Set network topology

函式 set_topology() 會根據 setting.py 的 get_hosts(), get_switches() 設定對應的裝置，get_ip(), get_mac() 設定設備的 ip 和 mac address，get_link() 則是會在設備之間建立連線。函式 run_net() 則是用來接收指令，並分析指令中關鍵字來執行對應功能。

Function in class host

```
def ping(self, dst_ip):
    # handle a ping request

    if dst_ip in self.arp_table:
        # send an ICMP packet
        self.send(Packet(self.ip, dst_ip, self.mac, self.arp_table[dst_ip], self.name, 'icmp_request'))
    else:
        # broadcast an ARP request
        self.send(Packet(self.ip, dst_ip, self.mac, 'ffff', self.name, 'arp_request'))
        self.send(Packet(self.ip, dst_ip, self.mac, self.arp_table[dst_ip], self.name, 'icmp_request'))
```

```
def send(self, packet):
    # determine the destination MAC here
    '''
        Hint :
            if the packet is the type of arp request, destination MAC would be 'ffff'.
            else, check up the arp table.
    '''
    node = self.port_to # get node connected to this host
    node.handle_packet(packet)
```

例子: h1 ping h2 時，h1 會先檢查 h2_ip 是否在自己的 arp_table 中，如果有的話，送 packet_type='icmp_request' 的封包給 h2。若 h2_ip 不在 h1 的 arp_table 中，那麼換先送一個 arp_request 給 h2，等到收到 h2 回復的 arp_reply，h1 會再送一個 icmp_request 給 h2。

```
def handle_packet(self, packet):
    # handle incoming packets
    if packet.dst_mac == self.mac:
        self.update_arp(packet.src_ip, packet.src_mac)


    if packet.packet_type == 'arp_request':
        # print(f'{packet.src_ip} to {packet.dst_ip} : arp_request')
        if packet.dst_ip == self.ip:
            self.send(Packet(self.ip, packet.src_ip, self.mac, packet.src_mac, self.name, 'arp_reply'))
    elif packet.packet_type == 'icmp_request':
        # print(f'{packet.src_ip} to {packet.dst_ip} : icmp_request')
        if packet.dst_ip == self.ip:
            self.send(Packet(self.ip, packet.src_ip, self.mac, packet.src_mac, self.name, 'icmp_reply'))
```

例子: 假設目前封包在 s1，要轉送給 h2。

s1 呼叫了 send() 之後，會通知 h2 去收，h2 會先檢查 dst_mac (destination mac address) 與自己的 mac address 是否相同，若相同的話，則更新 h2 的 arp_table，紀錄 src_ip 和 src_mac 的對應關係。

若封包中的 dst_ip 與 self.ip 相同，則依據封包的 packet_type 回傳對應的封包。

Function in class switch

```
def handle_packet(self, packet):
    # handle incoming packets

    # find the port to sending_device
    for i in range(self.port_n):
        if self.port_to[i].name == packet.sending_device:
            incoming_port = i
            break
    else:
        incoming_port = -1

    self.update_mac(packet.src_mac, incoming_port)
    # self.show_connected_devices()

    if packet.dst_mac == 'ffff':
        # broadcast the packet to all ports except the incoming port
        for i in range(self.port_n):
            if i != incoming_port:
                self.send(i, packet)
    elif packet.dst_mac in self.mac_table:
        self.send(self.mac_table[packet.dst_mac], packet)
    else:
        # broadcast the packet to all ports except the incoming port
        for i in range(self.port_n):
            if i != incoming_port:
                self.send(i, packet)
```

封包中 packet.sending_device 紀錄的是從哪個裝置送過來的，迴圈中檢查所有 port 連出去的裝置名稱是否與 packet.sending_device 相符，與名稱相符的 port 則稱為 incoming_port。

若 dst_mac='ffff' ，那麼就 broadcast 到 incoming port 之外的所有 port，若 dst_mac 在 mac_table 中，則根據 mac_table 送到對應的 port。假如不符

合上面兩種情形，還是 broadcast 到 incoming port 之外的所有 port。

Answer Questions

1. Difference between broadcasting and flooding
   Flooding may send the same packet along the same link multiple times, but broadcasting sends a packet along a link at most once.
2. The process of h1 ping h7 :
   因為 h1 中 arp_table 是空的(不清楚 h7 mac address)，所以先送 arp_request 出去(dst_mac='ffff')。而 s1 從 port=0 收到來自 h1 的封包，但 s1 不清楚送到 h7 要往哪個 port，所以用 broadcast 的方式送到 "除了 port=0" 的 所有 port (all ports except incoming port)。
   送往 h2 的那個封包，因為 dst_ip, dst_mac 與 h2 不符，因此被丟棄掉(後續到錯誤的 host 都會被丟棄掉)。

   送往 s2 的封包會再度被 broadcast 出去，抵達 s7 之後也會再度被 broadcast 出去，一路在經過 s5, s6，最終抵達 h7。

   路徑: h1 -> s1 -> s2 -> s7 -> s5 -> s6 -> h7

3. 連接 s2 和 s5 之後，若 s2 或 s5 broadcast 之後找不到目的地，會持續 broadcast，封包的數目會在 switching loop 中不斷增加，並消耗掉頻寬。Spanning Tree Protocol 可以暫時關閉某些 port，使得拓樸維持著 tree 的架構。