Algorithm for Source NF:

We consider a data structure named ***temp_list*** that will store packets from the input buffer of flow thread.
Let, load balancing event is triggered after processing ***y*** packets.

Start:

temp_list ← [ ]
flow_id ← *f*
processed_packets_count ← *y*
thread_id ← *(f, T)*
dest_NF_IP ← *p.q.r.s*

Step-1:

Copy packets from input buffer of *thread_id (f, T)* to *temp_list*

Step-2:

Kill thread *(f, T)*

Step-3:

Broadcast state up to packet ***y***

Step-4:

Notify SDN controller to change the rule to redirect **flow *f*** to the new destination member NF.

Step-5:

```
loop [ for each packet in input buffer of NF]
        if flow_id = f
                then copy packet from main input buffer of the NF to temp_list
        end if
end loop
```

Step-6:

Send ***(dest_NF_IP, flow_id, temp_list)*** to switch.

Algorithm for Destination NF:

Start:

Receive ***(flow_id, temp_list)*** from switch.

Step-1:

Loop [for each packet in temp_list]
    Copy the packet to the **main input buffer** of the NF
End loop

Create a new thread for ***flow f***

Step - 2:

Receive state updates up to packet ***y***
Set ***nextExpectedPktID*** ← ***y+1***

Concern: The state update broadcast for a flow needs to be FIFO. Otherwise the following problem may occur,

Suppose, destination NF received states up to ***x*** packets. Source NF updates packets up to ***y*** packets and after some time, up to ***z*** packets. Here, **z > y > x**.

Now, if the update for ***z*** reaches first, ***nextExpectedPktID*** will be set to ***z+1***. And after some time, the update for **y** will reach and ***nextExpectedPktID*** will be set to ***y+1***. So after ***nextExpectedPktID*** will reach ***z+1***, the NF processing will be halted for infinite time as the NF will expect ***z+1*** but ***z+1*** was already processed before, which will never come again.

We can try to maintain a separate module in between to ensure FIFO property.