1. There are two goals to this simulation.

The first goal is to decrease the variance in interarrival times of the packets. Packets may arrive at any time. Sometimes, a large number of packets arrive in a short span of time, and at other times a small number of packets take a long time to arrive. Variations in the delay between the arrival of one packet and the arrival of the next causes jitter.

To avoid this, we use this flow control mechanism. Since tokens are generated at a constant rate, packets will only be able to leave at a constant rate. If the rate of token generation is faster than the rate at which packets are arriving, then the problem of jitter will not be solved. However, if the token generation rate is slower, then we can force packets that are arriving too fast to essentially slow down and force all packets to depart at a constant rate, thus solving the problem of jitter. Tokens will not be generated if there are no packets in the queue.

The second goal is to decrease the queue length, and thus the queuing delay. When decreasing the token generation rate, we need to make it just slow enough to avoid jitter. The slower we make it, the longer the queuing delay and the slower the connection will seem. In fact, if we make it too slow, then the queue may become full and we may start losing arriving packets.

1. The state variables are the number of packets in the packet queue, , and the number of tokens in the token queue, .

The output variables are the mean packet delay, , the mean packet queue length, , the mean token queue length, and the mean inter-departure time of the packets, .

1. The events of the simulation are a packet arrival and a token generation. Notice that a packet departure is not a separate event. This is because a packet will depart as soon as it arrives if the token queue is not empty or will depart as soon as a token is generated.
2. If a packet arrives and the token queue is empty, the packet is added to the packet queue. If a token arrives and the packet queue is not empty, then a packet is removed from the packet queue. Thus,

If a packet arrives and the token queue is not empty, a token is removed from the token queue. If a token arrives and the packet queue is empty, the token is added to the token queue. Thus,

The output equations are:

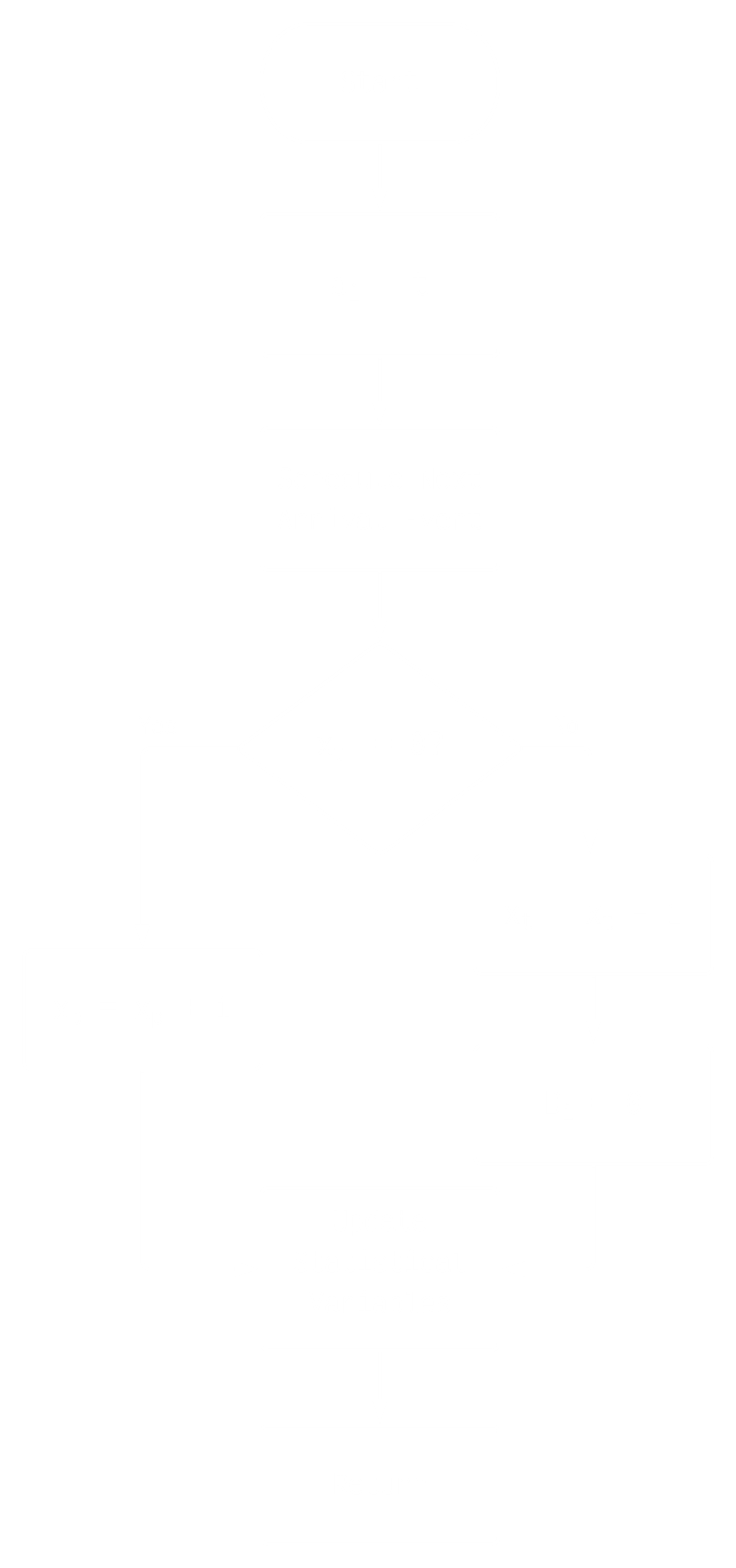
where is the delay for the th packet and is the total number of packets.

where is the packet queue length at time and is the total time for which the simulation runs.

where is the token queue length at time .

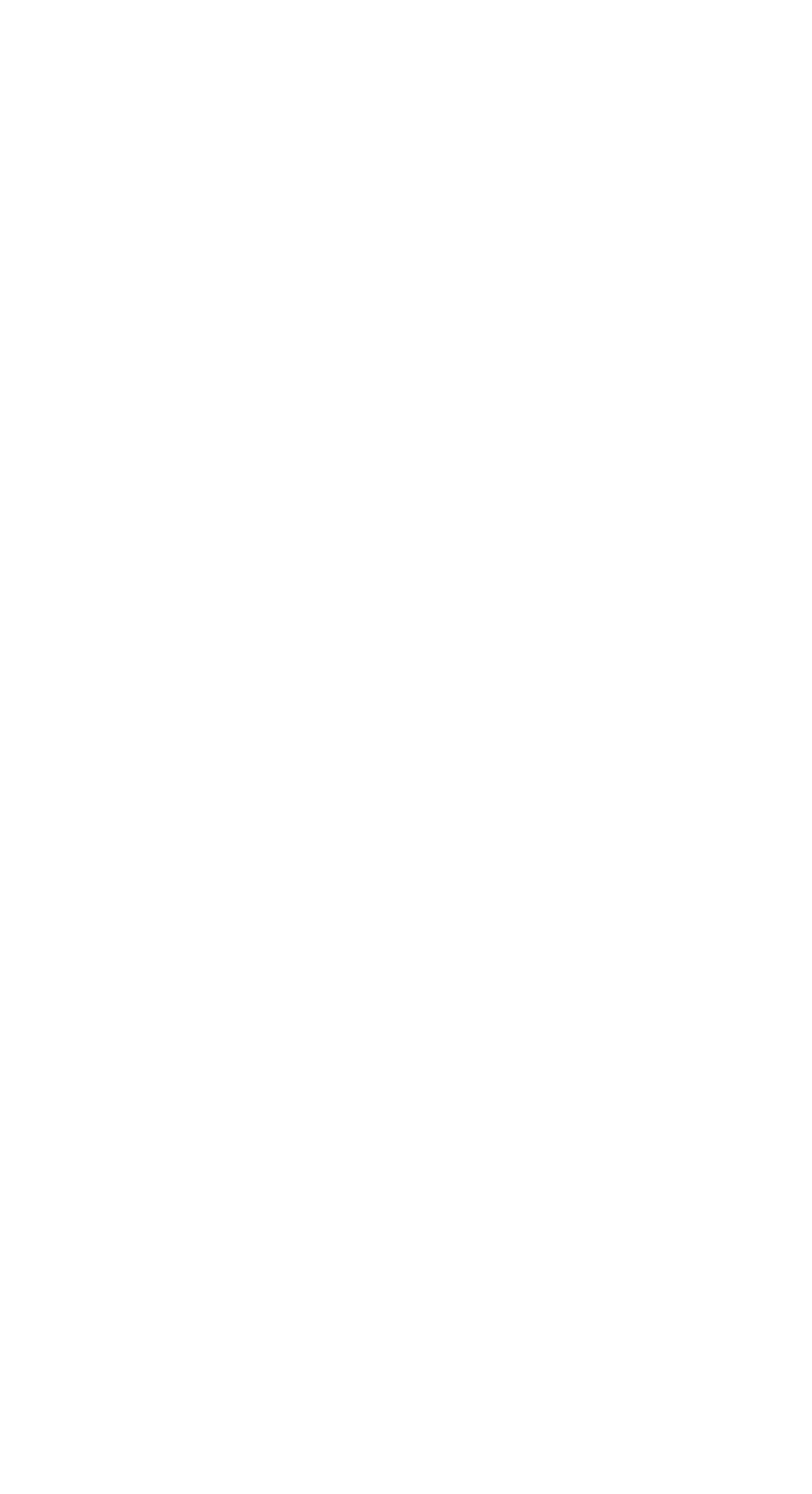
where is the departure time for the th packet.

1. The state space is:
2. Packet Arrival Event



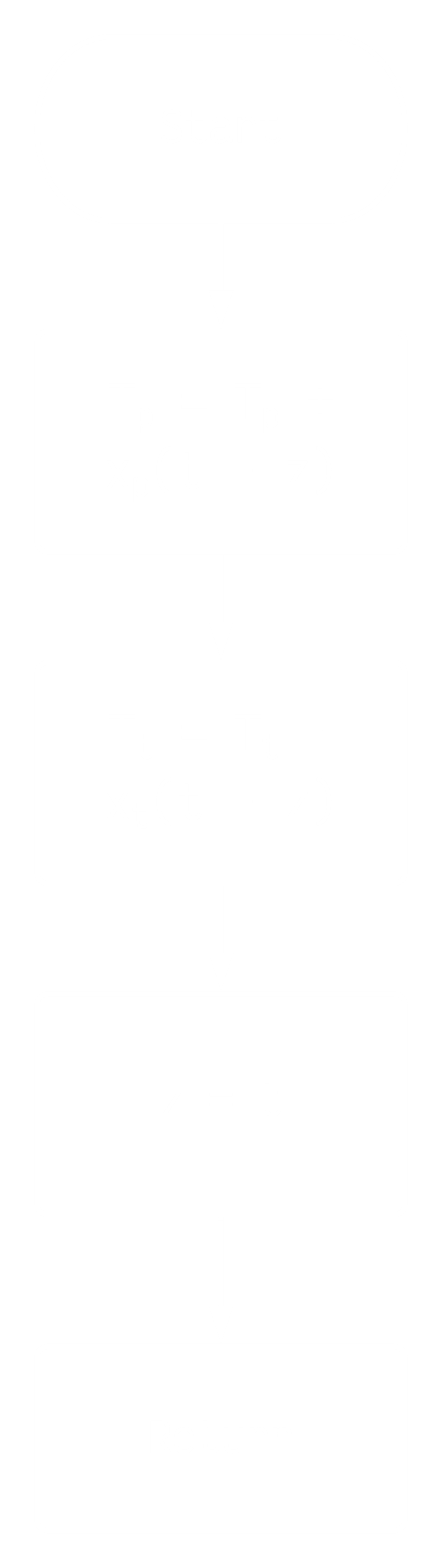
Here, the arrival time, , is being stored for future use. If the packet departs immediately, the departure time, , is set to .

Token Generation Event



Here, is being calculated and stored. It will be used to generate results.

1. When updating statistical variables, we do not need to be concerned about the average departure time or the average inter-arrival time. Since we are storing the departure times for all the events, we can directly calculate these values using the equations provided earlier when generating the final report. For now, we simply need to keep track of the cumulative packet queue and token queue delays, and respectively. We do this with the help of another variable, , which stored the time of the last event.



1. We would need to have two termination events. The first termination event occurs at time . This event would remove any scheduled arrival events. Arrival events are only scheduled under two circumstances, firstly at the beginning of the simulation, and secondly when an arrival event itself occurs, the next one is scheduled. Due to this, removing all scheduled arrival events means no more arrival events can occur.

Once all the packets in the packet queue have departed, the queue becomes empty. In the actual simulation program, the run function detects that there are no other scheduled events and ends the simulation.

Technically, we still need to deal with the token generation, since those events will continue to be produced. However, if we assume that tokens are not generated when the packet queue is empty, then this problem does not occur.

If we assume this though, we run into a further problem. The token generation process will repeatedly need to check that the queue is not empty, which decreases performance.

Instead, we could schedule a second termination event after the last packet in the packet queue has departed. Thus, if we have packets in the packet queue when the first termination event occurs, we can schedule the second termination event time units later, where is the fixed time difference between token generations and is the time of the next scheduled token generation event. This is not simply since the next token generation event might not be exactly time units after the first termination event. That termination event would simply end the simulation.

N.B.: The system given in this question won’t actually be able to stop jitter from occurring, since we could have a large number of tokens in the token queue and no packets in the packet queue. In that case, if a large number of packets arrive suddenly, they would all depart immediately, causing jitter. Thus, the real system would have to use the additional conditions mentioned in part h anyways.