Scenario

In a distributed system with Byzantine fault tolerance, there are four nodes N1, N2, N3, and N4, where up to one node can be faulty (f = 1). A client requests the values of keys *x* and *y* sequentially by sending two GET requests.

The system uses a quorum approach where each quorum requires at least three nodes (2f+1 = 3) and ensures an overlap of at least two nodes between consecutive quorums.

Here's what happens during each request:

- For GET x, the client queries N1, N2, and N3.
    - Responses: N1 and N2 return x = 10, while N3 (a faulty node) returns x = 7.
- For GET y, the client queries N2, N3, and N4.
    - Responses: N2 and N4 return y = 20, while N3 (still faulty) returns y = 15.

Question

Why is it important that there is an overlap of at least two nodes between the quorums for **GET x** and **GET y**? Describe how this overlap contributes to the reliability of the client's results despite node N3 being faulty.

Answer

Since each quorum is of 2*f*+1 nodes and we can have *f* faulty nodes at most, PBFT nicely ensures that we always have *f*+1 non-faulty nodes. For any configuration of nodes that is valid according to PBFT rules, this also means that we will have an overlap of *f*+1 nodes across requests. The overlap ensures that at least one non-faulty node is common across both requests, which means there will be consistent information despite N3's faulty behaviour. It also allows the client to cross-validate data across requests, ensuring that any discrepancies between responses can be attributed to the faulty node, not to inconsistencies in the system.

That said, the 2*f*+1 quorum already ensures the reliability of results returned by a majority, so this is an additional reassurance at best.