ASTES

Advances in Science, Technology and Engineering Systems Journal Vol. 5, No. 3, 46-54 (2020)

www.astesj.com

Special Issue on Multidisciplinary Innovation in Engineering Science & Technology

ASTESJ

ISSN: 2415-6698

Using Leader Election and Blockchain in E-Health

Basem Assiri*

Computer Science Department, Jazan University, 45142, Jazan city, Saudi Arabia

ARTICLE INFO

Article history:
Received: 13 March, 2020
Accepted: 20 April, 2020
Online: 03 May, 2020

Keywords:
Electronic Personal Health Record
Parallelism
Distributed Systems
Leader Election Algorithms
Blockchain Technology

ABSTRACT

The development of electronic healthcare systems requires to adopt modern technologies and architectures. The use of Electronic Personal Health Record (E-PHR) should be supported by efficient storage such as cloud storage which enables more security, availability and accessibility of patients' records. Actually, the increase of availability of E-PHR enhances parallel access, which improves the performance and the throughput of the system. Using distributed systems, users are able to communicate and to share resources to achieve specific goals. Such kind of access needs to have more coordination to maintain parallelism, which can be provided through leader election algorithms. In leader election algorithms, users elect one of them as a leader to coordinate the work and to prevent conflicts. This paper introduces an adoptive leader election algorithm (ALEA) that considers medical and healthcare specifications, since it uses leader election algorithm for E-PHR in the cloud environment. The use of ALEA improves performance by allowing more parallelism and reducing the number of coordinating messages within the system, as well as facilitating the medical specifications such as having a primary doctor or handling emergency situations. Moreover, the paper highlights the strengths and weaknesses of using Blockchain technology in the field of healthcare. In fact, the paper investigates the implementation challenges of ALEA concepts using Blockchain technology.

1. Introduction

Within the last decades, the development of technologies, the Internet and digital applications makes them essential components in some other fields such as education and healthcare. This paper focuses on the development of e-health systems using some supportive algorithms and modern technologies. Actually, this paper is an extension of work originally presented in the 2ndInternationalConference on New Trends in Computing Sciences [1].

The competition among healthcare organizations encourages them to involve modern and advanced technologies to increase stakeholders' satisfaction. These technologies help to improve the provided services. For examples, a patient can schedule an appointment online; doctors (physicians) can access, maintain, and transfer E-PHRs electronically anytime and from anywhere; doctors would be able to diagnose, complete the required treatment and even participate in surgery remotely; prescriptions are sent to the corresponding pharmacy electronically. In addition, these technologies enable costs and managerial efforts to be reduced.

*Basem Assiri, 45142 - Jazan University - Jazan City – Saudi Arabia, +966599933185 & babumussmar@jazanu.edu.sa, bas0911@hotmail.com.

The services costs can be time, physical space, energy and infrastructure. Besides that, it gives deep and clear insight for better administration and decision making. The use of E-PHR as a digital version of PHR allows information to be accessed, updated and transferred in an electronic manner [2], which enhances information accessibility, availability, security, privacy, completeness and consistency. It also helps to avoid the risk of having traditional PHR in case of natural disasters such as Hurricane Katrina [2]-[6].

Moreover, an efficient pattern of storage such as cloud storage is required to support the use of E-PHR in distributed systems. Actually the E-PHRs are stored in servers and can be accessed securely on the Internet [7], [8].

The presence of E-PHR, servers, cloud storage and many connected devices creates a parallel and distributed system. In distributed systems, devices are connected through networks to perform specific tasks. Thus, it helps to improve efficiency and throughput of the process of sharing resources but it also requires more control and coordination. Therefore, leader election algorithms can be used to coordinate the parallel tasks and to preserve the consistency of E-PHRs [9]. Indeed, having parallelism could result in conflicts, especially when a process tries to update an E-PHR, while another one is reading or updating it at the same time. In fact, having a leader allows for an exclusive access (token) to an E-PHR, which keeps it consistent.

In addition, Blockchain technology allows decentralized ledger to be managed within a distributed system. The ledger contains a chain of blocks (records). In Blockchain many nodes share and process distributed copies of the same ledger. In fact, when a node proposes a new block of transactions, the other nodes process it and vote to commit it if its valid or to abort to if it is not. Based of the votes of the majority (consensus), the block commits and every node updates its copy of the ledger or the block is ignored [10].

Nakamoto uses Blockchain to produce the Bitcoin as a first cryptocurrency, in which users execute electronic financial transactions without banks [10]-[12]. After that, Blockchain technology has been involved in many other areas such as judiciary, notary, copyrights, education and healthcare [12].

This paper introduces an adoptive leader election algorithm(ALEA) that is suitable for E-PHRs and healthcare systems. The paper proposes the principles of a primary and a secondary leader as well as having multiple tokens. ALEA allows the number of communication messages to be reduced in case of failures. Moreover, this work highlights the strengths and weaknesses of using Blockchain technology in the field of healthcare to implement the concepts of ALEA.

The rest of our paper is organized as follows: in Section 2, some related work is discussed. Sections 3 and 4, introduce our proposed system model and algorithm. Section 5 discusses many important issues and techniques such as algorithm correctness, consistency, synchronization, file sharing, traffic flow and replication. Finally, Sections 6 and 7 focus on the advantages and disadvantages of using Blockchain technology, while Section 8 provides conclusion.

2. Related Work

There are many techniques to preserve data consistency in cloud storage [13], [14]. Some research proposes strict consistency while the others relax this concept and accept weaker levels of consistency. Coppieters et al. provide an algorithm with strict consistency, where they order all concurrent processes on all replicas. Actually, the concurrent execution of processes should be matched with a correct sequential execution [15]. Zellag and Kemme introduce an efficient relaxed consistency model for cloud storage [16].

Some others use leader election algorithms for consistency, whereby a leader grants exclusive access to some memory objects to prevent conflict [9], [17]. For leader electing, a bully algorithm [9, 18], enables every user to have a unique identifier (Id) and every user sends its Id to all other users. So, they select the node with the maximum Id as a leader. The complexity of this approach reaches $O(n^2)$ messages, which is considered expensive. With a token ring algorithm [19], the users are structured in a linked-circle and every user sends its Id to the next one. After receiving the message, the user compares its own Id with the received one and sends the greater one to the next user. The complexity of this approach costs $O(n)$ messages. Numan et al. provide an algorithm that uses a centralized linked-list queue of all users. The leader is the head of the queue; and when it fails, another user acquires a lock and dequeues the old head. The complexity of this approach is $O(1)$ [20].

At the same time, many countries and institutions have started using E-PHRs. For example, at the beginning of 2014, the American Recovery and Reinvestment Act enforced all healthcare agencies (public and private) to use E-PHR. This facilitates accessibility, utilization and management. However, such a change requires specific technical and infrastructural support to be adopted [21].

In addition, Blockchain technology helps to allow decentralized management of E-PHRs, where there is no need for a third party such as hospitals or healthcare agencies. However, Blockchain has been enhanced with fairness and freedom [22]. Therefore, Blockchain provides many advantages for many areas such as health-care systems. First, it supports the availability, robustness and security of E-PHR. In fact, many projects and companies, such as Data Gateways, Gem Health Network, Deloitte and Guardtime have started using Blockchain to manage their E-PHR [11], [23]. Second, it supports all related financial operations such as funding, donations and insurance payments through cryptocurrencies. Third, Blockchain supports scientific clinical and biomedical research such as in the MedRec Health bank. Indeed, companies and organizations use Blockchain for data sharing and verification, ownership proofs and privacy of patients and organizations. In addition, it could apply the principle of "gain as you contribute" in scientific clinical and biomedical research. For example, while Bitcoin (which is the first cryptocurrency) is earned through solving puzzles, Gridcoin, is another cryptocurrency that is earned based on the contribution to scientific research [10], [11] and [24]. Fourth, Blockchain can be used to manage and process any kind of data such as the records of employees, healthcare facilities, medical instruments, medicines and pharmaceutical supply chains.

Laure and Martha use Blockchain as a control manager to manage the access of E-PHR. Actually, this work focuses on Blockchain's advantages such as scalability, security and privacy. Indeed, the work proposes using a Blockchain system as an access control manager, so it only has indexes of records, while the real records are stored in separate storage (out of the Blockchain). This would help to avoid the negative impact of data redundancy, where every node in the Blockchain has a copy of all indexes instead of having a copy of all records [21].

Kevin et al. suggest the use of Blockchain in healthcare to solve the issues of hardware and software heterogeneity. The work focuses on the quality of sharing E-PHR in an understandable and meaningful manner. So, they use the concept of Fast Healthcare Interoperability Resources with specific Application Programming Interfaces as standard for data formatting and presentation [25].

3. The System Model

Before you ALEA is designed according the concepts of in well-organized bully algorithm for leader election [20]. The well-organized bully leader election algorithm is implemented using a linked-list queue to minimize the cost of leader election to $O(1)$. ALEA is a modified algorithm that is efficiently applicable for medical and healthcare systems.

ALEA creates a linked-list queue *Queue* of size *Z*, where *Z* is the number of nodes in the *Queue*. The *Queue* is stored in shared memory, so it is always visible to all nodes. A node (processor/doctor) is denoted as *Nod* and is represented with a unique identifier (*doctorId*), a pointer that is pointing to the next node ,and a *token* flag. The new inserted node (doctor) can read the E-PHR but it has to get exclusive access to update it. The exclusive access is given by changing the *token* value from *0* to *1*. The queue *head* is the leader (*PL*). To change the leader, dequeue the current *head* node and move the leadership (*PL pointer*) to the next node.

For emergency cases where the *PL* is not accessible, a temporary queue *TQ* is created with a secondary leader called *SL*.

Figure *1* shows *Queue* where the list has three doctors, and *PL* points to the head of the queue. It also shows the emergency block which has another list of *TQ* and *SL*. In reality, *TQ* is not usually there.

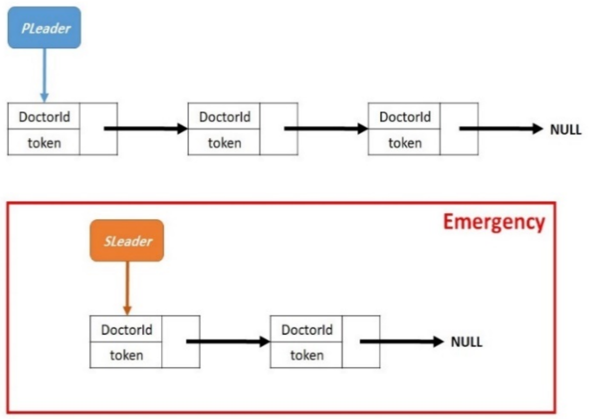

Figure 1: Leader election linked-list *Queue* with a *PL* pointing to the queue head; while the emergency linked-list *TQ* and a *SL* appear in the emergency block.

It is obvious that many doctors can read files in parallel, but conflict arises at the time of the update. For instance, the situation arises when one doctor is reading a file and another updating it. In this case, the data updated last should be visible to the reader. Also, when two doctors are writing to (updating) the same files, the two updating processes conflict with each others. To overcome this situation, it incorporates the idea of *TokenPtr*, which shows who holds the *token* flag to update the files.

4. Proposed Algorithm

- A hospital creates the patient's E-PHR with a primary doctor (leader) *PL*, (see Function 1). Actually, the node (doctor) has three attributes as follows: first, data to have the *doctorId*; second a *pointer* pointing to the next node; and third a *token* flag. When a new doctor is inserted into the *Queue*, the doctor can read the E-PHR directly, but for update permission, the *token* has to be changed to *1*. The *TokenPtr* is another pointer pointing to the node that has the *token* (*token=1*). Finally, it increases the size of *Queue*.
- If *PL* cannot be accessed for any reasons, except in the case of failure, it creates an emergence or temporary queue *TQ* with a secondary leader *SL*, (see Function 2). With the creation of a new node, the size of *TQ* is increased and the same procedures as in Function 1 are used.
- Now Function 3 shows the process of adding a new node to the *Queue*. Considering the medical needs, *PL* can add a new doctor to the team, by creating and enqueuing a new node to the *Queue*. Then it increases the *Queue* size. This is also applicable to *TQ*.
- Sometimes because of medical needs, the leader has to reorder doctors in the *Queue*, so it swaps the nodes as shown in Function 4. Based on the doctor's Id, *TempPtr1* starts searching from the head position, until it finds the first doctor. Then, *TempPtr2* starts searching from the head position, until it finds the second doctor. After that, it swaps the doctors by inserting *doctorId1* in the node of *TempPtr2* and *doctorId2* in the node of *TempPtr1*.
- When *PL* retires from leadership of the E-PHR's team, it follows the procedures for Function 5. If it is the only doctor who handles the E-PHR, which means *Queue* has one node only, then, the retirement is not allowed. Else, it uses *TPtr* to point to the *PL* node; moves the *PL* pointer to the next node; moves the *token* (if it is needed) and finally it dequeues the *TPtr* node and enqueues it again from the other end of the *Queue*. This is also applicable for *SL* and *TQ*.
- In Function 6, a doctor leaves the E-PHR's team (complete clearness). If it is the only doctor who handles the E-PHR, which means *Queue* has one node only, then, the clearness is not allowed. Else, it is dequeued from the *Queue* (as in Function 5), but there is no need to enqueue it again.
- Function 7, explains how to move the *token* among nodes. First, *PL* finds the targeted node according to its *doctorId*, then it gives the *token* (makes *token=1*), or gets it (makes *token=0*). In addition, *PL* allows parallelism by giving the *token* to many nodes simultaneously, which is explained in details later.
- Function 8, shows the case of a doctor requiring the *token*, so it sends an acquiring message to *PL* and waits for some time (*Timeout*). It should wait until the time becomes equal to *T* where *T=currenttime+Timeout*. In fact, it is supposed to receive a reply message (acknowledgment) from *PL*. However, if the timeout finishes without receiving the reply message, then *PL* fails, and it calls *PLFailure()*. Actually many nodes may detect *PL* failure simultaneously, so each node has to copy the *Id* of *PL* (failed leader's *doctorId*) (more details are given in Function 9).
- Function 9, shows the case of leader *PL* failing. Upon the detection of *PL* failure, the detector node calls *PLFailure(Id)*. It also passes *PL*'s *doctorId*. *PLFailure(Id)* moves the *PL* pointer to the next node and

```
1.  ‖ Algorithm 1: ALEA1.
2.  1. Initialization()
3.  //To create the linked-list
```

```
Queue Z=0; //The queue size
Nod=newnode();
Nod→data=doctorId;
Nod→next=NULL;
Nod→token=0;
PL←node;
TokenPtr=PL;
TokenPtr→token=1;
Z++;
Return
```

```
 2. Emergency()
//Adding a new doctor as SL & creating a temporary queue
Z=0; //The queue size
Nod=newnode();
Nod→data=doctorId;
Nod→next=NULL;
Nod→token=0;
SL←node;
TokenPtr1=SL;
TokenPtr1→token=1;
Z++;
Return
```

```
3. AddDoctor()
//Adding a new doctor to Queue
Nod=newnode();
Nod→data=doctorId;
Nod→next=NULL;
Nod→token=0;
Queue←enqueue();
Z++;
Return
```

```

4. SwapNodes(doctorId1, doctorId2)
//Swapping the doctors in Queue
TempPtr1=PL;
TempPtr2=PL;
i=1;
while(i ≤ Z) do
{
   if (TempPtr1→data, doctorId1) then
   {
   TempPtr1=TempPtr1→next;
   i++;
   }
   else
   {
   //First doctor is found, now find the other one
   Break;
  }
}
i=1;
while(i ≤ Z) do
{
   if (TempPtr2→data, doctorId2) then
   TempPtr2=TempPtr2→next;
   i++;
   }
   else
   {
   //Now Second doctor is found, so swap them
   TempPtr1→data=doctorId2;
   TempPtr1→token=0;
   TempPtr→data=doctorId1;
   TempPtr2→token=0;
   Break;
   }
}
Return
```

```
5. Rretirement()
//Retiring from the leadership
if (PL→next=NU LL) then
Return False;
else
{
TPtr=PL;
PL=PL→next;
   if (TokenPtr=TPtr) then
   {
   TokenPtr→token=0;
   TokenPtr=TokenPtr→next;
   TokenPtr→token=1;
   }
TPtr.dequeue();
T Ptr.enqueue();
}
Return
```

```
6. Clearness()
//Clearness
if (PL→next=NU LL) then
return False;
else
{
TPtr=PL;
PL=PL→next;
  if (TokenPtr=TPtr) then
   {
   TokenPtr→token=0;
   TokenPtr=TokenPtr→next;
   TokenPtr→token=1;
   }
TPtr.dequeue();
}
Return
```

dequeues the failed leader. If many nodes detect *PL* failure simultaneously, all of them invoke *PLFailure(Id*), which may cause multiple unnecessary dequeues. Thus, it is mandatory to use a Compare-and-Swap atomic operation (CAS statement), by which only one node changes the leader [26]. Using a CAS statement, one node checks if the *PL* is still in a failure (*PL's doctorId=Id*), and it invokes *Clearness()*. In *Clearness()*, *PL* (failed leader) is dequeued and another leader is elected. Therefore, the other nodes that detected the failure of *PL* also use CAS, but find PL's *doctorId*≠Id, since they find *doctorId* of the new *PL* that is not equal to the value of *Id*, and do nothing.

```
7. Leadership(doctorId1)
//Moving the token among nodes
//Getting the token
TokenPtr→token=0;
//Find the targeted node based on Id
i=1;
while(i ≤ Z) do
{
    if (TokenPtr→data, doctorId1) then
     {
     TokenpPtr=TokenPtr→next;
     i+ +;
     }
     else
     {
      //Give the token
     TokenPtr→token=1;
     Break;
     }
}
Return
```

```
8. Reminder()
//Node sends a message to acquire the token
Sendmsg(PL, "Acquire token");
//Waiting (Timeout)
T=CurTime()+Timeout;
while(receiveack() = false && CurTime() < T) do
wait();
//Timeout finishes and no response (PL fails)
if (receiveack()=false) then
{
Id=PL→data;
PLFailure(Id);
}
Return
```

```
9. PLFailure(Id)
//Leader still in failure
CAS (PL→data, Id, Clearness());
Return
```

Figure 2 is an example of a failed leader. Node 3 and 4 discover that the leader has failed. In Figure 2 (a), the two detector nodes elect a new leader in parallel, so both dequeue and move PL pointer. This causes one unnecessary dequeue. In Figure 2 (b) the two nodes elect a new leader in parallel using CAS (the lock can also be used), so both of them copy the doctorId of the PL in Id (Id=1). Both of them apply CAS such that one node will have successful CAS, and it dequeues the failed leader. However, since a new leader is already there, the doctorId of the new leader does not equal to Id anymore. Obviously, the doctorId is 2, while Id=1. As a result of this, the other node gets a failed CAS, so it does nothing.

5. Analysis

5.1. Correctness

For ALEA correctness, the correctness of concurrent operations, that are made by doctors, are proved by satisfying Linearizability [26]. This requires the concurrent operations to be ordered to match a correct (valid) sequential execution. Indeed, ALEA is considered an event-based model [9], where a doctor performs the operations(a read/update) on the E-PHR, and each operation is represented by two instantaneous events (*begin()* and *end()*). A complete execution is a sequence of operations where there is no pending operation and every operation has the two events. For the correctness and legality of all operations, it is not difficult to argue about the correctness and legality of a sequential execution where all operations are running in one processor and one after another. This helps to prove the memory consistency and to predict the situation of the file, before and after each operation. Since ALEA has two kinds of operations which are update and read, an update operation is legal if it appears instantaneously and all later reads read it, until another update takes place. A read operation is legal if it reads the last written data, and all later reads read the same data until another update takes place. A sequential execution is legal if all its operations are legal. Then, the concurrent execution is correct and the memory is consistent if the order of concurrent execution (including events of all concurrent operations) matches the order of a legal sequential one; this is known as Linearizability [26].

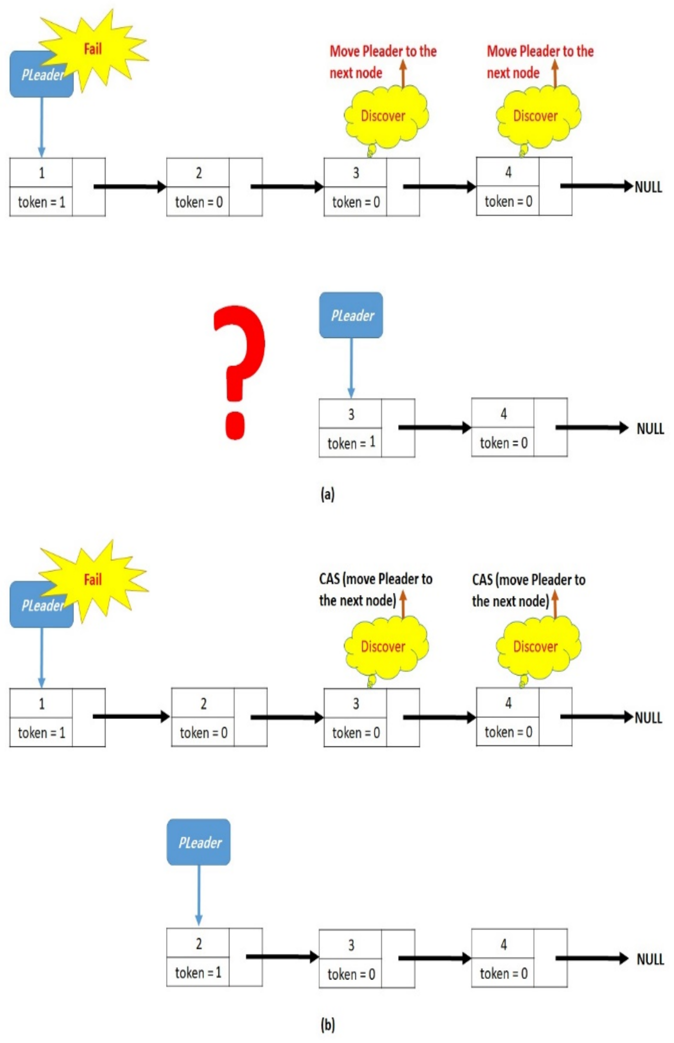

Figure 2: (a): As result of the failure of the leader, two nodes are electing a new leader in parallel, which causes one more unnecessary dequeue: (b) As result of the failure of the leader, two nodes use CAS to elect a new leader in parallel, so one node dequeues the failed leader and the other does nothing

In ALEA, there are two kinds of operations, some operations process the linked-list queue and the other operations process the E-PHR. The first kind is to modify the linked-list queue of doctors. To update the queue, there are operations such as enqueue, dequeue (even electing new leader is conducted through dequeuing) or move the token. In fact, ALEA maintains a queue of linked-list, so it follows the concepts of Michael and Scot [27], that is considered as one of the best lock-free algorithms in the field. Actually the enqueue and dequeue operations are conducted based on Michael and Scot's algorithm [27], which is linearizable. For the operation of moving the token, it is executed only by the leader, so it is serialized. The read operation simply tries to find the current leader which is always able to see the last change. In fact, the linked-queue is stored on a shared memory which makes all updates visible instantaneously. The second kind of operations is to update the E-PHR. Actually, some procedures that prove the correctness of the bully leader election algorithm [9, 18], and the well-organized bully leader election algorithm [20], are applicable in ALEA. To update the E-PHR it has to use the token which gives exclusive access, so it does not conflict with other operations and the operations can be ordered based on the token movement. On the other hand, all read operations are executed concurrently and are ordered with respect to the real-time order. Indeed, ALEA satisfies Linearizability.

5.2. Synchronization

Processes synchronization allows resources to be shared without causing any conflicts (it preserves consistency). Clearly, ALEA uses application level synchronization, but at some points it also relies on operating system primitives such as using locks and CAS atomic operation. Focusing on how to order events, Linearizability follows the real-time order of the concurrent operations (execution). In fact, when there is one thread, events are not interleaved on the same object. However, when there are many threads the synchronization satisfies a happens-before relationship. Therefore, the synchronization of events respects a well-formed clock. Actually, the order depends on the exact time in terms of where and when the operation takes effect. However, having multiple time zones of doctors and patients due to remote access or travel; challenges the use of physical clock. Therefore, a logical clock such as Lamport's logical clock [9, 23], is preferred. Lamport's logical clock is one of the famous approaches that uses the happens-before (|) relationship such that, for any two operations *a* and *b*, it is said *a* happens-before *b* ($a|b$) if and only if the *end()* of operation *a* occurred before the *begin()* of *b*. However, if there is an overlap between operations then the two possibilities are considered ($a|b$) or ($b|a$). Using ALEA, it orders the operations even if they are executed in multiple processors since all of them are executed on a single version of E-PHR. The read operation isordered easily, while the update operation gets exclusive access(token), so it is also ordered based on the token movements.

5.3. Parallel Access of E-PHR

In this part, it is suggested the E-PHR to divided into multiple sections *s*, and doctors are able to access different sections concurrently. Therefore, ALEA has to use multiple tokens, let say *k* tokens, where $k=s$ (a token for each section). Now, the leader gives a suitable token to a doctor according to the needed section. Thus, *TokenPtr* of the original ALEA is replaced with a two-dimensional array of pointers (with *k* rows and 2 columns). Each row of this array shows the doctor's id and the corresponding section.

5.4. Traffic Flow

Processes It is known that the election of a new leader requires a number of messages to passed and that may costn2messages for *n* nodes [17, 18]. On the other hand, having one leader has severe negative impacts on the system in case of leader failure. To handle this situation, decentralized leader election algorithms enable many replicas for each file and more than one leader. To access any replica, voting is conducted and access is allowed according to the majority (consensus). Such type of permission requires approximately *2n* messages [17, 18]. However, with ALEA the number of messages is reduced to *0*, as it dequeues the head node (leader) and the new head will be the new leader (no traffic). In addition, other messages are sent to acquire the token. In ALEA the leader can hold a token or move it to the targeted node as needed with no messages. Rarely, if a doctor needs to get the token, the doctor sends an acquire message (as shown in Function 8), and it receives a reply message from the leader, which causes no traffic.

5.5. Fairness and Starvation (timeout)

This part focuses on fairness of the leader election process and fairness of token movements. For the fairness of leader election process, the leadership appointment has to follow the queue property first-in-first-leader. However, this is relaxed to approximate-first-in-first-leader to handle medical and healthcare requirements. Indeed, in ALEA, the leader is able to swap the doctors in the queue based on the medical needs, which is completely fair from the medical point of view. On the other hand, ALEA allows the leader to move token among nodes as needed, which is also fair from the medical point of view.

5.6. Replication

This part It would never be advisable to use a single centralized copy of the E-PHR since there is no way for recovery in case anything goes wrong. In this regard, the idea of having multiple replicas of the same file is integrated, which is very important for reliability and performance. Firstly, many replicas allow recover of files if the reis an issue with a server, security problem, file corruption or failed operation (read/update). Secondly, the replicas allow improved performance as they can be distributed based on system capabilities, competences, load balancing or geographical distribution. However, having many replicas requires more effort to keep them consistent. For such an issue, it is suggested that the number of replicas be reduced to three copies. The main (permanent) replica is stored in a suitable place bearing in mind the geographical location of the patient and all doctors accessing this replica. The second replica (backup1) should be stored in the same system or in a very close one, so it can be used easily for recovery. The third replica (backup2) must be stored in a different server that is geographically located far away from the main one; so it can be used in case of natural disasters, for example. In order to maintain consistency in replicas, it is suggested to use the eventual consistency criteria where the consistency of file is relaxed [9, 15]. Eventual consistency is very suitable to E-PHRs and cloud storage. Using eventual consistency, read operations do not cause any harm, as the replicas remain consistent, but this scenario will be different in the update operation because the update operation changes the content of the replica. After confirmation of an update, the change should be reflected in all other replicas in an asynchronous manner (over the network). In ALEA, the token can be given to more than one doctor working on different parts of the file, which causes a write-write conflict. Therefore, every update is labeled with two timestamps, which are the timestamp when the update takes effect, and the timestamp of the last update before this one took place. This helps all replicas to order the changes and maintain consistency among all of them. Another way is to use Primary-Based Protocol to conduct remote write/update; in this scheme, the user holds a local copy of a file and after performing the update operation locally, it sends are quest to the server for final approval [9]. This way, the main server takes responsibility for preventing conflicts. To avoid any type of conflict, it updates the other replicas instantly. This protocol (of maintaining the local replica) is an actual support for mobile applications, where the user works on those selected files from different locations, which may affect the connection with the server. Besides this, a huge number of update operations on the system and the processes of maintaining replica consistency result in very high contention on the network. To solve this difficulty, the techniques below is effective:

- Directly, send the modified part of a file to replicas. This can be used in case of a considerable number of update operations.
- Only send a notification to invalidate the other replicas. The invalidation notification has a smaller size compared to the update messages. Indeed the invalidation can be more specific to tell which part of the replica is not valid anymore. Then, the system denies access to that part until it finds a suitable time to update it.
- For some kind of update, send the computation itself, so the others process it locally and make the update themselves.

6. The Use of Blockchain

Blockchain technology is a distributed ledger that is shared and processed by nodes according to consensus [10, 11]. In fact, Blockchain algorithms have three major phases which are proposing a new block, voting on the new block, committing or aborting it based the consensus [10, 11, 22]. The implementation of ALEA concepts using Blockchain technology requires some modifications. The creation of a new queue in Functions 1 and 2, will be based on the consensus of users, rather than hospitals or healthcare agencies. Moreover, adding, removing and swapping doctors (in Functions 3, 4, 5 and 6) will not be executed by the leader; instead, they have to make a proposal, vote and then take a decision according to the majority. The same thing is applicable to token movements in Functions 7 and 8, as well as in case of the failure of the leader in Function 9.

6.1. Transaction's Validation

The Blockchain consists of a number of nodes (processors). Each node can access the patients' E-PHRs and has a copy of the ledger. The ledger contains blocks and each block contains a number of transactions. The transaction contains some operations that are executed on E-PHRs. The operations on the E-PHR are either read or update (the update includes creating, editing and deleting E-PHR). After the execution of transactions, the miners validate the transactions by validating the output of the operations of those transactions. Indeed, the miners also consider the concurrency of transaction and can use some standard property for such issues such as Opacity [28]. According to this correctness property, a valid transaction commits and is placed on the blocks, while an invalid transaction aborts [29]. Note that more than one miner may validate the same transaction and every one places it in a different block. or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

6.2. Block's Validation

When the block is full (based on the size of the block), it is proposed to the validators (who are the team of doctors), so they validate the correctness of the block, considering the concurrent blocks as well. In fact, the content of the block is hashed to secure it. The hashing can be produced in many ways such as Proof-of-Work (PoW) or Proof-of-Stack (PoS) [30]. In addition, validators validate the signature of the proposer node to verify that the block has been created by a legitimate node. Actually, the validators validate the identity of the proposer through its digital signature, the PoW and content of the block. According to this, validators vote to commit the block or abort it.

6.3. Consensus

Blockchain uses consensus to consider the validation processes of all validators. If the majority votes to commit the block, then it is added to the ledger, otherwise it is ignored (and then there is no need for a leader like in ALEA). The majority of votes means more than 50% is needed, and in some systems they increase it to 67% [31, 22]. In addition, the order of the committed blocks in the ledger is decided based on who gets PoW first. Some work uses a unique timestamp for each block, so the one with a smaller timestamp is added first [22]. The blocks are chained in the ledger using the hash number, as every block has the hash of the previous one; thereforethe ledger is unchangeable.

7. Discussion on the Use of Blockchain

The implementation of ALEA's concept using Blockchain has some strengths and weaknesses. Thus, this section discusses the issues of security, privacy, immutability, decentralization, robustness, availability, ownership, computational costs and system traffic flow [10, 11, 24, 21].

7.1. Security, Privacy and Immutability

Using Blockchain technology the identities of doctors are hidden. The E-PHRs and the operations on them are hashed and encrypted. Such security and privacy are positive points that encourage everyone to use Blockchain. On the other hand, users will be untraceable and that is an issue for healthcare systems, especially in terms of tracking suspicious and illegal behaviors.

In addition, using Blockchain guarantees immutability, since the ledger (that has the records of all operations on E-PHRs) is unchangeable. Obviously, this is a positive point; however, such a system does not allow to rollback.

7.2. Decentralization

To avoid the presence and the control of the centralized third party such as a hospital, decentralization enables queue to be created or emergency to be handled using consensus. The negative impact of consensus is the voting delays; as well as having non-trusted nodes or misleading votes.

7.3. Robustness

To The distribution of Blockchain technology helps to prevent a single point of failure, especially in the case of leader failure. Instead, the work will continue as long as the majority of nodes is still working.

7.4. Availability

Blockchain provides high availability, since all nodes have replicas of the files. The large redundancy of replicas requires more space (memory), communication and processing (to preserve the consistency of the replicas). To avoid such an issue, it is proposed to use Blockchain as an access control manager that shares copies of indexes rather than the real records, and the records are stored in separate centralized storage [21], as shown in Figure 3.

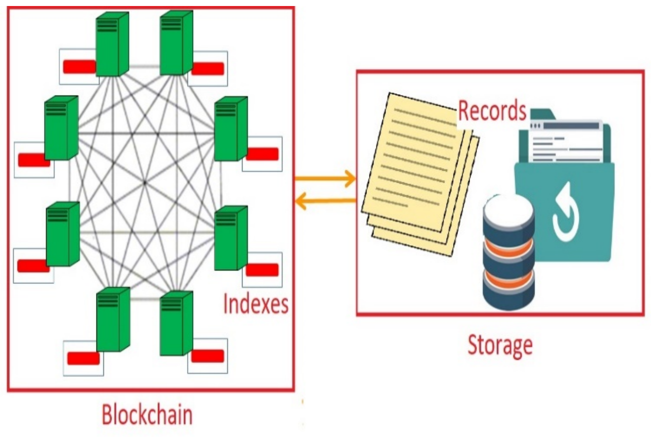

Figure 3: Blockchain with indexes and separated centralized storage of records.

7.5. Performance Cost

The use of Blockchain technology results in huge increases in computational and communication costs. In ALEA, the computation is executed by one node, while in Blockchain all validators execute the computation to confirm its correctness. In addition, when a node proposes a new block, it broadcasts to all validators, so if there are n validators, it broadcasts n messages. After the validators execute the computation, they send votes to all nodes, let's say m nodes, which costs m^2 messages. Actually, the set of the validators is a subset of all nodes in the Blockchain. Then, every node calculates the majority and broadcasts messages of the decision (commit/abort), which also costs m^2 messages. Finally, in case of commit, all nodes update their own copy of the ledger, rather than a limited number of replications. Thus, the use of Blockchain negatively affects the speed of the system and its traffic flow.

7.6. Fault Tolerance

The consensus is a fault tolerant correctness property, where some validators do not confirm the validity of the block. In medical cases, it is not suitable to ignore the votes of 49% or even 33% of doctors because they are not the majority. This means, the Blockchain may still allow for some errors, which is very critical for healthcare specifications.

8. Conclusion

This paper proposes an adaptive algorithm, for E-PHRs in a cloud environment, so it can be easily used with minimal infrastructure. ALEA enhances parallelism using an alternative and modified leader election technique that suits medical and healthcare systems. This work investigates the performance and the characteristics of the ALEA in comparison to Blockchain technology, which shows that the use of Blockchain technology may result in some negative impacts

Conflict of Interest

The authors declare no conflict of interest.

References

[1] Basem Assiri. Leader election and blockchain algorithm in cloud environment for e-health. In2019 2nd International Conference on new Trends in Computing Sciences (ICTCS), pages 1–6. IEEE, 2019.

[2] Paul C Tang, Joan S Ash, David W Bates, J Marc Overhage, and Daniel Z Sands. Personal health records: definitions, benefits, and strategies for over-coming barriers to adoption. Journal of the American Medical Informatics Association, 13(2):121–126, 2006.

[3] Selena Davis, A Roudsari, and Karen L Courtney. Designing personal health record technology for shared decision making. Studies in health technology and informatics, 234:75–80, 2017.

[4] Janet Woollen, Jennifer Prey, Lauren Wilcox, Alexander Sackeim, Susan Restaino, Syed T Raza, Suzanne Bakken, Steven Feiner, George Hripcsak, and David Vawdrey. Patient experiences using an inpatient personal health record. Applied clinical informatics, 7(02):446–460, 2016.

[5] Arloc Sherman and Isaac Shapiro. Essential facts about the victims of hurricanekatrina.Center on Budget and Policy Priorities, 1:16, 2005.

[6] Shayne Sebold Taylor and Jesse M Ehrenfeld. Electronic health records and preparedness: lessons from hurricanes katrina and harvey. Journal of medical systems, 41(11):173, 2017.

[7] Mu-Hsing Kuo. Opportunities and challenges of cloud computing to improve health care services. Journal of medical Internet research, 13(3):e67, 2011.

[8] Hoang T Dinh, Chonho Lee, Dusit Niyato, and Ping Wang. A survey of mobile cloud computing: architecture, applications, and approaches. Wireless communications and mobile computing, 13(18):1587–1611, 2013.

[9] Andrew S Tanenbaum and Maarten Van Steen. Distributed systems: principles and paradigms. Prentice-Hall, 2007.

[10] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot, 2019.

[11] Tsung-Ting Kuo, Hyeon-Eui Kim, and Lucila Ohno-Machado. Blockchain distributed ledger technologies for biomedical and health care applications. Journal of the American Medical Informatics Association, 24(6):1211–1220, 2017.

[12] Michael Crosby, Pradan Pattanayak, Sanjeev Verma, Vignesh Kalyanaraman,et al. Blockchain technology: Beyond bitcoin. Applied Innovation, 2(6-10):71, 2016.

[13] Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. The rise of "big data" on cloud computing: Review and open research issues. Information systems, 47:98–115, 2015.

[14] Divyakant Agrawal, Sudipto Das, and Amr El Abbadi. Big data and cloud computing: current state and future opportunities. In Proceedings of the 14th International Conference on Extending Database Technology, pages 530–533, 2011.

[15] Tim Coppieters, Wolfgang De Meuter, and Sebastian Burckhardt. Serializable eventual consistency: consistency through object method replay. In Proceedings of the 2nd Workshop on the Principles and Practice of Consistency for Distributed Data, pages 1–3, 2016.

[16] Kamal Zellag and Bettina Kemme. How consistent is your cloud application? In Proceedings of the Third ACM Symposium on Cloud Computing, pages 1–14, 2012.

[17] Gerard Tel. Introduction to distributed algorithms. Cambridge university press, 2000.

[18] George Coulouris, Jean Dollimore, and Tim Kindberg. Distributed systems: Concepts and design edition 3.System, 2(11):15.

[19] P Beaulah Soundarabai, J Thriveni, HC Manjunatha, KR Venugopal, and LM Patnaik. Message efficient ring leader election in distributed systems. In Computer Networks & Communications (NetCom), pages 835–843. Springer, 2013.

[20] Muhammad Numan, Fazli Subhan, Wazir Zada Khan, Basem Assiri, and Nasrullah Armi. Well-organized bully leader election algorithm for distributed system. In 2018 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET), pages 5–10. IEEE, 2018.

[21] Laure A Linn and Martha B Koo. Blockchain for health data and its potential use in health it and health care related research. In ONC/NIST Use of Blockchain for Healthcare and Research Workshop. Gaithersburg, Maryland, United States: ONC/NIST, pages 1–10, 2016.

[22] Basem Assiri and Wazir Zada Khan. Enhanced and lock-free tendermint blockchain protocol. In 2019 IEEE International Conference on Smart Internet of Things (SmartIoT), pages 220–226. IEEE, 2019.

[23] Marc Pilkington. Blockchain technology: principles and applications. In Research handbook on digital transformations. Edward Elgar Publishing, 2016.

[24] Usman W Chohan. Environmentalism in cryptoanarchism: Gridcoin case study. Available at SSRN 3131232, 2018.

[25] Kevin Peterson, Rammohan Deeduvanu, Pradip Kanjamala, and Kelly Boles. A blockchain-based approach to health information exchange networks. InProc. NIST Workshop Blockchain Healthcare, volume 1, pages 1–10, 2016.

[26] Maurice Herlihy and Nir Shavit.The art of multiprocessor programming. Morgan Kaufmann, 2011.

[27] Maged M Michael and Michael L Scott. Simple, fast, and practical non-blocking and blocking concurrent queue algorithms. In Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing, pages267–275, 1996.

[28] Rachid Guerraoui and Michal Kapalka. On the correctness of transactional memory. In Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming, pages 175–184, 2008.

[29] Nir Shavit and Dan Touitou. Software transactional memory. Distributed Computing, 10(2):99–116, 1997.

[30] Iuon-Chang Lin and Tzu-Chun Liao. A survey of blockchain security issues and challenges. IJ Network Security, 19(5):653–659, 2017.

[31] Basem Assiri and Costas Busch. Approximate consistency in transactional memory. International Journal of Networking and Computing, 8(1):93–123, 2018.