網路模擬器上之自動化程式開發

林孟璋
國家實驗研究院國家高速網路與計算中心
0303813@narlabs.org.tw

摘要

現今網路設備已經不是如從前功能單一的模式環境，使用模擬器平台日漸被網路工程師採用，透過 GNS3 模擬器平台架構，工程師可以載入設備的映像檔，供部屬前的規劃與測試使用，此外基於模擬器上的自動化開發工具模組 Netmiko，由於架構於 Python 程式語言，更是大受歡迎，透過這兩種開源軟體，以網路架構環境做自動化程式的開發，以利工程師做日常的維運處置與除錯，相信有兩者的加持，工程師更能得心應手於設備之間，減輕例行性的工作。

關鍵詞：Python、模擬器、Netmiko

一、前言

TWAREN 是高品質學術研究網路的簡稱，自建置100G 高頻寬骨幹網路以來，維運團隊除負責國內研究網路的建置規劃與維運外，更擴充國內及國際骨幹頻寬[1]，也陸續更換國際連網設備與執行前瞻計畫成立福爾摩沙開放網路交換中心 FOX(Formosa Open eXchange)，維運整個骨幹網路與交換中心，不僅僅是單一品牌路由器的環境，我們網路工程師也接觸到與之前不一樣品牌的設備，因此萌生了研究使用網路設備模擬器的念頭，希望藉此熟悉不同設備商的指令，以利後續維運所需。

因此，網路模擬器成了團隊的研究方向，在找尋目前市面所提供網路模擬器的同時，希望藉此在模擬器下達我們想要執行的指令，更希望以此環境，可以透過介面加以模擬我們欲規劃的網路架構，並自動化方式幫忙執行較例行公事，達到事半功倍的效果。

1.1、GNS3

GNS3是圖形化網路模擬器3(Graphic Network Simulator 3)的縮寫，是一個跨平台的模擬體平台，之所以稱呼它為平台，因為他結合了之前模擬器 Dynamips[2]的精髓，與現有虛擬機器(Virtual Machine)與容器(Container,Docker)的技術，創立了一個可以執行眾多網路設備廠商的模擬器。熟悉模擬器平台的網路工程師應該都知道，Dynamips 是以執行思科(Cisco)這家廠商的設備為主，但對於目前維運 TWAREN 骨幹網路是不夠的，起因在於其所提供的映像檔過於過時，不敷我們 TWAREN 骨幹網路設備模擬使用，因此選擇 GNS3可以透過載入 QEMU[3]虛擬機映像檔的方式，執行不同種廠商所提供的映像檔，以模擬這些設備。

除此之外，GNS3的最主要功能，它所設計的環境為主從(Client-Server)架構，當設計架構的工程師，在 GNS3所提供的圖形化介面規劃設計構圖時，各設備節點的環境是在伺服器上端運作，且這些伺服器可以是虛擬機器，並不需要真的一台實體機器，這樣對於建置更為省時省力。並且在部屬與規劃設備節點時，可以做到易於管理的好處。

圖1[4]就是以 GNS3所規劃出的一個複雜網路的拓樸架構，對於大型網路在做規劃時，不啻提供一個好的規劃環境，更明顯透過繪製圖形來標示個個區域網路所跑的路由通訊協定，以利辨別，並加以註解說明，對於日後工程師除錯是相當方便。

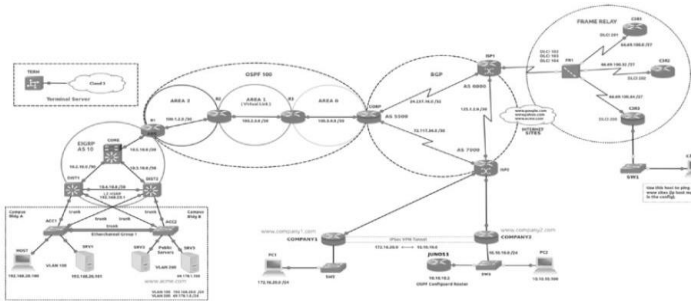

圖 1 .GNS3客端繪製拓樸圖

圖1所呈現的設備節點只少有25個，這對於管理這些設備是一大工程，GNS3也提供設備影像檔管理功能，可以輕鬆控管執行節點版本與映像檔。當我們規劃建置一個 專案時，有新的設備需加入時，GNS3介面提供模版的功能，針對設備做分類，並列出廠商型號與環境供規劃者選取，對於規劃者是一大利器，例如，我們欲模擬思科 IOS-XR 這類的模擬器，來規劃我們 TWAREN 骨幹路由器 ASR-9912，就可以選如下圖2所示之

模版功能管理其設備版本，對於日後升版之維護需求，可以說是非常實用。

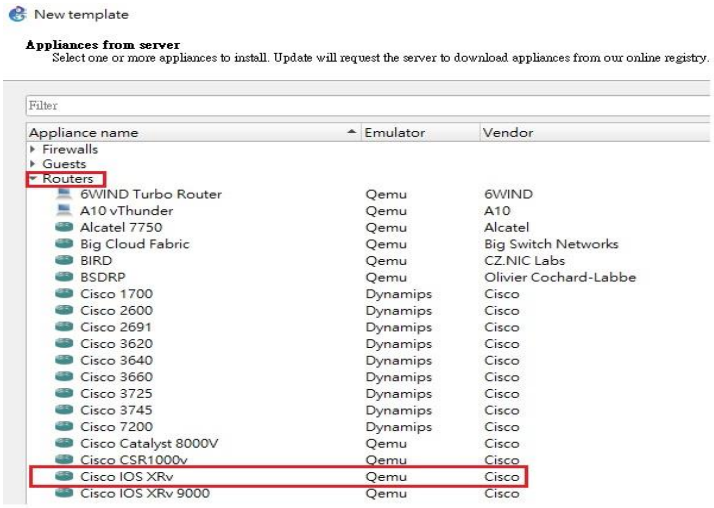

圖2. 新增模擬器之模板

雖然團隊研究不同的網路模擬器，例如 EVE-NG [5]、VIRL (Virtual Internet Routing Lab) 等，GNS3可以在眾多網路模擬器中脫穎而出可以說是自有其道理。

1.2 、Netmiko

Python 是當今流行的開源程式語言，不少開發者以自己的開發經驗，方便給其他人使用。模組(Module)就是一組特殊的函數或程序，藉以執行特殊的動作，以方便呼叫使用，但這些特殊模組需要另外安裝，以協順利完成主程式呼叫。此外，骨幹網路有不少設備，如果設備要接連登入執行指令，效率成為重要議題，如果要登入、執行、登出等動作，若每台路由器需花時10秒鐘的時間，若同時有20台路由器就需要200秒才可完成，進行更複雜的指令更是花更久時間，因此有平行運算與多執行緒的概念衍生而出，Python 提供多執行緒功能，以加快登自動化作業執行時間。

Netmiko 是一套以 Python 為基底的自動化開發模組，用來執行登入設備與執行指令之用，其前身為 Paramiko[6]，在開發者陸續開發開放源碼的自動化軟體趨勢下，Netmiko 模組化軟體孕育而生，Netmiko 更加相容多種設備的登入，此為選擇以此作為自動化模組最主要依據，以因應之後不同廠商的設備加入。GNS3是以主從式架構所設計，已經有伺服器端的環境可以作為開發平台，在虛擬機器環境下，可以登入 GNS3伺服器端，將伺服器端的 Python 版本升級到3.6版本以上即可使用此模組，其安裝過程可以說相當簡單。如下兩行之指令：

python3.8 -m pip install --upgrade pip

pip3 install netmiko(2)

雖網路上有其他自動化執行工具，例如 Ansible[7]，就是 RedHat 開發的自動化平台，但對於開發網路設備自動化而言，Netmiko 已足夠團隊使用，並不需要 Ansible 這樣大型的自動化軟體。

二、系統架構

因為 GNS3模擬器是跨平台的模擬軟體，可以在 Windows、Linux、MacOS 等作業系統下運作，客端軟體需求不需太高，且伺服器端不一定要與客端在同一台機器，大大減輕維護伺服器端的維運成本，且 GNS3的伺服器端已經虛擬化，可在遠端的一台裝有虛擬化平台的環境即可，系統開發的環境如下表1。

表 1: GNS3環境規格

硬體規格	CPU: Intel Core i5-2400 3.1GHz 記憶體:16G 網路卡:1Gb Ethernet
作業系統	Windows 10 64bits
虛擬化環境	VMware WorkStation Pro

由上表軟體需求中使用 VMware WorkStation Pro 這套軟體可看出，我們將使用主從式架構在同一台實體機器為目的，透過此虛擬化軟體產生一張僅有主機(Host-Only)的網路介面卡，並且藉由軟體，虛擬網路編輯器(Virtual Network Editor)達到從外面網路連線到內部虛擬機器的目的，架構如圖3。

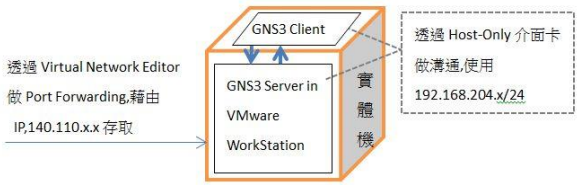

圖3. 架構圖

2.1、新增一台路由器

GNS3套件為可以主從端同時安裝的執行檔，執行該檔案同時可詢問用戶使用那個虛擬機器平台作為伺服器端環境，也順便將虛擬機映像檔一併下載，供之後安裝，操作相當親民，當我們要模擬 TWAREN 的路由器時，即可透過其客端 GUI 介面，設定虛擬路由器。在此同時，安裝好的軟體會在實體機器上產生 GNS3工作目錄，以 Microsoft Windows 10的平台下，內定安裝的工作目錄為 c:\Users\Users_name\GNS3。之後從網路下載的設備路由器映像檔就需放在 imags 下，images 下又可針對所下載的映像檔做分類，例如是透過 QEMU 虛擬環境執行的，即放在 images\QEMU 之下。之後新增模擬器樣版時，系統會透過 QEMU 目錄下的檔案自動去搜尋，以建立節點樣版。

如以建立 Cisco IOS XRv 9000 version 6.5.1這個節點來說明，可透過命名原則來規劃模擬器節點名稱，以利日後識別，例如依序是設備商名稱、作業系統、作業系統名稱[8] 、作業系統版本來規劃此模擬器命名。之後，透過 GUI 介面，如上頁圖2的新增模版介面做映像檔的匯入動作。當我們將網路下載的映像檔放入正確工作目錄後，亦或者，透過介面的匯入功能，將非工作目錄下的映像檔上傳到伺服器端，圖4即可看到匯入與安裝作業正在進行。

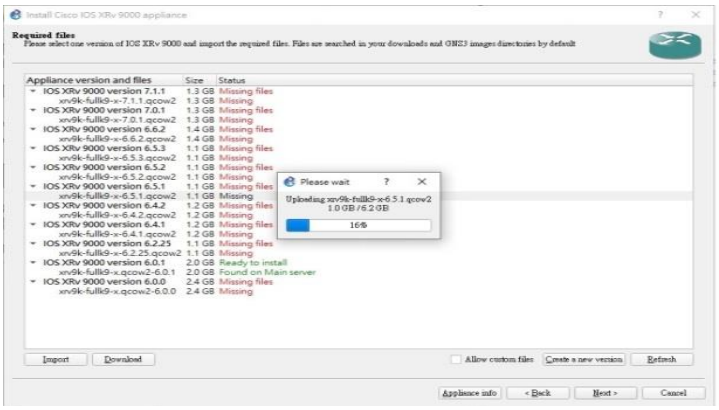

圖4. 匯入設備映像檔

2.2 操作模擬器介面

操作 GNS3的介面不外乎新增節點與設備之間的連接，以往設備工程師建置實體設備後，都需到實體機器做埠位(Port)拉線動作，指的是介面連接埠的連結，在 GNS3介面上即可將此功能予以實踐，首先我們從建立好設備模版上，做出一個節點(Node)，此節點代表一台模擬的設備，有兩台設備才可以拉線動作，介面呈現易於供設計者規劃，介面左邊呈現的即是設備選單，舉凡路由器、交換器、伺服器都、線路都在這邊表列，我們選取最上方的路由器，旁邊會顯示出我們剛剛建的路由器樣版清單，點選其中一台拉到左邊的工作建置區，就會產生一台路由器，以上述步驟產生兩台設備，並且點選設備按右鍵，按啟動(Start)，表示開啟設備電源的意思。之後點選最左邊選單的網路能圖示，表示拉線功能，之後點選設備，就可以透過拉線新增一條線路從路由器 CiscoIOSXRv6.1.2-5拉到 CiscoIOSXRv6.1.2-4，並選擇器所對應的埠位，如圖五。

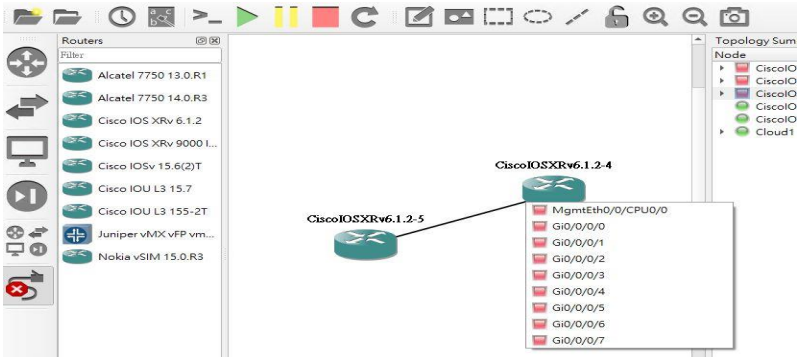

圖5. GNS3模擬拉線動作

操作完成後兩台路由器顯示綠色圓點於設備兩端，以表示設備間完成連線，並於右方顯示路由器 IP 與登入 Port 資訊，提供之後登入設備使用。如圖六。

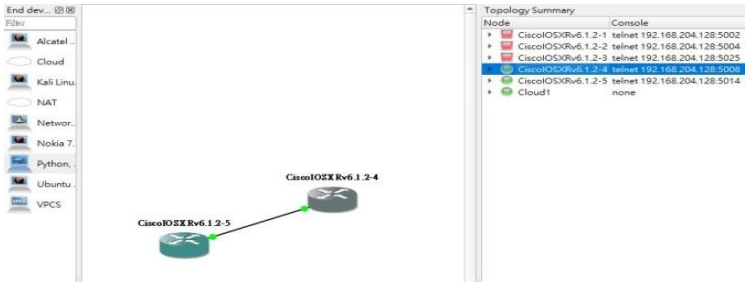

圖6. 完成路由器連接

2.3 登入路由器

如果要熟悉 Netmiko 這套自動化模組軟體，一定要參考作者 Kirk Byers[9]於 Github 的說明[10]，不啻是學習使用這個模組的好選擇，更提供了此模組可以連線哪些設備的清單，網站有明確的使用手冊，供開發者閱讀。使用此函示庫的步驟依序為登入設備、執行指令、切斷連線三步驟。我們舉這兩個函示說明如下，

```
from netmiko import ConnectHandler
device=ConnectHandler(device_type='cisco_ios',ip='192.168.248.249',user='user',passw
ord='test')
output= device.send_command("show interface gi0/0/0/0 ")
print(output)
device.disconnect()
```

第一行為透過 import 這個關鍵字載入模組。第二行為連接設備函數 ConnectHandler，參數依序為設備類型 cisco_ios、設備 IP、帳號、密碼，並將回傳值傳給變數 device 這個操作者(Handler)，第一個參數 cisco_ios 為思科 IOS 作業系統，在商業運轉環境下可透過 SSH 登入作業，在沒有資料安全疑慮的環境下可透過 telnet 模式登入，參數改成 cisco_ios_telnet，依不同廠商的設備在這個關鍵參數做修改。第三行為透過操作者做指令的下達，在此下達了列出有關介面 gi0/0/0/0的描述。第四行將下達指令之後並將結果予以印出。第五行結束設備連接。即可在程式輸出看到如圖7結果。

```
Tue Aug 16 05:57:10.318 UTC
GigabitEthernet0/0/0/0 is up, line protocol is up
  Interface state transitions: 1
  Hardware is GigabitEthernet, address is 0c43.c3c0.0001 (bia 0c43.c3c0.0001)
  Internet address is 10.1.1.2/24
  MTU 1514 bytes, BW 1000000 Kbit (Max: 1000000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation ARPA,
  Full-duplex, 1000Mb/s, unknown, link type is force-up
  output flow control is off, input flow control is off
  Carrier delay (up) is 10 msec
  loopback not set,
  Last link flapped 00:40:53
  ARP type ARPA, ARP timeout 04:00:00
  Last input never, output 00:40:50
  Last clearing of "show interface" counters never
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 0 multicast packets
              0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     1 packets output, 42 bytes, 0 total output drops
     Output 1 broadcast packets, 0 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
     1 carrier transitions
```

圖7. 執行指令後之輸出

工程師在下指令時不可能只下一條，有可能是連續指令，利用 Python 的串列(List)資料結構將要執行的指令儲存起來，透過 send_config_set 依序執行三行指令 conf t、logging buffered critical、logging buffered 20000，並透過 commit 函數寫進設定檔中。範例如下。

```
config_commands = ['conf t','logging buffered critical','logging buffered 20000']
output=device.send_config_set(config_commands)
device.commit()
```

以上是針對設定檔直接寫入動作，工程師在設定完成後，要將上述指令寫進設定檔。由於是寫的動作，比起讀取資料要求會比較高的權限，除需注意登入帳號是否有寫的權限外，建議使用交談(Interactive)方式確認是否做寫入動作。IOS-XR 系統可以透 end 指令做完成最後一個指令下達，並提出寫入要求，之後以交談方式詢問工程師是否將異動寫入設定檔。Netmiko 模組也針對對話方式予以設定。只要在 send_command 函數中，第二個參數加上 expect_string=r"" 即可針對回應字串做比對，如有出現雙引號之內容時，於下個指令做出回應指令，例如以下程式是針對送出 end 指令後，回覆應

該選擇 yes/no/cancel，如回答”y”，如下列兩行的陳述。

```
output=device.send_command(“end”,expect_string=r”Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]”)
output+=device.send_command(command_string=”y”,expect_string=r”#”)
```

2.4 輸出

在路由器上下達指令後，有較多且複雜的輸出資訊，但有些不是我們所需要的。之前程式中使用 send_command 送出的指令，可以輸出給 output 變數，可以使用字串處理的 find 功能，找出想要看的資訊，以下程式碼的範例，例如下達顯示作業系統版本的指令，但要找到開機至今的時間，即可使用 find(‘uptime is’)來做找尋，之後印出如圖8

```
output=net_connect.send_command("show version")
result = output.find('uptime is')
begin = int(result)
end = begin + 38
print(device['ip']+"=>"+output[int(begin):int(end)])
```

```
gns3@gns3vm:~/netmiko$ python3.8 show_uptimes.py
192.168.204.128:5002 => uptime is 18 hours, 23 minutes
192.168.204.128:5004 => uptime is 18 hours, 23 minutes
gns3@gns3vm:~/netmiko$
```

圖8. 找出執行時間

除了透過 Python 的處理字串函數找出想要的資訊，還透過名為 TextFSM 的模組將顯示內容有序的以 JSON 格式化輸出，方便我們過濾與取出想要的資料，因為 JSON 為熟悉的資料交換格式，更可透過 JSON 將資料拋給大數據資料庫的作法，來達到日後的資料分析。安裝模式，同如安裝 Netmiko 模組一樣，以 pip3 install textfsm 即可安裝，並以 import 載入模組。使用方式也是透過 send_command 之後加入參數 use_textfsm=True 來達成，於最後列印出來資訊時，使用 pprint(Pretty Print)模組最後看印出結果，如以下程式碼三行，執行結果如圖9。

```
from pprint import pprint
output = conn.send_command("show interfaces", use_textfsm=True)
pprint(output)
```

```
gns3@gns3vm:~/netmiko$ python3.8 show_interface_textFSM.py
[{'config_register': '',
  'hardware': [],
  'hostname': 'R1',
  'mac': [],
  'reload_reason': '',
  'restarted': '',
  'rommon': 'GRUB,',
  'running_image': 'disk0/xrvr-os-mbi-6.1.2/mbixrvr-rp.vm',
  'serial': [],
  'uptime': '21 hours, 39 minutes',
  'uptime_days': '',
  'uptime_hours': '21',
  'uptime_minutes': '39',
  'uptime_weeks': '',
  'uptime_years': '',
  'version': ''}]
```

圖9 .使用 textFSM 格式化輸出 JSON

三、案例研究

維運工程師在設定路由器時可先在模擬器上先行執行，測試完指令後確認是否正常

運作，在部屬到正式機器上，以確保指令正常運行，更可利用自動化程式幫忙處理日常事務，減輕每次登入設備辛勞。以下我們就針對 於 GNS3上架構一輕量級的 BGP 情境，BGP 是邊界閘道器協定（Border Gateway Protocol）的縮寫，是各大網路業者交換路由的通訊協定。在同一個 AS(Autonomous System)下，執行 IGP(Interior Gateway Protocol)通訊協定下的路由器，設定好彼此欲交換的路由，一般來說是以 OSPF、EIGRP 作為路由協定為主。以此例子做針對 BGP 的程式設計的架構。當然 Netmiko 程式也不僅於此，可以利用其提供針對不同設備的功能，處理各自因不同拓樸環境所產生的問題。

3.1 I-BGP 除錯情境

首先載入一個 BGP 範例[11]，此為 GNS3的可攜式專案檔，透過 GNS3匯入，並且安裝完成所需的映像檔，在此共有三台路由器，依序是最左邊的 R1(Router1)，依序至最右邊的 R3(Router3)。R1有一 Loopback IP 為1.1.1.1/32，且有一實體介面與 Router2連接，該介面配置 IP 為10.1.1.1/24。同理配置於 R2與 R3上也有相類似的 IP 設定。Loopback 為一虛擬的介面，可以使 BGP 會話(Session)一直保持存在，縱使實體的介面出問題也不會結束，這邊宣告相同 AS 號碼為65000，我們在 GNS3介面看到如圖10的拓樸圖，其展示 IP 與實體連接埠的配置狀況。

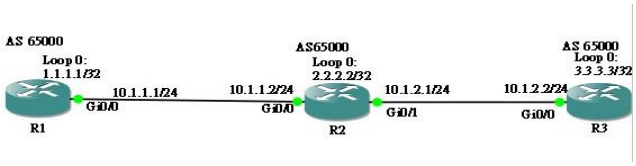

圖10 .BGP 情境架構圖

在除錯階段時，登入 R1與 R2的設備，下達檢查的指令，從回傳資訊看是哪裡有問題，提供給工程師參考，我們從 GNS3介面提供的登入主控台(Console Terminal)方式發現，R1沒辦法 Ping 到 R3的 Loopback IP，R3也沒法 Ping 到 R1的 Loopback IP 的狀況，這三台路由器之間的路由有問題。在這情況下，是透過 IGP 通訊協定作為媒介讓兩個或多個鄰居(Neighbor)連節，有可能其中 IGP 所跑的 OSPF 並沒有建立正確，導致 R1無法學到藉由 R2送過來的 R3所宣告的路由，反之亦然。在彼此無法正確學到彼此送出之路由，造成不通的情況，因此最後也以 ping 或 tracert 作為確認(Verify)程序來做為是否完成設定。透過指令查詢路由狀態與彼此鄰居的狀態，成為檢查的重點。於下個章節中，以 Netmiko 開發程式，自動登入路由器，完成檢查動做，並完成下達修正之正確指令，並做確認。以完成解決這個錯誤設定。

3.2 Netmiko 案例開發

在 TWAREN 維運開發中，當收到事件告警的同時，登入設備檢查哪裡出了問題，工程師透過指令查詢的方式，列出目前設備上的狀態資訊為何，以上章節 iBGP 拓樸架構來說，我們知道 R1與 R3之間有問題，除了路由器自己宣告的路由有異常之外，有可能各個路由器所的 OSPF 設定有問題，需要更詳細資訊以利除錯。進一步檢視相關資訊，可透過登入這三台路由器觀察其狀態，這些查詢的指令是結合維運工程師在維運上的經驗累積所常用的指令，涵蓋了大部分可找出原因的資訊，因此設計程式使之成為檢查模組(Check modules)，對於恢復 BGP 正常運作我們下的指令為(Set modules)，對於是否完成正常運作，設定為確認模組(Verify modules)，因此我們將這些模組依序

加入開發自動化程式之中。

依序針對 R1，我們依序下了 show ip protocol、show ip int brief、show ip bgp summary、show ip bgp neighbor、show run | sec bgp。這幾行的指令所查詢的相關資訊如表2。

表2:指令所代表意義

指令	指令所代表意義
show ip protocol	展現出所執行路由通訊協定資訊。
show ip int brief	展現介面狀態與資訊，查詢介面是否 Up 或 Down
show ip bgp summary	展現出 BGP 路由的狀態與資訊，提供路由筆數等相關資料
show ip bgp neighbor	展現出 BGP 鄰居的建立連結狀況
show run | sec bgp	展現出執行設定檔中有關 BGP 關鍵字的紀錄

在列印出這些指令之後，我們檢查是否為相同 AS、確認介面是 Up、BGP 狀態的摘要概述、BGP 相鄰的鄰居狀態概述、有關 BGP 的設定紀錄檔等資訊，此時查閱這些資訊後發現 R1宣告路由時有問題，其所宣告的路由不應該為1.0.0.0/8與10.0.0.0/8應該為1.1.1.1/32與10.1.1.0/24，此時進入設定 BGP 的程序，做錯誤的修正。邏輯架構如圖11

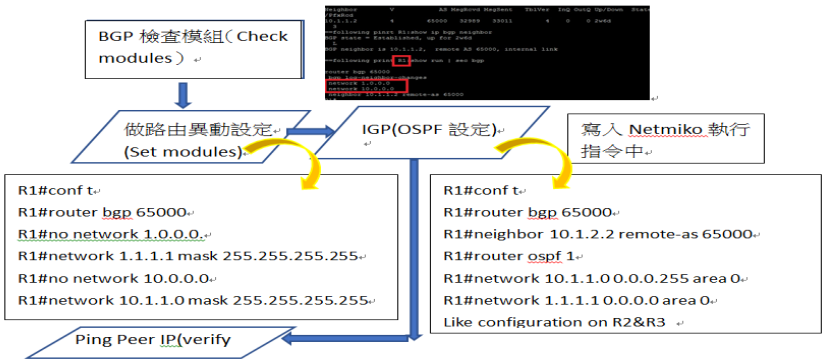

圖11.邏輯流程與指令

四、結論

GNS3是一套整合網路拓樸設計與映像檔維護的模擬器軟體，除了其免費的特性之外，更加持了介面可供網路架構工程師管理與模擬，對於日後除錯提供了好的環境。網路設備的支援也豐富有彈性，有別於之前的模擬器，其虛擬化伺服器功能讓我們可以於伺服器端開發 Netmiko 這套自動化軟體，Python 程式語言的盛行提升了這套軟體的可見度，工程師可以透過這套軟體規劃與處理例行性程序，而且 Netmiko 可以登入並執行指令的設備廠商也相當豐富多元，更可與 GNS3相同呼應。最重要的是其可程式化的步驟與邏輯技巧更易入手，對於跨平台的開發者也相當友善，不啻是一套可入手學習的套件。

此外對於網路設備除錯與設定上的應用，有鑑於網路架構與維運工程師除錯的技巧，僅以較常使用的指令加入命令列中，由 Netmiko 依序執行，是否透過更周詳或人工智慧的文字判斷來處理是未來一大課題，相信 Python 在機器學習上的優點，對於其

設備輸出資訊加以琢磨處理，已呈現出最好的除錯技術。

五、參考文獻

1. 張聖翊、 古立其、林書呈、謝欣叡, 2021 『TWAREN 國內及國際骨幹現況與發展趨勢』, TANet 2021 臺灣網際網路研討會。

2. Dynamips, (available online at https://github.com/GNS3/dynamips).

3. QEMU. (available online at https://www.qemu.org/).

4. Jason C.Neumann, “The Book of GNS3,” pp4,No Starch Press

5. EVE-NG, Emulated Virtual Environment-New Generation, (available online at https:// https://www.eve-ng.net/).

6. Paramiko, (available online at https://www.paramiko.org/).

7. Ansible, (available online at https://www.ansible.com/).

8. IOS-XRv,(available online at https://www.cisco.com/c/en/us/products/collateral/routers/ios-xrv-9000-router/datasheet-c78-734034.html).

9. PythonModuleMaintainer,KirkByers,(available online a https://pypi.org/project/netmiko/).

10. https://github.com/ktbyers/netmiko/blob/develop/EXAMPLES.md

11. David Bombal,https://www.youtube.com/watch?v=eu9iUE9gf2Y,G

NS3 Project download url, (available online at https://bit.ly/2wOGalc).