### Energy arrival subprogram

Fig. 4 - 6 illustrates the flow chart of the subprogram for energy arrival. At the start, we update the current simulation time as the energy arrival time and generate the next energy arrival time using the exponential variate function with . Additionally, we accumulate the total number of energy units that have arrived. We then check if the energy queue is full. If it is not, we increment the number of energy units in the queue by one. However, if the energy queue is full, the number of blocked energy units is incremented by one. Finally, the program describes the cases that follow.

1. We begin by checking the server status is "IDLE" or not and whether there are packets in the packet queue. If both conditions are met, we proceed to check whether the number of energy units in the energy queue is sufficient to serve the packet at the head of the queue. If either condition is not met, the subprogram will return to the main program without executing any further actions.
2. The second step involves checking whether there are enough energy units in the energy queue to serve the packet at the head of the packet queue. If there are enough units, the relevant statuses are updated, including the energy and packet queue sizes, server status, and waiting and service times. The departure time is then generated using an exponential variate function, and the packets in the queue are shifted one place forward. The next impatient time is updated or set to "infinite" based on the status of the packet queue.
3. If the energy queue doesn't have enough energy units to serve the packet, we generate a random value between 0 and 1 and compare it with the probability of using regular battery, represented by . If the random value is less than or equal to , we update the relevant statuses and accumulate the appropriate time, which includes the consumption of energy units from the regular battery that are not in the energy queue. Then, we generate the departure time using the exponential variate function with , and shift all the remaining packets in the packet queue forward one position. Finally, based on whether the packet queue is empty or not, we determine whether to update the next impatient time or set it to infinity.
4. If the number of energy units in the energy queue is insufficient and the random value is greater than , the subprogram will not execute any further action and return to the main program.

### Impatient subprogram

Fig. 4 - 7 illustrates the flow chart of the subprogram for impatient. It starts by updating the current simulation time to the next impatient time. Then, based on packet priority, we determine the packet to be removed from the queue due to impatience, and accumulate the total number of impatient packets and total impatient time. Subsequently, we decrease the number of packets in the queue by one and shift all packets after the th position forward one place, overwriting the original packet information. Finally, the subprogram considers different cases for further description.

1. Initially, we determine whether there are any packets in the packet queue. If not, the next impatient time will be set to an infinite value. If there are packets in the queue, we proceed to check if the server status is "IDLE" or not.
2. If the server is busy, we proceed to check whether the number of energy units in the energy queue is sufficient to serve the packet at the head of the packet queue or not. If the energy requirement is met, we update various statuses and accumulate relevant time based on the priority of the packet. Otherwise, we move on to the next step in the subprogram.
3. If the energy queue has enough energy units to serve the packet, we update several statuses and accumulate relevant times based on the packet priority. These include the number of energy units and packets in the queue, the server status, total waiting time in the queue, and total service time. Next, we generate a departure time for the packet using the exponential variate function with . Then, we shift all remaining packets in the queue forward one place. Finally, we check whether the next impatient time needs to be updated or set to infinity based on the packet queue's status.
4. If the energy queue does not have enough energy units, a random value between 0 and 1 is generated and compared to the regular battery usage probability . If it is greater than , no further action is taken, and the subprogram returns to the main program. However, if it is less than or equal to , various statuses are updated, and relevant times are accumulated. This includes the number of energy units consumed from the regular battery, excluding those in the energy queue. The departure time is then generated using the exponential variate function with . All remaining packets in the packet queue are moved forward one place. Finally, we determine whether the next impatient time should be updated or set to be infinite, depending on whether the packet queue is empty.
5. If the number of energy units in the energy queue is insufficient and the random value is greater than , no further action is taken, and the next step is to update the next impatient time based on the impatient time of packets in the queue.

### Departure subprogram

Fig. 4 - 8 illustrates the flow chart of the subprogram for departure. To begin with, the subprogram updates the next departure time as the current simulation time, and then counts the total number of served packets based on the server's status. After this, the subprogram considers different cases for further description.

1. To begin with, we first check whether the packet queue is empty. If it is empty, the server status is updated as "IDLE", and both the next departure time and the next impatient time are set to be infinite. However, if the packet queue is not empty, we need to verify if the number of energy units in the energy queue is sufficient to serve the packet at the head of the queue.
2. The second step involves checking whether there are enough energy units in the energy queue to serve the packet at the head of the packet queue. If there are enough units, the relevant statuses are updated, including the energy and packet queue sizes, server status, and waiting and service times. The departure time is then generated using an exponential variate function, and the packets in the queue are shifted one place forward. The next impatient time is updated or set to "infinite" based on the status of the packet queue.
3. If the number of energy units in the energy queue is insufficient to serve the packet, a random value between 0 and 1 is generated, which is compared with the probability of regular battery usage . If it is less than or equal to , we update the relevant statuses and accumulate the appropriate time, including the number of energy units consumed from the regular battery, which are not in the energy queue. Next, the departure time is generated using the exponential variate function with , and all the remaining packets in the packet queue are moved forward one place. Finally, depending on whether the packet queue is empty or not, we decide whether to update the next impatient time or set it to be infinite.
4. If the number of energy units in the energy queue is insufficient and the generated random value is greater than , the server status will be set to "IDLE" and the next departure time will be set to be infinite.

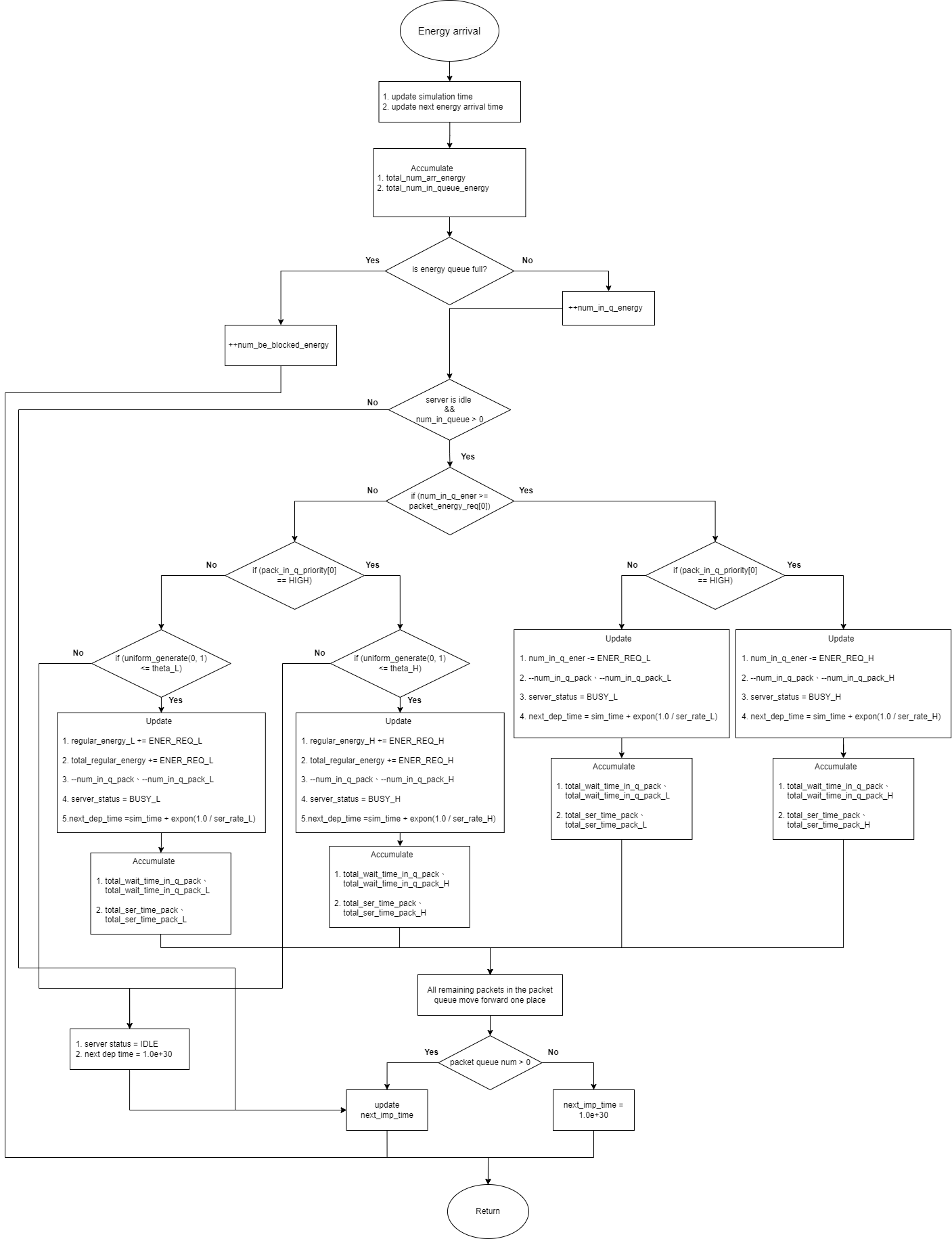


Fig. 4 - 6: Flow chart graphic of energy arrival subprogram for scenario 1

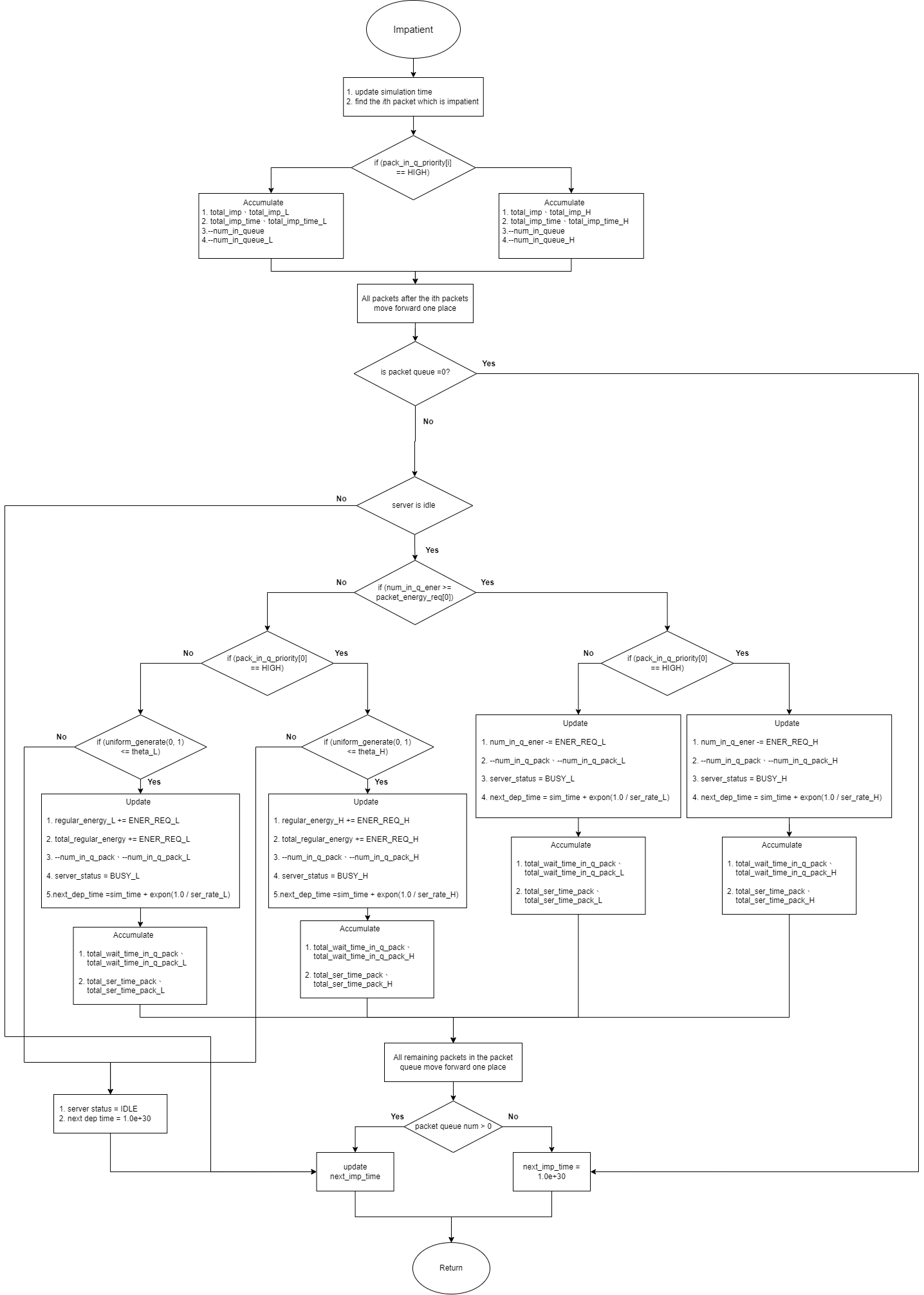


Fig. 4 - 7: Flow chart graphic of impatient subprogram for scenario 1

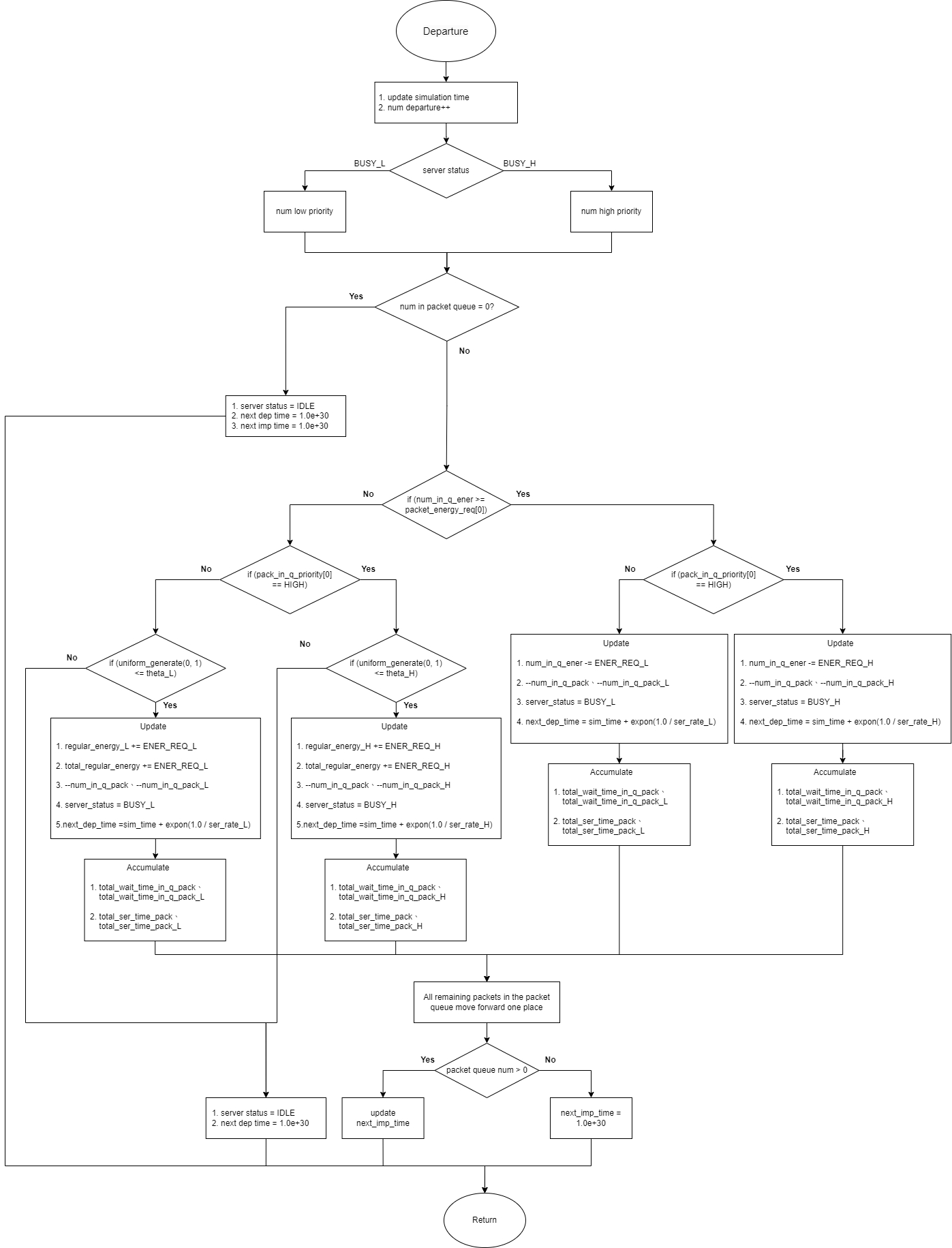


Fig. 4 - 8: Flow chart graphic of departure subprogram for scenario 1