### Packet arrival from node 3 to node subprogram

Fig. 4 - 15 illustrates the flow chart of the subprogram for packet arrival from node 3 to node . To make the flowchart easier to understand, we will use the notation "node " to represent node 1 and node 2 in the network since the flowchart for packet arrival from node 3 to node is similar. Initially, the program updates the current simulation time as the packet arrival from node 3 to node Then the next packet arrival time from node 3 to node is set to be infinite. The program then accumulates the total number of arrived packets for node , packets in the queue of node , and packets in node separately by the priority of the arrived packet. It checks whether the packet queue in node is full or not. If the packet queue is not full, the program adds one packet to the queue and inserts the related information of the HP packet at the front of all LP packets or stores the related information of the LP packet at the end of the queue, depending on the priority of the packet. It generates the impatient time using the exponential variate function with . Then the HP/LP packet is marked as "passed through node 3" for identification. On the other hand, if the queue is full, the program increases the number of blocked packets for node by one. The program then executes the situations divided into the following cases.

1. The first step is to check the status of the server in node . If it is not "IDLE", we update the next impatient time by considering the impatient time of packets waiting in the queue. However, if the server is "IDLE", we need to check whether the energy units in the energy queue of node are sufficient to meet the energy requirement of the packet at the head of the packet 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 of node are updated, including the energy and packet queue sizes, server status, and waiting and service times. The departure time for node is then generated using an exponential variate function with , and we temporarily store information regarding whether the packet at the head of the queue has passed through node 3. In addition, 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 regular battery usage probability of node , i.e., . 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 for node is generated using the exponential variate function with , and we temporarily store information regarding whether the packet at the head of the queue has passed through node 3. In addition, 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 of node is insufficient and the random value is greater than , the next impatient time needs to be updated based on the impatient time of packets in the queue.

### Node- impatient subprogram

Fig. 4 - 16 illustrates the flow chart of the subprogram for node- impatience. To make the flowchart easier to understand, we will use the notation "node " to represent all three nodes in the network since the flowchart for impatient is similar for each node. It starts by updating the current simulation time to the next impatient time of node . Then, based on packet priority, we determine the packet to be removed from the queue of node 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 impatient packet, say the th place, one place forward, 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 of node is set to an infinite value. If there are packets in the queue, we proceed to check if the server status in node is "IDLE" or not.
2. If the server is idle, we proceed to check whether the number of energy units in the energy queue of node 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 node 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 . For the specific node being considered, whether it is node 1 or node 2, we temporarily store the information about whether the packet at the head of its queue has passed through node 3. In the case of node 3, we store the information about which node the packet originated from. 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 of node i.e., . 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 for node is then generated using the exponential variate function with . For the specific node being considered, whether it is node 1 or node 2, we temporarily store the information regarding whether the packet at the head of its queue has passed through node 3. In the case of node 3, we store the information about which node the packet originated from. 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.

### Node-1 departure subprogram

Fig. 4 - 17 illustrates the flow chart of the subprogram for node-1 departure. To begin with, the subprogram updates the next departure time of node 1 as the current simulation time. Next, we make a decision on which node the packet is routed to, taking into account whether it has passed through node 3 and the routing probability, denoted as . Based on this decision, we update the arrival time for the packet accordingly. Then the total number of served packets is calculated based on the server status of node 1. After this, the subprogram considers different cases for further execution.

1. To begin with, we check whether the packet queue in node 1 is empty. If it is empty, the server status is updated as "IDLE", and both the next departure time for node 1 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 of node 1 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 of node 1 are updated, including the energy and packet queue sizes, server status, and waiting and service times. The departure time for node 1 is then generated using an exponential variate function with , and we temporarily store information regarding whether the packet at the head of the queue has passed through node 3. In addition, 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 regular battery usage probability of node 1, i.e., . 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 for node 1 is generated using the exponential variate function with , and we temporarily store information regarding whether the packet at the head of the queue has passed through node 3. In addition, 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 of node 1 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.

### Node-2 departure subprogram

Fig. 4 - 18 illustrates the flow chart of the subprogram for node-2 departure. To begin with, the subprogram updates the next departure time of node 2 as the current simulation time. Next, we make a decision on whether the packet should be routed to node 3 or exit the network, taking into consideration the routing probability denoted as and the packet's passage through node 3. Based on this decision, we update the arrival time for the packet accordingly. Then the total number of served packets is calculated based on the server status of node 2 and the priority of packet arriving at the next node. After this, the subprogram considers different cases for further execution.

1. To begin with, we check whether the packet queue in node 2 is empty. If it is empty, the server status is updated as "IDLE", and both the next departure time for node 2 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 of node 2 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 of node 2 are updated, including the energy and packet queue sizes, server status, and waiting and service times. The departure time for node 2 is then generated using an exponential variate function with , and we temporarily store information regarding whether the packet at the head of the queue has passed through node 3. In addition, 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 regular battery usage probability of node 2, i.e., . 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 for node 2 is generated using the exponential variate function with , and we temporarily store information regarding whether the packet at the head of the queue has passed through node 3. In addition, 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 of node 2 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.

### Node-3 departure subprogram

Fig. 4 - 19 illustrates the flow chart of the subprogram for node-3 departure. To begin with, the subprogram updates the next departure time of node 3 as the current simulation time. Based on the originating node of the packet, we redirect it back to the previous node while updating the arrival time accordingly. Then the total number of served packets is calculated based on the server status of node 3 and the priority of packet arriving at the next node. After this, the subprogram considers different cases for further execution.

1. To begin with, we check whether the packet queue in node 3 is empty. If it is empty, the server status is updated as "IDLE", and both the next departure time for node 3 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 of node 3 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 of node 3 are updated, including the energy and packet queue sizes, server status, and waiting and service times. The departure time for node 3 is then generated using an exponential variate function with , and we temporarily store information about the previous node. In addition, 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 regular battery usage probability of node 3, i.e., . 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 for node 3 is generated using the exponential variate function with , and we temporarily store information about the previous node. In addition, 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 of node 3 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.

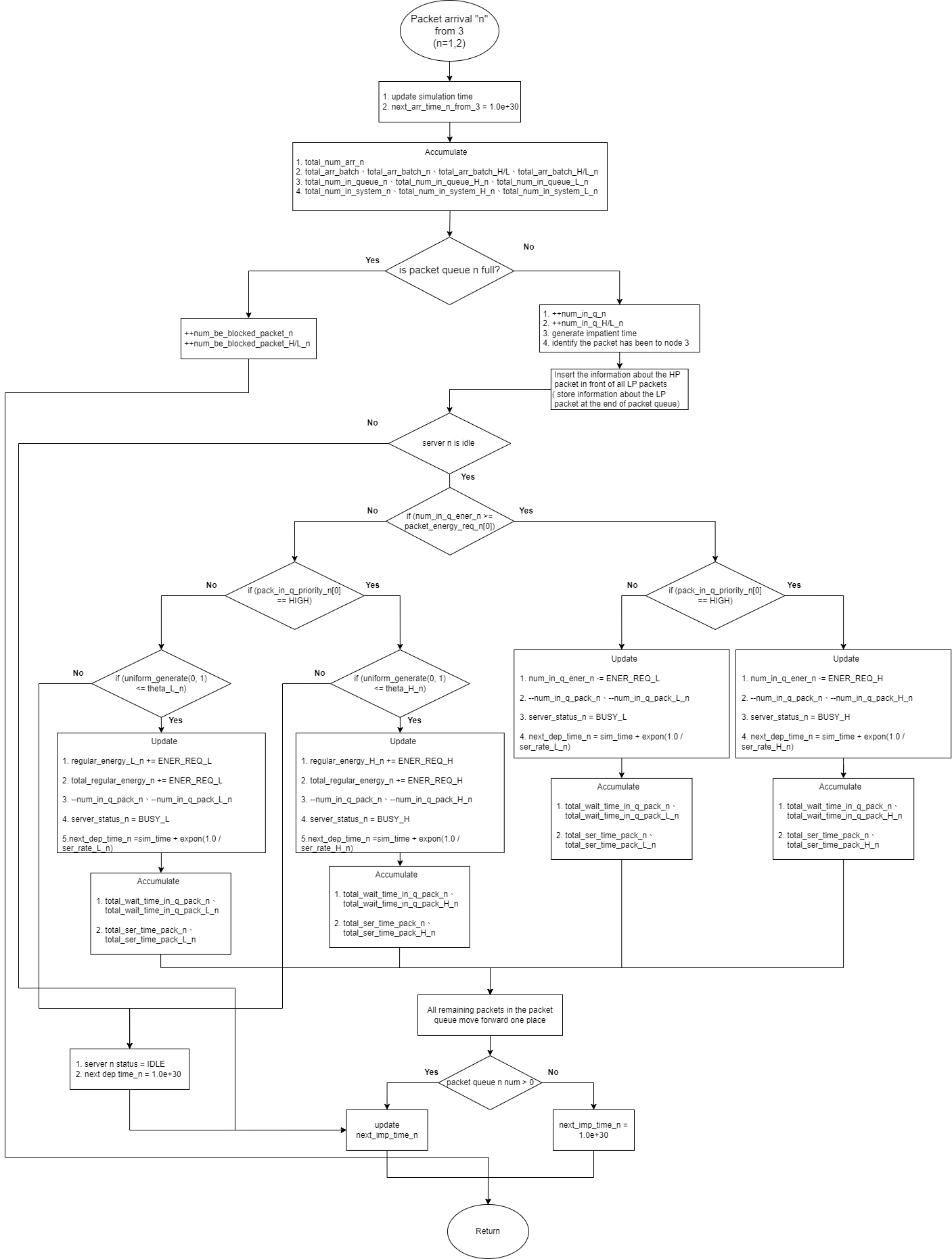


Fig. 4 - 15: Flow chart graphic of the packet arrival from node 3 to node subprogram for scenario 2

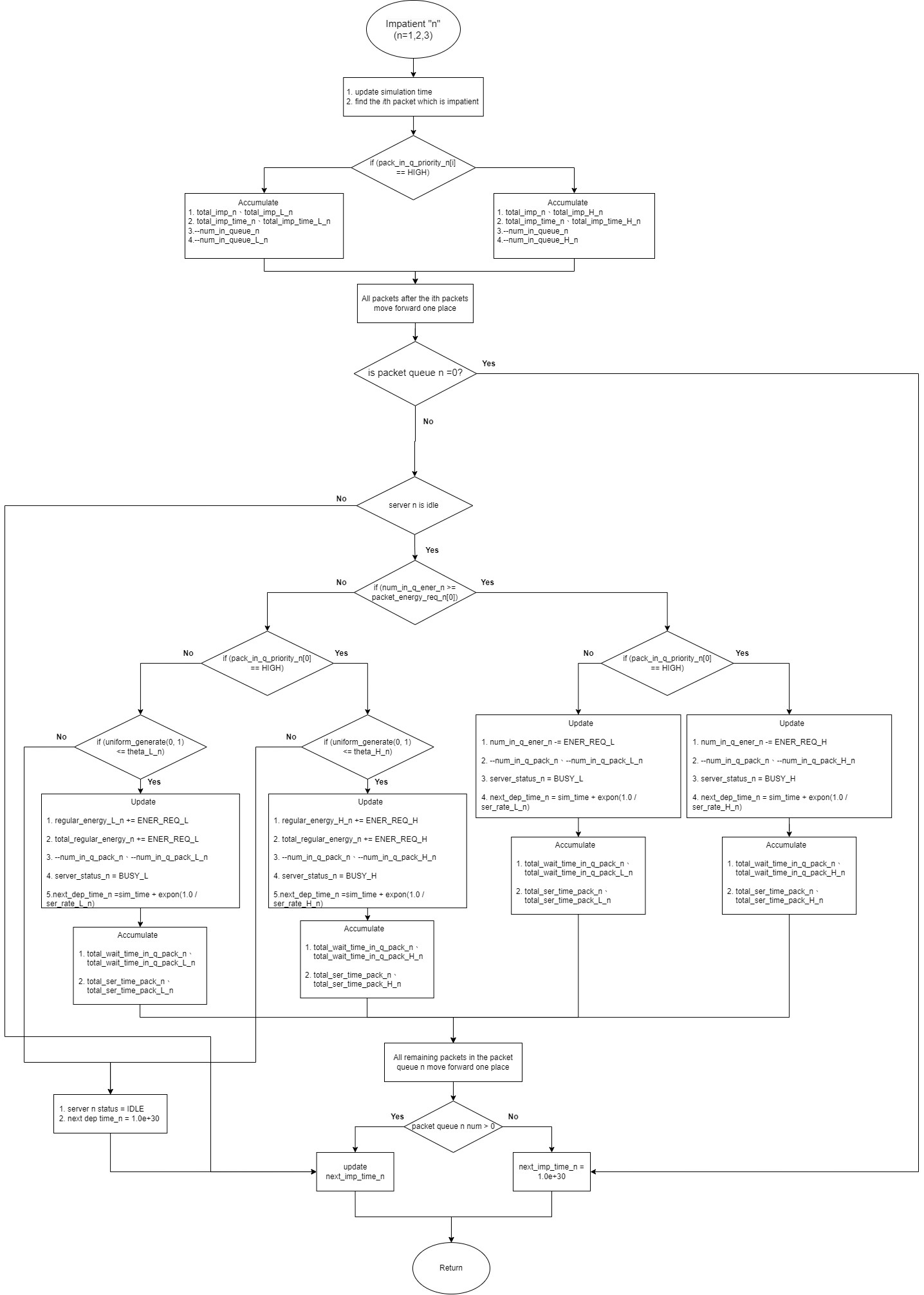


Fig. 4 - 16: Flow chart graphic of node- impatient subprogram for scenario 2

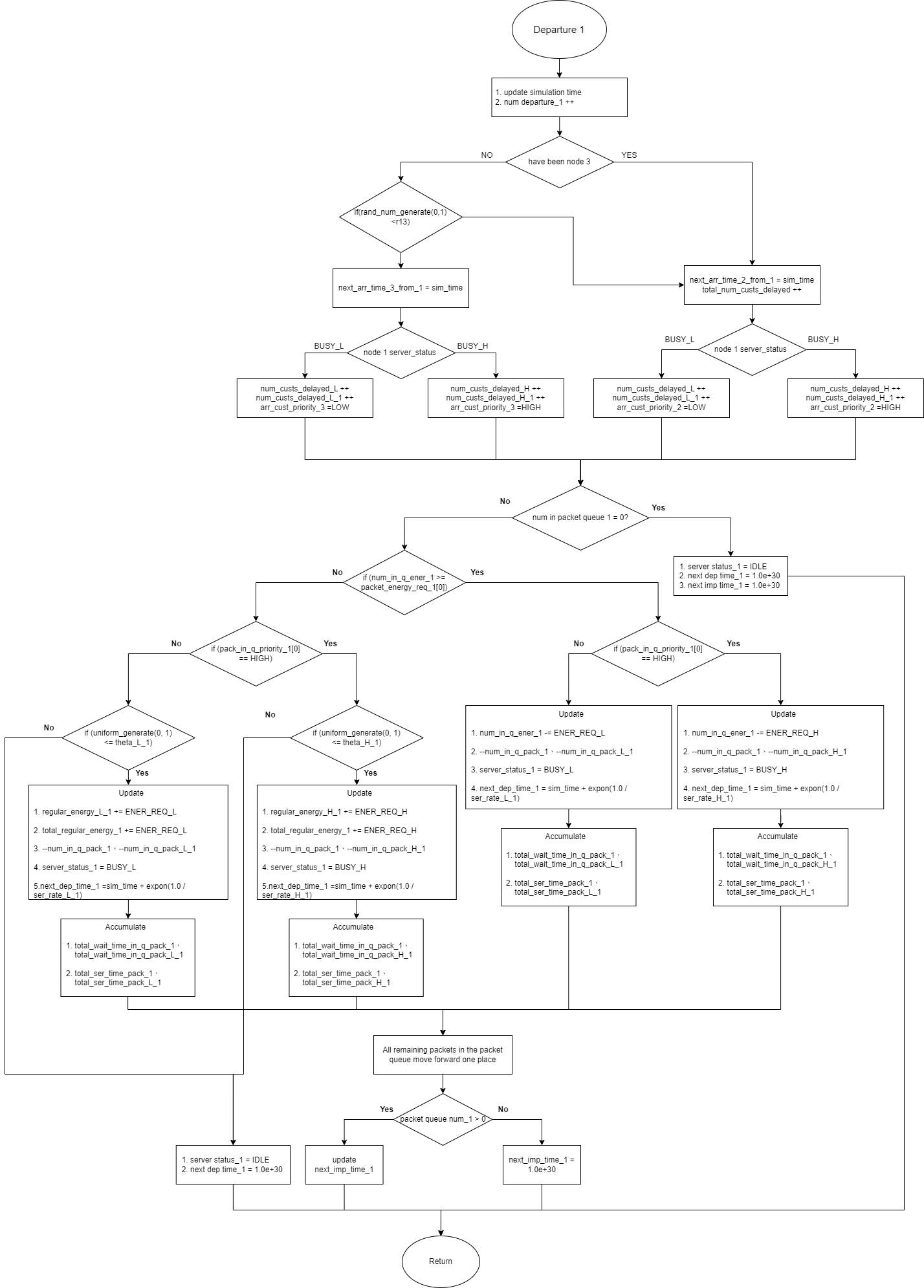


Fig. 4 - 17: Flow chart graphic of node-1 departure subprogram for scenario 2

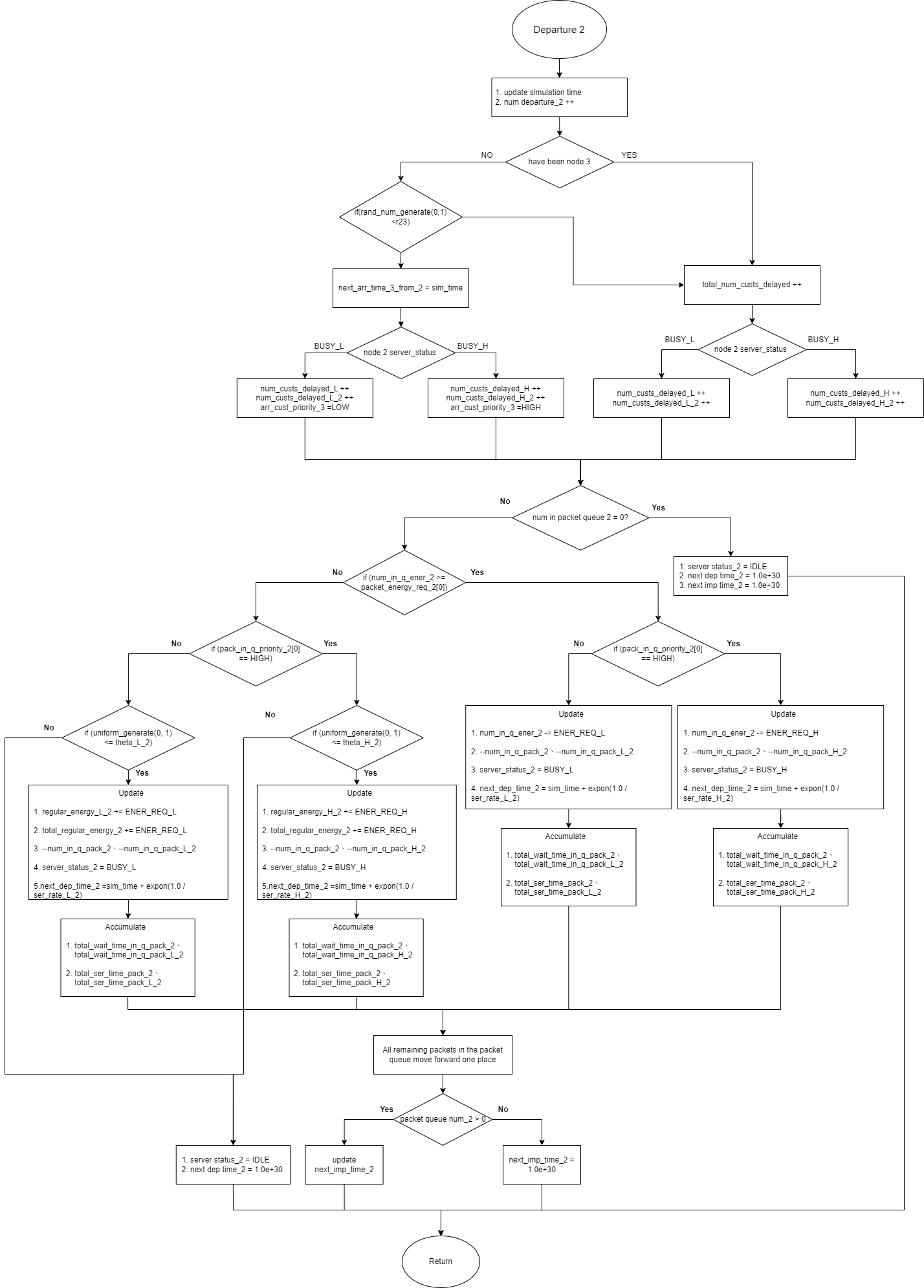


Fig. 4 - 18: Flow chart graphic of node-2 departure subprogram for scenario 2

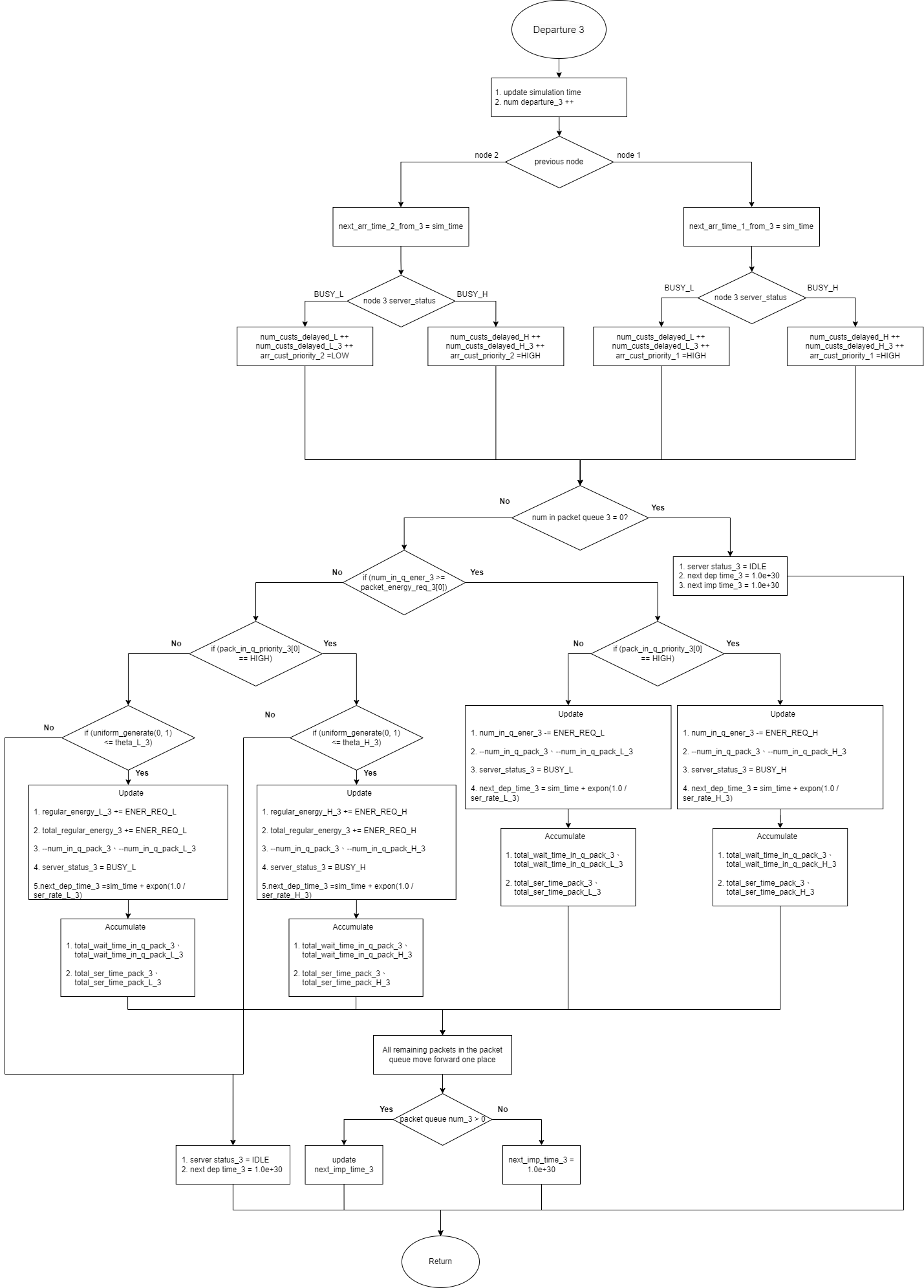


Fig. 4 - 19: Flow chart graphic of node-3 departure subprogram for scenario 2