مجلة جامعة بني وليد للعلوم الإنسانية والتطبيقية
Bani Waleed University Journal of Humanities and Applied Sciences
تصدر عن– جامعة بني وليد – ليبيا
Website:https://jhas-bwu.com/index.php/bwjhas/index
المجلد التاسع ـ العدد الخامس ـ 2024 ـ الصفحات (538 -555)
ISSN3005-3900

انتخاب لوغريثم المستأسد في الأنظمة الحاسوبية الموزعة

عامر صالح مصباح قدوعة *
قسم برمجيات الحاسوب، كلية تقنية الحاسوب طرابلس، طرابلس، ليبيا
amer_s_gadowa@cctt.edu.ly

Bully Algorithm for Election in Distributed Systems

Amer Saleh Musbah Gadowa[*]
Department of Computer Programming, College of Computer Technology Tripoli, Tripoli. Libya

تاريخ الاستلام: 2024-11-01 تاريخ القبول: 2024-11-28 تاريخ النشر : 2024-12-15

الملخص:
من أهم المصاعب التي تواجه الأنظمة الرقمية المتفرعة في تأدية العمل المناط بها هو تأخر أو فشل في وظيفة المنسق الرئيسي لتلك الأنظمة، مع ذلك يعتبر انتخاب منسق موثوق به هي العملية الضرورية لمعالجة الفشل الجزئي وكيفية الاستمرار بدون اي عوائق في إنجاز المهمة المستهدفة مع عدم القصور على أداء المهام الرئيسية.
من هذا المنطلق، يهدف هذا البحث الى دراسة وحل مشكلة تداعيات المنسق الرئيسي بعمل انتخاب للرسائل بتمريرها وتفاعلها بشكل كامل ما بين الأجهزة الحاسوبية الموجودة داخل تلك الأنظمة الرقمية الموزعة. استخدمت هذه الدراسة خوارزمية المستأسد (Bully Algorithm) مع أحدث التقنيات البرمجية (PVM) والتي تعني الألة الافتراضية المتوازية التي توفر بيئة تزامنية لحل هذه المشكلة. إن الغاية من ذلك هو خلق و البداية في عملية جديدة لاختيار منسق جديد و الذي يتكون من أكبر عدد من الأجهزة الحاسوبية. مع ذلك تعتبر (PVM) مصممة بشكل رئيسي لهذه المهام الا وهي استخدامها داخل بيئة متزامنة وغير متجانسة، ولا سيما إمكانية توفير صورة شاملة و متكاملة افتراضيا.
في القسم الخاص بالبناء و التنفيذ يُظهر المحاكاة الكاملة حالة فشل الاتصال و كذلك خطوات اختبار البرنامج المقترح، علاوةً على ذلك تم وضع وصف واضح وصريح لتفاعل الرسائل المنجزة لجميع الأجهزة الحاسوبية المتفاعلة في البيئة التشغيلية. نتيجةً لذلك تم تنفيذ وإنجاز الخوارزمية المقترحة والتي أسفرت على مجموعة من النتائج المهمة و المتنوعة، بما في ذلك بيئة لبنية شبكية متكاملة مع احتمالية حدوث عطب أثناء العملية التشغيلية للانتخاب، إضافةً لوغارثمية الانتخاب تمت بنجاح بانتخاب واختيار منسق جديد والذي يحتوي على أكبر رقم من الحواسيب التشغيلية التي تعتبر أحد الأنظمة الطرفية للأنظمة المتفرعة. لقد تم توضيح و توثيق جميع الشفرات المصدرية للبرنامج بالإضافة الى عمليات محاكاة (PVM) لتبادل رسائل الانتخاب ما بين الأجهزة المتفاعلة.
لقد تم كتابة الشفرة المصدرية لهذه الخوارزمية بواسطة لغة البرمجة C والتي تدعم الحواسيب التزامنية وايضاً البيئة التشغيلية لنظام UNIX من خلال الأنظمة العنقودية.

الكلمات المفتاحية: لوغارثميات الانتخاب، لوغارثم المستأسد، الأنظمة الموزعة، الألة الافتراضية المتوازية، الحواسيب التزامنية، البيئات التشغيلية الغير متجانسة

Abstract:
One of the most indispensable circumstances that distributed systems encounter is an inconvenience and delay in the leader process, however, electing an authoritative leader is the imperative procedure for recovering from partial failure and continuing to perform the goal without any repercussions for main duties until a proper redress has been accomplished. Therefore, this paper aims to investigate and resolve

the problem of the coordinator's failure with full interaction of message passing. However, for these spectacular distinguishing features of fault tolerance; the study utilized the bully algorithm with modern techniques of the PVM (Parallel Virtual Machine) software to think through this issue. The purpose of this, is choosing a new process with the highest number of coordinators, as PVM tool is mainly designed to be employed by a concurrent and heterogeneous environment, particularly, the possibility of providing a virtual view.
The implementation section shows the simulation of communication failure and the program testing steps. A clear description of the interaction of executed messages for all nodes is given. The proposed algorithm has been achieved and yielded varied outcomes, including full topology architecture with free crush during the election operations; moreover, the election algorithm has successfully elected the leader, the node which containing the highest figure. The source code program documentation was illustrated; as well as the entire PVM simulations of casting communications. The algorithm's ambiance has been written in the C language, which supports concurrent computers and UNIX environment on Clusters Systems.
Keywords: Election Algorithms, Bully Algorithm, Distributed Systems, Parallel Virtual Machine, Concurrent Computers, Heterogeneous Environment **.**

Introduction

A distributed system is multiple autonomous computers linked together by a network that has the capability of sharing resources of system hardware, software, and data, and also cooperating on their activities. Indeed, the dramatic and exasperating brain-teasers in paralleled computing, is a boss fiasco. The current quandary could be soluble of using the leader algorithm. This algorithm moves the interaction packages from an incipient stage, where whole interacted processes are in the similar tasking stage, to a renewed phase in which one of performing nodes is a coordinator, and the others are not. For example, for this critical issue, the leader election is used vastly in centralized computing for resolving single-point downfall problems, in this server the leader elections performed during there is misleading in meeting its objectives and there is a requisition to impart the command to another station. Furthermore, the coordinator election is assigned in a token ring too.
However, it has been the subject of miscellaneous research, since it is a fundamental difficulty that led to the devising of an algorithm by Hector Molina (Hector, 1982), which was based on the bullying election (the boy with the super power takeovers). It is vital for a collection of nodes that perform a role to continue functioning when a coordinator crashes. As a consequence, a new node leader is in demand to take the place of the previous coordinator to manage and perform the operations that are assigned to the system. In other words, the purpose of electing one node as a leader (node with the biggest id), where all nodes agree with it, is to maintain the continuity of the system's duties. However, the bully procedure has been deemed to be the most classic and sufficient selection algorithm which is simple to use for determining the node with the biggest number. For this purpose, it was adapted to fulfill and implement the core code of this research. This algorithm is applicable when each process of the member group knows each other's addresses and (*tIDs*), every process has a unique identity number, thus it commences the algorithm of the election, and which aim will be terminated by selecting one process acting as a leader with the highest priority number that all nodes agreed with. The algorithm sessions has the obligation to economize the message passing, especially for running time expense. Consequently, the obstacle occurs at leader failure is the case which this study has to cover. For this purpose, the circumstance has paramount aspects, results in impact intensification by diversifying as well as designing a mechanism using unprecedented techniques of electing algorithms. The aspect of bullying algorithm was being considered to be bully algorithm in the PVM tool. Therefore, the targets of this paper are to establish and investigate these comprehensive techniques using a concurrent execution environment that is divided into several PVM programs, which allow master and slave to interact with each other for the operation to elect one node as a leader during the master's failure. The Clusters Systems at Oxford's School of Technology were used for fulfilling the project mission.
To sum up, all needed materials have been provided for reaching the solution to the problem area that was investigated. To simulate communication messages between the driver and other workers that have been implemented, the PVM library was included with a set of functions that can be sent and received during the operation's communication. To consolidate and let these implemented code programs work and

function, they needed some arrangement in messages' passing synchronization and marvelouslly represented the final algorithms for the hypothesized problem. In addition, two main core programs have been produced in this paper, the first one is the sender procedure called (driver operation), and the other one is the receiver procedure, which is called (worker operation).

1. Related Studies

This section is decisively profitable in the methods of analysis, investigating a variety of leader election algorithms and understanding the agreement in distributed systems, as well as giving an undeniable notion for designing the crux algorithms. The area of leader election subject has been covered abundantly by a lot of research, with different paradigms, for instance (Andrea, 2001; Basu, 2011; Biswas et. al., 2021; Chen, et. al., 2023; Daymude, et. al., 2017; Kutten et. al., 2015; Marc, 2019; Rahman, 2009; Refai, 2010; Scott, 1999; Seema, et. al., 2014; Shital, et. al., 2023; Singh, 1992; Supase, et. al., 2021; Wei, 2005).

1.1 Distributed Algorithm

The algorithm of distributed system is a procedure mechanism, which has appointed for the attainment of interconnected processes to carry out some duties. The indispensable attention is how the processes communicate and exchange information. Similarly, the interactions between inter-processes communications in case of any stumbling block. In a diversity of applications, distributed computing systems have been used in and covered a very wide range of areas, for instance, education institutions, and scientific computing resources (Tanenbaum, et al., 2007). Many difficulties can be solved by distributed algorithms, such as coordinator election, mutual exclusion, propagation of spanning trees, consensus, and distributed searching. However, distributed algorithms can act concurrently, that is to say; independent processors can execute parts of algorithms concurrently, in addition to the challenges of coordinating synchronized jobs for each part separately, and unreliable communications would be avoidable, (Coulouris, et al., 2005). In this research, the leader election is the drawback, which was focused on and shed light on, and then Garcia’s Bully algorithm with the PVM tool is employed to investigate this area.

1.2 Election Algorithm Theory

Increasingly, distributed systems have proliferated everywhere, and tremendously competitive revolutions in information science that sparked off great prominence innovations. Indeed, the principal objective of distributed system is to include the competence of creating a connection via detachable resources remotely, for independent to hold some activities (Kordafshari, et. al., 2005). In this condition, the fabulous advantage of participation in particular place is diverse; the pronounced manner is financial, this deduction means cutoff the expenses for firms, for instance centralizing sites, additionally the cooperation is close at hand of exchanging exchange of information. Hence, the attributes spread over autonomous computers. With relation to that, the crucial emphasis point’s duty is to concentrate on the most splendid election algorithm techniques that have been implemented and how to solve the leader election problem (Supase, et. al., 2020).

1.2.1 Leader Election Algorithm Description

It is crucially valuable in some distributed systems algorithms to designate one of the nodes to act as a leader. That means the node has a unique role in carrying out extra responsibilities on some duties. The election of the leader is an intrinsic subject to give attention to and discusses this paramount problem, as the leader election applied in tremendous scientific fields for instance Berkeley algorithm and communication network. An example of using coordinator election algorithms is in client-server (centralized system) to solve the problem of single point failure, the election is used when the failure occurs in the server and there is a need for transferring the command to another station (Refai, n.d). Nevertheless, under no circumstances it is unattainable of a related system opts one node as a leader if the interconnection system has no distinction features. For that reason, their requirement is preferably specified a node to be unrepeatable. Pursuing this further, managing and choosing the exact node with the greatest identifier number and assigned as the bully guy. The communication interaction message in the system must be has the capacity and dealing with timeout interval during the occurring of downfall, (Hector, 1982). Coordinator communication algorithm is the technique distributed over all nodes to look out a method of dealing with situation obstacles, employing of breaking the symmetry of the distributed systems, the algorithm starts as a simple case when one process detects leader breakdown or as the worst when the failure is detected by all processes. Nevertheless, the election terminates when a newly elected node is recognized by all processes (Refai, (n.d); Tanenbaum, 2007). Ever since, substantial research has been revealed and devised, even though prosperous network paradigms of distributed systems like ring, bus, mesh, and ad hoc mobile network. Next, there are some diversified examples of leader election algorithms related to different paradigms:

- Wire interaction communication, to illustrate the ring and bully algorithms.
- Wireless communication, this can be seen in banking transaction services.
- Tremendous broadband system.

Furthermore, the study of (Shital, et. al., 2023) is a magnificent research that has given a colossal scope of nearly all existing coordinator election algorithms, in addition, the paper has given comparison explanations between their properties and pointed out the difference in such patterns (topology, fault tolerated, reliability, communication cost, and system type).

1.2.2 Bully Algorithm

For resolving this circumstance, this duty was presuming a procedure that relies on the Bully algorithm. Garcia's paper (Hector, 1982), whereby this is the paper of phenomenal reputation, which is rivaling the coordination election in all probability of distributed system, where it discussed and indicated an algorithm that rectifies and make the most of it tolerable against downfall, furthermore, ensure and assumption that all nodes had been obtained a distinguish value. In this point, a message of the utmost imperative has the smallest time period ***t***, no longer than a demand of tasking. In addition, ascertain inquired node ***n*** is animate. Even though, this performance of duty pattern intended to nominate a unique node as a boss for guiding the others. It follows that the initiation of the following steps if the connection is cut off:

1- Broadcast a message to all nearby nodes by a specific node ***n*** involving with the colossal value, but not including its figure; beside that ***n*** look for anticipation to receive acknowledgement of replying answer.
   - Crucially, the *time* ***t*** makes a successful pronouncing node ***n*** as a winner, if replying message package does not come along within a certain *time* ***t***.
   - If there is no replying from any other nodes, the holding-operation node ***n*** waits for tiny supplement *time* ***t*** before the nomination package has been announced.

2- Then the nomination electing package arrives in for the node *n*, hence the election message will be commencing the election procedure from the scratch, if there is no task performing before.

3- Finally, the leader-casting-message *is* received by a node n and registers the value included, to be the boss.

Election task procedure sets off when there is a deficiency at the leader node. However, if antecedent node which was function turns back again, it handles its duty as the leader over network communication. In (Figure 1) defines the operation of electing a leader *P2* when dismissed.

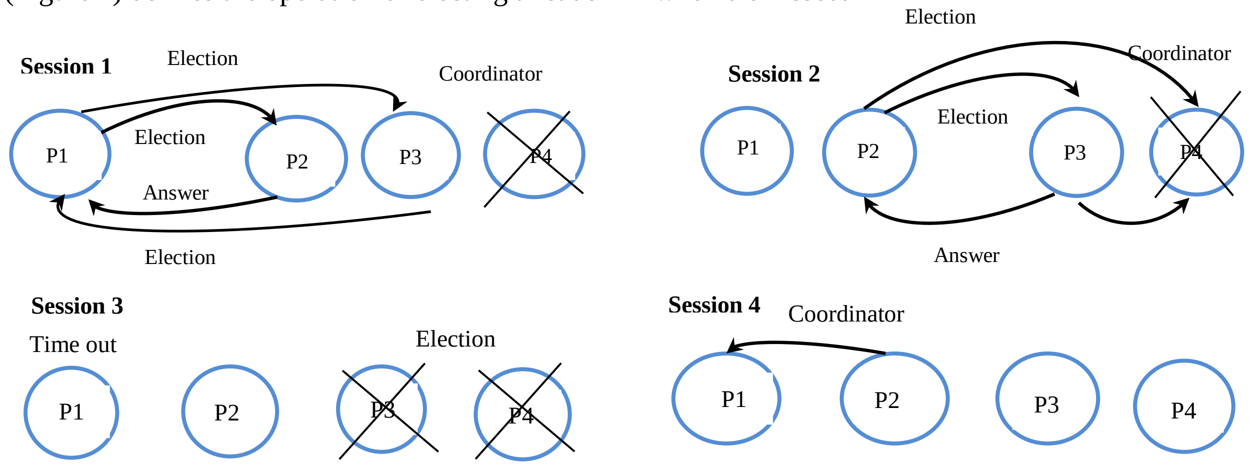

Figure (1) Bully Algorithm (from: Tanenbaum, 2007).

In above simulation represents four nodes interaction into communication and electing *begins* when *p1* detects the fall down. Herein, another theory (Seema, et. al., 2014) paper illustrates an election technique that enhances the Bully algorithm and allows announcing a new coordinator before the current leader collapses, as well as clarifies some of the disadvantages encountered.

1.2.3 Ring Algorithm

This algorithm, also known as Chang and Roberts's technique, in Chang and Roberts's paperwork (Chang, 1979) represents a set of processes that take an organised circular paradigm where its magnitude ***n*** is unfamiliar, and without a central controller. This election algorithm is harmonious for a set of nodes interdependent the utilizing of a ring paradigm with message clockwise movement, and it works for finding either the highest or the lowest numbered process, in which the average O(n log n) messages interaction. The algorithm assumes that each process knows its own number, as well as, a message initializing is generated with its own number that passes it to the left. When the message arrives in with acknowledgement to any node, that node kicks off to draw an analogy the value into with its own. If the node's number is greater, or corresponds to the message's value, then the process disposals of interaction-message package and it will be the highest-numbered process in the system. During the start-up condition, at least one process initiates a message. Since then, each process initiating a message declares itself; any message that arrives at an unmarked process will cause the process to proclaim itself and generate a message. This step is to ensure that all processes are involved in the operation. The election algorithm will succeed when it finds the highest number. The time requisite is O(n) when all processes are started, where (n) is the number of the nodes. However, the message will take one cycle if the node with the vast number is started first, and the time O(n) will be needed. There is only one process that initiates the election if it is furthest away from the longest, and the required time is O(n-1) for getting the largest process. The time, which is needed for the largest to be elected is O(n), therefore, the time required will be O(2n-1). The best case of message passes is that all processes are regulated in increasing sequence clockwise so that each message is only capable of going once. There are (n-1) messages and message (n) needs (n) passes. Eventually, the aggregate message node number is (n+n-1 = 2n-1). On the contrary, the unpleasant instance will be, that nodes are organized clockwise in diminishing sequence so that message (i) passes I time. Since then, the entire interaction-message number will be (i = n (n+1)/2).

1.2.4 Peterson Algorithm

The theory of Peterson's algorithm (Peterson, 1982) proposed a coordination procedure pattern in one direction around circle for all neighbours. In this algorithm, a certain operation paradigm it takes over as the master by participation processes in the ring, that is carrying a specific task. It is ensure and match the temporal identifiers. The algorithm works as supposed the messages are sent only clockwise along the ring as follows:

- All processes are in active situations, each process temporal identifier (*tID*) is the identifier of itself. Initially, each process sends its (*tID*) clockwise for two steps, so that each process will receive the (*tIDs*) of its predecessor, then next to (its next to last predecessor).
- Every process P makes a decision related to its current (*tID*) and other (*tIDs*) that have been received.
- Each and all efficient message-interaction package dispatches its (*tID*) along to the following two lively predecessors around the ring.
- It will repeat the previous steps, and in that time the left process numbers become less. That is to say, the only active process declares itself as a leader.

1.2.5 Franklin Leader Election

Leader election (Wm., 1982), which was based on a ring, presents a proficient state mechanism for seeking the greatest component via a rotary nodes schedule, where the interaction package have the capability to be passed in either direction. However, the algorithm begins with defining an inactive node as one known which means it is not the largest; the other nodes are active. An active node with two neighbours is the active node close by it and in direct link-up throughout both directions the ring sides. The poor quality plight of a ring procedure that has two involved nodes, each becomes the others' two

adjoining nodes; similarly, it will become both of its next doors process, if there is only one active node. Comparatively, the interconnection packages that are took apart in the communication are transmitted in the same direction throughout the circle, for the most part each process has the acceptability for proceeding the procedure of the election at first glance, since interaction participation will be dismissed for whole processes. For this purpose, if the leader notified by any nodes that it does not have sufficient leading part, it is establishing and broadcast the package with its value-number all over the communication channel, accordingly, when the message-package is joined up it examines the value embodied with its own number and sent it towards other nodes when the value included is bigger than the recent node number, if not, it will made the replacement and exchange the value by its identifier, hence again forward the package message to the next process.
In the same way, when the coming-up value reaches itself once again, admittedly, the package message figure has to be the master, it deploys interconnection elected message via the ring by announcing itself as the winner. The message total number passed requires (2N) for each message, without discretion about the number of currently active processes. Correspondingly, this source (Coulouris, et al., 2005), precisely Chapter 12 contains flourishing data and figures. The drawing (Figure 2) represents what have been discussed earlier above and illustrates bellow the simulation of message interaction when system tasking collapsed.

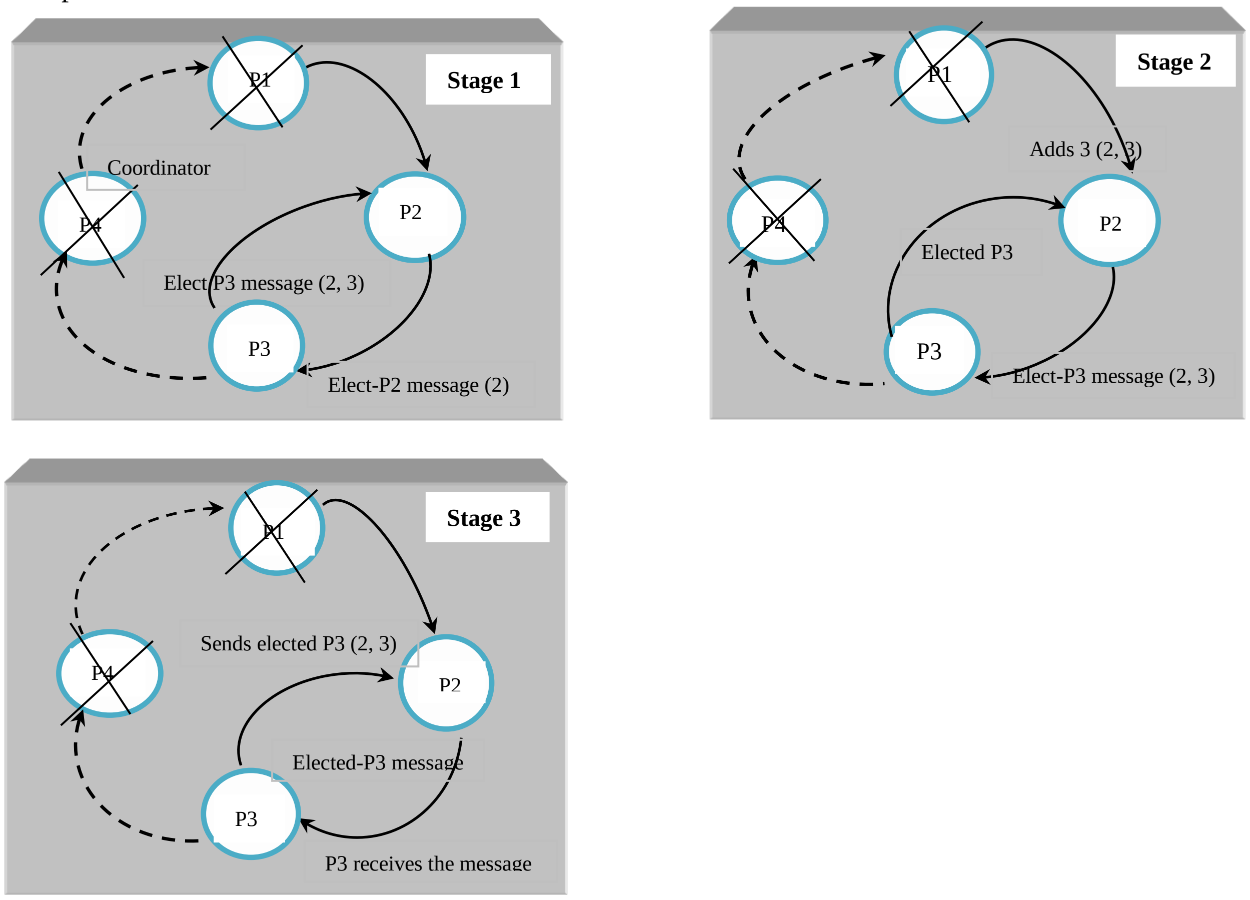

Figure (2) Ring Algorithm

1.3 Distributed Database Systems

This is an example (Riccardo, 1985) of application using the leader election. This system necessitates the dovetailing of splendid binary aspect, the reintegrating cross-over the components of system, as well as, circulating throughout the network, this characteristic make association of data available, where it is possibly to fix up a colossal organization predicament and let them not to be perplexed plight. The distribution of data over as a specific network is worthy to get in vastly within fruitful shared data, managed by one node and hence called the leader. As a case in point, the election algorithm of TEMPO functioning and performing on Berkeley UNIX 4.3BSD manifests the advantage of opting one node as a leader, where it has the capability of performing message-package interaction with two perfect features, the boss and the slave.

After the crash of the machine on which the original master is running, a new master is chosen by an election algorithm from among the slaves. In each slave, the election time is reset periodically when the master is working. However, the expiry timer of the slave first becomes a candidate for the new master if the original master disappears. TEMPO job is defined as measuring the interaction of the local time of running nodes and the other participation processes. The purposing time is computed by the master as the standard time supplied by non-faulty local-time-machine, and it transmits the redressing that have to be carried on the local-time of its process to each slave time daemon. The inconvenience delays impossibly intervention with the concurrence, while the rectifying is seemed as a local-time delay rather than absolute time; the process will turn back and link-up the system when it commences a slave-time procedure, which will demand the master for a redress-time and reconstructs the clock of the system beforehand the initiating of any user pursuit. However, to provide the continuity of TEMPO, service, and reliability, it is obligatory to form an election paradigm that has the ability for selecting a new leader, which machine should run when the current leader crashes. The election algorithms have a key role in distributed databases, as it able to play the main duty and perform some tasks as the following:

- It allows the time daemon, which means the candidate of the master has the opportunity to assemble information about the topology of the system.
- Mask the failures of communication, for instance, loss, delay, and message duplication.
- Has the potency to tolerate and withstand the failures of machines, which occur during the election.
- Network partitions dealing, with and handling abnormal cases.

1.4 Search Engines

Web search engines like Yahoo, Google, Alltheweb, Amazon, and Bing, (The Spider's Apprentice, n.d.; Marc, 2019; Yann, et al, 2003), are powered by a greater autonomy software for tasking called spiders that are designed for allowing the searching and survey for specific words throughout the web. Then, all data which is retrieved from the web pages are aggregated to the navigation engine index. Therefore, when the user enters a quest at a search engine, the contributed data will be examined against the engine’s index via web pages. Thus, the results are turned back to the user as ranking order with the optimum result at the topmost of the page. The simple and easy searching method can be accomplished by using the Boolean, or election algorithm sorts, where the engine sets off and tries to find the documentation related to the words included in the query. The search engine results are based on the algorithm that is used, searching and looking for text mining leads to retrieving all documents including mining text words. The evolutionary algorithm is used for optimizing application extraction where there are some attractive optimization tools in a framework of information retrieval. Search engine ranking algorithms are still secrecy guarded; the search engine companies need to make it difficult to manipulate their ranking website owners, as well as the company wanting to protect their strategies and methods from their competitors.

The prodigious most renowned search engines is Google, it is used for searching substantial websites and databases, and it is designed especially for crawling and indexing the web, this produces excellent results that can be more satisfying for the users. Moreover, the engine has the capability for searching a huge number up to tens of millions in limited times, (Sergey, n.d), there is an example of search quality and how it is improved. Google is entitled as a requisite tool for assorted applications. Google File Systems (Sanjay, et al, 2003), is a valuable technique designed to meet the fast-growing demand and can serve a brilliant useful attribute, such as scalable data application, provide the reliability of sustaining while tasking on cheap merchandise hardware, and also gives peak execution to a set of clients. Even the algorithms that are used in search engines have a key role in answering a query by retrieving hundreds of pages in a limited time and giving precision results connected to the web page quality ranking. Furthermore, the algorithm produces a high page ranking if the retrieved pages are sharply many. In other words, it yielded quality competency to the searched results.

Dramatically, (Marc, 2019) presents how Amazon Web Services diffused a leader across its database using a vast range of patterns from paradigms such as Apache Zookeeper, to Paxos is a byword for algorithms.

Fault Tolerance
Google search engines can deal with component failure. The quality and quantity are together making the problem normal rather than appear to be more complicated. The failure of the components is capable of serving a result with the worse corrupted data or unavailable system; the algorithm that the engine used entitles the engine to be the most popular one as the way used for ranking and searching the word query (Yann, et al., 2003); as well as (Rafailescu, 2017) which discussed Fault-tolerant and its effectiveness for prompt performance; besides (Shital, et. al., 2023) that analyzed the computation cost related to the size of system and Fault-tolerant.
Availability
In Sanjay, et al., (2003) paper's, the Google file system has the strength to be used in major applications, like in clusters. As a result, to keep the efficient system with high availability, some strategies have been adopted:
- Fast recovery
- Replication

2. Materials and Methodologies

The greatest extensive trouble in distributed systems is a leader's failure. The choice of the bully algorithm to be the main core algorithm for this work is backed by tremendous causes. The algorithm is simple to understand; it could be accomplished by any topology and is not complicated to provide a full joining channel, unlike the ring algorithm which is specified for the ring paradigm and needs more time and effort. Several studies have revealed how to modify the bully algorithms (Frans, 1992; Kordafshari, et al, 2005; Ozalp, 1995; Quazi, et al., 2004). The proliferation of these studies makes a fast decision to adopt this algorithm.

Proposed Methodology
This paper intends to develop the topic of the leader election algorithm in distributed systems that authorize a collection of processes for choosing one node to represent as a leader when an enigma occurs and produce a proper algorithm design that can deal with this problem. Nevertheless, to achieve this goal, PVM is used for the understanding of concurrent programming as a message passing, this facility has a strong capability for sending and receiving data messages between processes furthermore it supports a simultaneous interaction framework. Similarly, it is convenient and uncomplicated to generate a set of processes. The advantageous enchanting of PVM is yielding structure with respect to the unification throughout the parallel framework such as the cluster, where the user can apportion a chunk of data in various missions that are possible engaged in concurrently by the interactive role of data all over nodes. The purpose of this study focused on how to profuse and design an election procedure that can build a trust communication messages channel between a block of synchronized code, each message tagged by a unique number. This system was written in C language, and there was a main task known as (Driver), where the user is capable of initializing and making the announcing for the major attributes of the proposed executed algorithm. There is diversity of functionality for the message interaction package. What is more, a program called (Worker) can function after receiving the data from the driver program and makes suitable communications with other workers as needed. Since then, it bundles the outcome which included into message-package and transmits them back to the driver. The achievable messages-interaction purpose means, all messages are synchronized, and message-interaction will be assigned by integer number (msgtag). For the most part, this sequence number provides the reliability of task communication, as there will be a set of messages exchanged during the performance. The simulation of node fails attained by using the flag (disable). Accordingly when there is a failure and the election is needed, the flag will be disabled, which is conducive to operating and electing a new node to be a new coordinator.

2.1 Cluster Computing

A cluster can be defined as subdivision of associated processes that tasking relative to each other via brisk-speed networks, to constitute an unaccompanied system. This network normally has two computers or more, namely the nodes and sole boss that customarily has its scheduled obligation by arranging the workload of slaves to be executed. However, the website of the Scalable Computing Laboratory (Scalable Computing Laboratory, n.d.) supplies a terrific resource for cluster computing. The advantages of accomplishing combined repository aspect are for enhancing the rendition and make the services available of services all over the system in case one or two node crashing. This can be seen during the hardware amendment; besides, all nodes provide a proper agreement and availability among substantial sorts of computers from singled process to a machinery with vector process, and a parallel computing together with miscellaneous of micro processes. As an example being, shared memory multiprocessors, cluster computing, networks, vector processes, and internet computing.

2.2 PVM Tool

The interaction for a combination of paralleled computers in any architecture that performing consistently to solve vast predicament, owing to this, the emergence of software called PVM, and hence it accredits the authenticity for unified workstations to execute their duties. PVM comprises of a massive programming library. Furthermore, a bunch of functionalities those supply the interaction among the nodes. There are several benefits of PVM, to illustrate, the capability of drawbacks endurance, the users have the ability for imposing their knowledge to abate the overcharges of resolving a broad problems; as well as, the mobility, to be on a large scale, and the homogeneity.

The reason why establishing PVM for current study is due to its attributes, and advantages, its procedure is written in C language which supports the most paradigms. In the same way, the paralleled message interaction is quite simple for conception and not taking too much time to acquaint with. However, the reference book (Geist, et al., 1994), and then both (Index for PVM3 Library; PVM Home Page), all are rich of data that can knowledgeable for express. In addition, the technology school at the university accommodates PVM at its cluster, where the programmer has the ability to perform a massive range of paralleled functionalities those contained in the library. To illustrate, below some steps demonstrate using belonged cluster:

- Throughout the clustered and to kicks-off the mission, it needs the address (***`sots.brookes.ac.uk`***)
- Getting the authentication for the system by activates the accounts, entering details (user-name and pass-word). ***`(user-name@sots.brookes.ac.uk).`***
- ***`(ssh 161.73.146.51)`*** this is the head node address, which is compulsory to get in on and commences the activities, for example spawn more node and possibly start the interaction using PVM environment.

Now the infrastructure smooth for tasking, indeed, to deal successfully with the situation, this a valuable piece of information indicated below dealing with both writing and compilation of algorithm:

`$ ls` shows the whole of the tasks that the user has in its own cluster.

`$ cd` moving to the major algorithm that has two codes, the boss and servant.

`$ vi electiondri.c` goes into the master (driver) code.

`$ vi electionwor.c` enters the slave (worker) code.

Assuredly, if the user requisites to go into one more node just typing ***`$ pvm`*** at the head node, at that time is likely to be added by typing ***`pvm> node005`***, furthermore, to state the recent paralleled machine, which carried out by writing ***`pvm> conf.`*** This is the flourished complete reference for message interaction (Geist, et al., 1994). The compiling of algorithms was performed by using `UNIX Makefile`. Definitely, a set of miscellaneous functionalities that accredits a user proceeding writing PVM interaction messages and get acquainting within a shorter time. Following is a description of some main functions, and those who interact with them to design and build the project's core code:

All tasks of the PVM are discriminated by an integer task featured (*tID*), as the messages are dispatched to and received from (*tIDs*). Since (*tIDs*) entirely must be unique across the platform. When a task joins a group, it is assigned a unique (***instance***) number, when it commenced at and counts up.

- ***pvm_mcast()*** procedure is anticipated for deploying interacted package, lively transmits the package to a set of performed duties in the task id (*tIDs*) array, and with using a message tag (***msgtag***) the included values could be recognized. The master task supplied by this procedure, therefore the receiving nodes have the ability to contact the ***pvm_recv()*** function for get in the imitating of the package message interaction.
- ***pvm_spawn()*** performs for initializing new nodes duplicating of interacted processes. It is holding the path of the interactive nodes.
- ***pvm_recv()*** it is duty carries on a block of nodes until circulating a package with its recognized tag has come in. Then the function ***pvm_recv()*** function places the interacted package in a new live receive buffer, furthermore unpacks all values included in the package using the command ***pvm_upkint()***.
- ***pvm_send()*** command establishes and sends a package that contained in an active broadcast buffer to identify processes by (*tID*), where the message tag is used to label the included in the package.
- ***pvm_nrecv()*** command whereupon employed to ascertain whether the interacted package with its tag figure has come in. Within the election, the worker hold on an concession from the worker since the concession probably unattainable, a non-blocking receiver will use ***pvm_nrecv()***; else, the receiver will be blocked.
- ***pvm_mytid()*** function which pass back (*tID*) of current procedure and it possible to re-established various times. It registers the node inside PVM if just this is the first PVM call. All the call of PVM system "not only ***pvm_mytid***()" will record a performed interaction in PVM when the mission precluded from transcribed earlier than the call, nonetheless it is mutual discipline to demand ***pvm_mytid()*** primarily to achieve the enrolling.
- ***pvm_exit()*** function informs the topical PVM daemon that the current node is departing the PVM. The function impossible to knock off the node, which is able of performing the tasks just like the other UNIX node. Usually, the programmers calling the ***pvm_exit()*** before logout their C tasks.
- ***pvm_parent()*** routine function returns the (*tID*) of the parent process.
- ***pvm_tidtohost()***function sends back the host of the specified process
- ***pvm_halt()*** routine shuts down the entire parallel virtual machine system.
- ***pvm_delhosts()*** function dismisses the hosts from the virtual machine.

From the efficiency, and the performance standpoint, PVM has been affirmed to be adequate even for applications with a vast interaction to calculating ratio; although, it is significant and gives a great attention for consolidating the tasks with tremendous sizes and diminutive communication. The simplicity of facsimileing the PVM system; furthermore, the application of various programs in addition ameliorates its requisition and will participate to its growing intention.

3. Results

3.1 Implementation

The designed interaction messages comprised of triple stages. Every individual stage has assorted phase's package with vast communication between the nodes. The first stage pitches in during the discovering of at least by one node that the coordinator lost its duty and interrupted, and due to that election procedure commencing. However, the current stage diminishes the compute of collaborate nodes in the message interaction procedure to (n/2). The process that gets in the evidence of acknowledgement possibly receives the interacted package before a particular time; hence, the authentic package is disregard when the highest (*tID*) is received. Secondly, in this phase the acknowledged package comes to with ordinary. Pursuing this further, to afford the link crash possibility the node has authorization to initialize the message passing via the linking channel. In jointly situations, the processes that get in the message interaction consent forward an acknowledged message to the sender.

The second phase uses the minimizing all-to-one interaction to have the upshot in one node that has the address. The operation steps allocated to the (n/2) node that scored from stage one; collect the output in a sole node that has the address for swapping the bigger (*tID*) with the nodes with the similar aspects.

Finally, in the third stage, the node propagates the leader package to the whole nodes in the nearby area utilizing a one-to-many proclaiming communication and considering the connection crashing. Within every step in stage one and two, the received (*tID*) is contrasted with the local (*tID*). The higher (*tID*) is forwarded to the followed step. The leader message includes the new leader (*tID*) and position. Each node receives message-package contained the coordinator forwarded to it via the communication channel procedure. In addition, for presuming the drawback for current stage each node gets in the coordinator package and then circulates another package throughout the interaction channel to ascertain if the message attained to the whole nodes that participation in communication. The node that receives the additional package possibly acquaints the propagation; therefore, it disregards the additional package. If the node has no knowledgeable about the package it publishes the additional package via a communication channel.
The implementation of the Bully algorithm was added to the PVM tool as a part of this project. As declared before, the communications for sending and receiving messages used C language with PVM. The Library of PVM has a set of functions, and they are used to implement the election algorithm communication for this study. All of these functions allow different data to be sent and received. The Bully algorithm can be presented via driver and slave programs, as the driver will manage and initiate the requests and send them to the worker to perform some operations. PVM provides some messages passing and interaction between the worker and the driver as well. The worker program will contain a random array, where it is probable for any node to start the election during coordinator failure. Besides, the [*tIDs*] array holds 5 nodes specified by their identifiers. The worker program will contain a block of Bully algorithm code; every message is tagged by a unique number to keep messages continuously passing and performing correctly. The subsequent, are some of the essential operations that show and describe the implementation of the core algorithms, in the same way, the messages passing communication.

Sender procedure (Driver Operation):
First, the program commencing by spawning numbers of executable election workers copied in a virtual machine and then initializes a coordinator with the highest number. Every program on a virtual machine has a worker task name as a path for the executed program.
This line is to declare workers' distinguisher.
mytid = pvm_mytid();
Here, to find out my task-identifier number.
info = pvm_spawn ("homes/project/electionwor",
(char**) 0, PvmTaskDefault," ", PROCES_NO, tids);
coordinator = tids [4];
Here again, initialization of the coordinator.
coordinator = tids[4];
printf ("The Recent Coordinator is Process 5 which has Value of: %d \n", coordinator);
The code above clarifies the possibility of printing out every task identifier (*tIDs*) for the workers. Here are some parameters that are needed to call. The parameter [***PROCES_NO***] refers to the number of processes included in all programs. The array of [*tIDs*], which contains five numbers, represents the election workers' task identifiers.
This code below articulates the operation of the loop to send the initial coordinator to all other workers and notify them about the worker coordinator. Sending a step attached the sequence number to keep the message synchronizing by using a message tag (*msgtag).*
for (i = 0; i < PROCES_NO; i++) {
printf ("pvm_spawn=%d, electionwor tids= %d\n", info, tids[i]);
pvm_initsend (PvmDataDefault);
pvm_pkint (tids, PROCES_NO, 1);
pvm_pkint (&coordinator, 1, 1);
msgtag = 1;
pvm_mcast (tids[i], msgtag);}
while (info > 0) {
waiting the reply from slaves.
msgtag = 2;
pvm_recv (-1, msgtag);

```
pvm_upkint (&current_Node[i], 1, 1);
printf ("The Coordinator is: ",current_Node);}
```

In the next code, the driver disables the communication for the initial coordinator and broadcast the message with the series value, and then the interacted package will be received by the worker to set off some operation reflected in the disabling situation and send the result back to the driver.

```
tids [4] = 0;
pvm_initsend (PvmDataDefault);
pvm_pkint (&coordinator, 1, 1);
msgtag = 3;
pvm_send (tids [4], msgtag);
```

Again in the following code, the driver receives the newly elected coordinator message, accordingly the driver multicasts the message to all election workers, and unpack data as well.

```
msgtag = 4;
pvm_recv (-1, msgtag);
pvm_upkint (&result, 1, 1);
 printf ("The New Elected Coordinator is: \n", result);
While (info > 0) {
coordinator = disable;
pvm_initsend (PvmDataDefault);
pvm_pkint (&PROCES_NO, 1, 1);
pvm_pkint (tids, PROCES_NO, 1);
pvm_pkint (&coordinator, 1, 1);
msgtag = 5;
pvm_mcast (tids [4], msgtag); }
pvm_exit (); }
```

Receiver procedure (Worker Operation):

This is the first step for receiving the code that all workers receive from the driver to notify them by the initial coordinator, the message holds the coordinator, which is tagged with an integer message number. When the message is arrived to the workers, all the contents will be unpacked.

```
#define PROCES_NO 5
int mytid;
int msgtag ;
int current_Node = 0 ;
int coordinator ;
int new_Coordinator;
int driver_tid ;
int electedNode=0;
int info;
int driver_tid ;
int enable = TRUE ;
int disable = FALSE ;
int tids[PROCES_NO]={150,500,700,200,900};
mytid = pvm_mytid();
driver_tid=pvm_parent();
While (info > 0) {
msgtag = 1;
pvm_recv (-1, msgtag);
pvm_upkint (&PROCES_NO, 1, 1);
pvm_upkint (tids, PROCES_NO, 1);
```

The code underneath, determines the slave to carry-out the operation

```
for (i= 0; i < PROCES_NO ; i++ )
if ( mytid == tids[i] ) {
result = i;
coordinator = result;
```

The coming block of code deploys the outcome backwards to the master after carrying out some operations.

```
pvm_initsend (PvmDataDefault);
pvm_pkint (&result, 1, 1);
msgtag = 2;
pvm_send (result, msgtag);  } }
```
The forthcoming piece of source code illustrates the operation to start the election for a new coordinator, which checks the state for every node in the array [*tIDs*] with the current node. The largest numbered node will be elected after comparing the element of the array [*tIDs*] with the current node identifier.
```
msgtag = 3 ;
pvm_recv (-1, msgtag);
pvm_upkint (&coordinator, 1, 1);
pvm_upkint (&tids [0], 1, 1);
coordinator = disable;
if (coordinator == disable) {
srand (time (NULL) );
```
This code to Initialize random generator .
```
current_Node = (rand () % 6);
if (current_Node == 0)
current_Node = 1;
if (tids[current_Node - 1] == 0) {
If (current_Node == 5) {
current_Node = 4; }
else {
current_Node += 1; } }
printf ("The Current Node is: [Process %d] Which will Start Election.\n", current_Node);
```
The Operation to set up the electing procedure for a new coordinator.
```
new_Coordinator = current_Node;
for (i = current_Node; i < PROCES_NO; i++) {
if (tids[i]> current_Node && current_Node > 0) {
current_Node = tids[i]; }
if (current_Node > new_Coordinator) {
new_Coordinator = current_Node;
current_Node = tids[i]; }  }
Printf ("The Boss Node is: %d\n",  new_Coordinator);
```
The worker forward a message to notify the driver and other workers by a new (elected process).
```
for (i = 0; i < PROCES_NO; i++) {
pvm_initsend (PvmDataDefault);
pvm_pkint (&new_Coordinator, 1, 1);
msgtag = 4;
pvm_send (tids[i], msgtag); }
```
The code techniques below manifest the procedure of choosing a new node from a random array because the proposed algorithm design is free from failure. That means it is enabled for any node to crash during the election of the coordinator. Therefore, the duty will continue by starting with a new node, instead of the process that starts the operation is down.
```
srand ( time(NULL) );
new_pross_request = (rand () % 6);
if (new_pross_request == 0)
new_pross_request = 1;
tids [new_pross_request - 1] = 1 ;
printf ("A node that holding tids : %d is perform!! \n", new_pross_request);
if(new_pross_request > new_Coordinator)  {
printf ("And The Coming Coordinator is, Node : %d\n",new_pross_request);
new_Coordinator = new_pross_request; }
pvm_exit ();}
```

Program Testing

The tests of these core algorithms go through the specification and design. That is to say, the assumptions that lead to current production, coupled with the design and implementation. Reliably, to authenticate and

achieve the goal of failure for starting the election algorithm, the flag (*disable*) is employed. There are two stages of algorithms have been implemented:
The first program is called the ***electiondri.c***, likewise the second program has been designated as ***electionwor.c***. In order to test for core algorithms and to be executed, the main method has been implemented for both programs and contains all parameters for the verification of execution. The program is estimated to be run using a ***makefile*** program. A UNIX makes the program provides the simplicity and maintenance of programs, and creates a ***makefile*** to accompany the source files. In the final phase, the core code is invoked using the make command wherever a change is made to the system, as it is a directive that gives the relationship between sources, the target, and the compilation command.

4. Discussion

4.1 Source Code Program Documentation

For making an unambiguous program design, it is splendid to give and express data structures and identically the description of operation. As though, the program documentation can be used to document the core algorithm concerning providing a record of program module development. Thereafter, below, is the code documentation that possibly assists in implementing the obvious design.

Title: Election Algorithm in Distributed Systems
Files names: 1 - electiondri.c
2 - electionwor.c
Problem Statements: The program elects a new coordinator among a set of processors when the leader fails.
Requirements:

- Driver is spawning election workers.
- Initializes a coordinator.
- Inform all workers by the new coordinator.
- Disable the coordinator's communication.
- Communication is performed between drivers and workers, likewise between workers themselves.
- Elected a new coordinator that all nodes agree with (the node with the highest number).
- Every previous coordinator who is disabled needs to enable their communications.
- Print each step of the electing coordinator procedure on the screen to notice the message passing between nodes.

Constraints:

- Every node has a unique identifier number.
- The array of nodes [*tids*] consists of 5 nodes.
- In every election operation, the nodes choose a process as the coordinator with the highest number from the array [*tIDs*].

Operation:

Program: ***electiondri*** (Driver)

- Coordinator Initialization.
- Propagates a package to the boss for disabling its communication.
- Initializes a new coordinator.
- Circulates a package to the new master to disable its communication.
- Exchanges the messages with the workers, and enables the previous coordinator to continue subsequent elections.

Program: ***electionwor*** (Worker)

- Receives and interacts with all messages of the driver.
- Receives coordinator messages, and workers perform the communications to elect a new leader.
- Send back the message for every step of the election to the driver program.

4.2 The Problem Specification Design

The infrastructure of the communications takes the consideration the mechanism of the network (Ben-Ari, 2006). The algorithms that are represented in this paper presuming the complete associated paradigm, the proposed election algorithms need message interaction between the programs during the operation, and that was achieved by using the cluster, which PVM is installed within, the tool in which attaining message passing and it presents in each node. Additionally, the assumption means the interactions are errors complimentary of an abstraction and the presuming of definite limit, but tyrannical of transition times for packages is unlikely to change. The algorithms have a conscientious duty, as well as are insensitive to transform. There is unrepeatable distinguishing value for every process that participates in the package interaction. In short, there are double remarks for interaction:

- Send (*MessageType*, *Destination* [Parameters]).
- Receive (*MessageType*, [Parameters]).

As a case in point, here next in (Figure 3) indicates *node* 1 proclaims a package of type request to *node* 2 with two numbers of parameters *35* and *49.*

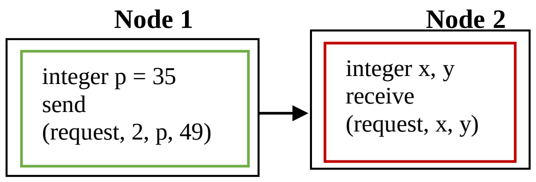

Figure (3) processes interactions.

When the package is come in at *node* 2, these figures are conveyed to the attributes *x*, and *y* announced at the *node*1.

4.3 Core Algorithm Architecture

The infrastructure of the algorithms uses an entirely joined manner and sufficiently every process able to transmit a package straight away to nearby processes. It is costly due to it lacks for extra interaction channels. This paradigm permits the node to dismiss while the tasks performing, the three types of messages are carried out. An elected package is broadcasted to declare a bullying, then an answer message is sent in response to the election package, and finally, the discriminated elected node is announced by the master sending the message. The drawing in (Figure 4) pointed out four nodes that contain the paradigm, that is, a node distinguished and the value chosen by each node.

Messages-Passing Time and Complexity

There is a paper work of modifying the bully algorithm using a different method such (Rahman, 2009; Basim, et. al., 2013), where it was presented what was called Election Commission; where it was possible for minimizing the complexity as well as scaling down the redundancy of election time. That was lead to diminishing the message-passing periods.

Here in the recent paper, the PVM used to control and manage the message-passing communication time, as PVM mechanism and design has the capabilities for dealing with the Time Limitation, and this is regarded as one of the most powerful advantage for this software

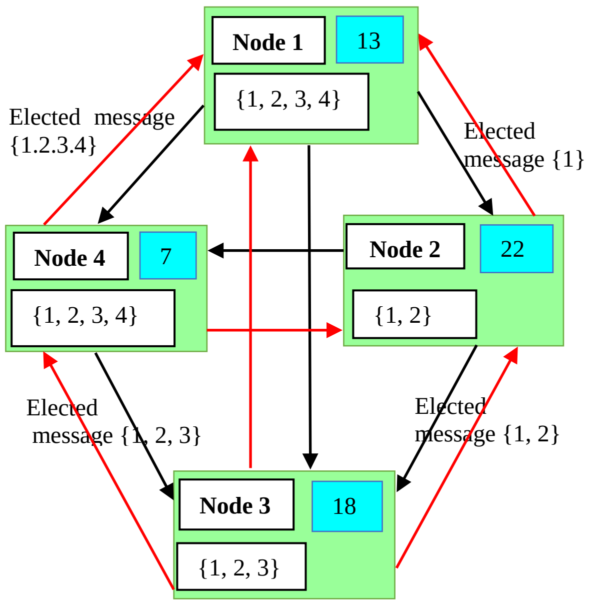

Figure (4) Entire Architecture of Interaction

Presuming the package interaction begins when node1 circulates message package to entire processes. Each node is benchmarked as not engaged in the bullying interaction when node 1 sets off the election package embracing the distinguished value inside it which is (35) in energetic list {1} and transmits it to the entire processes nearby. All participated nodes will get in the interaction message making comparison between the value (35) which contained in and their own values. It follows that, whole process will receive acknowledge message from node2 owing to the other nodes, due to the nodes have less identifiers than it does. In different circumstances, notably node 4 has no possibility to broadcast acknowledgement message, alternatively of appending all whole processes to its brisk list; furthermore node 3 backwards the acknowledgement message to node 1 and node 4 similarly, on the flip side node 2 will not receive any communication from node 3. From another side node 1 surely responds and backwards the package just for node4 only, because of the value of node 1 is greater than node 4. Again, during node 2 transmitting the acknowledgement message with its distinguished value where it is the greatest to the entire participated processes, and hence it is the bullying electing package (the leader). Since then it has a time interval if no replying message has arrived, with certainty it broadcasting the declaration announces itself as a leader.

Reliable Broadcast

To authenticate that all participated nodes announce the similarities of the interacted package when circulated messages will deploy by specific nodes, and similarly, for guarantee that there are no dissimilar interacted packages are transmitted holding the same identifier number, (Grama, et al., 2003; M. Ben-Ari, 2006). Therefore, during performing if the interacted message package of algorithm has been authenticated for execution, and deploying then the following are the assumptions:

- Reliable broadcast (1) *Validity*: whether the right node publishes the message (M), and some specific nodes finally forward (M).
- Reliable broadcast (2) *Agreement*: when a proper tasking node circulates a package-message M, hence proper nodes ultimately announce (M).
- Reliable broadcast (3) *Integrity*: In order to arrange the package tasking, every node *p* deploys at most one package message (M) with identifier *id* if the node *p* has the authenticity from other nodes, then if the sender (M) is succeeded, then *M* was formerly indicated by the sender (M).

The electing worker package employs the *Boolean*; hence it is announcing for attainable which node is performed and which does not.

5. Conclusion

The principal incentive for PVM function is obtained from its valuable for paralleled computing. Transferable skills with this software have shown clearly evidence that it possible to be equipped for executing on present platforms with acquiring the affecting of the interaction interfaces, that constructing the advantages for reliability of several heterogeneous resources. The imperative attributes of PVM in paralleled programming are that, providing the programmer with both the tolerance of crash as well the accuracy.

To sum up, how a set of interacted nodes performing a specific task to approved their action via contrasting distributed infrastructure, nevertheless of the failure. As the reliable channel delivers the messages to the recipient's input buffer; hence this explained neither the leader nor other nodes relied on for deploying the interacted package. Which gives the result, no issue will affect the capability of message package interaction.

In addition, the simulations of the package interaction algorithms between the driver and other workers programs have been implemented. PVM library has included a set of functions that capable of being transmitted and received during the undertaken of tasks' performing. Some arrangements in messages passing have synchronized; and the design stage has demonstrated the core of the algorithm that relied on the electing procedure, that commonly choosing single process among a group of nodes. Furthermore, the description of messages communication with using suitable function has been performed; the source code program documentation has been presented. As well as the requirements and constraints rules those control the exchange of the package-messages interaction between the processes for electing a new leader. Full communication's architecture is illustrated in drawing; in addition, the design of the package-interaction-messages is fully connected paradigm. Finally, the proper election algorithms were introduced, which were successfully performed.

References

Andrea, C. (2001). Implicit Co-scheduling: Coordinated Scheduling with Implicit information in Distributed Systems. *ACM Transactions on Computer Systems, 19*(3), 283–331.

Basim, A., Laith, H., Mohammed, H. & Mohammed A. (2013). Reducing Massage Passing and Time Complexity in Bully Election Algorithms Using Two Successors. *International Journal of Electronics and Electrical Engineering, 1*(1), 1-4.

Basu, S. (2011). An efficient approach of election algorithm in distributed systems. *Indian Journal of Computer Science and Engineering (IJCSE), 2*(1), 16-21.

Ben-Ari, M. (2006). Principles of Concurrent and Distributed Programming. 2nd ed. USA: Addison-Wesley.

Biswas, A., Tripathi, K. & Aknine, S. (2021). Lea-TN: leader election algorithm considering node and link failures in a torus network. *The Journal of Supercomputing, 77*, 13292-13329

Chang, E. & Roberts, R. (1979). An improved algorithm for Decentralized extrema-finding in circular configurations of processes. *Communications of the ACM, 22*(5), 281-283.

Chen, Z., Gul, M. & Kantarci, B. (2023). Practical byzantine fault tolerance-based robustness for mobile crowdsensing. Distributed Ledger Technologies: *Research and Practice, 2*(2), 1-24.

Coulouris, G., Dollimore, J. & Kindberg, T. (2005). Distributed Systems Concepts and design. 4th ed. USA: Addison Wesley.

Daymude, J., Gmyr, R., Richa, W., Scheideler, C. & Strothmann, T. (2017). Improved leader election for self-organizing programmable matter. *In Algorithms for Sensor Systems: 13th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, Vienna, Austria*, 127-140.

Frans, M. Kaashoek (1992). PHD Thesis*: Group communication in Distributed Computer Systems.*

Geist, Al., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., & Sunderam, V. (1994). PVM: Parallel Virtual Machine – A users' Guide and Tutorial for Networked Parallel Computing. USA: The MIT press-Massachusetts Institute of Technology.

Grama, A. Gupta, A., Karypis, G. & Kumar, V. (2003). Introduction to Parallel Computing. 2nd edition. USA: PEARSON Addison-Wesley.

Hector, G. (1982). Election in a Distributed Computing System. *IEEE Transactions on Computers, C-31* (1), 48-59.

Index for PVM3 Library (n.d.). Retrieved from: http://www.netlib.org/pvm3/

Kordafshari, S., Gholipour, M., Jahanshahi & Haghighat, T. (2005). Modified Bully Election Algorithm in Distributed Systems. Department of Electrical, Computer & IT, Islamic Azad University, Qazvin Branch, *Atomic Energy Organization of Iran (AEOI), NPPD*, Tehran, Iran.

Kutten, S., Pandurangan, G., Peleg, D., Robinson, P. & Trehan, A. (2015). On the complexity of universal leader election. *Journal of the ACM (JACM), 62*(1), 1-27.

Marc, B. (2019). Leader election in distributed systems. Amazon Web Services. Retrieved from: https://d1.awsstatic.com/builderslibrary/pdfs/leader-election-in-distributed-systems.pdf

Ozalp, B., & Andre, S. (1995). On Group Communication in Large-Scale Distributed Systems. *SIGOPS Operating Systems Review, 29* (1).

Peterson, L. (1982). An O(nlogn) Unidirectional Algorithm for Circular Extrema Problem. *ACM Transaction on Programming Language and systems*, *4*(4), 758-762.

PVM Home Page. (n.d.). Retrieved from: http://www.csm.ornl.gov/pvm/

Quazi, M., Salahuddin, M., & Mohammad, M. (2004). Modified Bully Algorithm for Electing Coordinator in Distributed Systems. *The 3rd WSEAS International Conference on Software Engineering, Parallel and Distributed Systems*.

Rafailescu, M. (2017). Fault-tolerant leader election in distributed systems. *International Journal of Computer Science and Information Technology*, *9*(1), 13-20

Rahman, M., & Nahar, A. (2009). Modified Bully Algorithm using Election Commission. *MASAUM Journal of Computing*, *1*(3).

Refai, M., Sharieh, A. & Alshammari, F. (2010). Leader Election Algorithm in 2D Torus Networks with the Presence of One Link Failure. *The International Arab Journal of Information Technology*, *7*(2).

Refai, M. & Naim M., A. (n.d.). Leader Election Algorithm in Hypercubes with the Presence of One Link Failure. Amman Arab University for Graduate Studies.

Riccardo, G. & Stefano Z. (1985). An Election Algorithm for a Distributed Clock Synchronization Program. University of California: Report No. UCB/CSD 86/275.

Sanjay, G., Howard, G. & Shunk-Tak, L. (2003). The Google File System. *ACM SIGOPS Operating Systems Review*. *37*(5), 29-43.

Scalable Computing Laboratory (n.d.). Retrieved from: http://www.scl.ameslab.gov

Scott, S. (1999). Leader Election in Asynchronous Distributed Systems. Computer Science Department, Indiana University, Bloomington, IN 47405, USA.

Seema, B. & Kavita K. (2014). Leader Election Algorithms in Distributed Systems. *International Journal of Computer Science and Mobile Computing*, *3*(6), 374-379

Sergey, B. & Lawrence, P. (n.d.). The Anatomy of a Large-Scale Hyper textual Web Search Engine. Computer Science Department, Stanford University. Retrieved from: http://infolab.stanford.edu/~backrub/google.html

Shital, S. & Jayshree, P. (2023). A Robust, Preference-Based Coordinator Election Algorithm for Distributed Systems. *International Information and Engineering Technology Association*, *28*(4), 843-851.

Singh, G. (1992). Leader Election in Complete Networks. *Department of Computing and Information Sciences, Kansas State University, Manhattan*, KS 66506.

Supase, S. & Ingle, B. (2021). A novel algorithm for secure and reliable coordinator election in distributed networks. *International Journal of Advanced Technology and Engineering Exploration*, *8*(85), 1682-1694

Supase, S. & Ingle, B. (2020). Are coordinator election algorithms in distributed systems vulnerable. *In 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 1-5

Tanenbaum, A. & Van, M. (2007). Distributed Systems Principles and Paradigms. 4th ed. USA: Pearson Prentice Hall.

The Spider's Apprentice: *Search Engine Ranking Algorithms*. (n.d.). Retrieved from: http://www.monash.com/spidap4.html

Wei S., Bouabdalla, A. & Pradip, S. (2005). Leader Election in Oriented Star Graphs. Networks, *45*, 169-179.

Wm, F. (1982). On an improved Algorithm for decentralized Extrema Finding in Circular Configurations of Processors. *ACM Communication*, *25*(5), 336-337.

Yann, L., Collet, P., Prost, T. & Lutton, E. (2003). Introducing Lateral Thinking in Search Engines with Interactive Evaluation Algorithm. *ACM symposium on Applied Computing*, 1-58113.