Logic for Loab-Balancing:

There are two components to the algorithm:

1. **The Load balancer:** Simply process the packet, fetch the keys and perform some logic on some parameters to find a specific host to send the packet to (for optimum resource utilization).
2. **The controller (Can be an entity external/internal to Load Balancer):** The job of the controller is to vary the parameters on which the load balancer will act upon . The parameters remain constant for a period of time. After the time period is over, the controller may vary some of them and then keep it constant for the next cycle.

Algorithm for Load Balancing:

1. Once the packet is received, fetch the slice id. If the slice id is 0 go to step 2, else go to step 5
2. Since slice id is 0, we know that this is a new connection. Fetch the value in the MME-ID.mme\_code header. If some value is present go to step 4, else go to step 3.
3. Since no MME-ID.mme\_code is present we need to choose a random server and store it in the mme\_code part of the MME-ID [ # using Algo 1] .
4. Send the packet to the host using the MME ID.mme\_code.
5. If slice ID is present it means , the device already has some context present and the request needs to be directed accordingly. We need to check if the MME-id is already present. If yes, send it to that MME.Else go to step 6
6. Choose the host GUTI ,Slice-ID and SLO-value as arguments.[# Algo 2]

Algo 1 (Choose Server Randomly):

1. Generate key = random()
2. Select host using consistent hash of “key”. Set MME\_ID.mme\_code = host.

Algo 2 (Choose specific server for handling control procedure):

1. Fetch the consistent hash of the given slice-id. Based on the SLO of the connection, fetch a list of H hosts by using the consistent hash and the key as GUTI.TMSI + (i -1) \* where i ranges from 1 to H.
2. Iterate through the list of hosts and use the queue usage value of the particular slice from the table to determine the lowest queue usage and choose that as the host.
3. Once the host is determined, set the MME-id.mme\_code as the host. This way all packets for a transaction go to the same host .For a new transaction, the procedures from step 1 would be repeated to find out the most optimum host

Approach adapted and things considered for load-balancing**:**

This algorithm deals with choosing the host based on the SLO, Slice ID and the GUTI.

1. **Host provisioning/scaling:** The same set of hosts will remain active for a specific time slice i.e., the controller may decide to scale down/up after this time slice (to avoid frequent state migrations).
2. **Replicas:** A user / IoT device’s state (or contextual information) can have multiple replica’s. We pick one specific host use the same host for one control procedure (i.e., each control procedure has some specific task to achieve and will have a set of control messages exchanged between the user/IoT device and the MME and other entities inside the packet core network), as the synchronization with its replica’s take place only after the complete signalling specific to that procedure is completed with MME. [more info in point “e”)]
3. **Choosing best possible or viable server from available list of servers/replicas:** We can choose the best possible server from the list of replicas. The number of replica’s can be decided based on the SLO requirement of the connection. We can fetch the queue size information of the MME at replica’s host and decide to choose the one with the lowest queue size value (Or should we choose the one with highest queue size but still be able to accommodate it without impacting the SLO requirements?).
4. **Slice Specific Consistent hash**: Each slice ID has a consistent hash associated with the list of hosts which support that slice.This is used to club together the hosts which support a particular slice and distribute load among them.
5. After a connection undergoes Attach Accept, SLICE-ID , SLO-value and the GUTI is created.Now based on the SLO-value we can create H replica’s. These replica’s are put in hosts which are decided by using the consistent hash of the slice-D and the key as GUTI.TMSI + (i -1) \* where i ranges from 1 to H. (I beleive number of replicas not only depend on the list of SLO requirement of that set of users/slice but also on the slice size. Consider for example, even if the slice size is only 5, but has tight SLO requirement. Do you wanted to create it three replicas, increasing the resource consumption or should we restrict to a max of two replicas in this case?)

Handling other scenarios/cases:

1. Once the time slot is completed, the connections can be migrated according to hosts being added and removed. For upcoming requests, the mme ID can be calculated using the GUTI.tmsi value and can be used for the time slot or duration.
2. Signaling occurring across time slots.This situation occurs when signalling takes place across two time slots. Since we already have the MME -id , the requests will be routed correctly. Once all signaling is completed, the migration can be performed.
3. We should consider the load on the hash table at the host .This means that we need to try and keep the hash tables at the host at most 50% usage. There could be degradation in performance of the Hash Table if there is more than 50% usage.

Controller Logic: The controller has the following data structures.

1. A Structure storing the following information for each host.  
   a) CPU load on each lcore.  
   b) For each slice-id supported, the following information.  
    i) NF being used -its queue usage and lcore being used.
2. A map between slice-id and its consistent hash ring.

Scaling (Adding or removing hosts to slices):

1. Fetch the expected load (currently the load can be considered as the queue usage of each slice) of the MME for each slice inside the host. If they are below the lower threshold value consider it for removal (Not sure if this is completely true). The threshold value is a percentage of queue size. Eg a lower threshold value could be 10%. Ie, If we think that less than 10% of queue usage will be present throughout the next cycle, we could consider the slice for removal (May be better approach is needed).The expected load can be calculated by taking into account the Traffic patterns of each connection within the slice.
2. If all the slices get removed from the host, we can remove the host.
3. Similarly check if the load is higher than an upper threshold value. If so we may need to add more hosts to that particular slice (is this load of slice or host? We need to be careful on at which level we are checking the host utilization i.e., either slice level or host level?).
4. Now we have a list of slices which require more hosts. We also have the information on how many more slices a host can accommodate (we can get this information based on the number of lcores being used) (We can add more than one slice to each lcore, hence it can not be completely static like this?).
5. We can now start iterating through this list of slices and see if we can allocate a host for it . Once a host is allocated, we can add the host in the consistent hash of the slice. After this the number of connections get redistributed (We can use the factor if the hosts are not able to handle any specific connection, we can also consider this as a factor to provision more resources i.e., slice creation on other host or migrating some connections to different hosts).
6. If at any point of time, there is no place in the existing hosts, we can spawn a new host and add that slice into that.

[Flow chart of Load balancer](https://drive.google.com/open?id=14wmNAnm5cF8uHp-Z8W8cilUb36PqpnEi)