CHAPTER FOUR

AN ELAPSED-TIME BASED SCHEME

4.1 Introduction

This chapter presents the detailed construction of the proposed scheme. At the beginning of the chapter, a framework of the construction of the scheme is given. Then, the phases of the scheme are explained. Next, the algorithm of detecting and mitigating DDoS attacks in SDN is presented. Finally, this chapter is wrapped with a conclusion.

This chapter is organized as follows. Section 4.2 introduces the phases of the proposed scheme. Sections 4.2.1, 4.2.2 and 4.2.3 explain these phases that used to construct the scheme. Section 4.2 ends with the computer algorithm used to detect and mitigate DDoS attacks. Finally, Section 4.3 provides conclusions.

4.2 Phases of the Proposed Scheme

The framework, which shows the processes used to build the proposed scheme, is illustrated in Figure 4.1. The three phases are presented and explained in the following subsections.

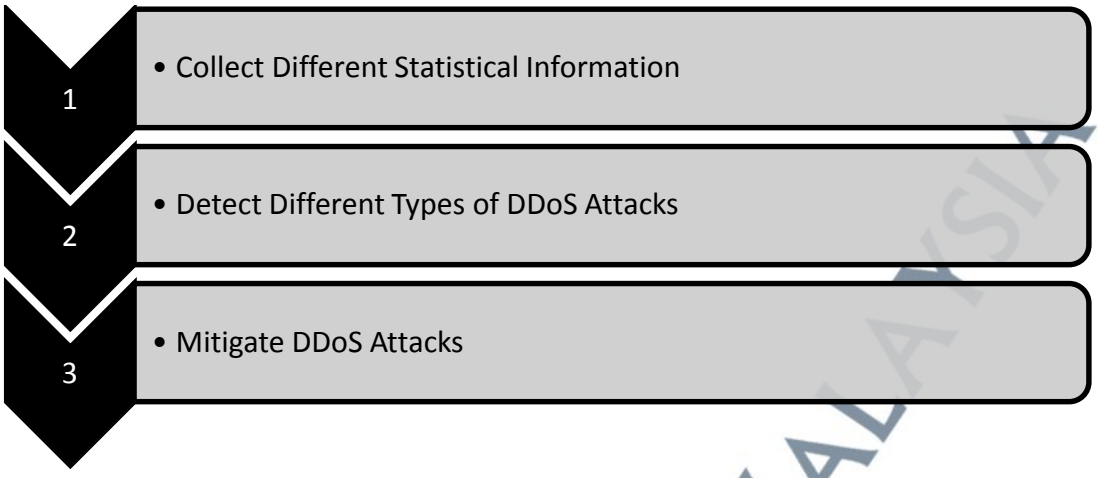

Figure 4.1: Scheme Process

4.2.1Phase One (Statistics Collection)

Upon arrive the packets at the switch; the switch automatically begins processing these packets based on the controller's instructions. The switch begins with applying its matching rules on these incoming packets to find if there is a match. If a match found, the switch applies the actions provided by the controller on these packets. The actions are either processing the packets in the current flow table or passing them to the next flow tables. If there is no matching, the switch applies the action of sending the headers of these packets to the controller. That means that these incoming packets are new and the switch needs to ask the controller how to deal with it. At this moment, our scheme takes place and the first phase begins. Figure 4.2 illustrates how the packet is processed in OpenFlow switch.

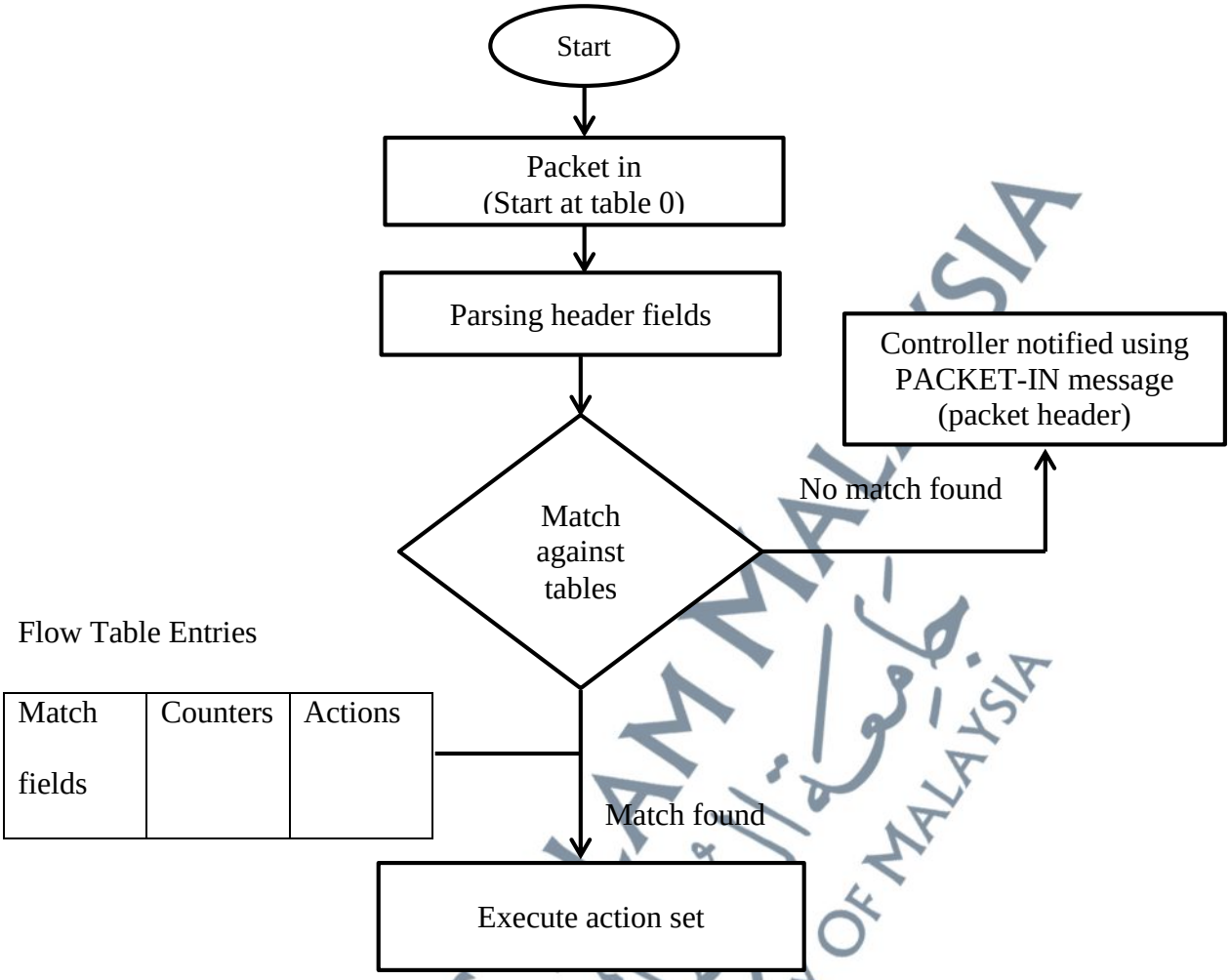

Figure 4.2: Packet Processing in OpenFlow Switch

In this phase, the scheme begins collecting the statistical information namely; number of packets, number of flows, arrival time of each packet forwarded to the controller, time elapsed between these arrival times and destination IP addresses of these forwarded packets every five seconds if the number of packets exceeds the threshold.

Before doing so, the scheme begins by instructing the POX controller to use its functionality of sending inquiry to the OpenFlow switch to know the exact cumulated number in the packet counter. This inquiry will be sent to the switch every five seconds which is a very short time. While the number of packets is collected by

inquiring the switch, the total number of flows is obtained from the switch based on the equation 4.1.

Flow_count = n_f + 1, ……(4.1)

Where n_f denotes the exact number of flows in the switch and starts with a given value 0, and number 1 used to incrementing it by one each time a new flow is created in the switch. The scheme collects the number of packets along with the number of flows from the switch every five seconds periodically.

The scheme applied in this research has two thresholds, one for the large volume of packets and the other is for the low volume of packets that create flows.

When the number of packets exceeds the first threshold, scheme begins collecting the destination IP address of every header of packet forwarded to the controller and save these IP addresses in a separate list called 'ipList' created for the purpose. Meanwhile, the scheme takes the packet arrival time. By subtracting the arrival times of the two packets the previous and the current, the scheme collects the elapsed-time between these successive packets as shown in equation 4.2.

ET = T_Last – T_Previuos,……(4.2)

Where T_Last denotes the time in seconds for the last packet header arrived and T_Previuos denotes the current time in seconds for the previous packet header arrived at the controller. To collect the current time in seconds, we used a Python function called 'time.time()'.

In addition to the importance of collecting the mentioned statistical information for the detection process, collecting these statistics in a short period of

time is important as well to verify any variation in the network traffic when it happens. All together make the controller notified with any sudden increase in both number of packets or number of flows once it occurs. This will lead to detect both attack packets and attack flows and, mitigate them before they become larger and larger.

4.2.2 Phase Two (Attack Detection)

This phase describes the steps taken by the scheme to detect the DDoS attacks at early stage. In this phase, scenarios applied in the all experiments are discussed. The three experiments are different in terms of the type of the DDoS attack, time of attack generating and scale of packets generating. The type of DDoS attack used in the first experiment is UDP flood attack, low rate SYN attack is the type of attack used in the second experiment. A mixture of a UDP flood attack and low rate SYN attack is used in the third experiment along with normal traffic. The attack traffic will be generated in a short and long terms. Two scales of packets generating are used in in the first and second experiments: a scale of generating packets in a large number and another scale in a small number. These two scales are used to represent the high and low attack traffic rates that the attacker uses to move from the high to low and vice versa during the time of the attack. These two scales are applied in the all second test cases in the first and second experiments. The test cases used in these experiments are: normal traffic generating, attack traffic generating and a mixture of normal and attack traffic generating. The three test cases are used to determining if the scheme works properly and to ensure the correctness of the results, particularly the third test case when the traffic is mixed between normal packets and attack packets. The third test case is the highest level of testing for the scheme where the functions of the scheme are putted under different scenarios to examine their

efficiency in distinguishing the attack packets from legitimates and keep the network resources available to the legitimate users.

The two scales of generating packets applied in the second test case are meant to determine if the scheme's detection function can detect the flooding DDoS attack packets weather they are coming in a large number of packets or a small number. In order to do that, monitoring any sudden increase in the number of packets in the network is important as same as monitoring the sudden increase in the number of the flows.

When the number of packets in packet_count parameter exceeds the threshold (1500 packets), the scheme begins with creating an empty list to contain the destination IP addresses. This number of packets has been selected as a threshold value after running a series of simulations to find the optimal threshold. The simulations cover an attack from one attack machine to one victim and a group of attack machines to one victim. After comparing the results, we found that this limit is the proper limit of the number of packets that fit our network topology as well as to give more protection to the controller. Moreover, this threshold can be changed to be higher or lower than this value based on the type of the network whether it is a large or a small network. These destinations IP addresses that were collected from the packets' headers arrived at the controller are going to be saved in this list. Meanwhile, the scheme takes the arrival times for each two successive packets and put them in two different variables called T_Last and T_Previuos. The scheme then begins with checking the similarity of the destination IP addresses in the list. Checking for similarity means comparing each IP address listed in the list with the previous IP to see if they are the same. If the similarity between the destinations IP

addresses under checking is positive (this means that these packets have been sent to the same destination), the process goes forward to check if the interval between arrival times is less than thirty seconds ($TE \leq 30$). TE represents the time elapsed between each two successive packets. Otherwise, the scheme will back to the packets threshold step.

It is important to mention that the number of seconds selected to be the maximum time elapsed between packets was selected after the network topology was tested under different conditions in terms of the number of attack devices and time of generation of these attacks. What has been observed from the different experiments conducted for the purpose is that the maximum distance in time between the successive packets is ten seconds. We have increased the maximum number of packets as an expectation of occurrence of bottlenecks or congestion in the network in some cases.

After the scheme checks the interval between arrival times, it determines if the time is less than 30 seconds or not. If it is less than 30 seconds, then these packets will be considered as flooding DDoS attack packets. These suspicious packets will be dropped directly from the controller and the flows contain these packets will be removed from the switch. In the case of interval between arrival times is greater than thirty seconds, the scheme will back to the step of checking the IP addresses similarity to apply it again on the next destinations IP addresses in the list. If the similarity is negative, the scheme will back to check the number of packets in the count_packet parameter.

If the number of packets in packet_count parameter does not exceed the threshold, the scheme will shift to the second part of the detection function which is

checking the number of flows in flow_count parameter. If the number of flows exceeds the threshold (flow_count ≥ 10), the scheme will proceed to the attack flows detection. This part of detection is meant to detect the attack flows that created by either the change in the volume of packets from high to low or the low rate attack packets. If the number is less than the threshold, the scheme will back to the step of checking the cumulative numbers of packets and flows in both packet_count and flow_count parameters respectively. The attack detection process is appeared in Figure 4.3.

Because DDoS attack sends packets with a forged IP address, the number of single-flows in the network is getting increased. Also, sending DDoS attack packets at low rates can make the number of flows skyrocket. After a series of experiments, we found that the optimal threshold for detecting single flows in the network must be greater than or equals ten flows. Keeping single-flows under this threshold will keep the OpenFlow switch safe. Protecting the switch flow tables from being filled up with attack flows that contain one packet is a full protection to the SDN controller from receiving unprocessed requests from all over the network.

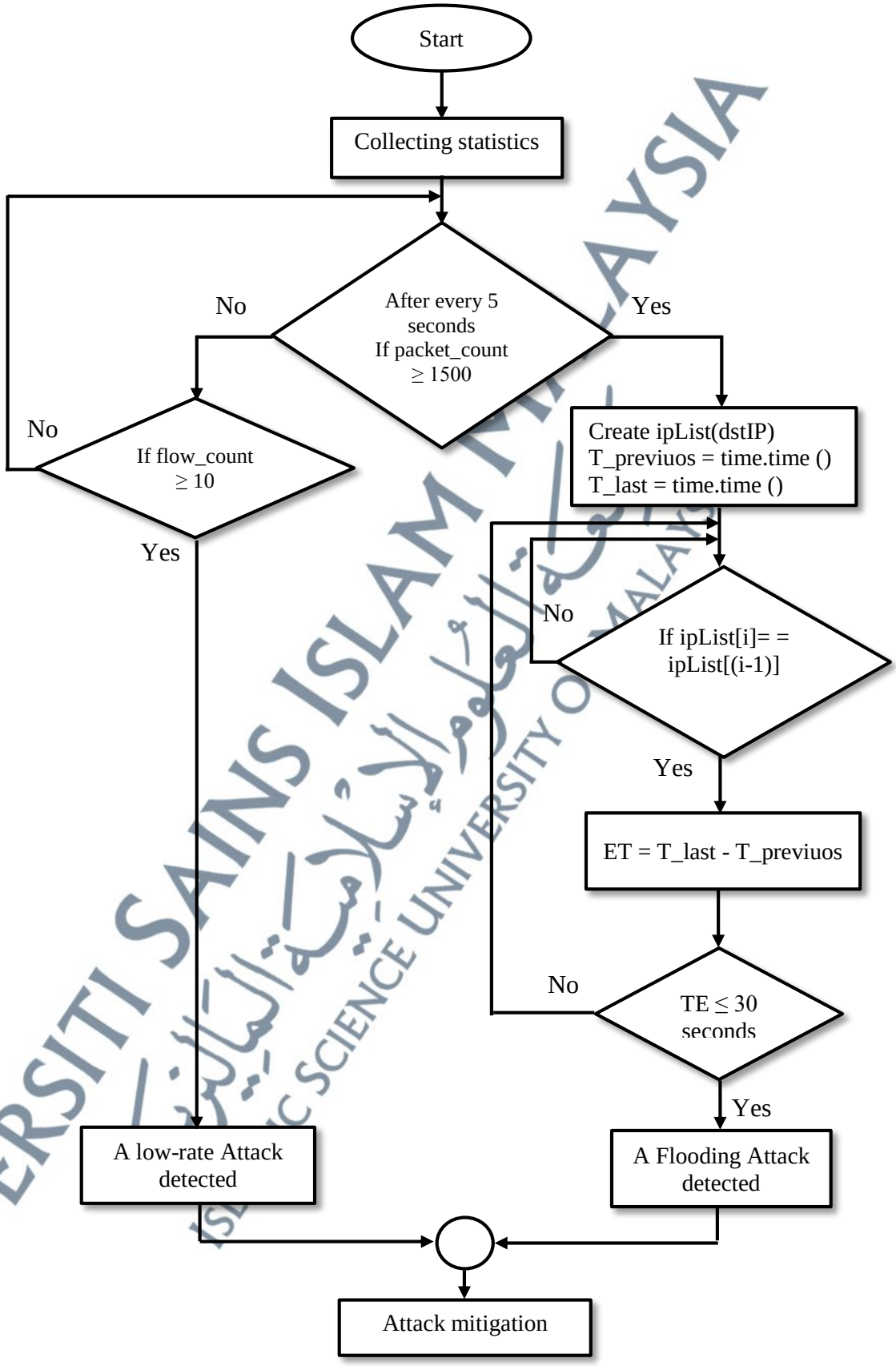

Figure 4.3: Attack Detection

4.2.3 Phase Three (Attack Mitigation)

Mitigation is an important step to make in order to protect the network from DDoS damages. Without mitigation the attack detection is useless. Mitigation process should be activated directly after the attack is detected and applied efficiently to achieve the purpose of protecting the SDN components.

The mitigation function in this scheme is meant to mitigate different DDoS attacks in early stage. Early mitigation means dropping the attack packets and removing the attack flows once the attack occurs. In this research, we concentrate on the increase in the number of new packets forwarded to the controller and the increase in the number of new flows created in the switch to mitigate the attack.

When the attack is detected as appeared in Figure 4.3, the mitigation function will be activated as shown in Figure 4.4. The suspicious packets detected will be dropped directly from the controller and the flows contain these packets will be removed from the switch. The scheme removes the flows by making the values of idle-timeouts and hard-timeouts flow entries in the switch flow table equal to zero and one respectively. Thus, the temporary memory reserved for these attack packets by the switch will be offered to the new coming packets instead. This will give both the controller and switch more protection.

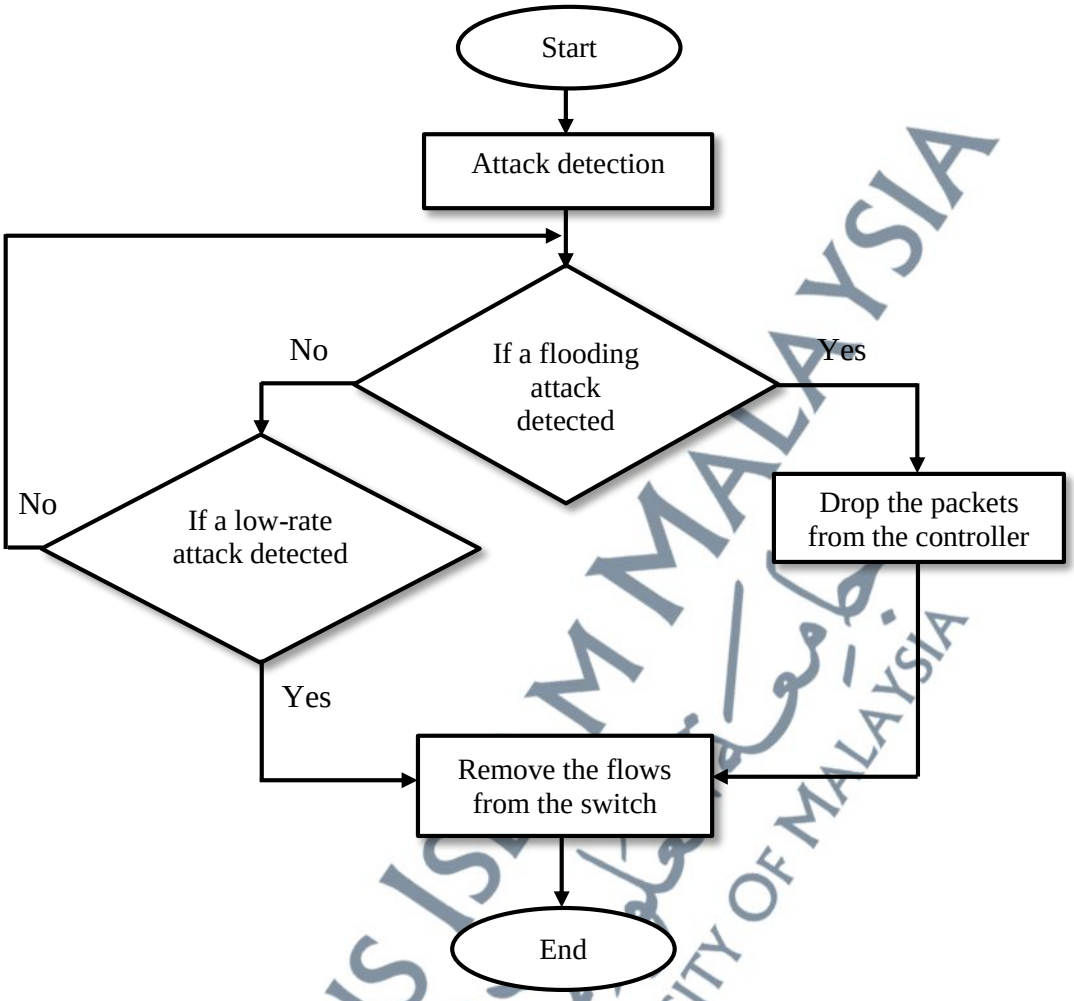

Figure 4.4: Attack Mitigation

The three phases of the proposed scheme are constructed by Python programming language and ran on a Linux Ubuntu operating system. The scheme constructed is implemented on POX controller. The algorithm used for detecting and mitigating DDoS attacks in SDN is shown in table 4.1. The three phases of the proposed scheme is appeared in a complete flowchart in Figure 4.5.

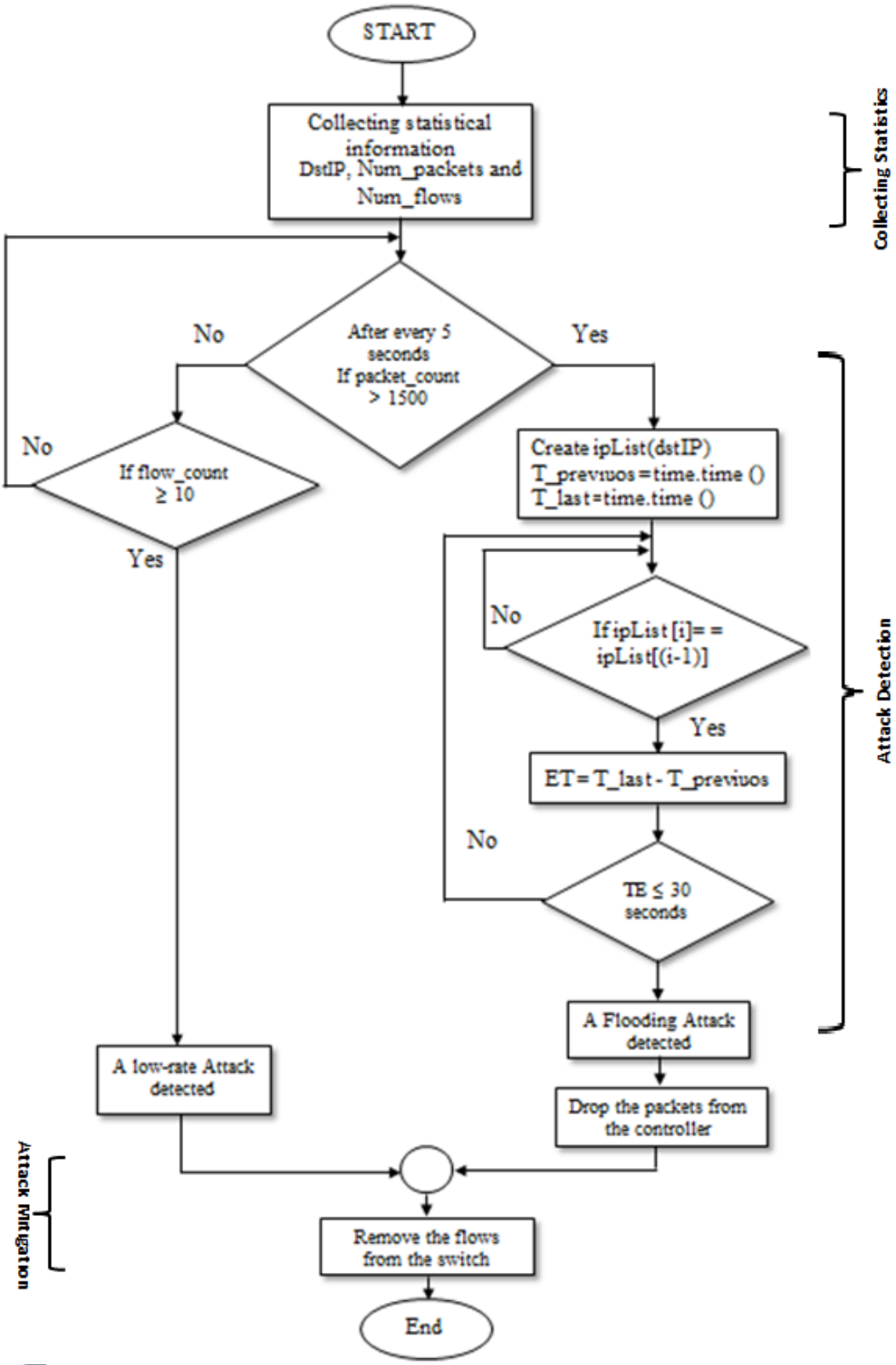

Figure 4.5: An elapsed-time based scheme

Table 4.1: The Algorithm of The Elapsed-Time Based Scheme for Detecting and Mitigating DDoS attacks in SDN

The Algorithm of Detecting and Mitigating DDoS attack in SDN

Input: dstIP, number of packets, number of flows, T_Last and T_Previuos
Output: flooding and low-rate DDoS attacks detected and mitigated

1. Pox.py/* turn on POX controller/
2. Pox.py forwarding.l2_learning/* invoking POX to make this component use OpenFlow switch as l2_learning switch/
3. Create a dstIP list set ipList:= ∅,
   Create a packet counter packet_count:= 0,
   Create a flow counter flow_count:= 0,
   Create packets arrival time T_Previuos and T_Last for each two successive packets
   T_Previuos:= 0, T_Last := 0
4. ∀ f in event.status:
   packet_count +:= f.packet_count
   flow_count +:= 1
   T_Previuos:= time.time()
   T_Last:= time.time()
   TE:= T_last - T_previuos
   if packet_count ≥ threshold 1:
   ipList:= ipList(dstip)
5. ∀ i in range(1, len(ipList)):
6. if ipList [i] = = ipList [i - 1]:
7. if TE ≤ max time between successive packets:
   print: attack is detected from srcIP() on dstIP()
8. Drop
9. elif n_flows ≥ threshold 2 /* p_count < 1500 */
   print: attack detected
10. Remove flows

4.3 Summary

The phases of the scheme construction are discussed in this chapter. The three phases are explained in details and the flowcharts of these phases are presented. Finally, the computer algorithm is presented.