NS Lab1 report

Part1:

1. 清空 switch s1, s2, s3 的 MAC Address table 並逐個顯示。

```
yucheng@ubuntu:~$ sudo ovs-appctl fdb/flush s1
table successfully flushed
yucheng@ubuntu:~$ sudo ovs-appctl fdb/show s1
 port  VLAN  MAC                Age
yucheng@ubuntu:~$ sudo ovs-appctl fdb/flush s2
table successfully flushed
yucheng@ubuntu:~$ sudo ovs-appctl fdb/show s2
 port  VLAN  MAC                Age
yucheng@ubuntu:~$ sudo ovs-appctl fdb/flush s3
table successfully flushed
yucheng@ubuntu:~$ sudo ovs-appctl fdb/show s3
 port  VLAN  MAC                Age
yucheng@ubuntu:~$
```

2. After h1 ping h4:

```
mininet> h1 wireshark &
mininet> h1 ping h4 -c 5
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=0.304 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.129 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.129 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.122 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=0.049 ms

--- 10.0.0.4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4083ms
rtt min/avg/max/mdev = 0.049/0.146/0.304/0.084 ms
mininet>
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::709a:71ff:fe6…	ff02::2	ICMPv6	70	Router Solicitation from 72:9a:71:60:0b:9b
2	0.000169627	fe80::acc2:7dff:fee…	ff02::2	ICMPv6	70	Router Solicitation from ae:c2:7d:ee:cf:a2
3	0.000209933	fe80::c4a5:5fff:fe0…	ff02::2	ICMPv6	70	Router Solicitation from c6:a5:5f:09:78:b7
4	15.717647894	fe80::acc2:7dff:fee…	ff02::fb	MDNS	203	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR …
5	16.041162591	fe80::e088:d2ff:fed…	ff02::fb	MDNS	203	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR …
6	16.105251663	fe80::c4a5:5fff:fe0…	ff02::fb	MDNS	203	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR …
7	16.384107781	fe80::74e2:10ff:fec…	ff02::2	ICMPv6	70	Router Solicitation from 76:e2:10:c8:7c:aa
8	43.149783057	72:9a:71:60:0b:9b	Broadcast	ARP	42	Who has 10.0.0.4? Tell 10.0.0.1
9	43.149974752	9a:dc:df:b9:e8:4b	72:9a:71:60:0b:9b	ARP	42	10.0.0.4 is at 9a:dc:df:b9:e8:4b
10	43.149978020	10.0.0.1	10.0.0.4	ICMP	98	Echo (ping) request id=0x1367, seq=1/256, ttl=64 (reply in 1…
11	43.150072267	10.0.0.4	10.0.0.1	ICMP	98	Echo (ping) reply id=0x1367, seq=1/256, ttl=64 (request in…
12	44.161075122	10.0.0.1	10.0.0.4	ICMP	98	Echo (ping) request id=0x1367, seq=2/512, ttl=64 (reply in 1…
13	44.161152339	10.0.0.4	10.0.0.1	ICMP	98	Echo (ping) reply id=0x1367, seq=2/512, ttl=64 (request in…
14	45.185057646	10.0.0.1	10.0.0.4	ICMP	98	Echo (ping) request id=0x1367, seq=3/768, ttl=64 (reply in 1…
15	45.185134360	10.0.0.4	10.0.0.1	ICMP	98	Echo (ping) reply id=0x1367, seq=3/768, ttl=64 (request in…
16	46.208133430	10.0.0.1	10.0.0.4	ICMP	98	Echo (ping) request id=0x1367, seq=4/1024, ttl=64 (reply in …
17	46.208204797	10.0.0.4	10.0.0.1	ICMP	98	Echo (ping) reply id=0x1367, seq=4/1024, ttl=64 (request i…
18	47.232481875	10.0.0.1	10.0.0.4	ICMP	98	Echo (ping) request id=0x1367, seq=5/1280, ttl=64 (reply in …
19	47.232511324	10.0.0.4	10.0.0.1	ICMP	98	Echo (ping) reply id=0x1367, seq=5/1280, ttl=64 (request i…
20	48.384075827	9a:dc:df:b9:e8:4b	72:9a:71:60:0b:9b	ARP	42	Who has 10.0.0.1? Tell 10.0.0.4
21	48.384083746	72:9a:71:60:0b:9b	9a:dc:df:b9:e8:4b	ARP	42	10.0.0.1 is at 72:9a:71:60:0b:9b
22	65.536939921	fe80::c840:f2ff:fe2…	ff02::2	ICMPv6	70	Router Solicitation from ca:40:f2:25:8a:7c
23	147.455907867	fe80::e088:d2ff:fed…	ff02::2	ICMPv6	70	Router Solicitation from e2:88:d2:dd:56:ec

從上圖中可以看到，10.0.0.1(h1)發送了一個 ARP 的廣播(No.8)，來詢問 10.0.0.4(h4)的 MAC Address。h2、h3 收到廣播發現不是自己，就會無視它；而 h4 看到廣播發現是自己，就會收下封包並得知 h1 的 MAC Address 就是 Source MAC Address。

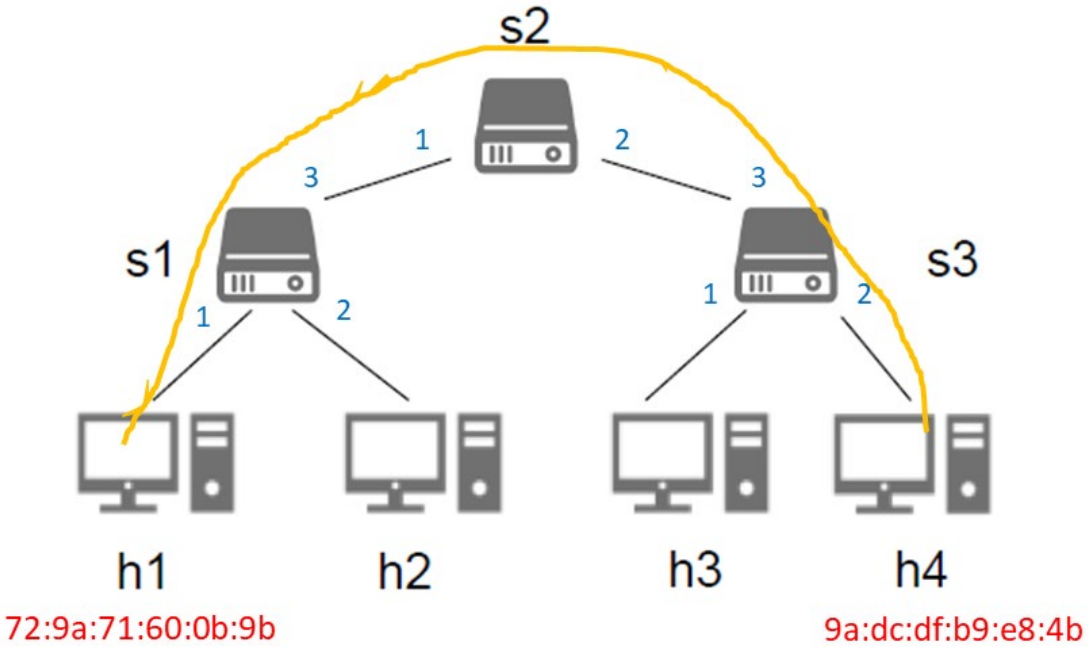
3. 承上題，收到廣播的 h4 會回覆了 h1 它的 MAC Address(No.9)。

4. 第一次 ping 會花比較長的時間是因為每個 switch 中的 MAC Address table 都沒有路徑資料。而 ping 成功一次之後，相對應的路徑資料就會被寫進 switch 的 MAC Address table，之後就只要依照 table 裡的資料進行封包轉送即可。因此第一次的 ping 和後面幾次相比，會花比較多時間。

5. 路徑建構:

```
yucheng@ubuntu:~$ sudo ovs-appctl fdb/show s1
 port  VLAN  MAC                Age
    1     0  72:9a:71:60:0b:9b    8
    3     0  9a:dc:df:b9:e8:4b    8
yucheng@ubuntu:~$ sudo ovs-appctl fdb/show s2
 port  VLAN  MAC                Age
    2     0  9a:dc:df:b9:e8:4b   12
    1     0  72:9a:71:60:0b:9b   12
yucheng@ubuntu:~$ sudo ovs-appctl fdb/show s3
 port  VLAN  MAC                Age
    2     0  9a:dc:df:b9:e8:4b   14
    3     0  72:9a:71:60:0b:9b   14
```

由上圖所示，我們可以得出路徑圖:

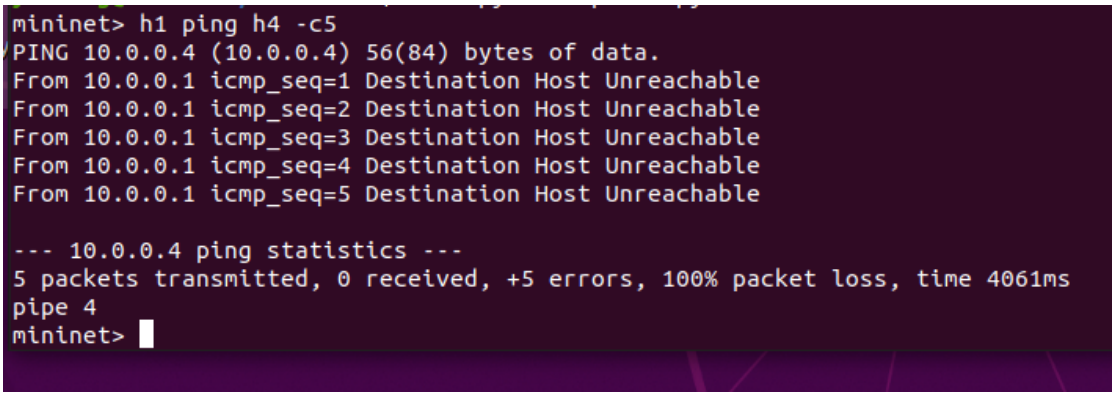


藍色為每個 switch 的 port number，紅色為 h1、h4 的 MAC Address，

黃色為 h1 ping h4 的所建構出的路徑。

Part2:

1. Before enable STP:



沒有 enable STP，無法連上。

2. After enable STP:

```
yucheng@ubuntu:~$ sudo ovs-vsctl set bridge s1 stp-enable=true
yucheng@ubuntu:~$ sudo ovs-vsctl set bridge s2 stp-enable=true
yucheng@ubuntu:~$ sudo ovs-vsctl set bridge s3 stp-enable=true
yucheng@ubuntu:~$ sudo ovs-vsctl set bridge s4 stp-enable=true
yucheng@ubuntu:~$
```

```
mininet> h1 ping h4 -c5
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=0.362 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.047 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.052 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.048 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=0.061 ms

--- 10.0.0.4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4084ms
rtt min/avg/max/mdev = 0.047/0.114/0.362/0.124 ms
mininet>
```

Enable 完之後，連得上。

3. s1 MAC Table:

Before enable STP:

```
yucheng@ubuntu:~$ sudo ovs-appctl fdb/show s1
 port  VLAN  MAC                Age
    4     0  9a:d6:76:4e:ee:72    0
    4     0  2e:79:9b:c8:bc:d3    0
    3     0  aa:48:fb:d8:25:9e    0
    4     0  5e:93:9a:fd:7a:8b    0
    4     0  22:52:d7:d0:b9:1b    0
    3     0  da:fb:f0:cf:0e:6a    0
    3     0  9a:4f:39:30:ce:cc    0
    3     0  3a:cb:57:82:1a:cc    0
    3     0  0e:c2:76:1d:e8:fe    0
    4     0  ae:35:53:b7:17:1d    0
    4     0  da:e4:35:a3:8d:6a    0
    3     0  a2:16:50:9a:fe:3c    0
yucheng@ubuntu:~$
```

After enable STP:

```
yucheng@ubuntu:~$ sudo ovs-vsctl set bridge s1 stp-enable=true
yucheng@ubuntu:~$ sudo ovs-vsctl set bridge s2 stp-enable=true
yucheng@ubuntu:~$ sudo ovs-vsctl set bridge s3 stp-enable=true
yucheng@ubuntu:~$ sudo ovs-vsctl set bridge s4 stp-enable=true
yucheng@ubuntu:~$ sudo ovs-appctl fdb/show s1
 port  VLAN  MAC                Age
    4     0  9a:d6:76:4e:ee:72   10
    4     0  2e:79:9b:c8:bc:d3   10
    3     0  aa:48:fb:d8:25:9e   10
    4     0  5e:93:9a:fd:7a:8b   10
    4     0  22:52:d7:d0:b9:1b   10
    3     0  3a:cb:57:82:1a:cc   10
    3     0  da:fb:f0:cf:0e:6a   10
    3     0  9a:4f:39:30:ce:cc   10
    3     0  0e:c2:76:1d:e8:fe   10
    4     0  ae:35:53:b7:17:1d   10
    4     0  da:e4:35:a3:8d:6a   10
    3     0  a2:16:50:9a:fe:3c   10
yucheng@ubuntu:~$
```

由上兩圖可發現，原本在沒有 enable STP 前，所有 table entries 的 Age 都是 0，代表相對應的 MAC Address 無法抵達；換句話說，那些 entries 都是無作用的，所以 h1 ping h4 才會 ping 不到。而在 enable STP 後，那些 entries 都有值出現，代表 entries 都有作用，MAC Address 也都連線的到。

4. What I observed and learned:

透過這次 lab 我觀察到，當網路拓樸結構中有迴圈時，在沒有 enable STP(Spanning Tree Protocol)前，封包是無法傳到特定位址的。在對每個受迴圈影響的 switch enable STP 後，網路能偵測迴圈並排除，讓封包不會卡在迴圈裡出不去。另外，我學會了如何使用 mininet 模擬網路環境，並使用 wireshark 偵測封包傳遞的詳細資料。我也學到了 switch 之間是如何透過網路環境學習、改變自己的 MAC Address table， 也

知道了 host 之間是如何透過 ARP 來得知對方的 MAC Address。