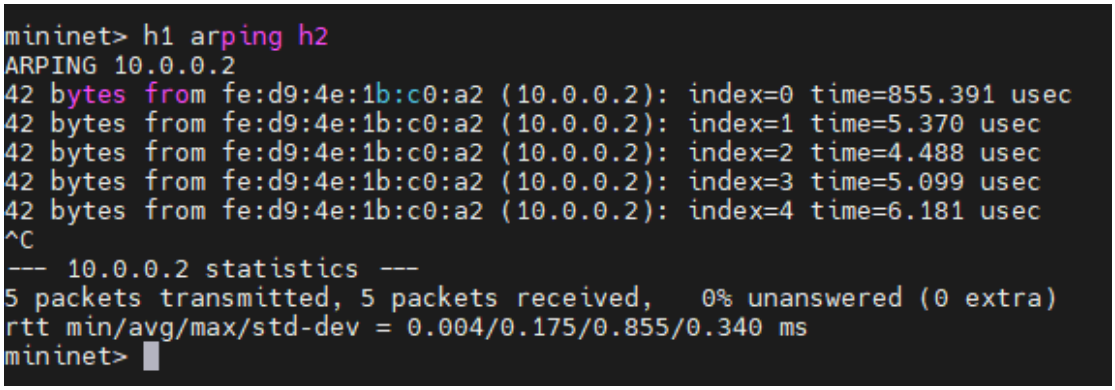
Part 1:

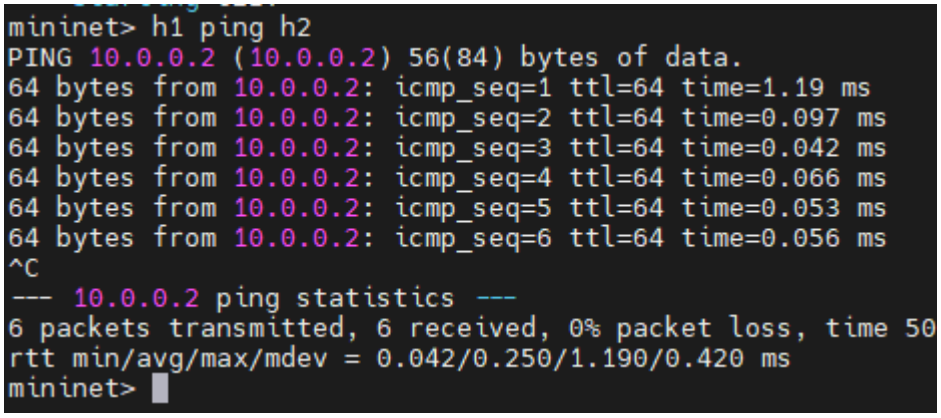
Match fields	Actions	Timeout values
ETH_TYPE=ARP	Output port=Controller	0
ETH_TYPE=BDDP	Output port=Controller	0
ETH_TYPE=LLDP	Output port=Controller	0
ETH_TYPE=IPV4	Output port=Controller	0
IN_PORT=2	Output port=1	0
IN_PORT=1	Output port=2	0

Part 2:

ARP PING



Ping



Part 3:

1. h1 Data plane: h1 準備 ping h2，但其 ARP cache 中沒有 h2 的 IP 到 MAC address，所以 h1 要發一個 arp 到 s1 跟 s1 問 h2 的資訊
2. s1 data plane: s1 收到 arp 但發現也沒有 h2 mac address cache，所以把 arp 存到 PacketIn 並透過 OpenFlow portal 去問 ONON controller
3. ONOS control plane: ReactiveForwarding 收到 packetin，拆開後發現是 ARP

boradcast，PacketOut 訊息給 s1，命令 s1 將這個 arp 請求從除了來自 h1 之外的 port 發送出去

4. s1, h2 Data plane: s1 根據 PacketOut 指令，將 arp request forward 給 h2，h2 arp response 它的 Mac address 會經過 s1，但 s1 也沒有 flow rule 到 h1，所以再次將 ReactiveForwarding 封包給 onos
5. ONOS control plane: onos 收到 ReactiveForwarding，ONOS 透過 HostService 已經知道 h1 的位置，因為 h1 剛剛送 arp request 時，ONOS 就記錄了其 MAC address、IP 和所連接的 switch port。現在收到 h2 的回覆，ONOS 也學習到了 h2 的位置
6. S1 h1 data plane: s1 根據 ONOS 的 Packet-Ou 指令，將 ARP 回覆封包送往 h1。h1 收到後，在其 ARP 表中記錄下 h2 的 IP 和 MAC 位址對應關係 ICMP Request
7. Data plane: h1 送 ICMP Echo Request，來源 MAC 是 h1，目標 MAC 是 h2
8. Data plane to Control plane: s1 收到這個 ICMP 封包。儘管它處理過 ARP 封包，但 flow table 中仍然沒有關於 h1 到 h2 的 IP 流量的 routing table rule，所以會再給 ONOS
9. Control plane: ReactiveForwarding 收到 ICMP。它檢查封包的來源和目的 MAC 位址。通過 hostService.getHost(id)，它能成功找到目標主機 h2 的位置，因為不是 ARP 所以 ONOS 會裝一個 flow table rule 以避免未來的封包都送到控制器。它會建立一個包含來源/目的 MAC 位址、port 等匹配欄位的 ForwardingObjective，ONOS 透過 flowObjectiveService.forward 將這個 flow rule 下發到交換器 s1。
10. Data plane: ONOS 會發送一個 Packet-Out 訊息給 s1，命令它將當前這個 ICMP 封包轉送到 h2 所在的 port
11. Data plane: s1 收到來自 ONOS 的兩個指令, Flow-Mod(用於安裝新的流表規則。交換器將此規則加入其流表中)、Packet-Out(用於處理當前的 ICMP 封包)
12. Data plane: h2 成功收到來自 h1 的 ICMP Echo Request 封包