# Distributed Systems Replication Architecture.

For this project we need to make our distributed system support some faults in the server’s.

The fault’s we want to resolve are silent faults and byzantine faults. We also need to take into account the fact that our messages can be received in different orders that they were sent.

Our solution consists in a set of 4 servers, we believe that this is the minimum number that is possible to have to solve the problems that were given.

## Why 1, 2 & 3 servers are not enough.

* With one server it is possible that is server goes silent (or *beserk’s*) and therefore it is impossible to resolve this fault.
* With 2 server’s we would have to receive 1 *ack* (due to the silent fault’s) each time we would make a write, therefore it is possible that, due to order of messages being able to change, that for a moment a server would have a different state from the expected one and that when reading (we wait for the first answer due to silent faults) we would read a wrong state.
* With 3 server’s there is the case of having one server out-of date, one byzantine and one right, if the byzantine answer’s the same as the out-of-date server then we might think that the right answer is the byzantine one when in fact it isn’t.

Note: These are just one example of many.

## Our logic.

Write:

To write an object we are going to send a write message to all 4 servers and wait for 3 *ack*, with this we know that in the worst case scenario we could have the message well-written in 2 server’s with one of the other having a fault (silent or byzantine) and the other with the message being delayed. When writing an object we have a value (*timestamp)* that will be incremented so we know that the out-of-date server will have a lower timestamp.

Read:

To read an object we are going to send read request to all servers and wait for 3 answers.

1. If they are all the same object then that’s the right object. (C,C,C) = C
2. If they are all the different we wait for the 4rt answer considering that object the right one. We can wait because know that there was a byzantine fault. (O,B,C,….C) = C
3. If we have two equal and one different we will first check the timestamp of the object’s.

* If the two equal have higher timestamp value then we consider that the right answer.(C,C,B) =C
* If the two equal have lower timestamp value then we wait for the forth answer.
  + If the forth answer is the same as the two equal objects then we know that this is the correct one. (C,C,B,C) = C
  + If it is equal to the one different we will chose the pair with higher timestamp.(C,C,B/O,B/O)=C

## Proof that we cover all problems:

One silent fault: (C, C, C, S)

* Our algorithm doesn’t wait for all answers.

One byzantine fault: (C, C, C, B)

* If this byzantine fault has a lower-timestamp and is one of the first 3 to arrive then we will choose the other object as the correct because it has the majority and higher timestamp. (C,C,B)
* If this byzantine fault has a higher *ts* our system knows that there was an byzantine fault (but not where) and therefore will wait for the last answer to arrive (since we don’t support one byzantine and silent fault at the same time)

One byzantine and one out-of-date. (C, C, B, O)

* If they are in the first 3 to arrive our system will wait for the 4rth answer which contains the correct object, note that this occurs even if the byzantine and the out-of-date have the same value, this is because their timestamp will be lower so our system know that there was a byzantine fault.

One Silent and one out-of-date. (C, C, O, S)

* The first 3 answers will contain the two correct and the out-of-date. The correct will contain the higher timestamp so we will choose it correctly.

One Silent and One Byzantine.

* Not supported, our system will wait forever for the forth answer.