Explanation of algorithm : D-ARPSpoof

1 Data Structures Used

1. HashMap<DatapathId,HashMap<OFPort,<Vlan, IP, Mac>> : portIpMap
2. HashMap<MacAddress,Switch-Port> : macPortMap

2 Handling Packet-IN DHCP Messages

2.1 Handling Packet-IN DHCP Messages

2.1.1 Updating Data Structures

- **DHCP Request**

  When we get a DHCP request from a port of switch it indicates that a switch is connected to any switch in the network. Because DHCP request is broadcasted all the times, for the first we observe DHCP request at controller from a specific MAC address, it is the switch whit which host is directly connected. Because DHCP request signifies that we are attepting to get a new IP address, that means old IP is not in use. So, we don't know if server will be able to perform its request or not. hence first we delete any entry related to that from data structure from the key as the current switch id.

  Now, the next time if we get the same request from other switch due to broadcast, we check if macPortMap contains any entry(of any switch id) with same source mac.

  - If it is not there, then we update the MAC table as this indicates this is the switch with which host is directly connected to.
  - If it is there, we don't do any updation because it is just same request broadcasted from other switch.

  This method ensures that there is no duplicate entry in data structure on any port binded with a mac address of a host in data structure.

- **DHCP ACKNOWLEDGEMENT**

  From the macTable we get the entry destination mac of DHCP acknowledgement, that should be host's MAC address which requested for it. Above mechanism ensures that we have the correct entry(at which switch, port the host is attached to) for host which has requested the IP address. We extract the information from the data structure and we update IP table with IP address dhcpPayload's your IP address field that is what DHCP server provides.

In this way, we make sure that we have correct entry for host's mac and IP and where it is attached to. We can use this information while forwarding ARP packets.

2.1.2 Updating Flow Rules

- **DHCP REQUEST**

  If ip table has the entry for current switch and inPort then, this indicates that we also have flow rules installed for them in switch. Because DHCP request indicates either connection of new host or acquiring a new IP address, these flow rules are outdated now, and should be removed. So, we remove any flow from the switch which has match with inPort and current inPort and ether type ARP, because we install flow rules for ARP only.

- **DHCP ACKNOWLEDGEMENT**

  When we get a DHCP-ACK it signifies that we get a new IP for the host, and now if we get any arp request other than this source IP address, that arp request should basically be blocked. And to achieve this we actually block any arp request from that port with priotiry 10, and allow only legal IP address with priority 20. Also, because we need to handle arp ourselves, the arp forwarding class does so, we need to again operate on the packet, we send those packet to table-1 of openflow pipeline. In the first table of openflow pipeline we would install the arp forwarding rules.

2.2 Handling DHCP ACK Packet-Out message

Because DHCP-ACK can be generated by a specific server in network as well as SDN itself, so if it is generated by a specific server, then we get DHCP ACK as packet-in otherwise as Packet-out, so, this is same as previous.

2.3 Handling ARP

As, genuine ARP packets are redircted to table 1 of openflow pipeline. We don't need to check the authenticity of those packets. Now, if we get any ARP packet at controller, we can directly utilize our data structure to get the attachement point of source and target. Also, routing service would provide us the optimal path from one switch to other switch. So,we don't need to mess up with the path much.

So, there are 2 cases can happen using routing service:

- If there is no path between the switch, we get path of infinite length.
- If there is path then we get a path with finite length.

So, first we check that if target IP is present in our data structure or not, and it is in the same vlan as request packet is. If it is not the case, we drop the packet, as communication is not possible.

Otherwise, we get the attachement point of target, and attachement point of source. And then we check patch between the source and target switches(that we got from our data structure) using Routing service.

If we get a path of infinite length, this signifies that there is no path between those switches, hence we drop the packet.

If we get a path of finite length, we install the rules in all the switches we got from routing service for forwarding.

NOTE: Routing service provides us a list of switch port pair in path

Also, we push a packet out to the first switch in path so, that packet is transmitted to correct switch and host does not have to generate the packet one more time.

Using this mechanism, we ensure several things:

- ARP packet of a certain kind is transmitted to the controller only once, because next time if packet goes to other switch, we already installed the rule on there.
- We eliminated the broadcasting case of ARP packets, as in classical forwarding module we needed to broadcast the packet because, we didn't know the attachement point of target. However, in our case, we know the attachement point already we utilize that and reduce the broadcasting case.

2.4 Handling Switch Added event

We add rule to forward all ARP packets to controller in flow table 1 of added switch with priority 0. Because ARP forwarding rules are there in flow table 1, and floodlight doesn't install this rule by default in flow table other than 0. So, we need to explicitly install it.

2.5 Handling Switch Removed event

Because, any entry related to that switch is now invalid in our data structure so, we just clean our data structure.

2.6 Handling link update

If we get any direct link between source and destination switch-port pair, and any of those exist in our portIPMap, it signifies that instead of a host a switch is connected to that port. So, we delete any flow regarding that port as inPort from switch, as those flows are now invalid as those were written as if those ports are connected with a host not with a switch. nd we clean our data-structures too for the same reason.