DATA in a cloud computing system should be highly available. That is, whenever you connect to the system, the data you stored there should be ready to use. The standard mechanism to accomplish this—data replication—involves maintaining multiple copies of each user’s data.

By itself, replicating data does not solve the problem, because the copy of data that is available might be stale. To see why, consider a system that maintains three copies of file $x$: $x_1$, $x_2$, and $x_3$. Suppose $x_1$ and $x_2$ are currently available and $x_3$ is down. If a program updates $x$, the system writes the update to $x_1$ and $x_2$, but not to $x_3$ because it is down. No problem, since the system still has two correct copies. Next, suppose $x_1$ and $x_3$ fail, and then $x_2$ recovers. Now there is a problem. If a user reads $x$, the best the system can do is return the value of $x_3$. But that copy is stale—it does not have the latest update.

An early solution to this problem is called majority consensus. To process a write operation on $x$, the system assigns a monotonically increasing timestamp to the write, and writes $x$’s value and timestamp to a majority of the copies of $x$. To process a read operation on $x$, it reads a majority of the copies of $x$ and returns one that has the largest timestamp. Since the intersection of any two majorities includes at least one copy, each read returns the result of the previous writes on the same data.

In our example, the read operation only reads one copy, which is not a majority of three. Therefore, it might not return the latest state of $x$. Using majority consensus, the system would have to wait until a second copy became available, so it could read two copies. Then, it could return the more up-to-date copy of the two that it reads, which must be the freshest.

The notion of majority was extended in Gifford to be a weighted major-