An Adaptive Bully Algorithm for Leader Elections in Distributed Systems

Monir Abdullah[1,2](✉), Ibrahim Al-Kohali[2], and Mohamed Othman[3,4](✉)

[1] Computer Science Department, University of Bisha, Bisha, Saudi Arabia
mkaid@ub.edu.sa

[2] Information Technology Department, Thamar University, Dhamar, Yemen
legend22013@hotmail.com

[3] Laboratory of Computational Science and Mathematical Physics, Institute for Mathematical Research, Universiti Putra Malaysia, UPM, 43400 Serdang, Malaysia

[4] Department of Communication Technology and Network, Universiti Putra Malaysia, UPM, 43400 Serdang, Malaysia
mothman@upm.edu.my, mothman@ieee.org

Abstract. Leader election is a classical problem in distributed system applications. There are many leader election algorithms, but we focus here on Bully Algorithm (BA). The main drawback of BA algorithm is the high number of messages passing. In BA algorithm, the message passing has order O (n^2) that increases heavy traffic on the network. In this paper, an Adaptive BA (ABA) is proposed to reduce the number of messages and make the leader election operation more flexible and safer. The proposed algorithm is based on the Highest Process Identification (HPI) and the Next HPI (NHPI) to facilitate the leader election operation. Moreover, the repetition of the leader election is stopped when the candidate coordinator fails. Our analytical equations show that the ABA algorithm is more efficient rather than BA algorithm, in both, the number of message passing and the latency, and the message passing complexity decreased to $O(n)$.

Keywords: Bully algorithm · Election system · Message passing

1 Introduction

Leader election is considered as an important problem, classical and fundamental problem which happens in distributed systems [1]. Leader election is to select one process or node in the system to become the new coordinator after the previous coordinator fail. The purpose of the leader election is to complete the same job as the ex-coordinator and to avoid any delay in tasks execution. Failures happen because of the occurrence of failures in the software, or hardware or maybe maintenance. Leader election operation occurred when there was no response from the coordinator, thus we were encouraged to start leader election. There are several algorithms had been introduced for electing coordinator process that based

© Springer Nature Switzerland AG 2019

on two basic algorithms, i.e. BA algorithm [2] and Token Ring algorithm [3]. In the coordinator election, our objective is to select a coordinator process among various processes that reside in a distributed environment. In this research, we are specifically focusing on BA algorithm. BA algorithm is an important algorithm used in leader election operation which is considered more popular [4]. Not only this, but it is recently used and implemented in Big Data and NoSQL [5] and IoT [6]. There is a plethora of research on BA algorithm and that helped in renewing related studies in this study [2,4,7,9–14]. The main drawback of BA algorithm is the high number of message passing. In this method, the message passing has order O (n^2) that increases heavy traffic on the network. Our proposed adaptive algorithm successfully reduced the number of message passing to O(n). The rest of the paper is organized as follows. Section 2 reviews the related works. In Sect. 3, the original BA algorithm is presented. Section 4 presents the ABA algorithm. The experimental results and discussion will be presented in Sect. 5. Finally, Sect. 6 concludes the paper.

2 Related Works

Several coordinator election algorithms have been proposed over the years some of the main election algorithms are BA algorithm, Ring algorithm. Garcia-Molina [2] proposed a BA algorithm in which they introduce an election mechanism for the selection of the coordinator. While undertaking this procedure the number of messages increased, i.e. the identification of the failed node, then starting an election procedure and the process that having the highest identification process number will be selected as a coordinator. After selecting a coordinator we make an announcement of the selection of new coordinator among various processes in the network. This whole procedure requires a number of messages is to be exchanged which increases the traffic in the network. The researchers discuss the shortcoming of synchronous BA algorithm and propose a modified version. They maintain that their modified algorithm is more efficient than the traditional BA algorithm because it decreases the number of passing messages, and it has fewer stages [9]. Some researchers added an additional feature to the original algorithm [10]. This method uses an assistant as a leader when ex-leader fails. Therefore, there is no need to stop the execution of tasks when a leader crashes. The performance increases when the numbers of node increase. The modified bully election proposes a linear time algorithm for leader election using heap structure that deals with the leader election algorithm for a set of connected processes like a tree network [11]. The researchers discuss the shortcomings of three algorithms of the original and modified BA algorithms. They propose the same traditional BA algorithm but using a new concept called election commission, with the addition of Failure Detector (FD) and a Helper processes (H) to have a unique election with the Election Commission (EC). This method is more efficient and decreases the number of passing messages [12]. A new method is based on electing a leader and an alternative is proposed [16]. In this method, if the leader fails, the alternative takes care of the leader's responsibilities. This way is more effective, messages will be less complexity in the fewer stages. The researchers proposed a new method that uses fault tolerant mechanisms to improve the BA and Ring algorithms [13]. They present a new algorithm called a heap tree algorithm, based on the max-heap data structure. Their results show a fewer number of passing messages. Furthermore, a new algorithm is proposed in which a new leader is elected immediately after the leader fails. It depends on a process status table which contains the number of each process and its status in the current system [14]. The researchers present a safety strengthened leader election protocol with an unreliable failure detector. By analysis, it appears as more efficient in safety and liveness properties in asynchronous distributed systems [15]. A new method which uses a flag that works to reduce the number of passing messages when a failure discovered by more than one process is presented [8]. The results show a relative success in decreasing the number of passing messages and the number of steps. In [7], the researchers proposed a new method reduced passing messages between the coordinator and processes. This mean, when a process starts sending a request to the coordinator, it stores them in a list. Every period the coordinator sends messages to other processes that it has the higher *id* number. But when the coordinator failed, we will compare the processes between *id* number of process and *id* number which sent by the coordinator [7]. The researchers in [17] proposed a new method that uses a proxy server for leader election by performing an analytical simulation. Their results show a decreasing in the number of passing messages and waiting time. A comparative study discussed the concept of four election algorithms, BA [2], Modified Bully Election [9], Improved Bully Election [20], Ring Election [18]. In [19], a slight modification in the classic BA algorithm is proposed which reduces the number of messages that are needed to elect the leader and also proposes new methods of how to react when the dead leader recovers again. The result of the modified BA algorithm is more efficient than the existing leader election algorithms. The researchers in [4] put forward a new method which depends on the distance. They assumed that there exist a node is called centroid. If the distance between a centroid and a node is short, the node has the highest priority and if the distance between the centroid and the node is long, the node has the lowest priority. Recently, BA algorithm is implemented on a specific and low-performance Internet of Thing (IoT) devices [6]. The implementation of the BA algorithm for leader election is achieved in a two-stage process.

3 Bully Algorithm

Based on message generation in the system, a comparative analysis of [2] and our proposed algorithm would be appropriate to determine which algorithm performs better than the others. BA algorithm requires $n - 1$ messages to elect a leader node in the best case, where n is the number of nodes. The best case happens when the node having the next highest *id* number detects the failure of the leader node and hence announces an election [4].

In the worst case, it requires $\mathrm{O}(n^2)$ messages to elect a leader node. The worst case happens when the lowest *id* node of the system detects the failure of the leader node. It will send election messages to $n - 1$ nodes having higher *id* than itself. Each of the nodes eventually initiates a separate election one by one. In this algorithm, a previously failed node which was not a leader node initiates an election after recovery. But if it was a former leader, it just broadcasts coordinator messages to other nodes to announce itself as the new leader. Hence, it requires $O(n^2)$ messages to elect a leader node in the worst case and $n - 1$ messages in the best case. The BA algorithm steps are as follows:

1. The process (Pd) that discovers a failure sends a message to all processes in the system. The message contains the *id* of a process (Pd).
2. When the process (P_i) receives the message, it starts comparing the received *id* with its *id*.
3. If the *id* of process (Pd) is lower than the *id* of process (P_i), Then process (P_i) returns a message: "Ok" to process (Pd).
4. the process (Pd) continues steps 1, 2, 3 even coordinator selected.
5. If process (Pd) does not receive a message: "Ok" from the other processes, and then it will be chosen as a coordinator (Fig. 1).

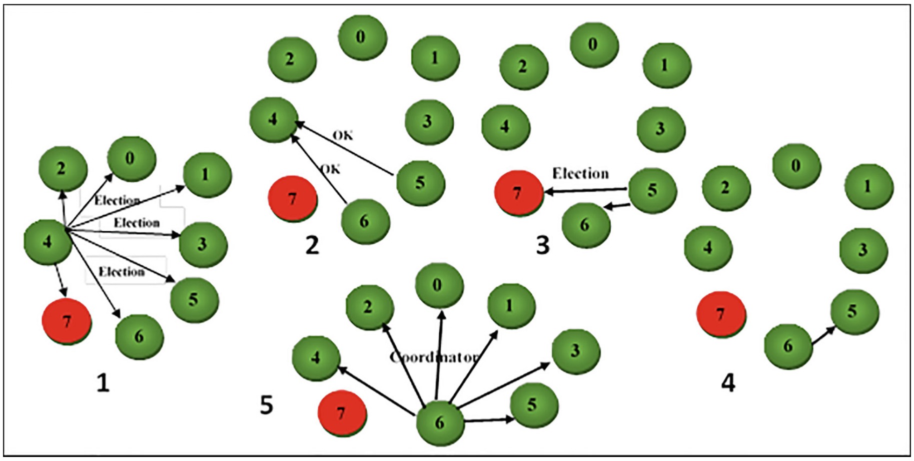


Fig. 1. Bully leader election algorithm.

The drawback of BA algorithm is that if the process that discovers the failure has a lower Id, this leads to the increase of the number of messages in the election operation. In this method, the message passing has order O (n^2) that increases heavy traffic on the network.

4 Adaptive Bully Algorithm

In this section, our proposed ABA algorithm is presented. Firstly, we will explain the four important variables:

1. The Election Variable (EV): is a variable that stores the node *id* of the coordinator.
2. Node ID: is a variable that stores the *id* number of the process itself. It cannot be modified.
3. The Highest Process Identification (HPI) and the Next HPI ($NHPI$): are variables which store the highest two numbers during election operation.

To implement our algorithm, we adapt a new structure for every node in the system which contains the above four variables as shown in Fig. 2:

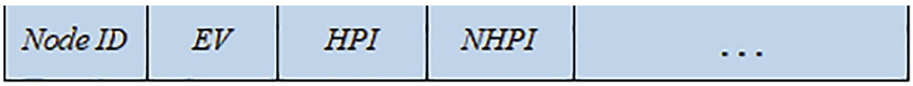


Fig. 2. ABA algorithm node structure.

4.1 Adaptive Bully Election Algorithm

When a process (P_i) requests any task from the coordinator and it does not receive any response within time ($T1$), this signifies the coordinator fails. This action is called: failure check. Failure Check "is a procedure that is immediately executed whenever any process makes a request to the coordinator. This procedure will detect a failure if it occurs". The failure check is the first step in any election operation. Afterwards, the election operation starts. Now, process Pd sends "Start Election" message to all the processes in the system: The message contains the *id* of the process that discovered the failure. Time $T2$ starts when this message is sent. During this time, the process Pd receives messages from the other processes. We have two cases:

1. If a process P_i does not receive a response within the specified time, it sends a message to all the processes in the system: "I'm Coordinator".
2. If a process P_i receives a response within the specified time, then the main operation, which stores the HPI and $NHPI$ starts.

When time ($T2$) finishes, process P_i sends a message to the winning process containing the highest NID: (Highest Value) and: "Tell everyone you are the coordinator". Time ($T3$) begins when process (P) receives the message. The winning process returns a message: "Ok" to process (P). If process (P_i) does not receive the message: "Ok" within time ($T3$), this means the process fails. Hence, process (P) sends to the second winning process, which has the second highest ID, a message contains $NHPI$ and: "Tell everyone you are the coordinator". Time ($T4$) begins when process (P_i) receives the message: "Ok". If process (P_i) does not receive the message "Ok" within time ($T4$), this means the process fails. The process (P_i) sends a message to all the processes in the system: "I'm Coordinator" as shown in Fig. 3.

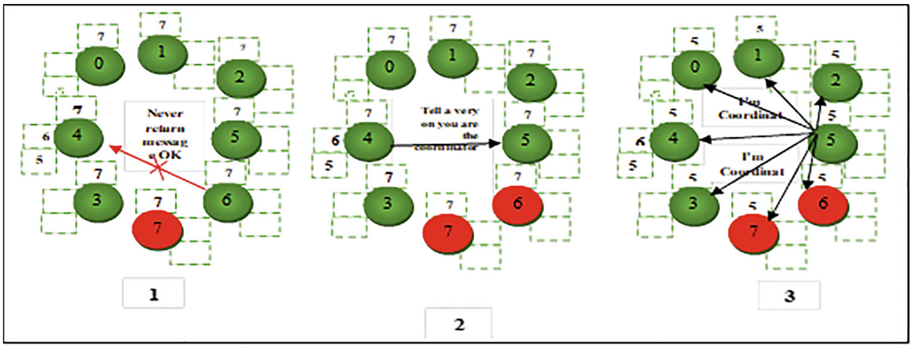
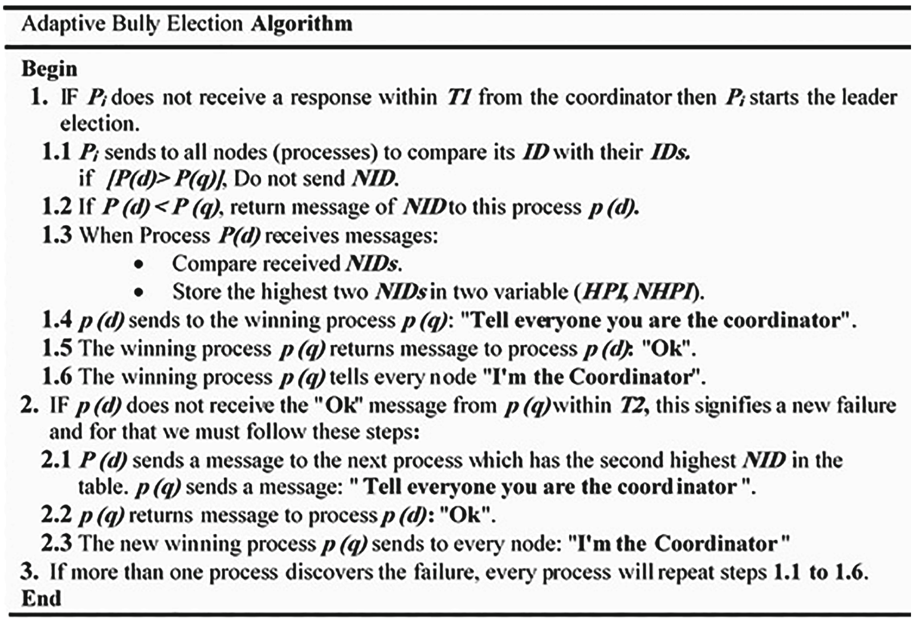


Fig. 3. Leader election operation in case of a failure.

When a process receives the message: "I'm the Coordinator", this signifies the end of the leader election operation, and the receiving process updates the value of EV which is attached to the message received. The ABA algorithm is shown in Fig. 4.



```
Adaptive Bully Election Algorithm
Begin
 1. IF $P_i$ does not receive a response within T1 from the coordinator then $P_i$ starts the leader
    election.
   1.1 $P_i$ sends to all nodes (processes) to compare its ID with their IDs.
       if [P(d)> P(q)], Do not send NID.
   1.2 If $P(d) < P(q)$, return message of NID to this process p (d).
   1.3 When Process P(d) receives messages:
           • Compare received NIDs.
           • Store the highest two NIDs in two variable (HPI, NHPI).
   1.4 p (d) sends to the winning process p (q): "Tell everyone you are the coordinator".
   1.5 The winning process p (q) returns message to process p (d): "Ok".
   1.6 The winning process p (q) tells every node "I'm the Coordinator".
 2. IF p (d) does not receive the "Ok" message from p (q) within T2, this signifies a new failure
    and for that we must follow these steps:
   2.1 P (d) sends a message to the next process which has the second highest NID in the
       table. p (q) sends a message: " Tell everyone you are the coordinator ".
   2.2 p (q) returns message to process p (d): "Ok".
   2.3 The new winning process p (q) sends to every node: "I'm the Coordinator"
 3. If more than one process discovers the failure, every process will repeat steps 1.1 to 1.6.
End
```

Fig. 4. Adaptive bully election algorithm.

Before ending the election, there are important points that should be tackled. These points relate to what happens to the other processes when they receive the messages: "Start Election" and "I'm the Coordinator". When a process receives the message: "Start Election", it starts comparing the EV and the received ID (NID):

1. If Node ID is 0 or less than the EV, then do not return a message.
2. If Node ID is higher than the EV, then update the value of the EV and return a message to the sender which contains the value of (NID).

4.2 Notations and Definitions

Before discussing the cost model and its related equations, it is necessary to clarify the notations and the definitions used throughout this paper as shown in Table 1.

Table 1. Notations and definitions

Notation	Definition
n	number of processes
P_d	process that discover the failure
P_w	wining process
id	process identification
EV	election variable
HPI	highest identification
$NHPI$	next highest identification
N_{MP}	number of message passing
P_{HPI}	the process that has the highest priority identification
P_{NHPI}	process that has the next highest priority identification
l	constant latency
L	latency cost

4.3 Cost Model

HPI and NHPI Variables. For the best case, the number of messages passing that we need to complete the election operation in our proposed algorithm is calculated by:

$$N_{MP} = (n-1) * 2 \tag{1}$$

where n is the number of processes that discovers the failure. Where the process that discovers the failure has a higher (id) number.

For the worse case, when the process that discovers failure has not the highest (id) number and there is more than one process discover the failure. Here, we will have two equations as follows:

When a process Pd discovers a failure, then the leader election starts:

1. Process Pd sends its id to all processes to compare it with their ids. If $Pd > Pq$, do not send your id. It needs $n - 1$ operations.
2. If $Pd < Pq$, then return a message of your id. It needs $n - Pd$.
3. When the process Pd receives the messages, the following steps take place:
   - Compare the received ids.
   - Store the highest two ids in two variables (HPI, $NHPI$).
4. Process Pd sends a message to the winning process $P - w$, which has the highest id, telling it that it is the coordinator.
5. Process Pw sends a message: "Ok" back to process Pq. It needs only 2 operations.
6. The winning process P_w sends to everyone: "I'm Coordinator".

Based on steps (1–6), N_{MP} will be calculated by Eq. (2):

$$N_{MP} = (n - 1) + [n - Pd] + 2 + (n - 1) \tag{2}$$

Equation (2) used when the election starts and there is no problem in the candidate coordinator.

However, when there is no response from Pd within ($T2$):

1. Process P_{NHPI} sends a message to process Pd that has the next highest priority id ($NHPI$) telling it that it is the coordinator now.
2. Process Pq sends a message: "Ok" back to process P_{NHPI}.
3. The winning process P_{NHPI} sends to everyone: "I'm the Coordinator". It needs $n - 1$ operations.

Based on (1–3), Eq. (3) will be used:

$$N_{MP} = 2 + (n - 1) \tag{3}$$

Latency. Another parameter used to compare our method is the latency (L). Latency is the time of sending a message from a source to the destination. However, the latency calculation in distributed system is difficult because of the different distances between devices. For this we assume the latency as stated in [21]. Equation (4) will be used to calculate the latency when using our algorithm:

$$L = [N_{MP} * l) \tag{4}$$

where N_{MP} is the number of message passing that calculated by Eqs. (1), (2) and (3) and l is a constant number (200 µs [4]).

The adaptive BA algorithm decreases the number of massages passing and latency. Four variables (VE, NID, HPI, NHPI) successfully decreased message passing complexity from O(n^2) to O(n). We can say when two processes discover failure, the election process is more flexible and safer.

5 Experimental Results and Discussions

In order to compare the performance of our algorithm with the other algorithms, we execute them in five test cases where the systems comprised 5, 10, 15, 20, and 25 nodes, respectively. We simulate our proposed algorithm using Java language on NetBeans editor. We used mesh topology to evaluate the cost model. Firstly, we will use Eqs. (1) and (2) mentioned above. We use Eq. (1) when the number of processes is equal to n. We assumed that the process $n - 1$ discovered the failure, which means that there is no process higher than it. Secondly, we use Eq. (2) when there are processes higher than the process that discovered the failure. The number of messages and latency is presented in Table 2.

Table 2. Number of passing messages and latency of the ABA algorithm.

No. of processes	Eqs. (1), (2)		Eq. (3)	
	Latency (μs)	Number of messages	Latency	Number of messages
5	1600	8	1200	6
10	5200	26	2200	11
15	8200	41	3200	16
20	11200	56	4200	21
25	14200	71	5200	26

As shown in Table 2, we observed that Eq. (3) produces better results compared with the results of Eqs. (1) and (2). In addition, when we compare our ABA with the BA algorithm [2] and Modified BA algorithm [8], it produces better results. The three algorithms are compared based on Messages passing and the results are shown in Table 3.

Table 3. Number of messages of the three algorithms.

No. of processes	BA	MBA	ABA	
			Eqs. (1), (2)	Eq. (3)
5	8	13	8	6
10	69	28	26	11
15	209	43	41	16
20	424	58	56	21
25	804	73	71	26

As shown in Table 3, it can be said that our method is better than Bully algorithm [2] and modified Bully algorithm [8] when there is no failure during the algorithm execution. That is because the number of passing messages in our method is less as clearly shown in Fig. 5.

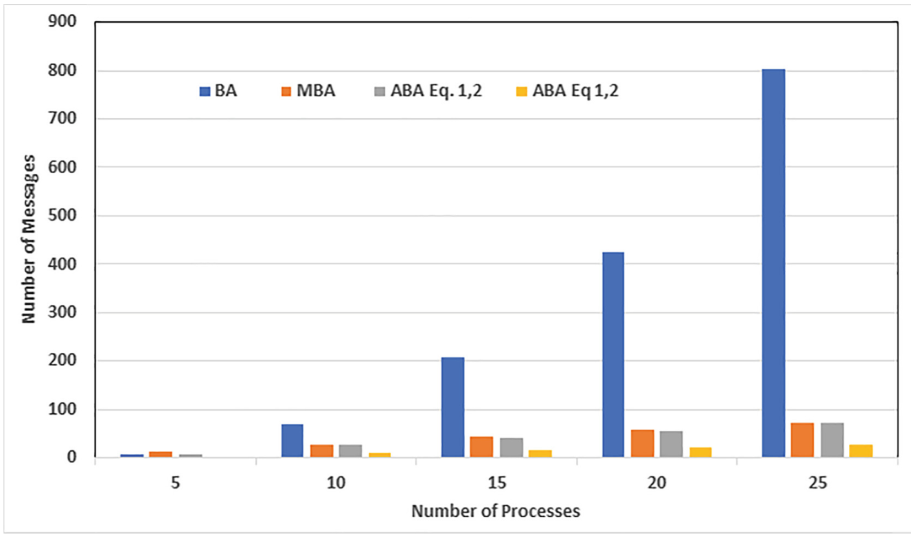


Fig. 5. Comparison between three algorithms.

As clearly shown in Table 3 and Fig. 5, it can be observed that our method is better than original Bully algorithm [2] and modified Bully algorithm [8] when repeating the leader election operation which occurs when the candidate coordinator fails too.

Latency. Another parameter compared in our work is latency. As shown in Tables 2 and 3, we created Table 4 and Fig. 6. Which contains the latency of the three algorithms.

Table 4. Latency (µs) of the three algorithms.

No. of processes	BA	MBA	ABA	
			Eqs. (1), (2)	Eq. (3)
5	1600	2600	1600	1200
10	13800	5800	5200	2200
15	41800	8600	8200	3200
20	84800	11600	11200	4200
25	160800	14600	14200	5200

As shown in Table 4, it can be observed that our method has a higher speed than the original Bully algorithm and modified Bully algorithm. When there is no failure during the algorithm execution it is safer. Overall, our experimental result shows that in the proposed algorithm, the number of messages and latency are very less as compared to the previous algorithms.

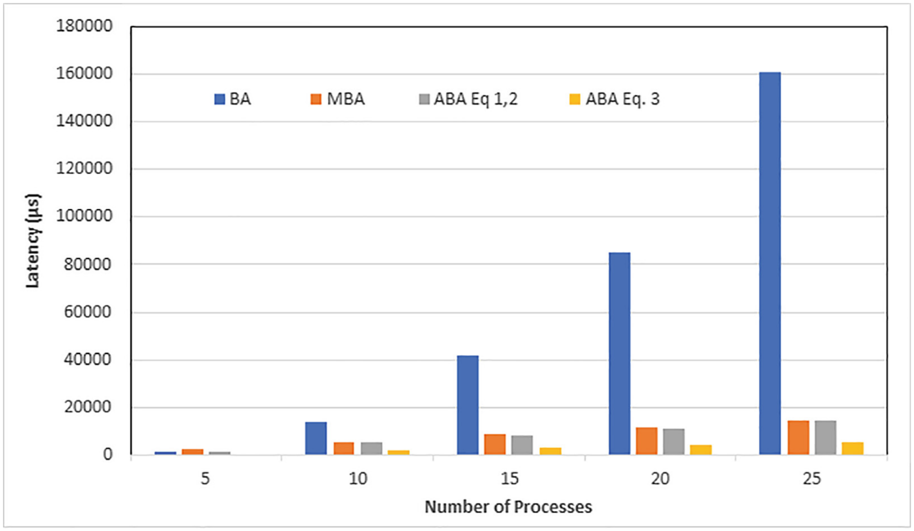


Fig. 6. Latency (µs) of the three algorithms.

6 Conclusion

In this paper, we successfully proposed ABA algorithm. Our ABA is better and more effective than BA algorithm and modified BA algorithm. It decreased the numbers of passing messages. Moreover, our ABA algorithm is safe (reliable) if failure for candidate coordinator happened. During the implementation of the algorithm, if errors occur for candidate coordinator, our method leads to stopping the repetition of algorithm implementation when failed in starting. In addition, four variables (VE, NID, HPI, NHPI) successfully decreased message passing complexity from $O(n^2)$ to $O(n)$.

Acknowledgment. The authors would like to thank everyone who provided valuable suggestions and support to improve the content of the paper. This research work is partial financially supported by the Malaysian Ministry of Education under the Fundamental Research Grant Scheme (FRGS/1/2018/STG06/UPM/01/2).

References

1. Coulouris, G., Dollimore, J., Kindberg, T., Blair, G.: Distributed System Concept and Design, 5th edn. Addison Wesley, USA (2011)
2. Garcia-Molina, H.: Elections in a Distributed Computing System. IEEE Trans. Comput. **100**(1), 48–59 (1982)
3. van Steen, M., Tanenbaum, A.S.: Distributed Systems. 3rd edn. CreateSpace Independent Publishing Platform (2017)
4. Murshed, Md.G., Allen, A.R.: Enhanced bully algorithm for leader node election in synchronous distributed systems. J. Comput. **1**(1), 3–23 (2012)

5. Distributed Algorithms in NOSQL Databases. https://highlyscalable.wordpress.com/2012/09/18/distributed-algorithms-in-nosql-databases/
6. Méndez, M., Tinetti, F.G., Duran, A.M., Obon, D.A., Bartolome, N.G.: Distributed algorithms on IoT devices: bully leader election. In: Proceeding of the International Conference on Computational Science and Computational Intelligence (CSCI), pp. 1351–1355, December 2017
7. Chhabra, S., Tyagi, G., Mundra, A., Rakesh, N.: Location based coordinator election algorithm in distributed environment. In: Proceedings of the International Conference on Computer and Computational Sciences (ICCCS), Noida, pp. 183–188 (2015)
8. Soundarabai, P.B., Sahai, R., Thriveni, J., Venugopal, K.R., Patnaik, L.M.: Improved bully election algorithm for distributed systems. Int. J. Inform. Process. **7**(4), 43–54 (2013)
9. Kordafshari, M.S., Gholipour, M., Mosakhani, M., Haghighat, A.T., Dehghan, M.: Modified bully election algorithm in distributed systems. In: Proceedings of the 9th WSEAS International Conference on Computers, Greece, pp. 1–6 (2005)
10. Zargarnataj, M.: New election algorithm based on assistant in distributed systems. In: ACS International Conference on Computer Systems and Applications (AICCSA), Amman, pp. 324–331 (2007)
11. Sepehri, M., Goodarzi, M.: Leader election algorithm using heap structure. In: 12th WSEAS International Conference on Computers, Heraklion, pp. 668–672 (2008)
12. Rahman, M.M., Nahar, A.: Modified bully algorithm using election commission. MASAUM J. Comput. (MJC) **1**(3), 439–446 (2009)
13. EffatParvar, M.R., Yazdani, N., EffatParvar, M., Dadlani, A., Khonsari, A.: Improved algorithms for leader election in distributed systems. In: Proceedings of the 2nd International Conference on Computer Engineering and Technology, Chengdu, China (2010)
14. Basu, S.: An efficient approach of election algorithm in distributed systems. Indian J. Comput. Sci. Eng. (IJCSE) **2**(1), 16–21 (2011)
15. Park, S.-H.: A stable election protocol based on an unreliable failure detector in distributed systems. In: Proceedings of the 8th International Conference on Information Technology: New Generations, pp. 979–984. IEEE Computer Society (2011)
16. Kordafshari, M M.S., Gholipour, M., Rahmani, A.M., Jahanshahi, M.: A New Approach for Election Algorithm in Distributed System, pp. 70–74 (2009)
17. Mishra, B., Singh, N., Singh, R.: Master-slave group based model for co-ordinator selection, an improvement of bully algorithm. In: Proceedings of the International Conference on Parallel, Distributed and Grid Computing, Solan, India, pp. 457–460 (2014)
18. Garg, D., Suman, N.: Study of assorted election algorithms in distributed operating system. In: Proceedings of the National Conference on Innovative Trends in Computer Science Engineering, pp. 132–134 (2015)
19. Sathesh, B.M.: Optimized bully algorithm. Int. J. Comput. Appl. **121**(18), 24–27 (2015)
20. Arghavani, A., Ahmadi, A.E., Haghighat, A.T.: Improved bully election algorithm in distributed systems. In: Proceedings of the 5th International Conference on Information Technology & Multimedia, pp. 14–16 (2011)
21. Fredrickson, G.N., Lynch, N.A.: Electing a leader in asynchronous ring. J. ACM **34**(1), 98–115 (1987)