1.

a). Suppose that the partition separates X and Y from Z when all data items have version v0:

X: A= 300 (v0) B=500(v0)

Y: A= 300 (v0) B=500(v0)

Z: A= 300 (v0) B=500(v0)

client reads the value of A and then writes it to B:

read quorum = 1+1 for A and B, so the client compares the version of A from X and Y and decided it to be 300.

Then the client get write quorum = 1+1 from X and Y for B and update B to be 300 at X and Y

b). If the client can only get access to Z, then the cilent can only get read quorum =1 and write quorum =1, so neither of the two operations will happen.

c).

after the operations in a), the new state of the servers are:

X: A= 300 (v1) B=300(v1)

Y: A= 300 (v1) B=300(v1)

Z: A= 300 (v0) B=500(v0)

Now the version of A and B at Z are out of date. The read request get a read quorum =1+1 from A and Z. By comparing the versions at X and Z, it’s obvious that Z is out of date, the version at A is selected and Z get updated. Then the write request gets a write quorum =1+1 from A and Z, the write operation can be done too.

2.

The quorum procotol is to validate the replication if the newly updated file from a certain SP. If the majority of SPs agrees with the update, that means that replication is the latest.

The role of the vector timestamps is like a counter. It counts the number of SPs that agrees with the update. If a certain SP receives a timestamps with the majority of bits set. It will know the majority SPs agrees with the update and will go ahead and update the changes.

3.

The following ideas are my thinking based my current code. It’s not yet reflected in my code yet:

Yes, it’s out of order. I think I can send the message to a monotonicly increasing queue in terms of time.

No, not the same order. I can use a intermediary to resolve this.

Yes. It may receive messages after leaving a group.

The weights and parameter definitely affect the result of the stats shown.

I need to add more markers, do more exception handling, etc.