

Comparative Evaluation of Approximate Byzantine Vector Consensus Algorithms

Shiyang Cheng, Kyle Sung
shiyangcheng@utexas.edu, kyle@utexas.edu

Abstract—This is an abstract.

I. INTRODUCTION

This paper describe two approximate multidimension consensus algorithms in distributed system. These algorithms are different from traditional consensus algorithms. They are meant to resolve the Byzantine problem which distributed system contains arbitrary failures. In traditional Byzantine problem: there are n processes in the system, several of them are faulty processes which can be considered generate any possible output in the system. Each non-faulty process will propose one value and then they will get one consensus value which need to meet several conditions:

- Termination: Every correct process eventually delivers some message
- Agreement: If a correct process delivers value m , then all correct processes deliver m
- Nontriviality: Both values should be possible outcomes. This property eliminates the protocols that returns a fixed value independent of the initial input

In multidimensional system, all the processes will propose one vector of values, and all non-faulty processes will get consensus on the n -dimensional value. The multidimensional input which is d -dimensional vector can be considered as a point in d -dimensional Euclidean space with $d > 0$. In the multidimensional Byzantine Consensus problem, the out come of each process should also be identical. And the output value need to be in the convex hull of the non-faulty processes' input in the d -dimensional Euclidean space.

To solve this problem, researchers also propose another problem named Byzantine Approximate Agreement problem. This problem also defined the out come of non-faulty processes will be in the convex hull. The outputs should be within the Euclidean distance ϵ of each other.

In these problem, the algorithms defined in a model include following property:

- All message will be eventually delivered
- Any two processes is connected to each other
- The communication channel is FIFO channel
- The processes can identify the sender by the sender ID in the message

From Vaidya and Garg's observation, simply performing scalar consensus on each dimension of the input vectors independently does not solve the vector consensus problem. In particular, even if validity condition for scalar consensus is

satisfied for each dimension of the vector separately, the above validity condition of vector consensus may not necessarily be satisfied. For instance, suppose that there are four processes, with one faulty process[7].

A. Multidimensional Byzantine Consensus

For synchronous system, the algorithms will run in round by round. In each round, processes will send messages and receive messages which were sent in this round.

A protocol solving the Multidimensional Byzantine Consensus problem need to satisfy following conditions[5]:

- Agreement. The output vector at all the non-faulty processes must be identical.
- Validity. The output vector at all non-faulty processes must be in the convex hull of the non-faulty inputs.
- Termination. Each non-faulty process must terminate within a finite amount of time.

This is known that $n > 3f$ is necessary and sufficient to solve the scalar consensus, under the condition that the communication model is a complete graph.

B. Multidimensional Byzantine Approximate Agreement

In asynchronous systems, the message deliver time is not guaranteed. The message may take unbound time to deliver. Also, there is not disjoint round in the algorithms. It is not possible to identify a process is faulty or slow[2]. And it is well-known that asynchronous scalar consensus is impossible in the presence of even a single crash failure[4]. Here we discuss the algorithms are also under the same condition, but the input and output switched to a vector values.

A protocol satisfying these conditions could be considered solving the Multidimensional Byzantine Approximate Agreement problem:

- Agreement. The output vectors of non-faulty processes should be within Euclidean distance $\epsilon > 0$, a constant defined a priori.
- Validity. The output vector at all non-faulty processors must be inside the convex hull of the input inputs.
- Termination. Each non-faulty process must terminate within a finite amount of time.

II. ASYNCHRONOUS COMMUNICATION PRIMITIVES

The multidimensional algorithms use two important communication primitives: the reliable broadcast and the witness technique.

A. Reliable Broadcast

The reliable broadcast technique avoids the situation where Byzantine processes convey different contents to different processes in a single round of communication. And the original paper is from Srikanth and Toueg[6] and Bracha[3].

The message sent by processes contains sender's identification. So one message contains the sender ID, receiver ID and content. And the reliable broadcast technique has the following properties:

- Non-faulty integrity: If a non-faulty processes p never reliably broadcasts one specific message, no other non-faulty process will ever receive it
- Non-faulty liveness: If a non-faulty process p does reliably broadcasts one message m , all other non-faulty processes eventually receive message m
- Global uniqueness: If two non-faulty processes reliably receive message m and m' , the messages are equal, even when the sender p is Byzantine.
- For two non-faulty processes p_1 and p_2 , if p_1 reliably receive m , p_2 also reliably receive m , even when the sender p is Byzantine.

The algorithms are following:

Algorithm 1 p .RBSend((p, r, c))

send(p, r, c) to all processes

Algorithm 2 p .RBEcho()

```

upon rcv( $q, r, c$ ) from  $q$  do
if never sent ( $p, qr\{echo\}, .$ ) then
    send( $(p, qr\{echo\}), c$ ) to all processes
end if

upon rcv( $., qr\{echo\}, c$ ) from  $\geq n - f$  processes do
if never sent ( $p, qr\{ready\}, .$ ) then
    send( $(p, qr\{ready\}), c$ ) to all processes
end if

upon rcv( $., qr\{ready\}, c$ ) from  $\geq f + 1$  processes do
if never sent ( $p, qr\{ready\}, .$ ) then
    send( $(p, qr\{ready\}), c$ ) to all processes
end if

```

Algorithm 3 p .RBRecv((q, r, c))

```

rcv( $., qr\{ready\}, c$ ) from  $n - f$  processes
return ( $q, r, c$ )

```

B. Witness Technique

The witness technique provide a method which can make two non-faulty processes get suitable overlaps values. This method is originally introduced by Abraham[1]. The method can make sure that non-faulty processes have $n - f$ common values, which is essential for our correctness and optimality

arguments. This witness technique will only wait for messages certain to be delivered.

The algorithms are shown:

Algorithm 4 p .RBReceiveWitness(r)

```

 $Val, Rep, Wit \leftarrow \phi$ 
while  $|Val| < n - f$  do
    upon RBRecv( $(p_x, r, c_x)$ ) do
         $Val \leftarrow Val \cup \{(p_x, r, c_x)\}$ 
end while
RBSend( $(p, r, Val)$ )
while  $|Wit| < n - f$  do
    upon RBRecv( $(p_x, r, c_x)$ ) do
         $Val \leftarrow Val \cup \{(p_x, r, c_x)\}$ 
    upon RBRecv( $(p_x, r, Val_x)$ ) do
         $Rep \leftarrow Rep \cup \{(p_x, r, Val_x)\}$ 
         $Wit \leftarrow \{(p_x, r, Val_x) \in Rep : Val_x \subseteq Val\}$ 
end while return  $Val$ 

```

III. THE SAFE AREA

In the multidimensional algorithms, non-faulty processes exchange messages containing vectors. Each process in the system will exchange their vectors in each round. And then they will compute one result just in the safe area. The safe area is convex hull of the all the input vectors[?].

We can consider to use the linear algorithm to compute the safe area. The goal is to find a vector w that could be expressed as a convex combination of vectors in C' for all choices $C' \in C$ such that $|C'| = n - f$. The linear program uses the $d + \binom{n}{n-f}(n - f)$ variables described below:

- w_1, \dots, w_d : variables for w_i -th element of vector w , $1 \leq i \leq d$.
- $\alpha_{C',i}$: coefficients multiplying vectors of C' that express w as their convex combination. We include here only those $n - f$ indexes i for which $v_i \in C'$.
- For every C' , the linear constraints are as follows.
 - $w = \sum_{v_i \in C'} \alpha_{C',i} \cdot v_i$ (i.e., w is a linear combination of vectors in C')
 - $\sum_{v_i \in C'} \alpha_{C',i} = 1$ (i.e., the sum of all coefficients for a particular C' is 1)
 - $\alpha_{C',i} \geq 0$ for all $v_i \in C'$ (i.e., all coefficients are nonnegative).

For all every C' , we get $d + 1 + n - f$ linear constraints, yielding a total of $\binom{n}{n-f}(d + 1 + n - f)$ constraints in $d + \binom{n}{n-f}(n - f)$ variables. Hence, for any fixed f , the vector w can be found in polynomial time by linear program with the number of variables and constraints that are polynomial in n and d (but not in f). However, when f grows with n , the computational complexity is high. Observe that we are interested in any feasible vector w that satisfies the above linear constraints and any deterministic optimization objectives function can be used in the linear program.

IV. SUFFICIENT CONDITION FOR MULTIDIMENSIONAL APPROXIMATE AGREEMENT

- A. *AAD*
- B. *MH*
- C. *VG*

V. COMPARASION

- A. *Time complexity*
Time complexity.
- B. *Running Time*
Running time.

VI. CONCLUSION

This is conclusion

REFERENCES

- [1] I. Abraham, Y. Amit, and D. Dolev. Optimal resilience asynchronous approximate agreement. In *OPODIS*, pages 229–239. Springer, 2004.
- [2] H. Attiya and J. Welch. *Distributed computing: fundamentals, simulations, and advanced topics*, volume 19. John Wiley & Sons, 2004.
- [3] G. Bracha. Asynchronous byzantine agreement protocols. *Information and Computation*, 75(2):130–143, 1987.
- [4] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.
- [5] H. Mendes, M. Herlihy, N. Vaidya, and V. K. Garg. Multidimensional agreement in byzantine systems. *Distributed Computing*, 28(6):423–441, 2015.
- [6] M. A. Perles and M. Sigron. A generalization of tverberg’s theorem. *arXiv preprint arXiv:0710.4668*, 2007.
- [7] T. Srikanth and S. Toueg. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. *Distributed Computing*, 2(2):80–94, 1987.
- [8] N. H. Vaidya and V. K. Garg. Byzantine vector consensus in complete graphs. In *Proceedings of the 2013 ACM symposium on Principles of distributed computing*, pages 65–73. ACM, 2013.