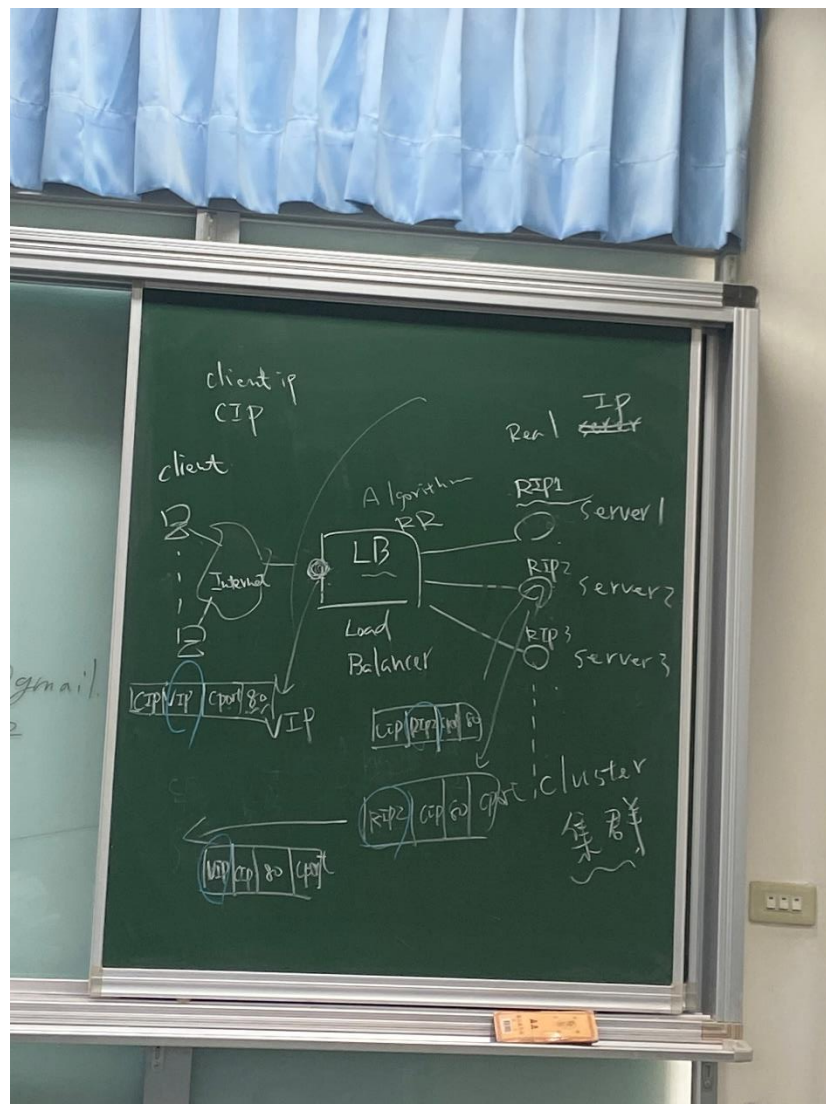


0625 Load Balancer 使用 p4 實作負載均衡器

我們現在所使用的網頁伺服器，一台機器如果性能比較好的，大概能服務一兩萬條連線，如果人數再更多，就沒有辦法負荷。所以，通常這種網頁伺服器都是用集群式的方式在服務，也就是說，它會開很多台，同時很多台在等待把使用者的請求服務分散掉，因為像一些購物節或者是搶票，使用者請求會非常多，幾台機器可能沒辦法應付，所以就有很多台機器。

Connection hash

當很多的客戶，如果集中在一台，容易爆掉，所以負載均衡就是把客戶的請求平均分散到不同伺服器上，這樣的話每一台伺服器的壓力就比較小，反應時間快，如果全都集中在單一台就會來不及服務，就會產生佇列，瀏覽器畫面就會在那邊等待，所以我們希望，當一鍵按下去可以快速得到我想要的東西。



因為這些伺服器每台都會有一個 ip，這些伺服器的 ip 叫做 real ip (rip)，這個負載均衡器會有一個對外 ip，叫做 virtual ip (vip)。

為什麼要有 vip？

因為當使用者要連線的時候，他們不會去記每一台的 ip 位址，所以就固定一個固定的 ip，他們只要連到這個 ip，就由它幫忙做分配。ex:選課系統客戶的 ip 叫做 client ip (cip)。

負載均衡器的概念就是，它會提供一個固定 ip，只要連線連進來這個固定 ip，它會自動幫你進行分發的動作。

只要來源相同，雜湊值就相同，如果來源不同，雜湊值相同的機率很小很小，如果真的相同，這種情況就稱為碰撞，那麼這個雜湊函數就不是很好。

p4-14 寫法轉換成 p4-16

執行步驟：

1. 打開終端機，切到 p4-test 資料夾
 2. 建立資料夾(mkdir hash-lb)，並切到 hash-lb 資料夾
 3. gedit hash-lb.p4 &，去 <http://csie.nqu.edu.tw/smallko/sdn/LBP4.htm> 把 load_balance.p4 複製並貼上
 4. 在終端機輸入指令 p4c-bm2-ss -p4v 14 -pp hash-lb16.p4 hash-lb.p4
 5. gedit hash-lb16.p4 & 就把 p4-14 寫法轉換成 p4-16 了
- p4-14 轉 p4-16 網站：<https://p4tw.org/%E5%B0%87-p4-14-%E5%BF%AB%E9%80%9F%E8%BD%89%E6%8F%9B%E8%87%B3-p4-16-%E6%96%B9%E6%B3%95/>

負載均衡器

[Topology]

H1 (10.0.1.1) -----(P4 switch: load balancer)----- H2 (Simple HTTP Server, 10.0.2.2).
----- H3 (Simple HTTP Server, 10.0.3.3).

Virtual IP: 10.0.0.1

第一個實驗：

1. 打開終端機，切到 `p4-test/hash-lb` 資料夾
2. `gedit commands.txt &` ，一樣去 LBP4.htm 網站把 `s1-command.txt` 複製並貼上
3. `gedit p4app.json &` ，把 `p4app.json` 貼上
4. 在終端機執行 `p4run`，開啟三個終端 `xterm h1 h2 h3`
5. 在 `h3` 執行 `python -m SimpleHTTPServer 80`
6. 在 `h2` 執行 `echo "hi" > hi.htm` 產生一個簡單的網頁，再執行 `python -m SimpleHTTPServer 80`
7. 在 `h1` 執行 `curl 10.0.0.1/hi.htm`

每次存取的時候，就會選擇不一樣伺服器

```
##### Welcome to the P4utils Mininet CLI #####
Your P4 program is installed into the BMV2 software,
and your initial configuration is loaded. You can interact
with the network using the mininet CLI below.

To inspect or change the switch configuration, connect to
its CLI from your host operating system using this command:
  simple_switch_CLI --thrift-port <switch thrift port>

To view a switch log, run this command from your host:
  tail -f /home/user/p4-test/hash-lb/log/<switchname>

To view the switch output pcap, check the pcap files
/home/user/p4-test/hash-lb/pcap:
for example run: sudo tcpdump -xxx -r s1-eth1.pcap

*** Starting CLI:
mininet> xterm h1 h2 h3
mininet>
```

量測效能指標的工具：apachebench

<https://blog.miniasp.com/post/2008/06/30/Using-ApacheBench-ab-to-to-Web-stress-test>

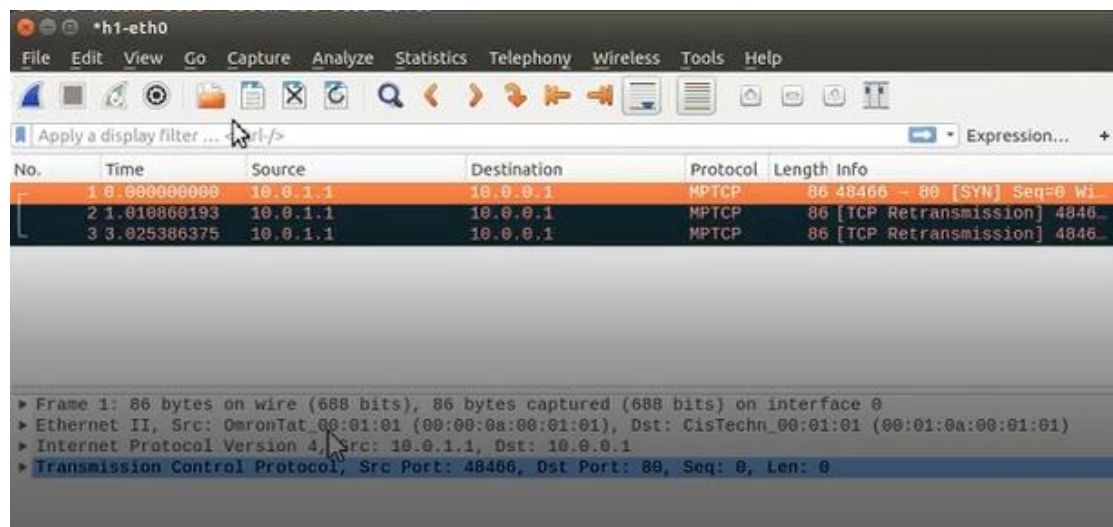
第二個實驗

先安裝 `ab`：`apt install apache2-utils`

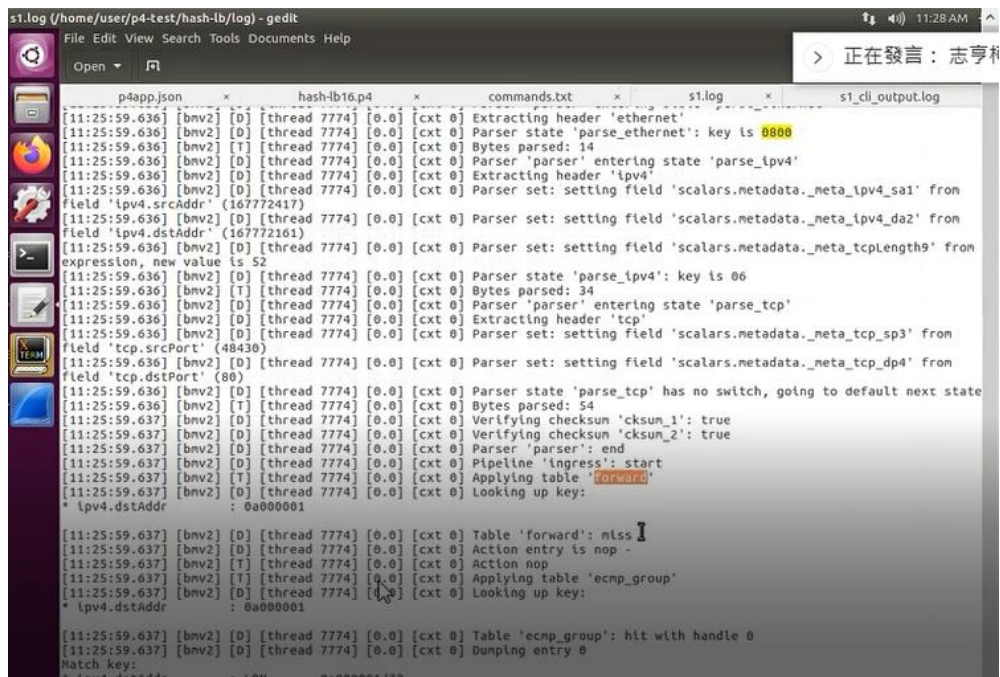
1. 在剛剛的 h1 輸入 `ab -n 1000 -c 10 http://10.0.0.1/hi.htm`
2. h2,h3 就會跑，h1 就會顯示 100,200 次的時間，最後做完會告訴你總共花了多久
3. 接下來測試 4 台機器，先去 `p4app.json,s1-commands.txt` 改規則
4. 再執行一次 `p4run`，`xterm h1 h2 h3 h4 h5`
5. h2,3,4,5 把伺服器都打開 `python -m SimpleHTTPServer 80`
6. 在 h1 執行 `curl 10.0.0.1/hi.htm`

模擬同時最多十條連線，總共要有一千次存取，看看需要花多少時間

若是失敗，可以再開一個 h1，打開 `wireshark`，選擇 `h1-eth0` 並點左上角鯊魚鰭啟動，然後在 h1 執行 `curl 10.0.0.1/hi.htm`，下面是問題畫面



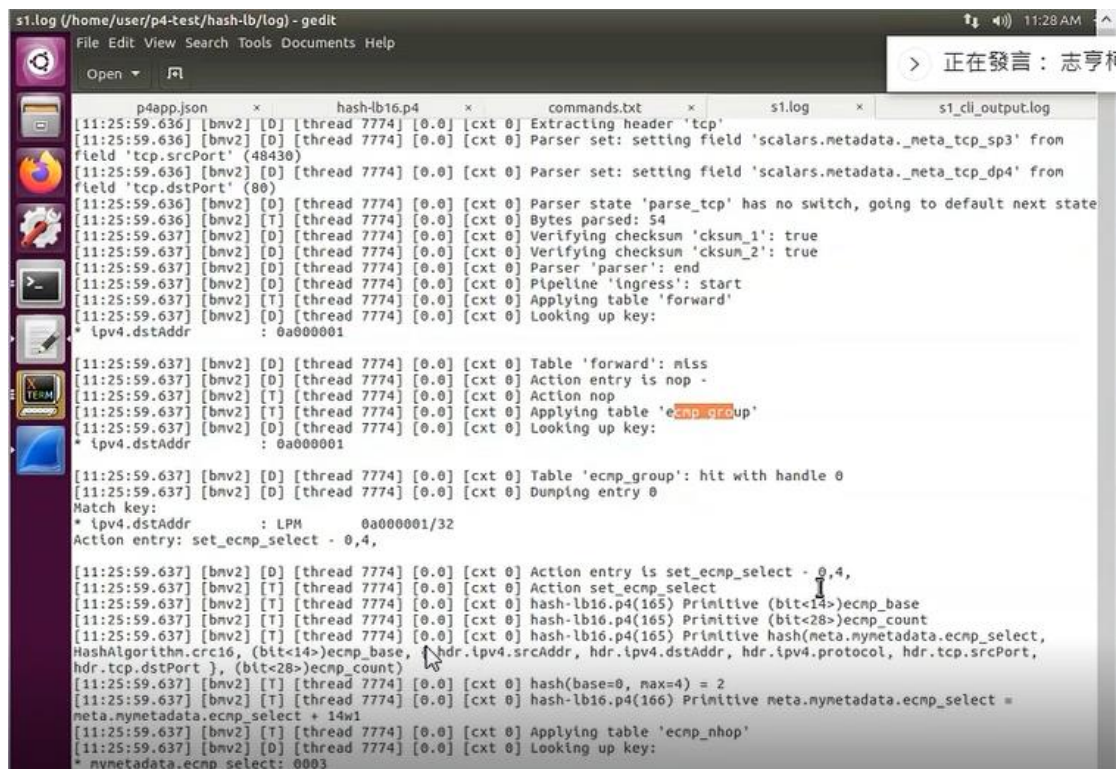
接著可以到log 資料夾(open/otherdocuments)把s1.log,s1_cli_output.log 打開在s1.log 用search/find 輸入0800，因為這是第一個ip 封包，所以從這邊開始查



```
s1.log (/home/user/p4-test/hash-lb/log) - gedit
File Edit View Search Tools Documents Help
Open 正在發言：志亨利

p4app.json hash-lb16.p4 commands.txt s1.log s1_cli_output.log
[11:25:59.636] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Extracting header 'ethernet'
[11:25:59.636] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Parser state 'parse_ethernet': key is 0800
[11:25:59.636] [bnv2] [T] [thread 7774] [0.0] [cxt 0] Bytes parsed: 14
[11:25:59.636] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Parser 'parser' entering state 'parse_ipv4'
[11:25:59.636] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Extracting header 'ipv4'
[11:25:59.636] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Parser set: setting field 'scalars.metadata._meta_ipv4_sa1' from
field 'ipv4.srcAddr' (167772417)
[11:25:59.636] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Parser set: setting field 'scalars.metadata._meta_ipv4_da2' from
field 'ipv4.dstAddr' (167772161)
[11:25:59.636] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Parser set: setting field 'scalars.metadata._meta_tcpLength9' from
expression, new value is 52
[11:25:59.636] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Parser state 'parse_ipv4': key is 06
[11:25:59.636] [bnv2] [T] [thread 7774] [0.0] [cxt 0] Bytes parsed: 34
[11:25:59.636] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Parser 'parser' entering state 'parse_tcp'
[11:25:59.636] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Extracting header 'tcp'
[11:25:59.636] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Parser set: setting field 'scalars.metadata._meta_tcp_sp3' from
field 'tcp.srcPort' (48430)
[11:25:59.636] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Parser set: setting field 'scalars.metadata._meta_tcp_dp4' from
field 'tcp.dstPort' (80)
[11:25:59.636] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Parser state 'parse_tcp' has no switch, going to default next state
[11:25:59.637] [bnv2] [T] [thread 7774] [0.0] [cxt 0] Bytes parsed: 54
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Verifying checksum 'cksum_1': true
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Verifying checksum 'cksum_2': true
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Parser 'parser': end
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Pipeline 'ingress': start
[11:25:59.637] [bnv2] [T] [thread 7774] [0.0] [cxt 0] Applying table 'forward'
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Looking up key:
* ipv4.dstAddr
: 0a000001
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Table 'forward': miss
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Action entry is nop -
[11:25:59.637] [bnv2] [T] [thread 7774] [0.0] [cxt 0] Action nop
[11:25:59.637] [bnv2] [T] [thread 7774] [0.0] [cxt 0] Applying table 'ecmp_group'
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Looking up key:
* ipv4.dstAddr
: 0a000001
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Table 'ecmp_group': hit with handle 0
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Dumping entry 0
Match key:
* ipv4.dstAddr
: LPM 0a000001/32
Action entry: set_ecmp_select - 0,4,
```

*Forward table 是反向的時候才會用到，所以第一個基本上不會用到就是miss

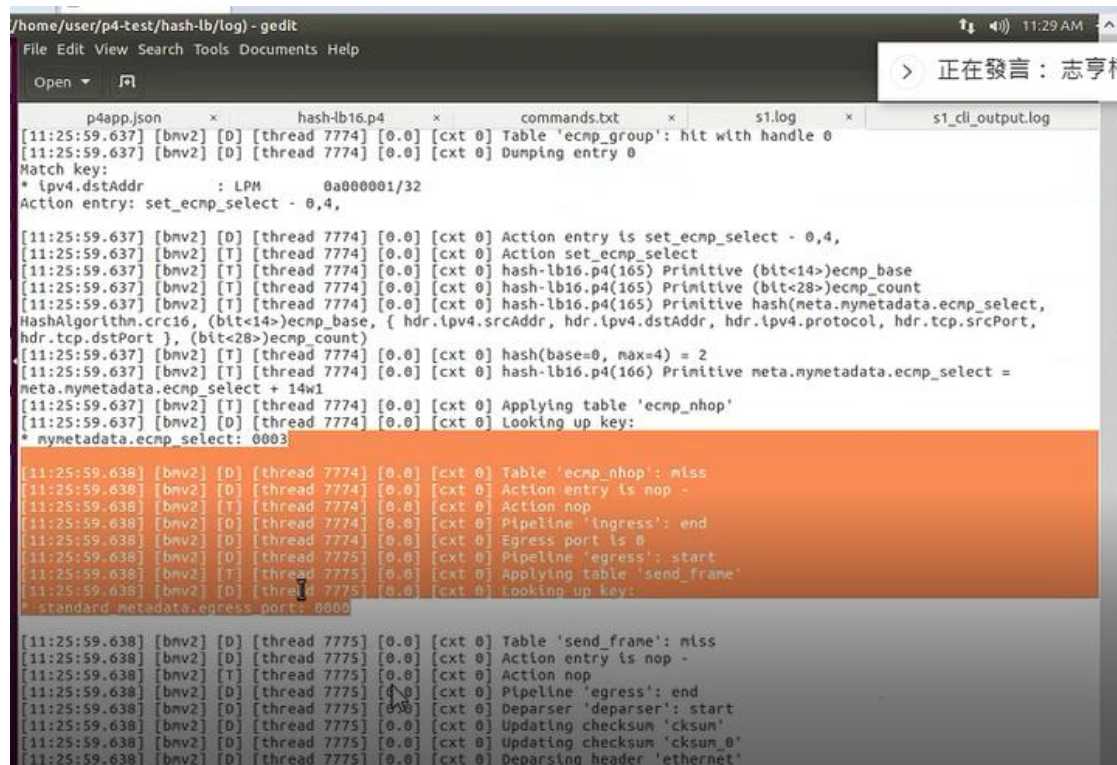


```
s1.log (/home/user/p4-test/hash-lb/log) - gedit
File Edit View Search Tools Documents Help
Open 正在發言：志亨利

p4app.json hash-lb16.p4 commands.txt s1.log s1_cli_output.log
[11:25:59.636] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Extracting header 'tcp'
[11:25:59.636] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Parser set: setting field 'scalars.metadata._meta_tcp_sp3' from
field 'tcp.srcPort' (48430)
[11:25:59.636] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Parser set: setting field 'scalars.metadata._meta_tcp_dp4' from
field 'tcp.dstPort' (80)
[11:25:59.636] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Parser state 'parse_tcp' has no switch, going to default next state
[11:25:59.637] [bnv2] [T] [thread 7774] [0.0] [cxt 0] Bytes parsed: 54
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Verifying checksum 'cksum_1': true
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Verifying checksum 'cksum_2': true
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Parser 'parser': end
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Pipeline 'ingress': start
[11:25:59.637] [bnv2] [T] [thread 7774] [0.0] [cxt 0] Applying table 'forward'
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Looking up key:
* ipv4.dstAddr
: 0a000001
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Table 'forward': miss
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Action entry is nop -
[11:25:59.637] [bnv2] [T] [thread 7774] [0.0] [cxt 0] Action nop
[11:25:59.637] [bnv2] [T] [thread 7774] [0.0] [cxt 0] Applying table 'ecmp_group'
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Looking up key:
* ipv4.dstAddr
: 0a000001
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Table 'ecmp_group': hit with handle 0
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Dumping entry 0
Match key:
* ipv4.dstAddr
: LPM 0a000001/32
Action entry: set_ecmp_select - 0,4,
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Action entry is set_ecmp_select - 0,4,
[11:25:59.637] [bnv2] [T] [thread 7774] [0.0] [cxt 0] Action set_ecmp_select
[11:25:59.637] [bnv2] [T] [thread 7774] [0.0] [cxt 0] hash-lb16.p4(165) Primitive (bit<14>)ecmp_base
[11:25:59.637] [bnv2] [T] [thread 7774] [0.0] [cxt 0] hash-lb16.p4(165) Primitive (bit<28>)ecmp_count
[11:25:59.637] [bnv2] [T] [thread 7774] [0.0] [cxt 0] hash-lb16.p4(165) Primitive hash(meta.nymetadata.ecmp_select,
HashAgorithm_crc16, (bit<14>)ecmp_base,
hdr.tcp.dstPort }, (bit<28>)ecmp_count)
[11:25:59.637] [bnv2] [T] [thread 7774] [0.0] [cxt 0] hash(base=0, max=4) = 2
[11:25:59.637] [bnv2] [T] [thread 7774] [0.0] [cxt 0] hash-lb16.p4(166) Primitive meta.nymetadata.ecmp_select =
meta.nymetadata.ecmp_select + 14w1
[11:25:59.637] [bnv2] [T] [thread 7774] [0.0] [cxt 0] Applying table 'ecmp_nhops'
[11:25:59.637] [bnv2] [D] [thread 7774] [0.0] [cxt 0] Looking up key:
* nymetadata.ecmp_select: 0003
```

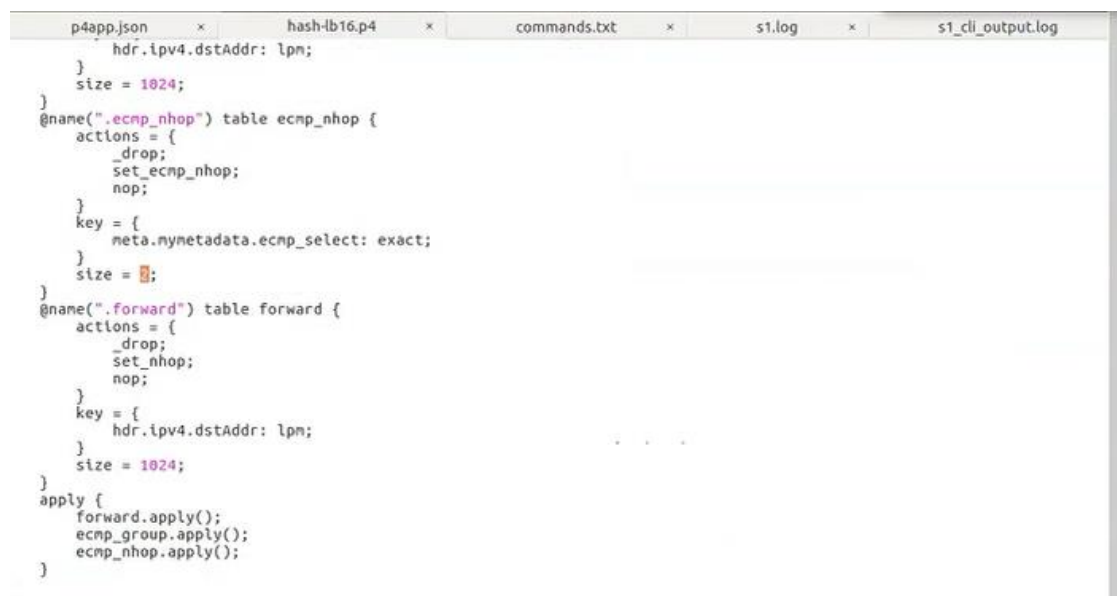
*第二個使用hash function 去挑1,2,3,4 裡面的值出來看看挑到什麼值，所以是hit，所以它就會去挑0~4 裡面的值，結果它挑2，規則上要+1 就變成3

加完以後，基本上 `select` 是 3，但下一個的 `egress_port` 是 0000，而且是 `miss`，這是問題所在！



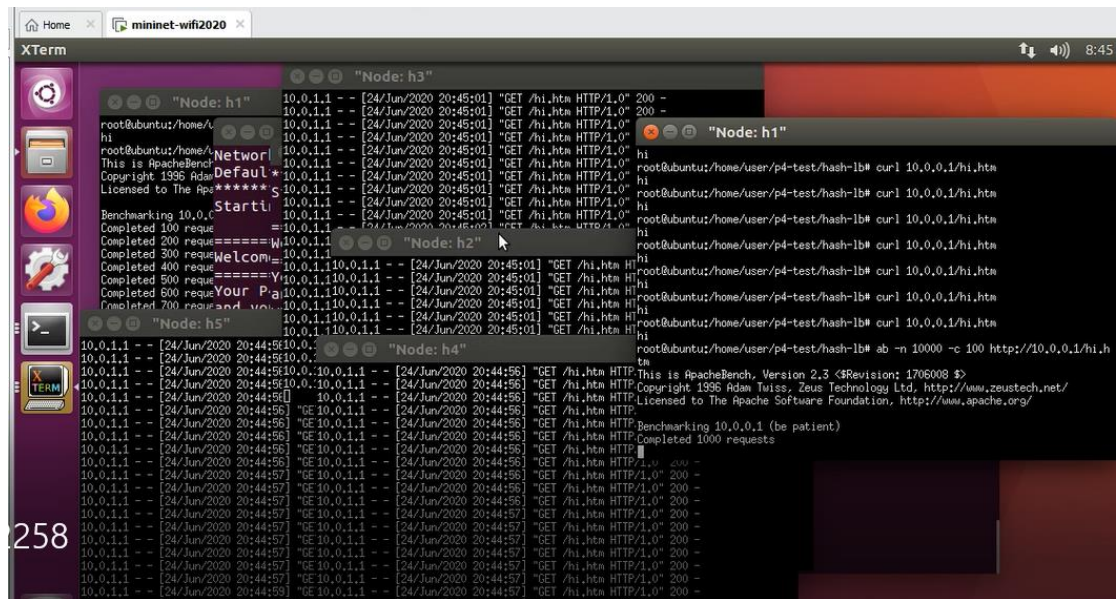
```
11:25:59.637 [bnv2] [D] [thread 7774] [0.0] [cxt 0] Table 'ecmp_group': hit with handle 0
11:25:59.637 [bnv2] [D] [thread 7774] [0.0] [cxt 0] Dumping entry 0
Match key:
* lpv4.dstAddr : LPM 0a000001/32
Action entry: set_ecmp_select - 0,4,
11:25:59.637 [bnv2] [D] [thread 7774] [0.0] [cxt 0] Action entry is set_ecmp_select - 0,4,
11:25:59.637 [bnv2] [T] [thread 7774] [0.0] [cxt 0] Action set_ecmp_select
11:25:59.637 [bnv2] [T] [thread 7774] [0.0] [cxt 0] hash-lb16.p4(165) Primitive (bit<14>)ecmp_base
11:25:59.637 [bnv2] [T] [thread 7774] [0.0] [cxt 0] hash-lb16.p4(165) Primitive (bit<28>)ecmp_count
11:25:59.637 [bnv2] [T] [thread 7774] [0.0] [cxt 0] hash-lb16.p4(165) Primitive hash(meta.mynetadata.ecmp_select,
HashAlgorithm.crc16, (bit<14>)ecmp_base, { hdr.lpv4.srcAddr, hdr.lpv4.dstAddr, hdr.lpv4.protocol, hdr.tcp.srcPort,
hdr.tcp.dstPort }, (bit<28>)ecmp_count)
11:25:59.637 [bnv2] [T] [thread 7774] [0.0] [cxt 0] hash(base=0, max=4) = 2
11:25:59.637 [bnv2] [T] [thread 7774] [0.0] [cxt 0] hash-lb16.p4(166) Primitive meta.mynetadata.ecmp_select =
meta.mynetadata.ecmp_select + 14w1
11:25:59.637 [bnv2] [T] [thread 7774] [0.0] [cxt 0] Applying table 'ecmp_nhop'
11:25:59.637 [bnv2] [D] [thread 7774] [0.0] [cxt 0] Looking up key:
* mynetadata.ecmp_select: 0003
11:25:59.638 [bnv2] [D] [thread 7774] [0.0] [cxt 0] Table 'ecmp_nhop': miss
11:25:59.638 [bnv2] [D] [thread 7774] [0.0] [cxt 0] Action entry is nop -
11:25:59.638 [bnv2] [T] [thread 7774] [0.0] [cxt 0] Action nop
11:25:59.638 [bnv2] [D] [thread 7774] [0.0] [cxt 0] Pipeline 'ingress': end
11:25:59.638 [bnv2] [D] [thread 7774] [0.0] [cxt 0] Egress port is 0
11:25:59.638 [bnv2] [D] [thread 7775] [0.0] [cxt 0] Pipeline 'egress': start
11:25:59.638 [bnv2] [T] [thread 7775] [0.0] [cxt 0] Applying table 'send_frame'
11:25:59.638 [bnv2] [D] [thread 7775] [0.0] [cxt 0] Looking up key:
* standard metadata.egress_port: 0000
11:25:59.638 [bnv2] [D] [thread 7775] [0.0] [cxt 0] Table 'send_frame': miss
11:25:59.638 [bnv2] [D] [thread 7775] [0.0] [cxt 0] Action entry is nop -
11:25:59.638 [bnv2] [T] [thread 7775] [0.0] [cxt 0] Action nop
11:25:59.638 [bnv2] [D] [thread 7775] [0.0] [cxt 0] Pipeline 'egress': end
11:25:59.638 [bnv2] [D] [thread 7775] [0.0] [cxt 0] Deparser 'deparser': start
11:25:59.638 [bnv2] [D] [thread 7775] [0.0] [cxt 0] Updating checksum 'cksum'
11:25:59.638 [bnv2] [D] [thread 7775] [0.0] [cxt 0] Updating checksum 'cksum_0'
11:25:59.638 [bnv2] [D] [thread 7775] [0.0] [cxt 0] Deparsing header 'ethernet'
```

原因是：剛開始設定的時候，這個表格只能記錄兩筆記錄，但我在 `commands` 裡面寫了四筆，也就是說，前面兩筆是有用的，後面兩筆是無用的，會導致後面這兩筆加不進去，所以只要把 2 改成 1024(調大一點)就行



```
hdr.lpv4.dstAddr: lpm;
size = 1024;
}
@name(".ecmp_nhop") table ecmp_nhop {
actions = {
_drop;
_set_ecmp_nhop;
_nop;
}
key = {
meta.mynetadata.ecmp_select: exact;
}
size = 8;
}
@name(".forward") table forward {
actions = {
_drop;
_set_nhop;
_nop;
}
key = {
hdr.lpv4.dstAddr: lpm;
}
size = 1024;
}
apply {
forward.apply();
ecmp_group.apply();
ecmp_nhop.apply();
}
```

7. 解決之後，重複上面 4~6 步驟，不斷在 h1 執行 `curl 10.0.0.1/hi.htm`
8. 發現幾乎都是在 h4 & h5 跑，所以改用 `ab -n 1000 -c 10 http://10.0.0.1/hi.htm` 跑看看，但還是一樣
9. 只要把 1000 改 10000，10 改成 100，就可以成功了



s1-commands.txt

```
table_set_default forward nop
table_set_default ecmp_group nop
table_set_default ecmp_nhop nop
table_set_default send_frame nop
table_add forward set_nhop 10.0.1.1/32 => 00:00:0a:00:01:01 1
table_add forward set_nhop 10.0.2.2/32 => 00:00:0a:00:02:02 2
table_add forward set_nhop 10.0.3.3/32 => 00:00:0a:00:03:03 3
table_add ecmp_group set_ecmp_select 10.0.0.1/32 => 0 2(2 改成 4)
table_add ecmp_nhop set_ecmp_nhop 1 => 00:00:0a:00:02:02 10.0.2.2 2
table_add ecmp_nhop set_ecmp_nhop 2 => 00:00:0a:00:03:03 10.0.3.3 3
table_add ecmp_nhop set_ecmp_nhop 3 => 00:00:0a:00:03:04 10.0.4.4 4
table_add ecmp_nhop set_ecmp_nhop 4 => 00:00:0a:00:03:05 10.0.5.5 5
table_add send_frame rewrite_sip 1 => 10.0.0.1
```

原來是 0 要改成 a

如果目的地是 10.0.0.1(vip 位址)，就用 hash 去挑，2 代表有兩個選擇，一個是 1 一個是 2。如果定義 hash 的結果是 1(bit)，代表要把請求丟給伺服器 2，如果 hash 值是 2 就丟到伺服器 3(第二台)
0 代表 base 值是 0，hash 從 0 開始(參照 load_balance.p4)

當封包回去的時候，要把原本的來源 ip 轉換成 vip(原本是 rip2，跑回來變 vip 如黑板圖)

p4app.json

```
{
  "program": "hash-lb16.p4",
  "switch": "simple_switch",
  "compiler": "p4c",
  "options": "--target bmv2 --arch v1model --std p4-16",
  "switch_cli": "simple_switch_CLI",
  "cli": true,
  "pcap_dump": true,
  "enable_log": true,
  "topo_module": {
    "file_path": "",
    "module_name": "p4utils.mininetlib.apptopo",
    "object_name": "AppTopo"
  },
  "controller_module": null,
  "topodb_module": {
    "file_path": "",
    "module_name": "p4utils.utils.topology",
    "object_name": "Topology"
  },
  "mininet_module": {
    "file_path": "",
    "module_name": "p4utils.mininetlib.p4net",
    "object_name": "P4Mininet"
  },
  "topology": {
    "assignment_strategy": "manual",
    "default_bw": 10,
    "default_delay": "1ms",
    "auto_gw_arp": true,
    "links": [{"h1", "s1"}, {"s1", "h2"}, {"s1", "h3"}, {"s1", "h4"}, {"s1", "h5"}],
    "hosts": {
      "h1": {
        "ip": "10.0.1.1",
        "gw": "10.0.1.254"
      },
    },
  },
}
```

```
"h2": {  
  "ip" : "10.0.2.2",  
  "gw": "10.0.2.254"  
},  
"h3": {  
  "ip": "10.0.3.3",  
  "gw": "10.0.3.254"  
}  
  "h4": {  
    "ip": "10.0.4.4",  
    "gw": "10.0.4.254"  
  }  
  "h5": {  
    "ip": "10.0.5.5",  
    "gw": "10.0.5.254"  
  }  
  
},  
"switches": {  
  "s1": {  
    "cli_input": "commands.txt",  
    "program": "hash-lb16.p4"  
  }  
}  
}  
}
```

Load_balance.p4

```
#include <core.p4>
#include <v1model.p4>
```

```
struct meta_t {
    bit<1>  do_forward;
    bit<32> ipv4_sa;
    bit<32> ipv4_da;
    bit<16> tcp_sp;
    bit<16> tcp_dp;
    bit<32> nhop_ipv4;
    bit<32> if_ipv4_addr;
    bit<48> if_mac_addr;
    bit<1>  is_ext_if;
    bit<16> tcpLength;
    bit<8>  if_index;
}
```

//在資料處理的過程當中，如果有一些比較重要的東西，可以放到 **metadata** 裡面

```
struct mymetadata_t {
    bit<14> ecmp_select;
}
```

```
header arp_t {
    bit<16> htype;
    bit<16> ptype;
    bit<8>  hlen;
    bit<8>  plen;
    bit<16> opcode;
    bit<48> hwSrcAddr;
    bit<32> protoSrcAddr;
    bit<48> hwDstAddr;
    bit<32> protoDstAddr;
}
```

```
header ethernet_t {
    bit<48> dstAddr;
```

```
    bit<48> srcAddr;  
    bit<16> etherType;  
}
```

```
header ipv4_t {  
    bit<4>  version;  
    bit<4>  ihl;  
    bit<8>  diffserv;  
    bit<16> totalLen;  
    bit<16> identification;  
    bit<3>  flags;  
    bit<13> fragOffset;  
    bit<8>  ttl;  
    bit<8>  protocol;  
    bit<16> hdrChecksum;  
    bit<32> srcAddr;  
    bit<32> dstAddr;  
}
```

```
header tcp_t {  
    bit<16> srcPort;  
    bit<16> dstPort;  
    bit<32> seqNo;  
    bit<32> ackNo;  
    bit<4>  dataOffset;  
    bit<4>  res;  
    bit<8>  flags;  
    bit<16> window;  
    bit<16> checksum;  
    bit<16> urgentPtr;  
}
```

```
header udp_t {  
    bit<16> srcPort;  
    bit<16> dstPort;  
    bit<16> length_;  
    bit<16> checksum;  
}
```



```

struct metadata {
    @name(".meta")
    meta_t      meta;
    @name(".mymetadata")
    mymetadata_t mymetadata;
}

```

```

struct headers {
    @name(".arp")
    arp_t      arp;
    @name(".ethernet")
    ethernet_t ethernet;
    @name(".ipv4")
    ipv4_t     ipv4;
    @name(".tcp")
    tcp_t      tcp;
    @name(".udp")
    udp_t      udp;
}

```

```

parser ParserImpl(packet_in packet, out headers hdr, inout metadata meta, inout
standard_metadata_t standard_metadata) {
    @name(".parse_arp") state parse_arp {
        packet.extract(hdr.arp);
        transition accept;
    }
    @name(".parse_ethernet") state parse_ethernet {
        packet.extract(hdr.ethernet);
        transition select(hdr.ethernet.etherType) {
            16w0x800: parse_ipv4;
            16w0x806: parse_arp;
            default: accept;
        }
    }
    @name(".parse_ipv4") state parse_ipv4 {
        packet.extract(hdr.ipv4);
        meta.meta.ipv4_sa = hdr.ipv4.srcAddr;
    }
}

```

```

    meta.meta.ipv4_da = hdr.ipv4.dstAddr;
    meta.meta.tcpLength = hdr.ipv4.totalLen - 16w20;
    transition select(hdr.ipv4.protocol) {
        8w6: parse_tcp;
        8w17: parse_udp;
        default: accept;
    }
}
@name(".parse_tcp") state parse_tcp {
    packet.extract(hdr.tcp);
    meta.meta.tcp_sp = hdr.tcp.srcPort;
    meta.meta.tcp_dp = hdr.tcp.dstPort;
    transition accept;
}
@name(".parse_udp") state parse_udp {
    packet.extract(hdr.udp);
    transition accept;
}
@name(".start") state start {
    meta.meta.if_index = (bit<8>)standard_metadata.ingress_port;
    transition parse_ethernet;
}
}

```

```

control egress(inout headers hdr, inout metadata meta, inout standard_metadata_t
standard_metadata) {

```

```

    @name("_drop") action _drop() {
        mark_to_drop(standard_metadata);
    }
    @name(".rewrite_sip") action rewrite_sip(bit<32> sip) {
        hdr.ipv4.srcAddr = sip; //如果 sip 是 1(参照 commands.txt)出口是 1
    }
    @name(".nop") action nop() {
    }
    @name(".send_frame") table send_frame {
        actions = {
            _drop;
            rewrite_sip;

```

```

        nop;
    }
    key = {
        standard_metadata.egress_port: exact;
//如果出口是 1，也就是要離開 load balancer，離開的方向是 1，就要把來源 ip
改成 10.0.0.1(vip)
    }
    size = 256;
}
apply {
    send_frame.apply();
}
}

```

```

control ingress(inout headers hdr, inout metadata meta, inout standard_metadata_t
standard_metadata) {
    @name(".drop") action _drop() {
        mark_to_drop(standard_metadata);
    }
    @name(".set_ecmp_select") action set_ecmp_select(bit<8> ecmp_base, bit<8>
ecmp_count) {
        //這個地方要來做 hash
        //count 值選完以後，hash 完的值放在這(meta.....select)
        hash(meta.mymetadata.ecmp_select, HashAlgorithm.crc16, //演算法
(bit<14>)ecmp_base, //hash 的值的基準點從哪裡開始(預設值 0,base=0，參照
commands.txt)
//hash 五個欄位：來源 ip，目的 ip，通訊協定，來源埠號，目的埠號
{ hdr.ipv4.srcAddr, hdr.ipv4.dstAddr, hdr.ipv4.protocol, hdr.tcp.srcPort,
hdr.tcp.dstPort }, (bit<28>)ecmp_count); //最多挑選挑的上限，從 0 開始挑到
count-1，出來的值最多是多少(2 就是選擇 0 & 1，3 就是 0,1,2)
meta.mymetadata.ecmp_select = meta.mymetadata.ecmp_select + 14w1;
//我們挑出來的值雖然是從 0 & 1，但我希望它的值可以再+1，變成 1 & 2
    }
    @name(".nop") action nop() {
    }
    @name(".set_ecmp_nhop") action set_ecmp_nhop(bit<48> nhop_mac, bit<32>
nhop_ipv4, bit<9> port) {
        standard_metadata.egress_spec = port;
    }
}

```

```

        hdr.ipv4.dstAddr = nhop_ipv4;
        hdr.ethernet.dstAddr = nhop_mac;
        hdr.ipv4.ttl = hdr.ipv4.ttl - 8w1;
    }
    @name(".set_nhop") action set_nhop(bit<48> dmac, bit<9> port) {
        standard_metadata.egress_spec = port;
        hdr.ethernet.dstAddr = dmac;
        hdr.ipv4.ttl = hdr.ipv4.ttl - 8w1;
    }
    @name(".ecmp_group") table ecmp_group {
        actions = {
            _drop;
            set_ecmp_select;
            nop;
        }
        key = {
            hdr.ipv4.dstAddr: lpm;
        }
        size = 1024;
    }
    @name(".ecmp_nhop") table ecmp_nhop {
        actions = {
            _drop;
            set_ecmp_nhop;
            nop;
        }
        key = {
            meta.mymetadata.ecmp_select: exact;
        }
        size = 1024;
    }
    @name(".forward") table forward {
        actions = {
            _drop;
            set_nhop;
            nop;
        }
        key = {

```



```

        hdr.ipv4.dstAddr: lpm;
    }
    size = 1024;
}
apply {
    forward.apply();
    ecmp_group.apply();
    ecmp_nhop.apply();
}
}

control
DeparserImpl(packet_out
packet, in headers hdr) {
    apply {
        packet.emit(hdr.ethernet);
        packet.emit(hdr.arp);
        packet.emit(hdr.ipv4);
        packet.emit(hdr.udp);
        packet.emit(hdr.tcp);
    }
}

```

Ingress 的部分有三個 table :

forward.apply(); : table_add forward set_nhop 10.0.1.1/32 => 00:00:0a:00:01:01 1

ecmp_group.apply(); :

table_add ecmp_group set_ecmp_select 10.0.0.1/32 => 0 2

如果他選擇 vip，就 set_ecmp_select;

ecmp_nhop.apply();

根據 select 值，去挑選伺服器

```

control verifyChecksum(inout headers hdr, inout metadata meta) {
    apply {
        verify_checksum(true, { hdr.ipv4.version, hdr.ipv4.ihl, hdr.ipv4.diffserv,
hdr.ipv4.totalLen, hdr.ipv4.identification, hdr.ipv4.flags, hdr.ipv4.fragOffset,
hdr.ipv4.ttl, hdr.ipv4.protocol, hdr.ipv4.srcAddr, hdr.ipv4.dstAddr },
hdr.ipv4.hdrChecksum, HashAlgorithm.csum16);
        verify_checksum_with_payload(true, { hdr.ipv4.srcAddr, hdr.ipv4.dstAddr,
8w0, hdr.ipv4.protocol, meta.meta.tcpLength, hdr.tcp.srcPort, hdr.tcp.dstPort,
hdr.tcp.seqNo, hdr.tcp.ackNo, hdr.tcp.dataOffset, hdr.tcp.res, hdr.tcp.flags,
hdr.tcp.window, hdr.tcp.urgentPtr }, hdr.tcp.checksum, HashAlgorithm.csum16);
    }
}

control computeChecksum(inout headers hdr, inout metadata meta) {
    apply {

```

```

        update_checksum(true, { hdr.ipv4.version, hdr.ipv4.ihl, hdr.ipv4.diffserv,
hdr.ipv4.totalLen, hdr.ipv4.identification, hdr.ipv4.flags, hdr.ipv4.fragOffset,
hdr.ipv4.ttl, hdr.ipv4.protocol, hdr.ipv4.srcAddr, hdr.ipv4.dstAddr },
hdr.ipv4.hdrChecksum, HashAlgorithm.csum16);
        update_checksum_with_payload(true, { hdr.ipv4.srcAddr, hdr.ipv4.dstAddr,
8w0, hdr.ipv4.protocol, meta.meta.tcpLength, hdr.tcp.srcPort, hdr.tcp.dstPort,
hdr.tcp.seqNo, hdr.tcp.ackNo, hdr.tcp.dataOffset, hdr.tcp.res, hdr.tcp.flags,
hdr.tcp.window, hdr.tcp.urgentPtr }, hdr.tcp.checksum, HashAlgorithm.csum16);
    }
}

```

```

V1Switch(ParserImpl(), verifyChecksum(), ingress(), egress(), computeChecksum(),
DeparserImpl()) main;

```