Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Technology 6 (2012) 57 – 66

Procedia Technology

2nd International Conference on Communication, Computing & Security

An optimal novel Byzantine agreement protocol (ONBAP) for heterogeneous distributed database processing systems

Dharavath Ramesh[a,*] and Chiranjeev Kumar [a]

[a]*Indian School of Mines (ISM), Dhanbad, Jharkhand 826004, India*

Abstract

Reliability deserves utmost position in processing the transaction effectively in distributed environment. Conservatively, the problems of consensus, Byzantine Agreement, and interactive consistency are studied in a fully connected network with multi nodes (processors) in malicious failure only. Such problematic instances are re-examined with the assumption of malicious faults on the side of both the nodes. The proposed ONBAP protocol use the minimum number of message exchanges and can endure the maximum number of allowable faulty components to make each fault-free processor reach a common agreement for the cases of processor failure to reach a goal effectively.

Key words: Fault tolerance, Distributed database systems, Agreement protocols

1. Introduction

In general, in a distributed processing system each node (processor) communicates with the others. Under such difficulties, providing some protocols defined by Colon, Siu, Yan, and Wang to help all fault-free nodes or sites reach an agreement and then do some related activities, even if some instances caused by faulty nodes exist, becomes necessary. The Byzantine agreement (BA) problem was extensively studied by Lamport in 1982. This problem depicts that each fault-free nodes can reach a common value,

* Dharavath Ramesh. Tel.: 91-326-2235795.
E-mail address: ramesh.d.cse@ismdhanbad.ac.in.

doi:10.1016/j.protcy.2012.10.008

and that, in an n-node distributed environment, at most [(n−1)/3] nodes are insane. The Byzantine Agreement problem has been defined by Lamport as follows.

1) There are n nodes, and at most [(n−1)/3] nodes that could fail without breaking down a workable network.
2) The nodes communicate with each other through message exchange (one with other) in a fully connected area.
3) The message's sender (called root node) is always identifiable by the receiver (participated nodes).
4) A random node is chosen as a source (root node), and its initial value, *v(SN)* is broadcasted to other nodes (participated nodes) and itself to execute the protocol

1.1 A Simple application approach

A distributed database comprises a set of databases stored at multiple sites or nodes that work together and appear as a single database to the user (Distributed scenario). Consider an example of making reservation through node n_1 (reservation site or node) to other nodes n_2 ($bank_1$) and n_3 ($bank_2$).

Scenario 1: If n_1 is fault-free then it completes the agreement algorithm by having the conversation with n_2 and n_3 and books the ticket successfully within the stipulated time. Here the conversation will occur by the exchange of messages between all the three nodes.

Scenario 2: If n_1 is a fault one then it completes the algorithm by taking much time and it diverse the nodes n_2 and n_3, i.e. amount is deducted from the bank accounts but the ticket will not be booked.

This is one of the difficult instances to make the case of agreement protocol possible.

```
Debit (amount)                          ------- Request made by node n1
if(balance>=amount)                     -------- Processed at node n2
  {
    balance=balance-amount;             ---------Operation status at n2
   return ("success", balance);         ---------Reverted to node n1
    }
else
  {
    return("failure", balance);         ---------- Reverted to node n1
    }
end;

----- Same request is processed at node n3 simultaneously
```

Table 1. A simple Fault-tolerant transaction approach

Based on the assumptions above, various agreement protocols by Lamport, Wang, Pease, Chan, Yan, and Liang for the BA problem have been developed to satisfy the following requirements.

- (Agreement): All fault-free nodes agree on a common value.
- (Validity): If the source node is fault-free, then the common value should be the source's initial value.

- (Message passing1): If all fault-free processors agree on a common value then they have to complete the algorithm with minimum number of required rounds.
- (Message passing2): If some of the nodes are fault-free and some are faulty then they have to complete the algorithm with sufficient number of rounds.

While there can be no other option that the proposed protocols of Pease, Wang, Lamport, and Fischer can allow all correct nodes to reach an agreement value with [(n-1)/3 + 1] rounds of message exchange, those protocols are somewhat inefficient due to the large number of messages resulting in a large protocol overhead.

Fig 1. (a) The Case where source node is faulty (b) the case where participant node is faulty

Hence, the instance we must consider next is improving the computation efficiency. In other words, if all correct nodes can reach an agreement value within fewer rounds of message exchange, the processing time and communication time can all be reduced in a distributed processing system? As a result, the protocol ONBAP is proposed in this paper.

According to the previous results by Pease, Wang, Lamport, and Fischer, there are two important parameters to consider when solving the BA problem in a distributed environment. At first, there will be at least $n - [(n-1)/3]$ fault-free node in a distributed system, and these nodes will always send the received values to others honestly. Second, the total number of messages received from fault-free node is more than the number of messages received from faulty nodes (processors or sites). Hence, if these two features can be utilized properly, the proposed protocol ONBAP can be improved efficiently.

2. Common methodology

A heterogeneous distributed database comprises a set of databases stored at multiple nodes that work together and appear as a single database to the user (Distributed scenario). We consider a scenario of heterogeneous nodes and one of the nodes becomes as source to lead the entire message passing system.

In this proposal, a protocol called an optimal novel byzantine agreement protocol (ONBAP) is proposed to solve the traditional BA problem. The ONBAP protocol can compare and count the received values from the message exchange to discover the reliable nodes. Subsequently, the values received from insane nodes can be replaced by the values received from reliable nodes. In other words, the influences caused by the faulty nodes can be reduced after applying the replacing procedure. Finally, all correct nodes can take the simple majority value to get the agreement value. The majority value is decided at ELECT function.

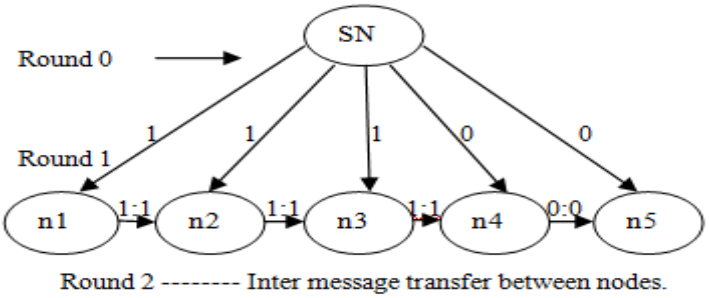


Fig 2. The case where Source node (RN) is faulty and passing the values between the nodes.

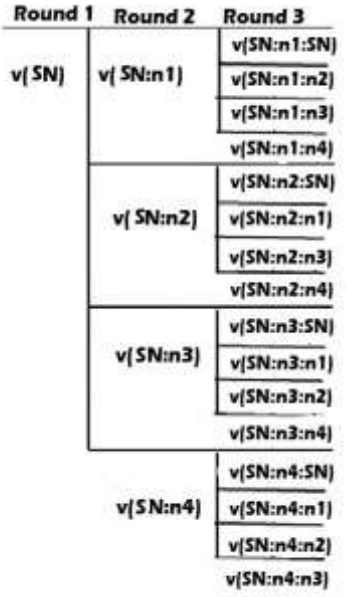
For an instance an n-node network requires [(n-1)/3] +1 rounds of message exchange to reach a common, and the message complexity is O ($n^{[(n-1)/3]}$). ONBAP only requires three rounds of message exchange and the complexity is O (n^2). In this way the proposed ONBAP protocol is said to be optimal why because the ONBAP is reducing the number of rounds up to the minimum level. In this study, we convey this protocol is to make all fault-free nodes reach an agreement. This is comparatively less than that of the traditional protocols proposed by Pease, Wang, Lamport, and Fischer.

2.1 The mg-tree used by the ONBAP protocol

During the exchange procedure, each node will use a tree structure to keep the messages received from others. Also, in each round of exchange, these messages are saved into the corresponding level of the tree. In this manuscript, this tree structure is called as mg-tree (a message gathering tree), which is similar to what Bar- Noy and Dolev's proposal. The concept of mg-tree is depicted in Fig. 3. Here, each fault free node maintains the received messages into its mg-tree during the execution of the ONBAP protocol. In the first round, the source node SN broadcasts its initial value v (SN) to the others and itself. Each node knows where the message comes from. Therefore, each fault-free node can identify the received message (here denoted as v(SN)) from the source and stores it in the root of its mg-tree. However, each processor cannot identify whether the source one is a correct node or faulty one. So each sane node requires additional rounds of message exchange to eliminate the faulty occurrences generated by a faulty source node. This gathering makes the time complexity of the ONBAP protocol O(n^2), why because the protocol completes the occurrence of faulty nodes in three rounds only. That's why the proposed protocol is said to be optimal one compared to the existing [Lamport, Wang, Pease, Chin]. During the message sending and exchange of values, ONBAP utilizes the portion of mg-tree.

Sender : n1		n2		n3		n4		n5	
Dest	Msg	Dest	Msg	Dest	Msg	Dest	Msg	Dest	Msg
n1	1	n1	1	n1	1	n1	0	n1	0
n2	1	n2	1	n2	1	n2	0	n2	0
n3	1	n3	1	n3	1	n3	0	n3	0
n4	1	n4	1	n4	1	n4	0	n4	0
n5	1	n5	1	n5	1	n5	0	n5	0

Table 2. Messages sent by all participated nodes in round1



In the second round, each participated node broadcasts the root's value of its mg-tree to all other nodes. Subsequently, each participant receives the values from other nodes and keeps the received values into the second level of its mg-tree. If the value is received from the node n1, the vertex name will be v(SN: n1). The method of the third round is the same as the second round, and the received values must be stored into the third level of the mg-tree. If the value is received from the participated node n2, the vertex name will be v (SN: n1:n2). Here, the vertices with repeated names of the root and other participated nodes (v(SN:SN), v(SN:n1:n1), v(SN:n2:n2), v(SN:n3:n3), v(SN:n4:n4),and v(SN:n5:n5)) are eliminated from the mg-tree to eliminate the cyclical instances from the faulty nodes. The cyclical instances are caused by the messages from faulty participant nodes, and they may be stored repeatedly in the mg-tree, resulting in an incorrect value caused by taking a simple majority. The configuration of each and every participated node is shown in the table 2. This example, using a sample of 6 nodes (One is the Source one and other 5 are participating n odes), depicts how to execute the exchange process. We assume that, in Round 0 the source node SN, which is faulty one told n_1, n_2, and n_3 that the command value was 1, and told n_4 and n_5 that the command value was 0. In round 1, the following messages would be sent:

While sending, each node distributes its message along with identification (ID) number. Assume that the id's of the participated nodes are {(n_1=12), (n_2=13), (n_3=14), (n_4=15), (n_5=16)}.

The five messages, that node n_1 received in round 1 were {1, 12}, {1, 13}, {1, 14}, {1, 15}, and {1, 16}. According to the earlier definition, n_1 will append its process ID to the path and forward each resulting message to all other participated nodes. The possible messages it could store in round 2 are {1, 12:12}, {1, 12:13}, {1, 12:14}, {0, 12:15}, and {0, 12:16}. The first message, {1, 12:12} contains a cyclical occurrence in the path value, so it is tossed out, leaving four messages to be stored at that tree level.

2.2 The improvement of the ONBAP (A comparison)

In the traditional BA protocol proposed by Lamport, all correct nodes need to run the algorithm OM ([(n−1)/3]) recursively to solve the BA problem. This means that all nodes require [(n − 1)/3] + 1 rounds of message exchange for collecting messages. Finally, the agreement can be reached by taking the majority value of the received values. The proposed protocol is compared with Yan and Wang in order to eliminate the deficiencies. Yan followed the method of Lamport to execute the scenario of BA which requires 2 * [(n-1)/3] + 3 rounds for agreement and [t_n+1] rounds for finding the faulty occurrences. But the proposed one is found to be less in all rounds.

	Required Rounds (for agreement)	Required Rounds (finding Faulty nodes)	Idea
Kuo-Qin Yan	2 * [(n-1)/3] + 3	$[t_n + 1]$	By product of the alg.
ONBAP	3	3	Comparison

Table 3. The comparison between the ONBAP protocol and others

2.3 The majority of the ELECT function utilized by the ONBAP

The ELECT (α) function is applied from the third level to root of mg-tree. After applying this, all the correct nodes can find an exact agreement value, and it is reached. Here, the majority of the ELECT function is the value which is sent by the nodes at root and child levels of the mg-tree up to maximum times, i.e. maximum number of nodes have sent the value 1 to the remaining.

The Function ELECT (α)=

1. V(α), if α is a leaf
2. The max value in the set of {ELECT (α_i) $1 \leq i \leq n$, and vertex α is a child of vertex α}, if max value exists.
3. A default value $ is chosen, otherwise.

2.4 Required rounds for the ONBAP protocol is three

The protocols of Pease, Wang, Lamport, Fischer, and Dolev needs $[(n - 1)/3] + 1$ rounds of message exchange for gathering messages. Under such protocols, each correct node always requires $[(n - 1)/3] + 1$ rounds of message exchange to reach agreement even if all participated nodes are correct. Moreover, the methods of Pease, Wang, and Dolev can make all correct nodes reach agreement based on only exchanging message with others continuously without comparing the characteristic values sent by the n-[(n-1)/3] fault-free nodes or revising the characteristic values sent by the faulty nodes. Hence, the previous protocols always require $[(n - 1)/3] + 1$ rounds of message exchange, and will result in a large overhead in the network environment. If one node is the correct one, then it will send a correct value to others. Since there are at least $n - [(n - 1)/3]$ correct nodes in the distributed environment, this also mean that n-[(n−1)/3] fault-free nodes must agree on this value and send this value to others again. Thus, after taking the majority value of these values in $(i + 1)^{th}$ $(1 \leq i \leq [(n - 1)/3])$ level of the mg-tree, the final agreement value will equal the values in the i^{th} level of the mg-tree if the values are sent by a correct node. In this paper, the proposed ONBAP protocol only requires three rounds of message exchange and it compares the values which it received.

2.5 Number of allowable processors

According to the existing protocols proposed by Pease, Wang, Lamport, Fischer, and Dolev, the liberal number of faulty nodes is $[(n - 1)/3]$ for the agreement. When the total number of faulty nodes exceeds the count, the correct nodes cannot reach an agreement. For example, if there are 10 processors in the system, then the number of allowable faulty nodes in our proposed protocol is $[(n - 1)/3] = [(10 - 1)/3] = 3$. The more nodes we have, the greater the efficiency of the proposed protocol. Because the traditional protocols of Pease, Wang, Lamport, Fischer, and Dolev always require $[(n - 1)/3] + 1$ rounds of message exchange, the required number of rounds of message exchange grows when the number of nodes becomes more. So, the ONBAP protocol is more efficient than those in previous works proposed by Shostak, Yan, Lamport, and Lynch in solving the agreement problem with three rounds of message exchange.

3. The Proposed ONBAP protocol

In this, the ONBAP is introduced to solve the BA in an n-node system. Our proposed protocol can bear [(n-1)/3] malicious faulty nodes, and only requires three rounds of messages to reach an agreement. There are two phases in the ONBAP protocol:

- Message exchange phase
- Decision making phase

The responsibility of the message exchange phase is to collect and store the values in the mg-tree at each round. Upon message exchange, the decision making phase is invoked.

The ONBAP protocol:

Preprocessing:

If the condition of the network is given, then

1) If a node fails, then t_n , is set to [(n - 1)/3], else $t_n = 0$; else
   if the link is frail, then X = 1, else X = 0;
2) $S = t_n + X + 1$; (the number of required rounds); otherwise, S is set to [(n - 1)/3] + 2.

Message Exchange Phase:

r = 1 do:

1) The source node broadcasts its initial value v, to each other and itself; and each participant (participated node) stores its received value in the root of its mg-tree.
2) If r = S, then GOTO Decision making phase

For r=2,….S do:

1) Each node broadcasts the values at level r -1 to the others and itself, and stores the values to the corresponding vertices at level r of the mg-tree.
2) Function MAX is used to each vertex at level r – 1 for each node's mg-tree and the function value is broadcasted to others and itself.

Decision Making Phase:

1) Each node's mg-tree is reorganized into a corresponding IC-tree. (collection)
2) Function ELECT is applied to root s of each node's IC-tree so that the common value (α) is obtained; and then halts the protocol. At that time, each correct node shall be agreed on a common value ELECT (α).After completing the protocol; every node will agree on a common value ELECT (α).

3.1 Executing the ONBAP (Working Procedure)

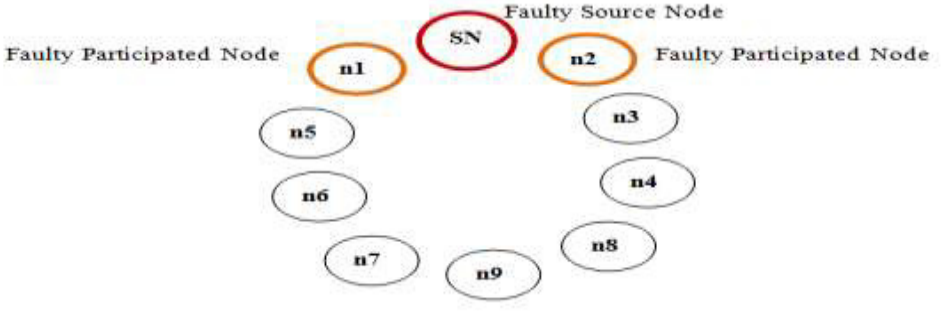


Fig 4. The 10 – node environment

In this, two examples are shown, in Figs. 4 and 5, to illustrate how the ONBAP protocol can make each correct node reach an agreement. In the first, ten nodes in a distributed environment only require three rounds of message exchange to reach an agreement even when the source one is faulty (shown in fig 2). (The previous tasks require four rounds $[(n-1)/3] + 1 = [(10 - 1)/3] + 1 = 4$) of exchange under the ten-processor environment.) We assume that the source node SN is the faulty one, which sends different values to the other nodes. Node n_1 and n_2 are also assumed to be faulty nodes.

The environment is shown in Fig. 4. Furthermore, to check the condition of the ONBAP protocol, we design a scenario (the numbers of and the behaviour of the faulty node is shown in Fig. 5(a). At the beginning, the source node SN broadcasts its initial value to all nodes in the first round of the message exchange phase. Unfortunately, source node SN is a faulty one; it sends different values 0, 1, 0, 1, 0, 1, 0 nodes n3, n4, n5, n6, n7, n8, and n9.

	n3	n4	n5	n6	n7	n8	n9
SN	0	1	0	1	0	1	0
n1	1	0	1	0	1	1	1
n2	0	0	0	1	1	1	1

Fig 5(a). The behaviour of faulty nodes

n3	v(SN)=0
n4	v(SN)=1
n5	v(SN)=0
n6	v(SN)=1
n7	v(SN)=0
n8	v(SN)=1
n9	v(SN)=0

Fig 5(b). The value stored at each root node

Here, each fault-free node stores the received value v (SN, n1, and n2) in the root of its mg-tree in the round 0, as shown in Fig. 5(b). However, the results of faulty node do not need to be discussed; thus this only shows the results of reliable nodes. Subsequently, each node exchanges the value from the first round of the message exchange phase with all nodes in the round1 of the message exchange phase. Similarly, the received values are loaded in the second level of their mg-tree and the name for each vertex will be given following the concept given above. For example, vertex v(XY) means that this value is received from node Y and the value was first sent by node X. The overall results for the round1 of message exchange are shown in Fig. 5(c).

In $round_3$ exchange phase, each node exchanges the received values from the $round_2$ of the message exchange phase with all nodes and keeps the received values in the third level of tree. Actually, the principles of forming the third level of the mg-tree are the same as for the round2 of message exchange.

In the coming phase, the decision making phase, each node must first recognize which nodes are reliable. For instance, the nodes SN, n2, n3, n4, n5, n6, n7, n8 and n9 can go for examining when the following conditions are satisfied:

1. $v(SN:n_3) = max_3 (SN:n_3)=0$
2. $\#max_3 (SN:n_3) = 8 >= (n-[(n-1)/3]-1) = 6$ (where # is the total number of values that are equal to $(SN:n_3)$ for each sub-tree which is expanded from the vertex $v(SN:n_3)$)
3. $v(SN:n_3:X) = max_3(SN:n_3)$ { such as , $v(SN:n_3:SN)$,v(SN:n3:n2),v(SN:n3:n4),v(SN:n3:n5), v(SN:n3:n6), $v(SN:n_3:n_7)$, $v(SN:n_3:n_8)$, and $v(SN:n_3:n_9) = max_3(SN:n_3) = 0$}.

	Node = n3		Node = n4		Node = n5		Node = n6
	v(SN:n1)=1		v(SN:n1)=0		v(SN:n1)=1		v(SN:n1)=0
	v(SN:n1)=0		v(SN:n1)=0		v(SN:n1)=0		v(SN:n1)=1
	v(SN:n1)=0		v(SN:n1)=0		v(SN:n1)=0		v(SN:n1)=0
v(SN)=0	v(SN:n1)=1	v(SN)=1	v(SN:n1)=1	v(SN)=0	v(SN:n1)=1	v(SN)=1	v(SN:n1)=1
	v(SN:n1)=0		v(SN:n1)=0		v(SN:n1)=0		v(SN:n1)=0
	v(SN:n1)=1		v(SN:n1)=1		v(SN:n1)=1		v(SN:n1)=1
	v(SN:n1)=0		v(SN:n1)=0		v(SN:n1)=0		v(SN:n1)=0
	v(SN:n1)=1		v(SN:n1)=1		v(SN:n1)=1		v(SN:n1)=1
	v(SN:n1)=0		v(SN:n1)=0		v(SN:n1)=0		v(SN:n1)=0

	Node = n7		Node = n8		Node = n9
	v(SN:n1)=1		v(SN:n1)=1		v(SN:n1)=1
	v(SN:n1)=1		v(SN:n1)=1		v(SN:n1)=1
	v(SN:n1)=0		v(SN:n1)=0		v(SN:n1)=0
	v(SN:n1)=1		v(SN:n1)=1		v(SN:n1)=1
v(SN)=0	v(SN:n1)=0	v(SN)=1	v(SN:n1)=0	v(SN)=0	v(SN:n1)=0
	v(SN:n1)=1		v(SN:n1)=1		v(SN:n1)=1
	v(SN:n1)=0		v(SN:n1)=0		v(SN:n1)=0
	v(SN:n1)=1		v(SN:n1)=1		v(SN:n1)=1
	v(SN:n1)=0		v(SN:n1)=0		v(SN:n1)=0

Fig 5(c). The mg-tree of nodes n3, n4.n5, n6, n7, n8, and n9

3.2 Number of allowable faulty nodes is [(n-1)/3] for ONBAP

If the number of faulty nodes is greater than (n/2), then all may send different values to each node. Correct nodes cannot have common vertices. Thus, one cannot be sure that all correct nodes can reach agreement. If the total number of faulty nodes is equal to (n/2), and n is an even number, then the number of 0s and 1s in the second level may be the same after applying the ELECT function. Under such conditions, all correct nodes cannot get a common value. According to the assumptions and limitations of the BA problem, the number of faulty nodes cannot exceed $[(n - 1)/3]$. These are identical to our constraints. So, the total number of allowable faulty nodes is $((n-1)/3)$, i, e. The optimality is proven.

3.3 Message complexity is O (n^2)

In the round 0 of the exchange phase, the source node (SN) sends its value to the other nodes. Hence, one message must be generated. In the round 1 of the exchange phase, all nodes must send the received values in $round_0$ of message exchange to others, and n messages must be generated. In the $round_2$, $(n*n)$ messages must be generated. So, the total number of messages to be generated during the execution of the ONBAP protocol is ON $(1 + n + n * n)$. After the acceptance testing we found that, the message complexity for the proposed protocol is O (n^2). We considered and tested a 7 node methodology to find out the majority function which is optimal according to the mg-tree structure.

3.4 Optimality of ONBAP Protocol

Let $M^1 \subseteq M$ be the set of receiving nodes which receives more than three messages in each round and the sender which receives more than one message in each round of the ONBAP. Let λ be either 1 or 0, according to the sender is in M or not. Then the total number of messages in each round runs of N at least

$|R0| + |R1| + 2\,(|M| - |M^1|) + 3\,(|M^1|) - \lambda = (n - 1 + \lambda - |M|) + 2\,(|M - M^1|) + 3\,(|M^1|) - \lambda = n - 1 + |M| + |M^1|$.

Since N has optimal message complexity with the proposed, we have $n + t - 1 > n - 1 + |M| + |M^1|$, which implies that $|M| + |M^1| \leq t$., where t is the stipulated time consistency for receiving and sending the messages. With this consideration the proposed protocol is having optimality according to the majority of the ELECT function of each round.

4. Conclusion

To ensure that all correct nodes reach agreement and do required actions in distributed computing are an important research consideration. In previous literatures of Pease, Wang, Lamport, Fischer, Reischuk, and Dolev, each fault-free node could reach an agreement and tolerate $[(n - 1)/3]$ faulty occurrences, using $[(n - 1)/3] + 1$ rounds of message exchange. However, these protocols only apply the majority function to eliminate the influences caused by the $[(n - 1)/3]$ faulty nodes by gathering messages from a large number of round levels. In this study, we revisit the characteristics that there will always be $n - [(n - 1)/3]$ correct nodes in a distributed system and that these nodes will always send the received values correctly. Furthermore, the values sent by the fault-free nodes will always be in the majority. Based on these two, we propose a novel agreement protocol, which we termed the ONBAP protocol. This protocol requires only three rounds of message exchange to collect the values sent by the participated fault-free nodes for finding the correct one, and then it uses the majority (max function) values of the correct nodes to replace the values sent by the faulty nodes.

References

F.C. Colon Osorio, Using Byzantine agreement in the design of IPS systems, in: *IEEE International Conference on Performance, Computing, and Communications Conference*, 2007, pp. 528–537.

H.S. Siu, Y.H. Chin, W.P. Yang, Byzantine agreement in the presence of mixed faults on processors and links, *IEEE Transactions on Parallel and Distributed Systems* 9 (4) (1998) 980–986.

K.Q. Yan, S.C. Wan, Grouping Byzantine agreement, *Computer Standard & Interfaces* 28 (1) (2005) 75–92.

K.Q. Yan, S.C. Wang, M.L. Chiang, Optimal agreement in a scale-free network environment, *Informatica International Journal* 17 (1) (2006) 137–150.

L. Lamport, R. Shostak, M. Pease, the Byzantine generals problem, *ACM Transactions on Programming Languages and Systems* 4 (3) (1982) 382–401.

L. Lamport, P. Melliar-Smith, Byzantine clock synchronization, in: *ACM 3rd PODC Conf. Proc*., 1984, pp.10–16.

S.C. Wang, S.C. Liang, K.Q. Yan, G.Y. Zheng, Efficient malicious agreement in a virtual subnet network, in: *The Second International Conference on Availability, Reliability and Security, ARES2007,* Vienna, April, 2007, pp. 10–13.M.

Pease, R. Shostak, L. Lamport, Reaching agreement in presence of faults, *Journal of the ACM* 27 (2) (1980) 228–234.

S.C. Wang, Y.H. Chin, K.Q. Yan, Byzantine agreement in a generalized connected network, *IEEE Transactions on Parallel and Distributed Systems* 6 (4) (1995) 420–427.

S.C. Wang, M.L. Chiang, K.Q. Yan, Streets of consensus under unknown unreliable network, "*ACM Operating Systems Review"* 39 (4) (2005) 80–96.

M. Fischer, N. Lynch, A lower bound for the assure interactive consistency, *Information Processing Letters* 14 (4) (1982) 183–186.

A. Bar-Noy, D. Dolev, C. D work, R. Strong, Shifting gears: changing algorithms on the fly to expedite Byzantine agreement, in: *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, 1987, pp. 42–51.

D. Dolev, R. Reischuk, H.R. Strong, Early stopping in Byzantine agreement, *Journal of the ACM* 37 (1990) 720–741.

D. Dolev, M. Fischer, R. Fowler, N. Lynch, R. Strong, An efficient algorithm for Byzantine agreement without authentication, *Information and Control* 52 (3) (1982) 257–274.