1. What is Leader-Based Replication?  Give some examples and discuss some problems. (4 pts)

In leader-based replication, for any *n* nodes in an environment, one is selected as a leader node, to which all incoming writes are written. Upon completion of a write, the leader then sends the writes (either synchronously or asynchronously) to the follower nodes. Leader-based replication systems can be found in a system where reads are much more common than writes (like, for example, in a company that processes data in batches, as opposed to needing to be constantly available for writes from different users – consulting comes to mind).

Problems arise in leader-based systems in two primary circumstances that I’d like to mention: failure of a single leader and conflicts between multiple leaders. When a leader fails and it is the only leader in the ecosystem, there must be a protocol for failover – that is, the promotion of a follower node to leader and the recapturing of its writes. While there are several ways to handle this (and too many to into detail on here), this must be a consideration when creating the system.

When there are multiple leaders in different availability zones (which can be useful for reducing latency to distributed end users), the failure of one leader may lead to a break in the replication between leaders. This also requires failover, but must take in the additional consideration of reconfiguring the new leader in relation to the leaders in other zones.

1. Define, compare, and contrast the following types of Multi-Leader Replication Topologies (3 pts):
   * **Circular Topology –** Replication between leaders in which a “stamp” is made on a particular piece of committed data and then passed in a set order to the other leaders. When the data comes back to a leader whose stamp is already on it (the leader which processed the original write), the information is halted and ceases to pass on since all leaders have replicated it. This may break in the case of a failover of any leader, which breaks the chain, blocking proper replication across leaders until failover is performed.
   * **Star Topology –**  Replication method in which one central leader acts as a passthrough for all other leaders, maintaining two-way replication feeds with each of them (while the secondary leaders do not maintain replication feeds with one another). The failure of the central leader is the most detrimental (until failover), but in the case of failure of any leader besides the central one, the other leaders are not affected.
   * **All-to-All Topology –** Similar to the star topology, with the exception that every leader has a read/write relationship with every other leader. The failure of one leader does not affect the performance of any other leader, as all other leaders can send and receive data from any non-failed leader node.
2. Explain what is going on in the following architecture and what we can do to fix it. (3 pts)

A screenshot of a map

Description automatically generated

In this process flow, User 1234 is sending a write to the leader node, which uses asynchronous replication to send it to two follower nodes (replicas). The write commits to Follower 1, which is then queried for the key that is written by User 1234. User 2345 receives a valid response because Follower 1 has received the write from the leader, while Follower 2 has not yet (endemic to asynchronous replication – User 1234 has already received a success message because the Leader has processed the write).

User 2345 then queries the key again, but her query is directed to the node Follower 2, which has not yet received the write from the leader. Because no data yet exists in Follower 2 for the newly-written key, User 2345 receives an empty return (no results, to distinguish from no return whatsoever) – effectively, a stale response.

This can be fixed by establishing a quorum for reading and for writing. A quorum assures that there is sufficient overlap between the nodes to which a value is written and the nodes from which that value can be read. It increases fault tolerance in the case of unavailable follower nodes, meaning that User 2345 never receives the stale read (an empty result set).