propagates the new state to them (Rao, 2008, Cecchet et al., 2008, Lang et al., 2010, Effatparvar et al., 2010, Budhiraja et al., 1993, Mostafa and Youssef, 2014a).

Classic replication techniques have similar issues in that they have a server that is a bottleneck, which means that management responsibilities are cohesively done by a single server and not distributed between servers in a cell as effectively as they could be. This leads to an imbalance in the communication pattern, which in turn leads to limiting the available network bandwidth. In addition, in the case of a leader or primary server failure, re-electing a new one can cause performance degradation. This is because in situations where a leader or primary server needs to be pulled out of the system for any reason, it is not possible to perform this operation until the server is shut down or

Altukhaim, F. and Mostafa, A.
Challenges and New Avenues in Existing Replication Techniques.
In *Proceedings of the 6th International Conference on Cloud Computing and Services Science (CLOSER 2016) - Volume 1*, pages 147-154
ISBN: 978-989-758-182-3

another protocol that is designed to do this particular job is used, such as the Primary Replacement Protocol (Mostafa and Youssef, 2014b) and the Primary Shift Protoco (Mostafa and Youssef, 2013). Thus, without a structured solution programmed to be a part of the replication protocol itself, the reconfiguration process to the leader or primary server may not be easy or effective. On the other hand, modern replication techniques are focusing on distributing responsibilities of the single point of failure that the classic ones struggle with.

There are many issues that can be found in a replication cell which affect the cell's performance. For instance, having an unbalanced load distribution between servers in a single cell due to the lack of an inherent load balancing technique; the lack of latency awareness between these servers and their clients and between the servers themselves; poor reconfiguration planning; the absence of techniques that allow the system to be aware of the resources capabilities (such as CPU or memory) of its servers; and the inability of the system to be aware of external environmental factors such as network changes and clients' behavior.

The rest of the paper is structured as follows: Section 2 discusses, reviews, and analyzes classic replication techniques and shows a detailed discussion regarding challenges that these techniques struggle with in addition to comparing their strengths and weaknesses. Section 3 discusses the same aspects that have been discussed in section 2 but with modern replication techniques. Section 4 proposes suggestions that could be utilized to improve replication techniques. Section 5 proposes the Dynamic Object Ownership Distribution Protocol and shows some of its preliminary results. Section 6 concludes the paper.

2 DESCRIPTION AND CHALLENGES OF CLASSIC REPLICATION TECHNIQUES

This section reviews classic replication techniques including Paxos, the Primary-Backup Replication, and Chain Replication, and some of the challenges associated with them.

2.1 Paxos

Paxos is a consensus protocol that uses state machines in its replication process (Bolosky et al., 2011, Lamport, 1998, Lamport, 2001, Lampson, 2001). It is the first example that comes to mind when thinking of a consensus protocol. It results in an agreement on the order of inputs between several servers, even if some of the servers crash and restart or the minority of them fail permanently. More details about Paxos in the paper (Tan et al., 2014). The Chubby lock service for loosely coupled distributed systems (Burrows, 2006) and Spanner as Google's globally distributed database (Corbett et al., 2013) are services that utilize Paxos as their core replication protocol.

The challenge that Paxos is facing is that the server leading the backups (followers) does all the management work without including any of the other servers to do that work. If the leader fails, then one of the backups is chosen, depending on a consensus decision to be the next leader. During the election process, any operations that are trying to reach this data partition need to stop working for a while until a new leader is elected (throughput drops to zero).

2.2 The Primary-Backup Replication

The Primary-Backup Replication (Cecchet et al., 2008, Lang et al., 2010, Effatparvar et al., 2010, Budhiraja et al., 1993, Mostafa and Youssef, 2014a, Schneider and Zhou, 2005) has a single server which is the primary one that is responsible for organizing and managing the locking/unlocking operations of objects. It is exclusively designated to do this job to make sure that consistency and serialization are applied all the time. Any other server in the system is a backup and does not have any of the management responsibilities. One of the services that sometimes utilizes the Primary-Backup Replication is Zookeeper (Hunt et al., 2010).

The challenge that the Primary-Backup Replication has is similar to the one that we have discussed with Paxos. It includes an unfair distribution of the management responsibilities between servers in a cell in addition to having difficulties when applying the election process which includes dropping the throughput to zero.

2.3 Chain Replication

It is one of the Primary-Backup Replication approach forms that improves throughput and availability (Van Renesse and Schneider, 2004). In a paper entitled "Chain Replication for Supporting High Throughput and Availability", the two researchers, Can Renesse and Schneider, described the Chain Replication process to be a group of

various master/slave replications where servers that store replicas of an object are linearly ordered to form a chain. All decisions about object updates are made by the head of the chain in a strict order and these decisions are propagated down the chain. All read-only queries are processed by the tail of the chain which is the server that is positioned last. One of the services that utilizes Chain Replication is Hibari (Fritchie, 2010). Figure 1 portrays how Chain Replication works.

The challenge that Chain Replication is facing is similar to the one that we have discussed in Paxos and the Primary-Backup Replication before. It includes not having a fair distribution of the management responsibilities between servers in its cell, since the head server is responsible for the update operations and the tail server for the read operations. In addition, there is a process that is employed in Chain Replication called the master process which deals with a server failure depending on the position of the failing server. So, it is difficult to apply reconfiguration to the head or tail server and having to apply the master process without preventing clients from accessing the data partition for a while.

The reason for these issues in these three replication protocols is the lack of an inherent load balancing technique that is built into the replication protocol itself. Table 1 shows a comparison between the classic replication techniques.

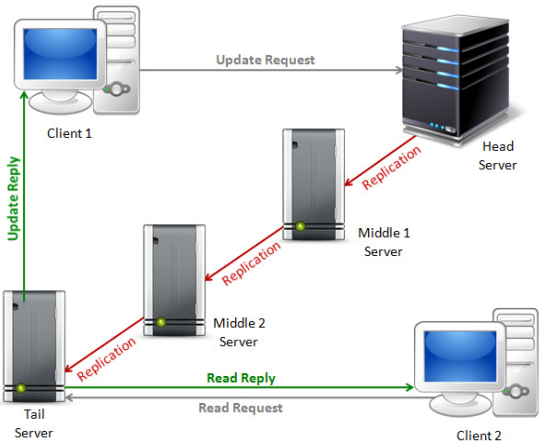

Figure 1: "Update request" and "Read request" in Chain replication (Fritchie, 2010).

3 DESCRIPTION AND CHALLENGES OF MODERN REPLICATION TECHNIQUES

This section reviews modern replication techniques including Mencius, Egalitarian Paxos, and Object Ownership Distribution Replication and some of the challenges associated with them. Their main objective is to improve a cell's throughput by distributing management responsibilities among servers.

Table 1: Comparison between Paxos, Primary-Backup Replication, and Chain Replication.

Type of Replication	Important Strengths	Important Weaknesses
Paxos	- It is preferable when dealing with byzantine faults. - Disseminating updates is a parallel process.	- The need for a leader election protocol. - No consensus can be made during the leader election process. - Most of the real-world servers are non-deterministic. - Determinism requires ordering which makes the process harder. - Cell's throughput equals the throughput of the leading server (bottleneck problem).
Primary-Backup Replication	- Cheap. - Disseminating updates is a parallel process.	- The need for a leader election protocol. - Cell's throughput equals the throughput of the primary server (bottleneck problem). - No transactions can be processed during the primary election process.
Chain Replication	- The primary role is split between two servers (head and tail). - It guarantees a strong consistency and high throughput. - For query requests, it provides low latency.	- The need for the master process. - For update requests, it provides high latency. - Cell's throughput equals the head's throughput (write) and the tail's throughput (read). - Disseminating updates is a serial process. - During the master process run, specific transactions cannot be processed after the failure of the head or tail server.

3.1 Mencius

Numerous protocols extend Paxos for the purpose of improving consensus performance, two of its variants being Fast Paxos (Lamport, 2006), which reduces latency, and CoRefP (Dobre et al., 2006), that simultaneously runs Paxos and Fast Paxos to improve the problem of collisions. Neither of these two variants are the most suitable choices when thinking of wide-area applications. A better solution can be found in Mencius (Mao et al., 2008) which is a multi-leader state machine replication protocol that comes originally from Paxos.

The idea of Mencius is to distribute the sequence of the instances of the consensus protocol between servers in a cell. For instance, if we have five servers in a cell, the first server is responsible for instance 0, 5, 10 and so on, the second server is responsible for instance 1, 6, 11, and so on, the third server is responsible for instance 2, 7, 12 and so on, and in a similar fashion with the rest of the servers. Figure 2 clarifies that.

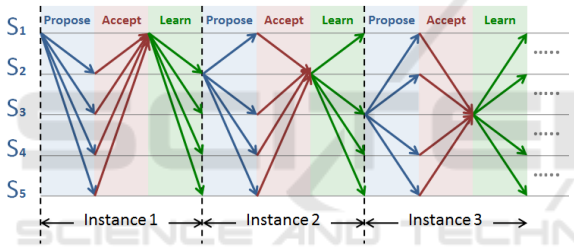

Figure 2: Rotating the leadership between five servers in Mencius (Wei et al., 2013).

Mencius is designed to eliminate some of the challenges that Paxos has. First, it decentralizes the responsibilities between servers by having an inherent load balancing technique that rotates leadership between all servers in a cell, thus eliminating the need for leader election. Second, it lowers latency since their clients use a local replica, which leads to increasing latency-awareness between clients and servers. Third, it smoothes the reconfiguration process due to the fact that when pulling away a server from a cell, another server arises to be the new leader. This can be achieved by having a specific time period for each server to hold the responsibility of leadership. However, the challenge is that, at any point in time, there is still one leader, which can have a negative effect on performance.

3.2 Egalitarian Paxos

Egalitarian Paxos (Moraru et al., 2013), based on Paxos, is a new consensus protocol that aims at a) reaching an optimal latency to commit a command in wide-area applications, b) having the best possible load balance between all servers so the throughput can be increased, c) having a graceful degradation in performance when some of the servers fail for some reason. One of the main properties of this protocol is that there is no server that is designated to be a leader and clients are able to select which server to send a command to.

To clarify the approach of Egalitarian Paxos, Figure 3 portrays an example of command A that is sent by client C1 to server S1 which is one of five servers in a cell. S1 then sends a preAccept message to the majority of servers which represents the quorum servers to S1. These servers incorporate S1 itself, S2, and S3. After that, S2 and S3 send acknowledgment messages to S1. Thus, S1 can commit locally and asynchronously notify the rest of the servers. In this case, S1 is the leader of command A.

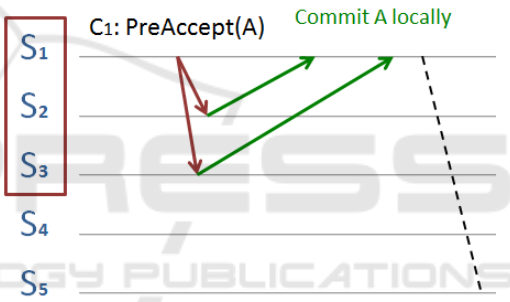

Figure 3: Executing command A and B in Egalitarian Paxos (Moraru et al., 2013).

Egalitarian Paxos has an input into eliminating some of the challenges that classic replication techniques have. First, as Mencius, it decentralizes the responsibilities between servers by having an inherent load balancing technique. This approach involves having clients choose any server as the command leader, which eliminates the need for electing a new leader. Second, it has latency-awareness between servers in the cell. This is done by taking into consideration the latency between servers in a cell so each server knows its quorum when it needs the majority of servers to execute a command. Third, it smoothes the reconfiguration process because the execution of commands is not disrupted when pulling away a server from a cell since clients can choose any other server in the cell. The challenge that this protocol has is that it has to go through two round trips instead of one when there are interfering or concurrent commands.

3.3 The Object Ownership Distribution

The Object Ownership Distribution Replication (Mostafa and Youssef, 2014a) aims to improve scalability and availability in Primary-Backup Replication systems. It decentralizes the exclusive management role which is the responsibility of the primary server in the Primary-Backup Replication. This approach utilizes the concept "object ownership", which can be defined as the exclusive right a server can have over an object to permit update transaction. This approach depends mainly on ownership distribution between servers, so the server that created the object has the right to own it. Figure 4 portrays the system architecture for the Object Ownership Distribution Replication Protocol.

This protocol has also eliminated some of the challenges faced by classic replication techniques.

First, we can see that it solves one of the challenges that we have discussed earlier, which is the lack of an inherent load balancing technique by having the ownership distribution mechanism that makes each server responsible for managing some of the objects in the system so all servers are involved in the management job. Second, when a server needs to be reconfigured, all the objects that it owns are owned by another server and there is no need for an election process that can affect the performance to such an extent. The challenge is that the ownership distribution is static in a dynamic environment, which is impractical and does not reflect the reality of the problem.

Table 2 shows a comparison between the modern replication techniques.

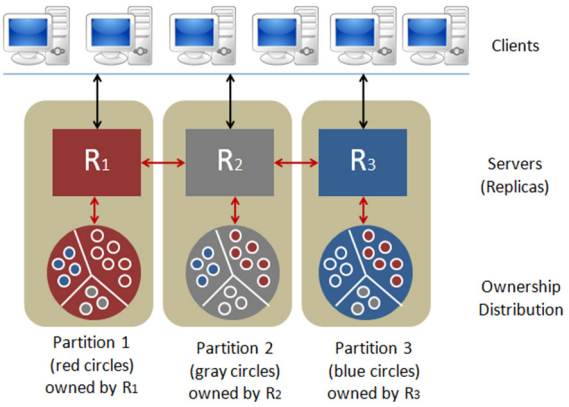

Figure 4: Object Ownership Distribution Replication system architecture (Mostafa and Youssef, 2014a).

4 OPPORTUNITIES FOR IMPROVEMENTS IN REPLICATION TECHNIQUES

The system to which we are trying to provide a replication technique can have several characteristics added to either the classic or modern replication techniques to improve their performance. One way to improve classic replication techniques is by injecting an inherent load balancing technique to them. This contributes to eliminating the bottleneck issue caused by having a single point of failure. Partitioning is one technique that has been utilized to solve scalability challenges; this technique started the problem of having imbalanced distributed systems. More information about partitioning is available in (Quamar et al., 2013; Ishikawa, 2013).

Table 2: Comparison between Mencius, Egalitarian Paxos, and Object Ownership Distribution Replication.

Type of Replication	Important Strengths	Important Weaknesses
Mencius Replication	- No single point of failure since there is no leading server. - Higher throughput than Paxos since it uses a partitioned leader scheme. - Full utilization of bandwidth. - Lower latency than Paxos when there is a small number of clients since they can use a local server to be the leader of their requests.	- There is only one leader at any point in time. - A server must hear from all other servers prior to committing any command.
Egalitarian Paxos	- No single point of failure since there is no leading server. - Improving load balance which increases throughput and scalability.	- It takes two round trips to deal with interfering or concurrent commands.
Object Ownership Distribution (OOD) Replication	- No single point of failure since there is no primary server. - Improving load balance which increases throughput and scalability.	- Static ownership.

So, when using replication techniques, each data partition needs to be replicated into a number of servers (normally 3 or 5 servers) which form a replication cell. In cells where there is a high load and great pressure is placed on the leader or primary server, this definitely impairs load balancing within a cell since this server has to process all clients' requests (Mao et al., 2008).

Latency-awareness can contribute in lowering latency which is an issue that is barely investigated by researchers when designing a replication protocol. It means that taking into consideration the latency between a client and server, a client can be aware of the closest server to it and then sends it a request. In Mencius, latency is reduced by using a local server to be the request leader. Latency-awareness can be improved between servers themselves, as in Egalitarian Paxos. Each server in a cell chooses its quorum based on the lowest latency between itself and the rest of the servers.

Reconfiguration (Aguilera et al., 2010) is an important aspect to which some replication protocols pay little attention. Whether planned or unplanned, it needs to be dealt with practically without having to stop accessing the data partition, when dealing with clients' requests, as this slows its performance. Classic replication techniques struggle with this issue. However, Modern replication techniques ease the process of reconfiguration by eliminating the need for electing a new leader or primary server.

One aspect that the available replication protocols can take into consideration is the process of testing the resources capabilities of each server and giving them responsibilities depending on the results of these tests, thereby improving performance in a cell.

Another aspect to consider is assigning responsibilities to servers based on network changes or the continuously changing number of clients' requests that each server receives.

Table 3 summarizes issues that we have discussed in this section, whether addressed or not, by the six common replication techniques that we have reviewed earlier in this paper.

5 THE DYNAMIC OBJECT OWNERSHIP DISTRIBUTION PROTOCOL

The Dynamic Object Ownership Distribution Protocol is a fault-tolerant protocol that aims to improve the operations load balance in scalable distributed systems for the purpose of increasing throughput and lowering latency. While the Object Ownership Distribution Protocol has a static ownership mechanism to own objects, this protocol has a dynamic way to do that. In an environment where the behavior of clients is extremely dynamic, it is only convenient to have a dynamic solution to a problem in a continuously changing environment.

The protocol gains its dynamicity from the fact that it allows object ownership to change from one server to another. The change depends on the number of update operations that are performed by each server on a certain object. The server that performs the highest number of operations on this object has the right to be its owner. The proposed protocol is also able to handle failures. Once a server fails, its objects are owned automatically by the server that has the highest ratio of operations after the failed server. When a server recovers, a

Table 3: Addressed and unaddressed issues in classic and modern replication techniques.

	Classic Replication Techniques			**Modern Replication Techniques**		
Issues	**Paxos**	**Primary-Backup Replication**	**Chain Replication**	**Mencius**	**Egalitarian Paxos**	**Object Ownership Distribution**
Inherent load balancing techniques	No	No	No	Yes	Yes	Yes
Latency awareness (between a client and a server)	No	No	No	Yes	No	No
Latency awareness (between servers themselves)	No	No	No	No	Yes	No
Smooth reconfiguration process	No	No	No	Yes	Yes	Yes
Awareness of the capabilities of servers' resources	No	No	No	No	No	No
Reactive nature to network changes	No	No	No	Yes	No	No
Reactive nature to the amount of clients' requests	No	No	No	No	No	No

comparison operation must be performed to check which server has the highest ratio of update operations between the recovered server and the new owner.

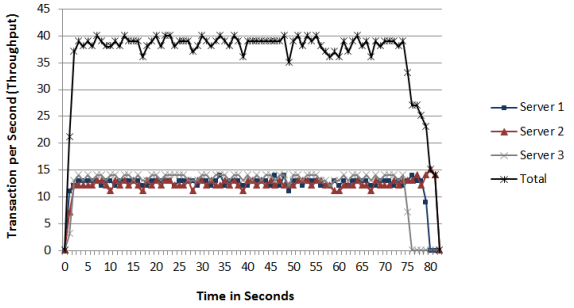

Figure 5: The throughput graph in the Object Ownership Distribution Protocol.

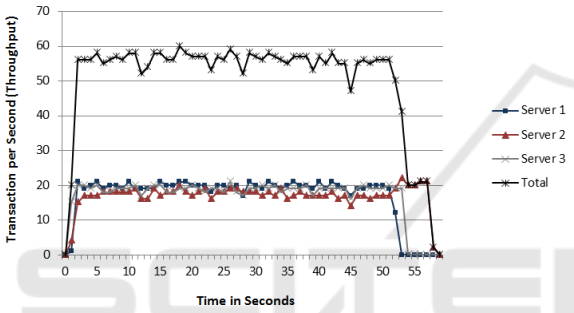

Figure 6: The throughput graph in the Dynamic Object Ownership Distribution Protocol.

In an experiment where the replication factor is three, each object is owned by a different server, and each client updates an object that did not create, the results of the throughput of the Object Ownership Distribution Protocol and the Dynamic Object Ownership Distribution Protocol are shown in Figure 5 and 6. As we can see, the result is in favor of the protocol that can perform the largest number of local transactions which is, in this experiment, the Dynamic Object Ownership Distribution Protocol.

6 CONCLUSIONS

Classic replication techniques suffer mainly from having the bottleneck issue that makes one server take on all the management responsibilities, which lowers throughput and increases latency. Furthermore, they struggle with performing effective reconfiguration operations. On the other hand, modern replication techniques decentralize the management responsibilities among servers in a cell. Numerous approaches to improve current replication techniques have been discussed in this paper. In addition, the Dynamic Object Ownership Distribution protocol is briefly discussed.

REFERENCES

Aguilera, M. K., Keidar, I., Malkhi, D., Martin, J.-P. & Shraer, A. 2010. Reconfiguring Replicated Atomic Storage: A Tutorial. *Bulletin Of The Eatcs*, 84-108.

Bolosky, W. J., Bradshaw, D., Haagens, R. B., Kusters, N. P. & Li, P. Paxos Replicated State Machines As The Basis Of A High-Performance Data Store. Nsdi, 2011.

Budhiraja, N., Marzullo, K., Schneider, F. B. & Toueg, S. 1993. The Primary-Backup Approach. *Distributed Systems,* 2, 199-216.

Burrows, M. The Chubby Lock Service For Loosely-Coupled Distributed Systems. Proceedings Of The 7th Symposium On Operating Systems Design And Implementation, 2006. Usenix Association, 335-350.

Cecchet, E., Candea, G. & Ailamaki, A. Middleware-Based Database Replication: The Gaps Between Theory And Practice. Proceedings Of The 2008 Acm Sigmod International Conference On Management Of Data, 2008. Acm, 739-752.

Charron-Bost, B., Pedone, F. & Schiper, A. 2010. *Replication: Theory And Practice*, Springer.

Corbett, J. C., Dean, J., Epstein, M., Fikes, A., Frost, C., Furman, J. J., Ghemawat, S., Gubarev, A., Heiser, C. & Hochschild, P. 2013. Spanner: Google's Globally Distributed Database. *Acm Transactions On Computer Systems (Tocs),* 31, 8.

Dettoni, F., Lung, L. C., Correia, M. & Luiz, A. F. Byzantine Fault-Tolerant State Machine Replication With Twin Virtual Machines. Computers And Communications (Iscc), 2013 Ieee Symposium On, 2013. Ieee, 000398-000403.

Dobre, D., Majuntke, M. & Suri, N. 2006. Corefp: Contention-Resistant Fast Paxos For Wans. Technical Report, Tu Darmstadt, Germany.

Effatparvar, M., Yazdani, N., Effatparvar, M., Dadlani, A. & Khonsari, A. Improved Algorithms For Leader Election In Distributed Systems. Computer Engineering And Technology (Iccet), 2010 2nd International Conference On, 2010. Ieee, V2-6-V2-10.

Fritchie, S. L. Chain Replication In Theory And In Practice. Proceedings Of The 9th Acm Sigplan Workshop On Erlang, 2010. Acm, 33-44.

Hunt, P., Konar, M., Junqueira, F. P. & Reed, B. Zookeeper: Wait-Free Coordination For Internet-Scale Systems. Usenix Annual Technical Conference, 2010. 9.

Ishikawa, K.-I. 2013. Asura: Scalable And Uniform Data Distribution Algorithm For Storage Clusters. *Arxiv Preprint Arxiv:1309.7720*.

Lamport, L. 1998. The Part-Time Parliament. *Acm Transactions On Computer Systems (Tocs),* 16, 133-169.

Lamport, L. 2001. Paxos Made Simple. *Acm Sigact News,* 32**,** 18-25.
Lamport, L. 2006. Fast Paxos. *Distributed Computing,* 19**,** 79-103.
Lampson, B. The Abcd's Of Paxos. Podc, 2001. 13.
Lang, W., Patel, J. M. & Naughton, J. F. 2010. On Energy Management, Load Balancing And Replication. *Acm Sigmod Record,* 38**,** 35-42.
Mao, Y., Junqueira, F. P. & Marzullo, K. Mencius: Building Efficient Replicated State Machines For Wans. Osdi, 2008. 369-384.
Moraru, I., Andersen, D. G. & Kaminsky, M. There Is More Consensus In Egalitarian Parliaments. Proceedings Of The Twenty-Fourth Acm Symposium On Operating Systems Principles, 2013. Acm, 358-372.
Mostafa, A. M. & Youssef, A. E. 2013. A Primary Shift Protocol For Improving Availability In Replication Systems. *International Journal Of Computer Applications,* 72**,** 37-44.
Mostafa, A. M. & Youssef, A. E. 2014a. Improving Resource Utilization, Scalability, And Availability In Replication Systems Using Object Ownership Distribution. *Arabian Journal For Science And Engineering,* 39**,** 8731-8741.
Mostafa, A. M. & Youssef, A. E. 2014b. Prp: A Primary Replacement Protocol Based On Early Discovery Of Battery Power Failure In Manets. *Multimedia Tools And Applications***,** 1-12.
Özsu, M. T. & Valduriez, P. 2011. *Principles Of Distributed Database Systems*, Springer Science & Business Media.
Quamar, A., Kumar, K. A. & Deshpande, A. Sword: Scalable Workload-Aware Data Placement For Transactional Workloads. Proceedings Of The 16th International Conference On Extending Database Technology, 2013. Acm, 430-441.
Rao, S. 2008. Distributed Systems: An Algorithmic Approach. *Ieee Distributed Systems Online,* 11**,** 3.
Schneider, F. B. & Zhou, L. 2005. Implementing Trustworthy Services Using Replicated State Machines. *Security & Privacy, Ieee,* 3**,** 34-43.
Sousa, J. & Bessani, A. From Byzantine Consensus To Bft State Machine Replication: A Latency-Optimal Transformation. Dependable Computing Conference (Edcc), 2012 Ninth European, 2012. Ieee, 37-48.
Tan, Z., Dang, Y., Sun, J., Zhou, W. & Feng, D. 2014. Paxstore: A Distributed Key Value Storage System. *Network And Parallel Computing.* Springer.
Van Renesse, R. & Schneider, F. B. Chain Replication For Supporting High Throughput And Availability. Osdi, 2004. 91-104.
Wei, W., Tian, H., Fengyuan, G. & Li, X. Q. Fast Mencius: Mencius With Low Commit Latency. Infocom, 2013 Proceedings Ieee, 2013. Ieee, 881-889.