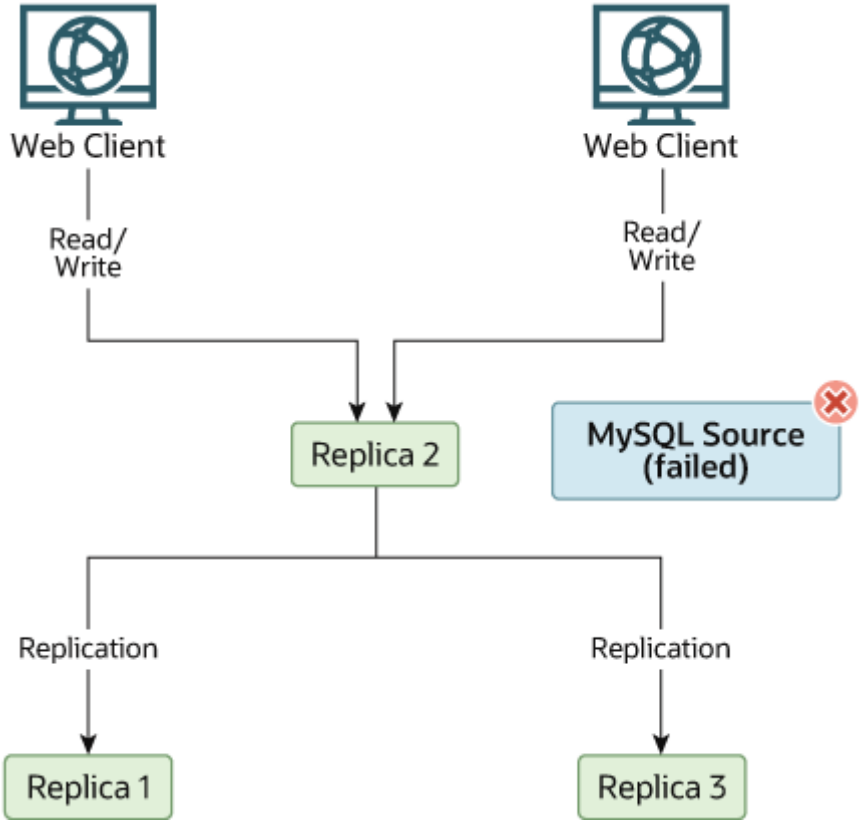
Figure 17.5 Redundancy Using Replication, After Source Failure



When `Source` becomes available again, you should make it a replica of `Replica 1`. To do this, issue on `Source` the same `CHANGE REPLICATION SOURCE TO` | `CHANGE MASTER TO` statement as that issued on `Replica 2` and `Replica 3` previously. `Source` then becomes a replica of `Replica 1` and picks up the `Web Client` writes that it missed while it was offline.

To make `Source` a source again, use the preceding procedure as if `Replica 1` was unavailable and `Source` was to be the new source. During this procedure, do not forget to run `RESET MASTER` on `Replica 1` before making `Replica 1`, `Replica 2`, and `Replica 3` replicas of `Source`. If you fail to do this, the replicas may pick up stale writes from the `Web Client` applications dating from before the point at which `Source` became unavailable.

You should be aware that there is no synchronization between replicas, even when they share the same source, and thus some replicas might be considerably ahead of others. This means that in some cases the procedure outlined in the previous example might not work as expected. In practice, however, relay logs on all replicas should be relatively close together.

One way to keep applications informed about the location of the source is to have a dynamic DNS entry for the source server. With `bind` you can use `nsupdate` to update the DNS dynamically.

17.4.9 Switching Sources with Asynchronous Connection Failover

Beginning with MySQL 8.0.22, you can use the asynchronous connection failover mechanism to automatically establish an asynchronous (source to replica) replication connection to a new source after the existing connection from a replica to its source fails. The asynchronous connection failover mechanism can be used to keep a replica synchronized with multiple MySQL servers or groups of servers that share