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 concensus algorithms in distributed system. These algorithms are different from traditional concensus 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 concensus 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 protocals 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 concensus on the n-dimensional value. The multidimensional input which is d-dimensional vector can be considered as a point in d-dimensinal Euclidean space with $d > 0$. In the multidimentional Byzantine Concensus 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-dimensinal Euclidean space.

To solve this problem, reasearchers 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 shold 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 obversation, 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[9].

A. Multidimensional Byzantine Concensus

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 protocal solving the Multidimensional Byzantine Concensus problem need to satisfy following conditions[6]:

- 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 commnunication model is a complete graph.

B. Multidimensional Byzantine Approximate Agreement

In asynchronous systems, the message deliver time is not guarenteed. The message may take unbound time to deliver. Also, there is not disjoint round in the algorithmes. 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 commnunication 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 commnunication. And the original paper is from Srikanth and Toueg[8] 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 reilable broadcast technique has the following properties:

- Non-faulty integrity: If a non-fauty 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 $m\prime$, 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 recv(q, r, c) from q do
if never sent $(p, qr\{echo\}, .)$ **then**
send($(p, qr\{echo\})$, c) to all processes
end if

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

upon recv(., qr{ready}, c) from $>= f + 1$ 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))

recv(., 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 whilereturn 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[7].

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 conbination of vectors in $C\prime$ for all choices $C\prime \in C$ such that $|C\prime| = n - f$. The linear program uses the $d + \binom{n}{n-f}(n - f)$ variables described below[6]:

– $w_1, ..., w_d$: variables for w_i-th element of vector w, $1 <= i <= d$.

– $\alpha_{C\prime,i}$: coefficients multiplying vectors of $C\prime$ that express w as their convex combination. We include here only those $n - f$ indexes i for which $v_i \in C\prime$.

For every $C\prime$, the linear constraints are as follows.

– $w = \Sigma_{v_i \in C\prime} \alpha_{C\prime,i} \cdot v_i$ (i.e., w is a linear combination of vectors in $C\prime$)

– $\Sigma_{v_i \in C\prime} \alpha_{C\prime,i} = 1$(i.e., the sum of all coefficients for a particular $C\prime is 1$)

– $\alpha_{C\prime,i} >= 0$ for all $v_i \in C\prime$ (i.e., all coefficients are nonnegative).

For all every $C\prime$, 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 ploynomial 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 compulexity is high. Observe that we are interested in any feasible vector w that satisfies the above linear constaints and any deterministic optimization objectives function can be used in the linear program.

IV. Sufficient Condition for Multidimensional Approximate Agreement

This paper introduces two different algorithms which are meant to resolve the multidimensional approximate agreement problem. Both of the algorithms were obtained by suitably modifiying Abraham's algorithm for approximate agreement over scalars, which called AAD algorithm[1].

A. The AAD Algorithm

In the AAD algorithm, each non-faulty process p_i maintains a scalar variable v_i that changes between multiple discrete rounds. The scalar value in process p_i at the end of round r is denoted by v_i^r. The input value of process p_i is denoted by v_i^0. In each round, non-faulty processes[6]:

1. Reliable broadcast the current value v_i^{r-1};

2. Using the witness technique , receiving M, a message set containing values from existing proecesses;

3. Compute a new state v_i^r, based on Content(M)

B. The Mendes-Herlihy Algorithm

Medes-Herlihy's algorithm will approximate agree over vectors, originally present in [5]. From the algorithm it seems that the algorithm compute dimension by dimension but each dimension do not compute independently. The algorithm using another sub-procedure to compute the number of iterations. For each dimension, indexed by m, it execute a specific time to get convergence. The algorithm will converge after $accept > f$ halt messages, accumulated in H.

The Mendes-Herlihy algorithm is:

Algorithm 5 p.AsyncAgreeMH(I)

$(R, v) \leftarrow CalculateRounds(I)$
for i **do** 1... d
  $H \leftarrow \phi$
  $r \leftarrow 1$
  while $|H| <= f$ **do**
    RBSend(($p, m.r, v$))
    upon $V \leftarrow RBReceiveWitness(m.r)$ do
      $S \leftarrow Safe_f(V)$
      $v \leftarrow v \in S$ such that $v[m] = Midpoint(S(m))$
    if $r = R$ **then**
      RESend(($p, m.r, \{halt\}$))
    end if
      $r \leftarrow r + 1$
    upon RBRecv(($p\prime, m.r\prime, \{halt\}$)), with $r\prime >= r$ do
      $H \leftarrow H \cup \{(p\prime, m.r\prime, \{halt\})\}$
  end while
end for
return v

C. VG

The sub-procedure of computing iteration times algorithm:

Algorithm 6 p.CalculateRounds(I)

RBSend(($p, 0, I$))
$(V, W) \leftarrow (Val, Content(Wit))$ from RBReceiveWitness(0)
$U \leftarrow$ {barycenter of $Safe_f(W\prime)$: $W\prime \in W$}
$v \leftarrow$ barycenter of $Safe_f(U)$
$R \leftarrow \left\lceil \log_2(\sqrt{d}/\epsilon \cdot max\{\delta_U(m) : 1 <= m <= d\}) \right\rceil$
return (R, v)

D. The Vaidya-Garg Algorithm

This algorithm works just like AAD algorithm. And it use simpler gemetric primitives. This algorithm is present in [9]. The algorithm is following[6]:

Algorithm 7 p.AsyncAgreeVG(I)

$R \leftarrow 1 + \left\lceil \log_{1/(1-\gamma)} \frac{\sqrt{d}(U-\nu)}{\epsilon} \right\rceil$
for i **do** 1... R
  RBSend((p, r, v))
  upon $M \leftarrow$ RBReceiveWitness(r) do
    for $M\prime \subseteq M, |M\prime| = n - f$ do
      $S_{M\prime} \leftarrow Safe_f(M\prime)$
      $Z \leftarrow Z \cup DeterministicallyChoosePoint(S_{M\prime})$
    $v \leftarrow (\Sigma_{z \in Z} z)/|Z|$
end for
return v

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 and M. Herlihy. Multidimensional approximate agreement in byzantine asynchronous systems. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 391–400. ACM, 2013.
[6] H. Mendes, M. Herlihy, N. Vaidya, and V. K. Garg. Multidimensional agreement in byzantine systems. *Distributed Computing*, 28(6):423–441, 2015.
[7] M. A. Perles and M. Sigron. A generalization of tverberg's theorem. *arXiv preprint arXiv:0710.4668*, 2007.
[8] T. Srikanth and S. Toueg. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. *Distributed Computing*, 2(2):80–94, 1987.

[9] 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.