Weighted Byzantine Agreement Problem

Rishabh Singhal
Roll No. 20171213

Samhita Kanaparthy
Roll No. 2018121005

Vani Sancheti
Roll No. 20171179

April 2020

1 Byzantine Agreement

The Byzantine Agreement (BA) problem is a famous consensus problem in distributed systems, discussing how a set of individual processes in a network can reach an agreement in the presence of failures.

1.1 Goal

In the classic Byzantine Agreement problem, there are N individual processes in a distributed system, f out of them are faulty, which may arbitrarily present Byzantine faults. The goal here is to make a protocol in which all the correct processors can agree on a binary value by following it, regardless of the behavior of the faulty nodes. Lamport, Shostak, and Pease [2] proved this problem is solvable if and only if more than $\frac{2}{3}$ of the generals are correct and follow the protocol. They further proved that the algorithm requires $f + 1$ rounds of communication to terminate.

This solution is based on the following assumptions:

1. Messages sent by a non-faulty processor is delivered correctly.
2. A processor can determine the originator of the message it receives.
3. Process can detect the absence of the message.

The result should satisfy the following two conditions:

1. All non-faulty processors must use the same input value, to produce the same output.
2. If the input unit is non-faulty, then all non-faulty processes use the value it provides as their input, to produce the correct output.

2 Weighted Byzantine Agreement

Weighted Byzantine Agreement Problem was proposed by Garg and Bridgman [1]. In this version, Every process is assigned a weight w_i. Instead of limiting the number of faulty processes to f, the goal is to make the consensus reachable under the condition that the total weight of faulty processes is not greater than $\frac{f}{N}$.

2.1 Model

A distributed system with N processes, $P_1, \ldots, P_N$ and a completely connected topology is taken. The system is assumed to be *Synchronous* with a reliable communication and satisfies FIFO(first-in-first-out). The weights w_i associated with each process P_i are *non-negative*. All processes in the system have complete knowledge about weights of others processes. The weights of all the processes are normalised (i.e. $\sum_{i=1}^{N} w_i = 1$). The sum of weights of all faulty nodes is denoted by ρ. we define *anchor* of the system α_ρ as the minimum number of processes such that the sum of their weights is strictly greater than ρ.

$$\alpha_\rho = \min\{k | \sum_{i=1}^{k} w_i > p\}$$

The Weighted Byzantine Agreement protocol should satisfy the following conditions:

- **Agreement**: Two correct processes cannot decide on different values.
- **Validation**: The value decide must be proposed by some correct process.
- **Termination**: All the correct processes decide in finite number of steps.

Lemma 2.1.1. *There is no protocol to solve the weighted byzantine agreement problem for all values of w when $\rho \geq \frac{1}{3}$*

Proof. Any protocol to solve WBA can be used to solve standard BA by setting $w_i = \frac{1}{N} \forall i$. For this weight assignment, $\rho \geq \frac{1}{3}$ implies that the number of failed processes f in standard BA protocol is atleast $\frac{N}{3}$. We know that no protocol exists for standard BA when $3 * f \geq N$. □

Lemma 2.1.2. *Any protocol to solve the WBA problem for a system with $\rho < 1$ takes at least α_ρ rounds of messages, in the worst case.*

Proof. Any protocol to solve WBA can be used to solve BA by setting $w_i = \frac{1}{N} \forall i$ where, $\rho < 1/N$. No protocol exists to solve BA in less than f + 1 rounds. □

There are two algorithms for weighted byzantine problem: weighted-Queen algorithm and weighted-King algorithm. In the next section we discuss about weighted-Queen algorithm.

3 Weighted Queen Algorithm

3.1 Algorithm

This algorithm takes α_ρ rounds, each round of two phases. This algorithm uses constant size messages but requires that $\rho < \frac{1}{4}$. Each process has a preference for each round, which is initially its input value. This algorithm is based on the idea of a *rotating queen*, Processor P_i is assumed to be the queen for round i.

Algorithm 1 Weighted-Queen Algorithm

For a Process P_i

Parameters:

$V : \{0, 1\}$ initially proposed value

w:const array$[1 \dots N]$ of weights

Initially $\forall j : w_j \geq 0 \wedge (\sum_j : w_j = 1)$

for $q =: 1$ to α_ρ **do**

float $s_0, s_1 := 0.0, 0.0$;

First phase:

if$(w_i > 0)$ **then**

send V to all processes including itself;

forall j such that $w_j > 0$ **do**

if 1 received from P_j **then**

$s_1 := s_1 + w_j$;

else if 0 or no msg received from P_j **then**

$s_0 := s_0 + w_j$;

if $(s_1 > \frac{1}{2})$ **then**

myvalue $:= 1$; *myweight* $:= s_1$;

else *myvalue* $:= 0$; *myweight* $:= s_0$;

Second phase:

if $(q = i)$ **then**

send *myvalue* to all other processes;

receive *queenvalue* from P_q;

if *myweight* $> \frac{3}{4}$ **then**

$V :=$ *myvalue*;

else $V :=$ *queenvalue*;

endfor;

Output V as the decided value;

There are two phases for the algorithm:

1. **First Phase**: Each process broadcasts its initial value and correspondingly receives messages from every other process. Based on the values received and the weights of the processes sending the value, the process decides the value of *myvalue*.

2. **Second Phase**: In this step, each process receives value from the queen. Each process decides whether to select its own value or the value proposed by the queen based on the sum of the weights of the process that proposed value *myvalue*. The process decides on *myvalue* if the total weight is at least $\frac{3}{4}$ otherwise it takes the value proposed by the queen.

3.2 Properties

Lemma 3.2.1. *Persistence of Agreement Assuming $\rho < \frac{1}{4}$, if all correct processes prefer a value v at the beginning of a round; then, they continue to do so at the end of the round.*

Proof. If all correct processes prefer v, then the value of *myweight* for all correct processes will at least be $\frac{3}{4}$; because, ρ is at most $\frac{1}{4}$. Hence, they will choose *myvalue* in the second phase and ignore the value sent by the queen. □

Lemma 3.2.2. *There is at least one round in which the queen is correct.*

Proof. By assumption, the total weight of faulty processes is ρ. The "for loop" is executed α_ρ times. From the definition of α_ρ, There exists at least one process from $P_1 \dots P_\alpha$, which is guaranteed to be correct. Therefore, there has to be one round in which the queen is correct. □

3.2.1 Correctness

Theorem 3.1. *The Weighted Queen's Algorithm solves the agreement problem for all $\rho < \frac{1}{4}$.*

Proof. As shown in the lemma 3.2.2, there is at least one round in which the 'queen' is correct. Each correct process either decides on the value sent by the queen in that round or its own value w. It chooses its value w only when it's *myweight* is at least $\frac{3}{4}$. As the weight of all the faulty process can be at most $\frac{1}{4}$, the queen of that round must have weight of at least $\frac{1}{2}$ corresponding to that value. Thus, In the First phase, the value selected by the queen process would have been w. Hence each process decides on the same value at the end of a round where the queen is non-faulty. From 'persistence of agreement' lemma 3.2.1, As each correct process has decided on the same value, they would continue to do so at the end of the round. Therefore, in the end, the value decided by every correct process would be the same, and it is the value preferred by the queen when it was non-faulty. □

Lemma 3.2.3. $\alpha_{\frac{f}{N}} \leq f+1$ *for all* w *and* f

Proof. It is sufficient show that $\forall f$, $\sum_{i=1}^{f} w_i \geq \frac{f}{N}$. Suppose $\sum_{i=1}^{f} w_i < \frac{f}{N}$ for some f, implies that the sum of the remaining weights is $\sum_{i=f+1}^{n} w_i > \frac{(N-f)}{N}$, because all weights add up to 1. Since w is in non-decreasing order, $w_{f+1} > \frac{1}{N}$; otherwise, the sum of the remaining weights would be at most $\frac{(N-f)}{N}$. This implies that $\sum_{i=1}^{f} w_i > \frac{f}{N}$, because $w_i \forall i \leq j$ is at least w_{f+1}. This contradicts our assumption that $\sum_{i=1}^{f} w_i < \frac{f}{N}$ for some f. Thus, $\forall f$, $\sum_{i=1}^{f} w_i \geq \frac{f}{N}$. □

3.2.2 Message Complexity

There are α_ρ rounds, each with two phases. In the first phase, all the processes with positive weight send messages to all the processes. This phase results in pN messages where $p \leq N$, the number of processes with positive weight. The second phase uses only N messages. Thus, total number of messages is $\alpha_\rho(pN+N)$.

4 Dynamic Weighted Byzantine Agreement

A Dynamic Weighted Byzantine Agreement is needed when the system needs to solve BA various times. Each time a BA is executed, Dynamic weighted byzantine provides feedback about the process. If a process didn't follow the protocol, then the process is considered less reliable for the future. Below we discuss a fault-tolerant method to update weights in the discussed Weighted-Queen Algorithm.

Lemma 4.0.1. *In the Weighted-Queen algorithm, a correct process* P_i *can detect that* P_j *is faulty if any of the following conditions are met:*

1. *If* P_j *either does not send a message or sends a message with wrong format in any of the rounds, then* P_j *is faulty.*
2. *If* myweight $> \frac{3}{4}$ *in any round and the value sent by the queen in that round is different from* myvalue*, then the queen is faulty.*

Proof. In the first part the process is faulty. In the second part, if *myweight* $> \frac{3}{4}$; then, the weight of the queen for that value in that round is at least $\frac{1}{2}$ as the weight of all the faulty process is at most $\frac{1}{4}$. Therefore, if the queen were correct, the value sent by the queen would have matched *myvalue*. □

4.0.1 Dynamic version of the Weighted Queen Algorithm

Each process keeps track of all the faulty processes detected by it using the lemma 4.1. The basic notion for the dynamic version is, at every execution of the BA, the weights of the reliable process should increase, and correspondingly those of the faulty process should decrease. This algorithm requires the weight assignments done by all processes to be identical. All processes need to finally agree on the *faultySet*. The algorithm consist of three phases:

1. **First Phase**: Also known as the learning phase, where each process broadcast their *faultySet*. The main idea is that if a process with total weight at least $\frac{1}{4}$, informs process P_i that process P_j is faulty, then P_j has to be in the *faultySet* of at least one correct process.

2. **Second Phase**: For each process j, if$j$ is in the *faultySet* of P_i, then P_i invokes Weighted-Queen-BA algorithm with '1' as its proposed value; otherwise, it invokes '0' as its proposed value. The output variable value denotes the decided value by the Weighted-Queen-BA algorithm. Therefore, the set of faulty processes that all correct processes agree upon is *consensusFaulty*.

3. **Third Phase**: Weights are updated according to the *consensusFaulty*. The weights of the processes identified as faulty are reduced to zero. After that, the weights among all the processes are normalized.

Lemma 4.0.2. *All the correct process before the execution have same weights assigned after the execution of the algorithm.*

Proof. As the weights are assigned using the *consensusFaulty*, which is identical for all the correct process therefore the weights assigned are identical for all the correct process. □

Theorem 4.1. *A correct process can never be there in* consensusFaulty*. Any faulty process that is in initial* faultySet *of correct processes with total of at least 1/4 will be in* consensusFaulty *of all processes.*

Proof. A correct process can never be identified as a faulty process by any correct process. Therefore the total weight corresponding to that process being faulty can be at most $\frac{1}{4}$ as it can be suspected only be non faulty processes. Therefore that process wouldn't be in the *faultySet* of any correct process after the learning phase, consequently it won't be there in the consensus*faultySet* also. Any faulty process that is in the initial *faultySet* of correct processes with total weight of at least 1/4 will be in *faultySet* of all correct processes after the learning phase. And From the validity of WBA, the faulty process will be in *consensusFaulty*.

□

5 Conclusion

This report has discussed the famous byzantine agreement problem and a weighted version of BA, the 'Weighted-Queen algorithm' which allows, the total weight of the faulty nodes to be at most $\frac{f}{N}$. We have also explained the Updating weights algorithm, an extension to the WBA, guarantees that the weight of a correct process is never reduced, and that of a faulty process is reduced to 0. This weighted algorithm requires fewer messages and is more fault-tolerant than the unweighted version.

Some challenges that we faced while implementing the above described algorithm

1. Handling sequential message passing between processes.

2. For showing the working with faulty nodes, we manually marked few process as faulty and allowed them to reply randomly to different processes.

References

[1] Vijay K. Garg and John Bridgman. The weighted byzantine agreement problem. In *Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium*, IPDPS '11, page 524–531, USA, 2011. IEEE Computer Society.

[2] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.