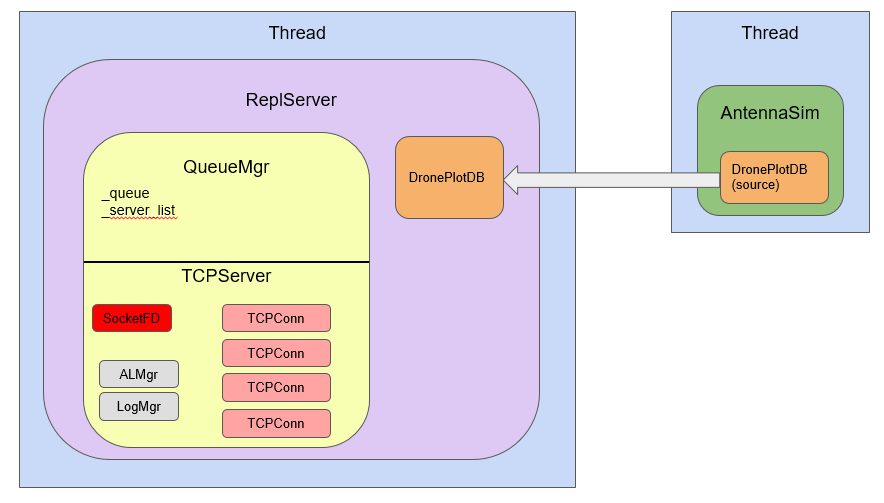
Mark Demore, 2d Lt

CSCE689 – HW4

**CODE:**

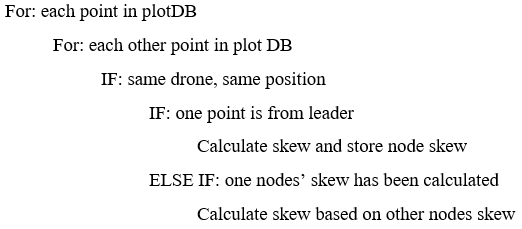
<https://github.com/mdemore2/AFIT-CSCE689-HW4-S>

**QUESTIONS:**



**Figure 1: System Architecture**

The image above shows the basic architecture given for this assignment. The consistency model for this system makes use of the queue manager to pass new plot points between the other servers to be replicated and then deconflicted for time skew and duplicate points. The system uses basic TCP communication protocols handled within the TCPConn class using Socket File Descriptors. Messages are passed using AES Encryption after authentication, shown in the images below. The authentication steps occur in conjunction with the s\_connecting and s\_connected states, making use of the new states s\_authenticate and s\_handshake. Reflection attacks are protected against with a simple check that the client/server is not sent its own SID by the other client/server to encrypt. Data is deconflicted on the replication server with a series of checks. The points are looped through and compared. If the two points have the same location and one of the points came from the elected leader or its time skew has already been calculated, the time skew is calculated for the other node. If the skew could not be calculated yet, it was ignored and revisited again once replicate() was called. Once the skew has been detected for all nodes, the times are adjusted for all points of those nodes in the database. Then, the points are checked for duplicates. If two points have the same timestamp after the skew correction, the point from the node with the lower priority is deleted. The leader is selected simply using a priority queue of the existing servers. This assignment makes use of a client-consistency model, with each client being responsible for maintaining its own data.



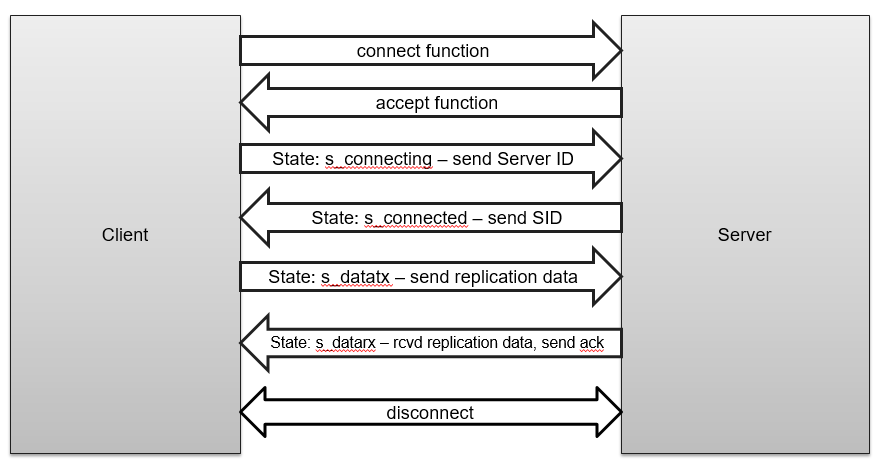
**Figure 2: Skew check psuedocode**

The order in which data arrives should only impact the algorithm when determining priority, in that a skew may be calculated using data from a node with higher priority but is not necessarily the leader. This should only occur when the node with priority has already had its skew calculated based on the leader to ensure that there is not a conflict in skew calculations. Data that needed to be adjusted for skew but was not replicated was adjusted before replication. This should work because either the node receiving this new data will no longer need to adjust it locally or will delete a potential duplicate if it later adjusts uncorrected data for the same point.

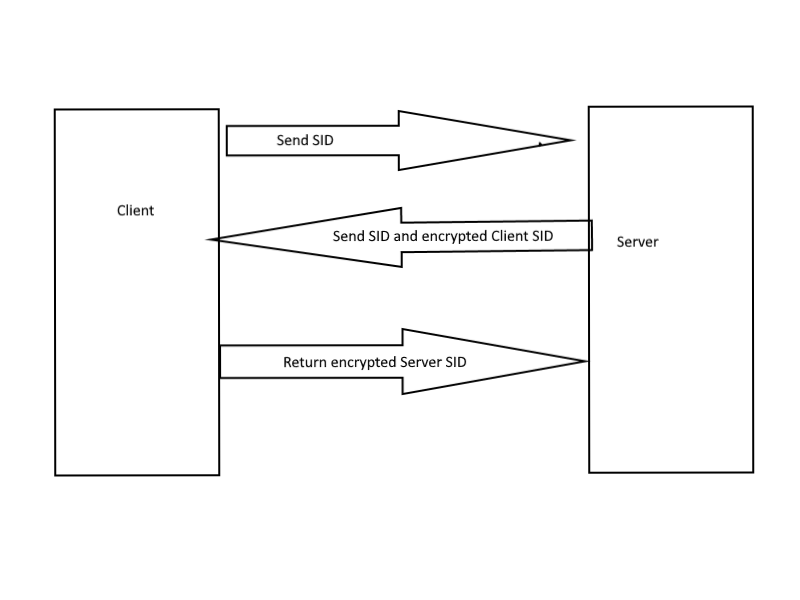
If a node’s offset is not yet known, it should be able to be calculated based on relative skew as long as the leader has one common report and skew can be calculated for a common node. Until this can be calculated, no skew is adjusted. Since each server can locally adjust for skew and remove duplicates, this should not be an issue as it could even be finished at the very end if the last point to be replicated was from the leader.

In order to move towards a more data-consistent model for the server, a mutex would be needed between the servers, with notifications and updates being pushed when necessary. This would require more overhead than the current replication every so often method.

All of these design decisions were selected based on the provided architecture and the inherent abstraction of the system in this assignment. This also made the project a bit difficult to understand and interesting in assessing what work needed to be done. It was also difficult to debug and interpret output.



**Figure 3: Original Communications Protocol**

****

**Figure 4: Authentication Steps**