## Scenario 3

Under this scenario's conditions, we consider a three-connected-node network with two priorities. Both priorities packets require one energy. The packets at node one in this scenario are in batch arrival, which arrive with one or two packets at a time. The energy required for operation can be supplied by either harvested energy or a regular battery. For clarity, we have divided the simulation into one main program and fifteen subprograms.

### Main program

Fig. 4 - 9 displays the flow chart diagram of the main program. The first step involves initializing all the variables that will be utilized. This includes setting the statistics parameters to "0", assigning "infinite" as the next departure time for all nodes, setting the server status of each node to "IDLE," and so on. The following parameters for each node and the network should then be inputted through the keyboard: the packet queue capacity , the energy queue capacity , the mean arrival rate of energy units , the mean service rates of HP and LP packets , the mean impatient rates of HP and LP packets , the probabilities of HP and LP packets using a regular battery , the routing probabilities of and , and the required number of packets.

Furthermore, it is important to note that incoming packets from external sources will only enter the network through the entry node. As a result, we only need to consider the batch arrival rates of one-packet/two-packet HP and LP packets for the first node initially. Subsequently, using the exponential variate function, the next energy arrival time for each node and the next one-packet/two-packet HP and LP packet arrival time for the first node are generated based on the input values. Moving on to the second step, we compare the times of the fifteen events to determine the earliest one, and then execute the corresponding subprogram. In the third step, a "while loop" is implemented to repeat the second step until the total number of served packets matches the required number of packets. Once this condition is met, the simulation concludes, and the performance results are obtained.

### One-packet HP batch arrival subprogram

Fig. 4 - 10 illustrates the flow chart of the subprogram for one-packet HP batch arrival. Initially, the program updates the current simulation time as the one-packet HP batch arrival time and generates the next one-packet HP batch arrival time using the exponential variate function with the arrival rate . The program then accumulates the total number of arrived packets for node 1 and the network, packets in the queue of node 1, and packets in node 1 separately. It checks whether the packet queue in node 1 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. It generates the impatient time using the exponential variate function with . On the other hand, if the queue is full, the program increases the number of blocked packets for node 1 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 1. 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 1 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 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 is insufficient and the random value is greater than , then the next impatient time needs to be updated based on the impatient time of packets in the queue.

### One-packet LP batch arrival subprogram

Fig. 4 - 11 illustrates the flow chart of the subprogram for one-packet LP batch arrival. To begin with, the current simulation time is updated as the one-packet LP batch arrival time and the next one-packet LP batch arrival time is generated using the exponential variate function with . Additionally, the total number of arrived packets for node 1 and the network, packets in the queue of node 1, and packets in node 1 are accumulated separately. The program checks whether the packet queue in node 1 is full, and if not, it adds one packet to the queue and stores the related information of the LP packet at the end of the queue. The impatient time is generated using the exponential variate function with . If the queue is full, the number of blocked packets for node 1 is increased by one. The program then executes the cases that follow.

1. The first step is to check the status of the server in node 1. 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 1 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 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 is insufficient and the random value is greater than , then the next impatient time needs to be updated based on the impatient time of packets in the queue.

### Two-packet HP batch arrival subprogram

Fig. 4 - 12 illustrates the flow chart of the subprogram for two-packet HP batch arrival. Initially, the program updates the current simulation time as the two-packet HP batch arrival time and generates the next two-packet HP batch arrival time using the exponential variate function with the batch arrival rate . The program then accumulates the total number of arrived packets for node 1 and the network, packets in the queue of node 1, and packets in node 1 separately. It checks whether the packet queue in node 1 is full or not. If the packet queue has at least two places available, the program adds two packets to the queue and inserts the related information of two HP packets at the front of all LP packets. It generates the impatient time using the exponential variate function with . If the packet queue has only one place available, the program adds one packet to the queue and inserts the related information of one HP packet at the front of all LP packets, and increases the number of blocked packets for node 1 by one. If the queue is full, the program increases the number of blocked packets for node 1 by two. The program then executes the situations that are divided into the following cases.

1. The first step is to check the status of the server in node 1. 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 1 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 of node 1 to serve the packet at the head of the packet queue in node 1. 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 is insufficient and the random value is greater than , then the next impatient time needs to be updated based on the impatient time of packets in the queue.

### Two-packet LP batch arrival subprogram

Fig. 4 - 13 illustrates the flow chart of the subprogram for two-packet LP batch arrival. Initially, the program updates the current simulation time as the two-packet LP batch arrival time and generates the next two-packet LP batch arrival time using the exponential variate function with the batch arrival rate . Additionally, the total number of arrived packets for node 1 and the network, packets in the queue of node 1, and packets in node 1 are accumulated separately. The program checks whether the packet queue in node 1 is full, and if not and still has at least two places available, it adds two packets to the queue and stores the related information of the two LP packets at the end of the queue. The impatient time is generated using the exponential variate function with . If the packet queue in node 1 has only one place available, the program adds one packet to the queue at the end of the queue, and increases the number of blocked packets for node 1 by one. If the queue is full, the number of blocked packets for node 1 is increased by two. The program then executes the cases that follow.

1. The first step is to check the status of the server in node 1. 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 1 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 of node 1 to serve the packet at the head of the packet queue of node 1. 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, 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 regular battery usage probability of node 1, i.e., . If it is less than or equal to , we update the relevant statuses of node 1 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 is insufficient and the random value is greater than , then the next impatient time needs to be updated based on the impatient time of packets in the queue.

### Node- energy arrival subprogram

Fig. 4 - 14 illustrates the flow chart of the subprogram for node- energy arrival. 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 energy arrival is similar for each node. At the start, we update the current simulation time as the energy arrival time of node and generate the next energy arrival time using the exponential variate function with . Additionally, we accumulate the total number of energy units for node that have arrived. We then check if the energy queue in node 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 for node is incremented by one. Finally, the program executes the cases that follow.

1. We begin by checking the server status in node is "IDLE" or not and whether there are packets in the packet queue of node . If both conditions are met, 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 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 in node . 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 for node is then generated using an exponential variate function with . Then, for the specific node in question (either node 1 or node 2), we temporarily store the information regarding whether the packet at the head of the queue has passed through node 3. On the other hand, for node 3, we store the information about which node the packet originated from. 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 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 node 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 for node using the exponential variate function with . Then, for the specific node in question (either node 1 or node 2), we temporarily store the information regarding whether the packet at the head of the queue has passed through node 3. On the other hand, for node 3, we store the information about which node the packet originated from. In addition, the packets in the queue are shifted one place forward. 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.

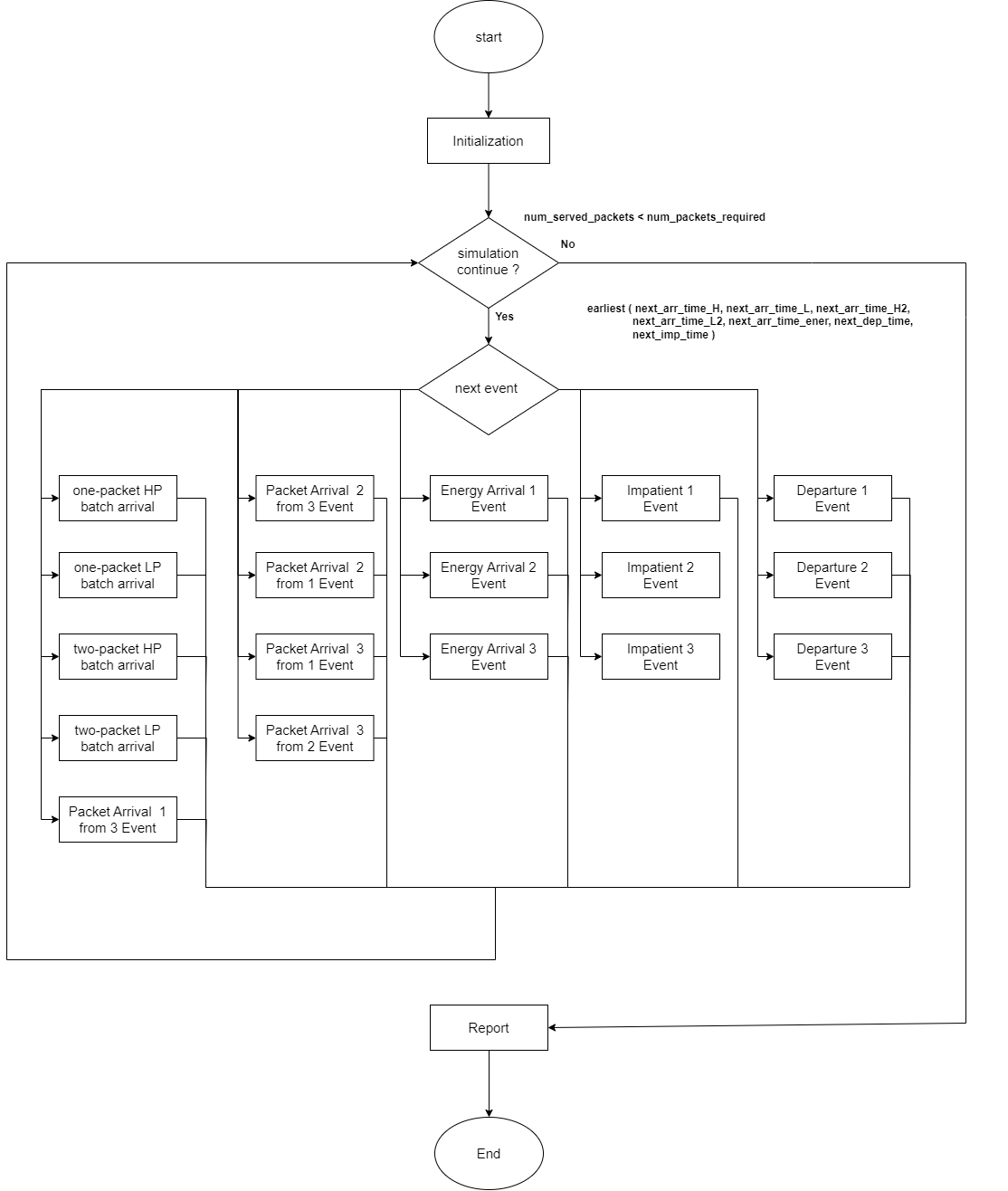


Fig. 4 - 9: Flow chart graphic of main program for scenario 2

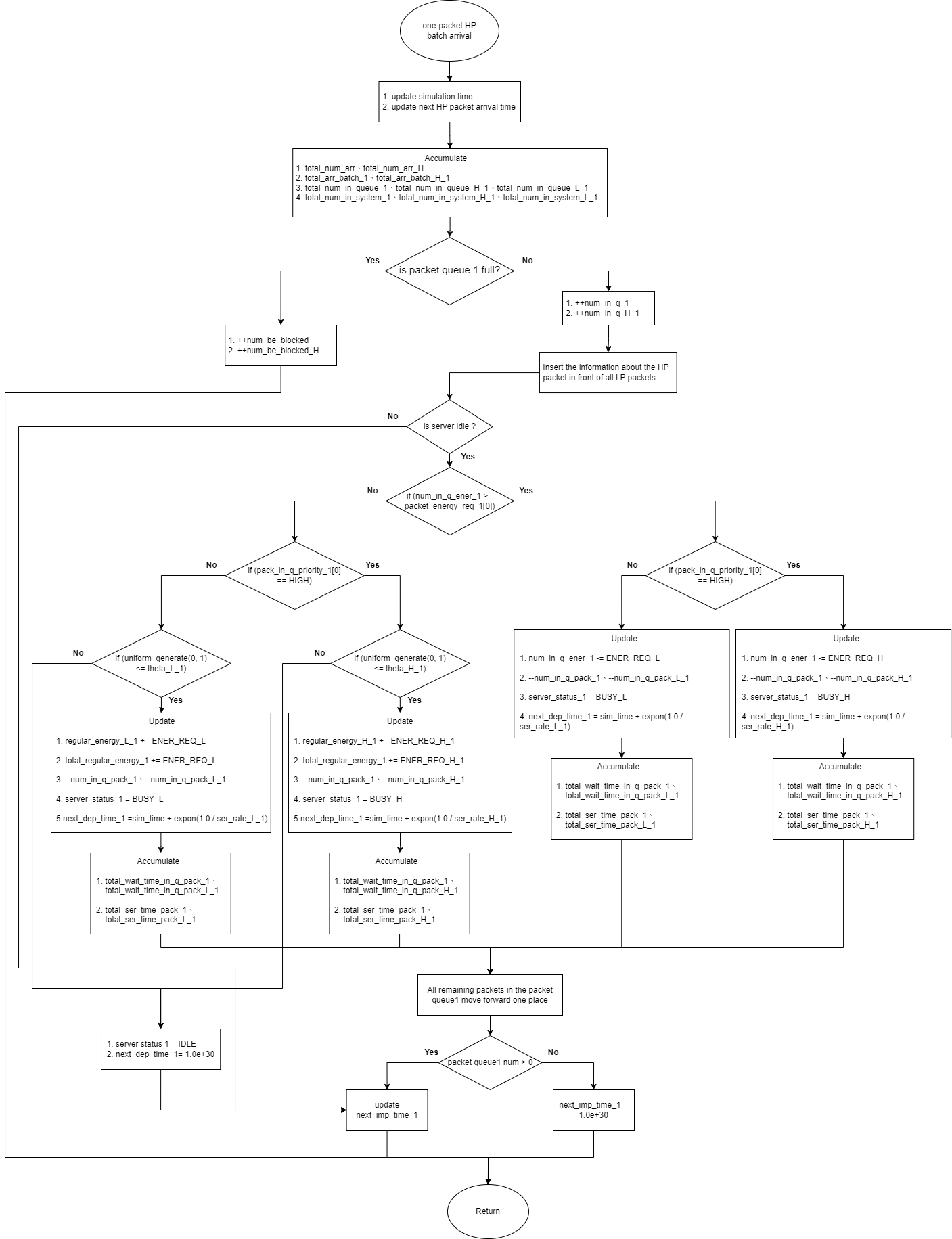


Fig. 4 - 10: Flow chart graphic of one-packet HP batch arrival subprogram for scenario 2

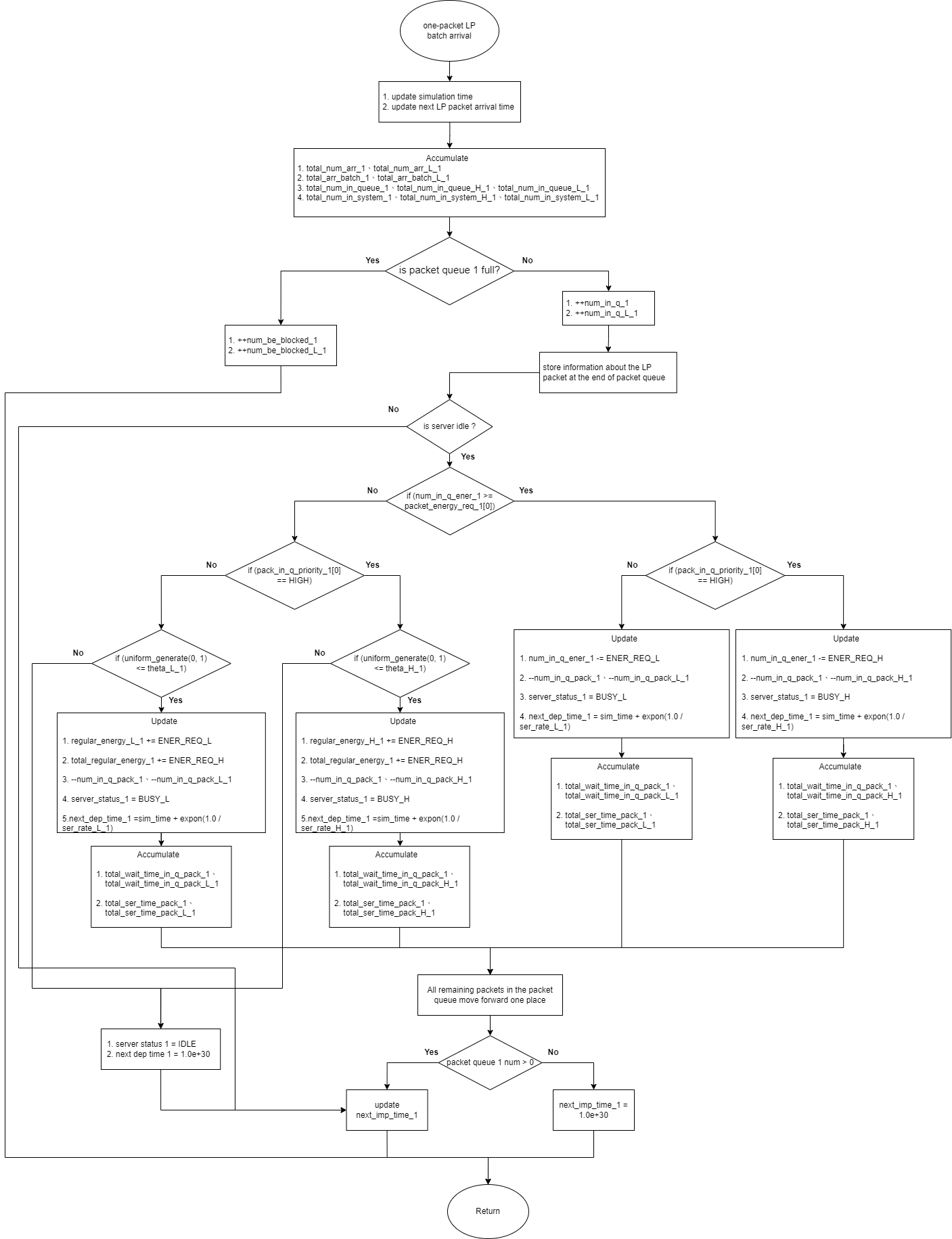


Fig. 4 - 11: Flow chart graphic of one-packet LP batch arrival subprogram for scenario 2



Fig. 4 - 12: Flow chart graphic of two-packet HP batch arrival subprogram for scenario 2

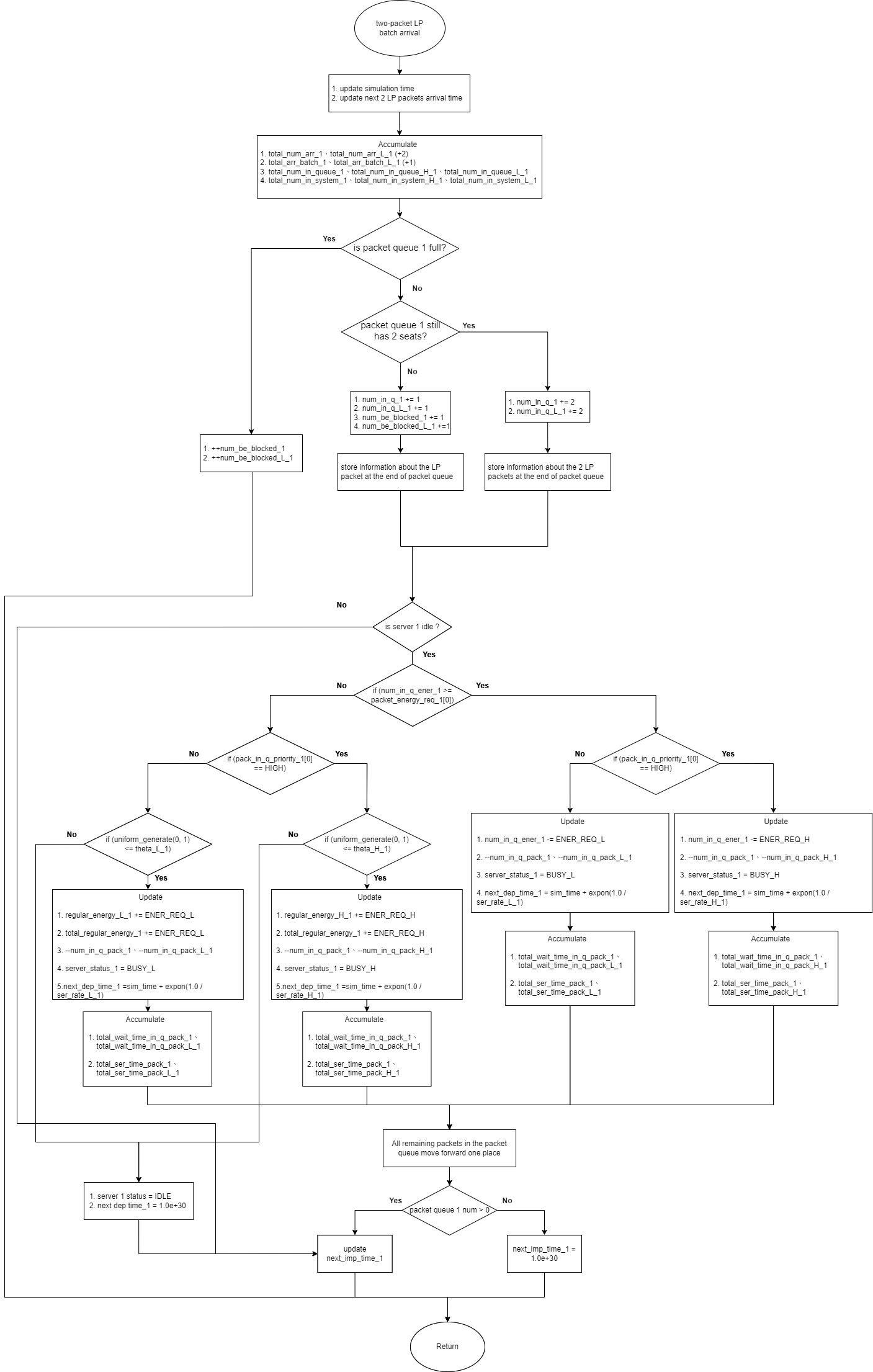


Fig. 4 - 13: Flow chart graphic of two-packet LP batch arrival subprogram for scenario 2

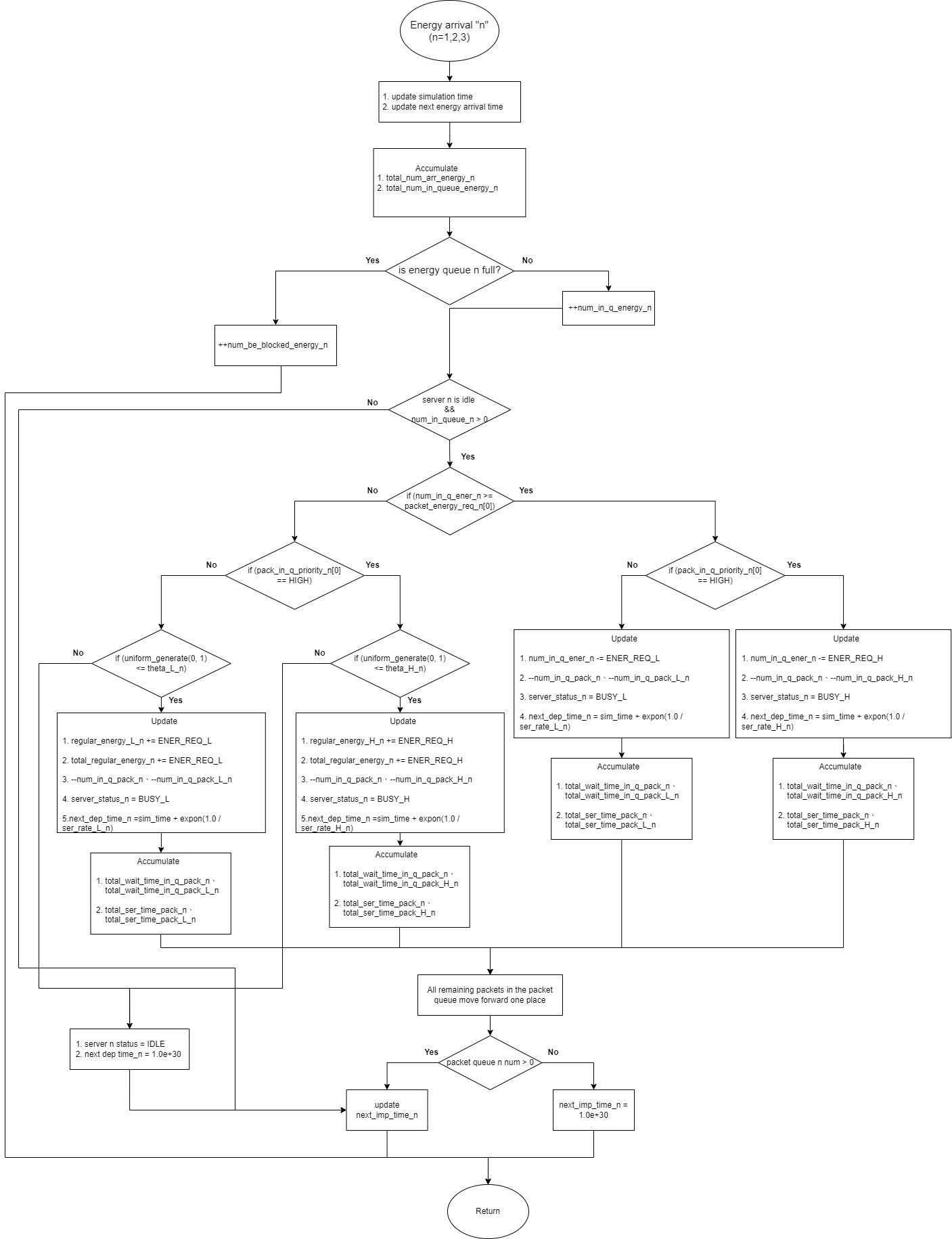


Fig. 4 - 14: Flow chart graphic of node-n energy arrival subprogram for scenario 2