**動態加入規則**

1.開終端機切到p4-test複製檔案：cp 1 –r 1-2 拷貝到1-2資料夾，切到1-1(或1-2)資料夾

2.輸入gedit basic.json p4app.json & 編輯這兩個檔案 (p4app.json用來描述環境拓樸)並儲存，然後p4run

3.開兩個新的終端機，切到1-1(或1-2)資料夾，分別輸入：

simple\_switch\_CLI --thrift-port 9090 (connect to s1)

simple\_switch\_CLI --thrift-port 9091 (connect to s2)

4.在s1，可以打help查看支持的命令是什麼 ex:help table\_dump 檢視match-table的entries(每筆紀錄)

5.table\_dump phy\_forward(phy\_forward是表格名稱)：顯示表格資訊

6.一開始的預設是drop，因為還沒寫規則，h1 ping不到h2，所以要加入規則

7.使用table\_add，若不會用就使用help table\_add

指令：table\_add phy\_forward forward 1 => 2 (1:match field 2:action parameter)

table\_add phy\_forward forward 2 => 1 (是對稱的，1進2出，2進1出)

8.這樣就把兩筆紀錄加上去了，可以用table\_dump phy\_forward查看有沒有規則

9.在s2也執行

指令：table\_add phy\_forward forward 1 => 2

table\_add phy\_forward forward 2 => 1

10.在最開始的終端機(mininet那個)輸入：h1 ping –c 3 h2)就可以ping了

**Basic.json:**

/\* -\*- P4\_16 -\*- \*/

標頭檔(寫所有p4程式記得加

#include <core.p4>

#include <v1model.p4>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* H E A D E R S \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

做資料處理的時候，需要把一些資料先暫存到暫時的變數，就可以在這裡面設定暫時性的變數(這裡沒有用到但留著)

struct metadata {

/\* empty \*/

}

這個範例不需要分析任何封包表頭，因為1號進來就從2號出去，2號進來就從1號出去，封包裡有什麼東西不需要關心，完全沒有考慮到封包的結構，所以不需要parse(沒有用到但留著)

struct headers {

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* P A R S E R \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

parser MyParser(packet\_in packet,

out headers hdr,

inout metadata meta,

inout standard\_metadata\_t standard\_metadata) {

state start {

transition accept;

封包進來，parser不需要做任何動作，所以只需要做一件事：transition accept，代表分析已經結束

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\* C H E C K S U M V E R I F I C A T I O N \*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

如果有些情況之下，需要去檢查標頭裡面的checksum，封包在傳輸過程中有沒有對跟錯(這裡沒有用到但要留著)

control MyVerifyChecksum(inout headers hdr, inout metadata meta) {

apply { }

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\* I N G R E S S P R O C E S S I N G \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

control MyIngress(inout headers hdr,

去匹配規則，做對應的事，這塊要從apply開始看

inout metadata meta,

inout standard\_metadata\_t standard\_metadata) {

action drop() { //定義動作1:drop，把他丟棄

mark\_to\_drop(standard\_metadata);

}

action forward(bit<9> port) { //定義動作2:forward

standard\_metadata.egress\_spec = port; //把想要轉發的埠號寫到egress\_spec，就會從egress\_spec轉發出去

//當我把值(2)設定過去給它的時候，到時候那個封包就會從第幾號port(2)送進去

// metadata.egress是action parameters要把他丟到?(2)好埠

}

table phy\_forward { //phy\_forward：表格名稱(實體層\_轉發)

key = { //key：根據什麼去做匹配

standard\_metadata.ingress\_port: exact;

// (standard\_metadata.ingress\_port)是系統內建參數，這個參數是用來存放，記錄這個封包是從哪個port號進來

//exact:匹配方式有很多種，這種是完全匹配，要一模一樣，不能有不一樣的

// metadata.ingress是match fields，封包從哪裡進來，要符合這個規則

}

在p4裡，規則是有一種結構存放的，叫table，所以在設定規則的時候，要先準備一個table，然後把規則還有要做的事情都放在這裡面，所以它是一個資料結構

actions = { //匹配完要做的事

forward; //轉發

drop; //丟棄

}

size = 1024; //每條就是一個規則，數字就是這條最多可存放幾個(1024)規則

default\_action = drop(); //如果沒有指派，規則表裡沒寫的預設

}

apply { //真正的程式執行從這開始看

phy\_forward.apply(); //套用這個表格執行程式

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* E G R E S S P R O C E S S I N G \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

control MyEgress(inout headers hdr,

Egress剛開始沒有寫

把它留空

inout metadata meta,

inout standard\_metadata\_t standard\_metadata) {

apply { }

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\* C H E C K S U M C O M P U T A T I O N \*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

如果封包在處理過程當中，有可能需要重新計算checksum，錯誤檢查碼可以在這做(這個範例沒做，留空)

control MyComputeChecksum(inout headers hdr, inout metadata meta) {

apply {

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* D E P A R S E R \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

control MyDeparser(packet\_out packet, in headers hdr) {

apply {

重組回來，因為沒有做解析，所以必須要重組

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* S W I T C H \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

V1Switch(

MyParser(),

MyVerifyChecksum(),

MyIngress(),

MyEgress(),

MyComputeChecksum(),

MyDeparser()

) main;

**P4app.json**

{

"program": "basic.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": "AppTopoStrategies"

},

"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": "l2", //l2是layer2第二層，把h1跟h2放在同一個區域網路，分配ip的時候，他們是同個網域的ip位址

"links": [["h1", "s1"], ["h2", "s2"], ["s1", "s2"]],

"hosts": {

"h1": {

},

"h2": {

}

},

"switches": {

"s1": {

"program": "basic.p4"

},

"s2": {

"program": "basic.p4"

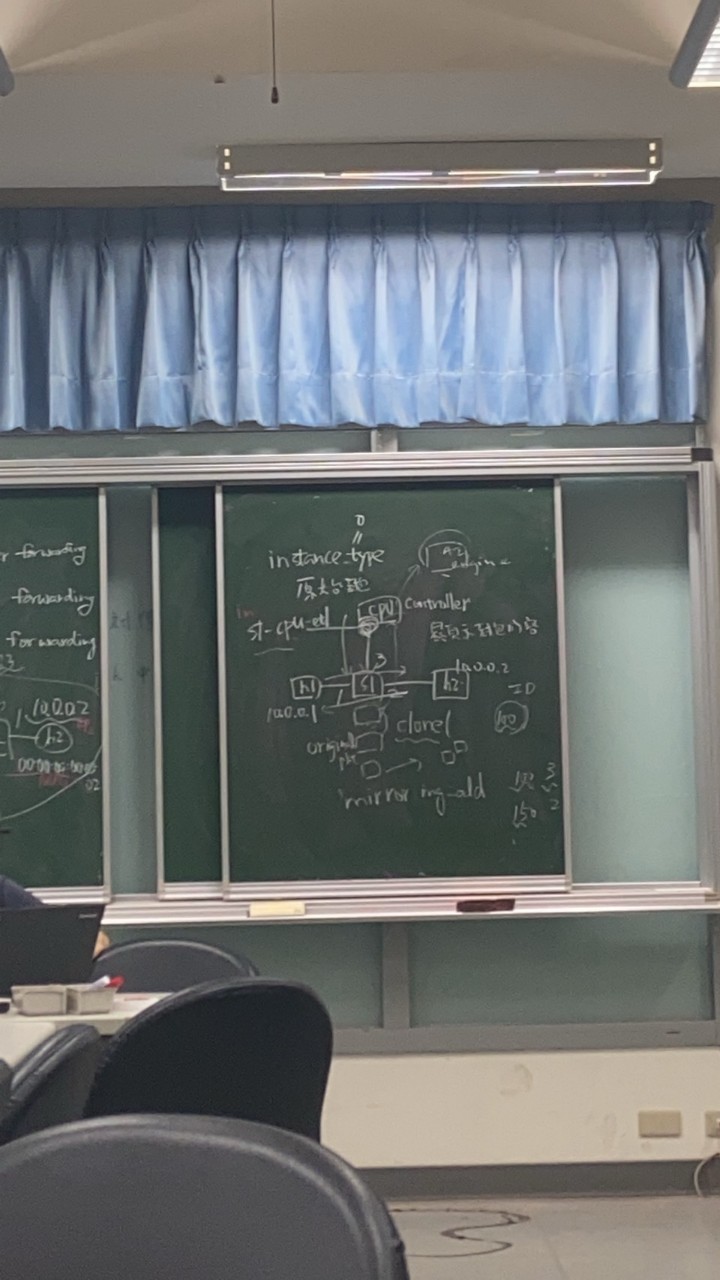
}

}

}

}

**Copy to cpu**



**P4app.json**

{

"program": "basic.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": "AppTopoStrategies"

},

"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": "l2",

"links": [["h1", "s1"], ["h2", "s1"]],

"hosts": {

"h1": {

},

"h2": {

}

},

"switches": {

"s1": {

"cli\_input": "cmd.txt",

"program": "basic.p4",

"cpu\_port": true //代表到時候會有多一個CPU的埠號，要送出去給CPU的地方

}

}

}

}

**Cmd.txt**

table\_add phy\_forward forward 1 => 2

table\_add phy\_forward forward 2 => 1

mirroring\_add 100 3 //拷貝完的送到3號埠 100是ID

**basic.p4**

/\* -\*- P4\_16 -\*- \*/

#include <core.p4>

#include <v1model.p4>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* H E A D E R S \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct metadata {

/\* empty \*/

}

struct headers {

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* P A R S E R \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

parser MyParser(packet\_in packet,

out headers hdr,

inout metadata meta,

inout standard\_metadata\_t standard\_metadata) {

state start {

transition accept;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\* C H E C K S U M V E R I F I C A T I O N \*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

control MyVerifyChecksum(inout headers hdr, inout metadata meta) {

apply { }

}

//拷貝的時候可以在兩個地方做，一個是ingress，或者是出去的時候做

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\* I N G R E S S P R O C E S S I N G \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

control MyIngress(inout headers hdr,

inout metadata meta,

inout standard\_metadata\_t standard\_metadata) {

action drop() {

mark\_to\_drop(standard\_metadata);

}

action forward(bit<9> port) {

standard\_metadata.egress\_spec = port;

}

table phy\_forward {

key = {

standard\_metadata.ingress\_port: exact;

}

actions = {

forward;

drop;

}

size = 1024;

default\_action = drop();

}

apply {

phy\_forward.apply();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* E G R E S S P R O C E S S I N G \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

control MyEgress(inout headers hdr, //在出口的地方做複製

inout metadata meta,

inout standard\_metadata\_t standard\_metadata) {

apply {

if (standard\_metadata.instance\_type == 0 ){

//.instance\_type == 0代表這是原始的封包

clone(CloneType.E2E,100); //原始的封包做複製，100就是給他一個ID，E2E:E就是在出口的部分，複製完他還是直接在出口的部分

}

clone(CloneType.E2E,100);拷貝的指令

if (standard\_metadata.instance\_type != 0 ){

truncate(34);

}

}

原本的封包要去做一次複製，對clone的來講，要去進行截斷(truncate)的動作，只截34byte，再傳出去，其他的都不要，也就是說，可以把部分資料丟上去就好

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\* C H E C K S U M C O M P U T A T I O N \*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

control MyComputeChecksum(inout headers hdr, inout metadata meta) {

apply {

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* D E P A R S E R \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

control MyDeparser(packet\_out packet, in headers hdr) {

apply {

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* S W I T C H \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

V1Switch(

MyParser(),

MyVerifyChecksum(),

MyIngress(),

MyEgress(),

MyComputeChecksum(),

MyDeparser()

) main;

**Receive.py**

#!/usr/bin/env python

import sys

import struct

import os

from scapy.all import sniff, sendp, hexdump, get\_if\_list, get\_if\_hwaddr, bind\_layers

from scapy.all import Packet, IPOption, Ether

from scapy.all import ShortField, IntField, LongField, BitField, FieldListField, FieldLenField

from scapy.all import IP, UDP, Raw, ls

from scapy.layers.inet import \_IPOption\_HDR

class CpuHeader(Packet):

name = 'CpuPacket'

fields\_desc = [BitField("device\_id",0,16), BitField('reason',0,16), BitField('counter', 0, 80)]

bind\_layers(CpuHeader, Ether)

print pkt.summary()

封包的一些部份資訊顯示出來，可以改成print pkt.show()會出現比較完整的資訊

def handle\_pkt(pkt):

print "Controller got a packet"

print pkt.summary()

def main():

if len(sys.argv) < 2:

iface = 's1-cpu-eth1'

else:

iface = sys.argv[1]

最重要的是這個：sniff(iface = iface,

prn = lambda x: handle\_pkt(x))

他要做一件事情，他要在監聽端口s1-cpu-eth1，做監聽的動作，聽到東西以後，就會把封包丟給handle\_pkt這個函式去處理

print "sniffing on %s" % iface

sys.stdout.flush()

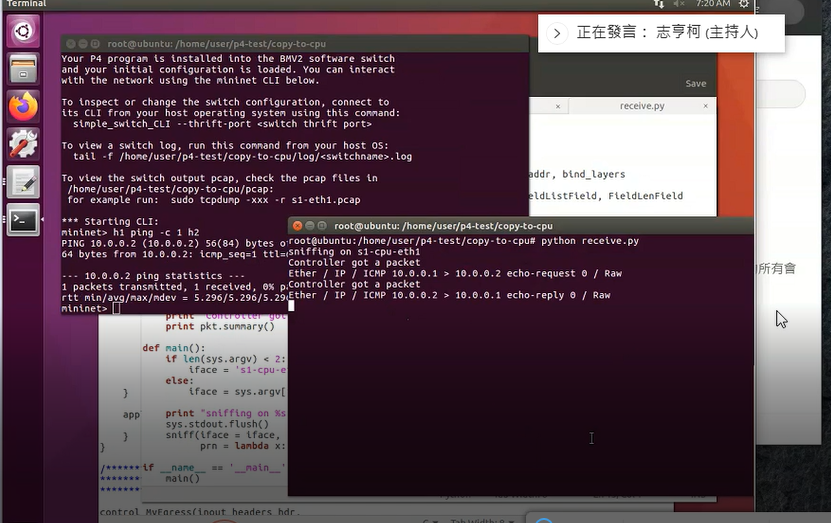
sniff(iface = iface,

prn = lambda x: handle\_pkt(x))

if \_\_name\_\_ == '\_\_main\_\_': //程式的入口點

main() //看到main，呼叫main函式

執行：

1. 在一個終端機(a左)輸入p4run
2. 另一個終端機(b右)輸入python reveive.py
3. 在a輸入h1 ping –c 1 h2
4. B就會出現兩個封包

**add\_cmd.sh** (在1-1(或1-2)資料夾)

規則很多的時候，這方法適合單一，例如只需要新增一筆規則

#!/bin/bash

//把前面結果輸出當作後面的輸入

echo ”table\_add phy\_forward forward 1 => 2” | simple\_switch\_CLI - -thrift-port 9090

echo ”table\_add phy\_forward forward 2 => 1” | simple\_switch\_CLI - -thrift-port 9090

simple\_switch\_CLI - -thrift-port 9091 < cmds2.txt

如果是比較大量，可以先把所有要做的規則寫到一個檔案(cmds2.txt)裡，再用導向(<)的方式丟到s2(9091 port)

用./add\_cmd.sh就把規則匯進去了