基於 P4之對稱性 Weighted-LACP 概念實作研討

胡乃元　周大源　曾惠敏　劉德隆
財團法人國家實驗研究院國家高速網路與計算中心
2103081, 1203053, 0303118, tlliu@narlabs.org.tw

摘要

因應軟體定義網路（SDN）[3]的技術演進，開發者除了針對控制層的 OpenFlow 也逐漸往針對 Data Plane 的 P4語言來進行研究，在本論文將會提出針對預計實現的基於 P4之對稱性 Weighted-LACP 進行概念上的講解以及相關判斷機制與運用，未來的目的將是可以透過 Control Plane 演算法以及 Data Plane 的配合來讓當 Load Balance 的群組成員失效時如何確保當成員狀態恢復時可以盡量降低連線的影響以及透過 P4程式進行權重的分配以及去回同路等等之功能性。

關鍵詞：SDN、P4、Tofino、TWAREN。

1. 前言

Programmable Protocol- Independent Packet Processors，簡稱為P4[1]，P4語言是一種與協定無關、可程式化的開放式交換器架構。網路開發者可以依照需求，針對封包Header之格式以及處理封包之程序等等部份，自行開發撰寫相對應之程式碼，並佈署至交換器上運行。相較於傳統網路，可程式化交換網路擁有更多的彈性。未來能夠發展的應用也會更多。

在2017年，Barefoot Networks公司開發出支援 P4 可程式化的交換器特殊應用晶片（Application Specific Integrated Circuit，ASIC） Tofino[2]，並於 2018 年改良為Tofino 2。目前已有多家網路硬體公司推出商用的可程式化交換器，並採用 Tofino/Tofino 2 ASIC，讓網路開發者可以直接撰寫 P4 可程式化網路程式，讓網路功能具有高度客製化的特性。在2019 年Barefoot Networks 加入 Intel，成為 Intel 的可程式化網路技術團隊。

在本次基於P4之對稱性Weighted-LACP報告中，將講解整體架構以及製作理念，將會簡單講解P4原先Action Selector的功能，因Action Selector是類似於Load Balance的機制，雖然只要會撰寫該函式以及下發Flow即可輕易的實現該功能，但是可能因為斷線等等原因而更改Port，待線路修復後，該函式將會更動回原本之Port，此種情況將會有可能導致TCP連線中斷或是Time out等等情形發生，所以我們會講解如何利用類似機制去在利用P4的情況下實作LACP的功能，利用P4程式靈活性的操作來克服問題，未來將會搭配演算法來進行Weighted-LACP的動態調整以及去回同路的功能。

本次論文將在章節2講解了P4相關的背景知識，章節3講解了基於P4之對稱性Weighted-LACP相關之架構，章節4講解了本次論文所提出之改進架構以及相關演算法機制，章節5則是結論。

2. 背景知識

本章節將會講解論文所用到之背景知識。

2.1 P4(Programming Protocol-independent Packet Processors)

P4[5]是一種用於可程式化資料層的高階語言，提供比傳統 SDN 的 OpenFlow 更為彈性的功能，透過 P4，開發者可以精確定位任意封包裡面的 Header 欄位，P4主要宣傳是可支援任何通訊協議，可支援任何平台，可隨時更改交換規則，支援任何通訊協議代表的是我們可以非常有彈性的去處理所有封包，可支援任何平台則代表可以在 FPGA、DPDK、Tofino、Smart NIC 等等上支援，不過目前仍以 Tofino 晶片為主，但是不同晶片上之Pipeline[4]略有差異，在可隨時更改交換規則這點上是允許我們對各交換器設備去重新定義對封包處理的方式，這也代表我們可以指定網路設備如何去處理網路封包，由於以前網路晶片的限制，推出一種新功能都需要數年的時間來讓晶片支援，透過 P4語言以及強大的 Tofino 晶片，達到為了實現特定的封包行為，開發者可以在幾分鐘之內完成一種網路協議的更動而並非耗費幾年的時間。

2.2 P4 WorkFlow

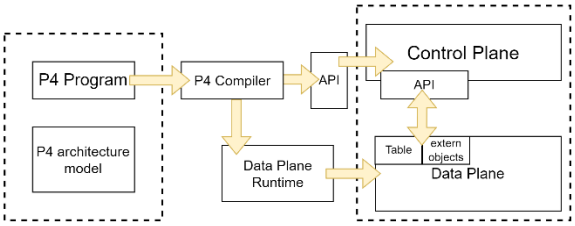

圖1. P4 WorkFlow[5]

P4的運行流程如圖1，將 P4程式透過 P4c 編譯，接下來將 P4讀取到 Data plane 設備，例如各種支援 P4的軟硬體交換機：BMv2、Tofino Model、實體 Tofino 交換器等等，然後透過控制層利用 P4 runtime 等 API 向 Data plane 下發相關的 Entry。

2.3 Tofino

Tofino 是世界上第一款用戶可程式化的乙太網路交換器 ASIC，專門設計給資料中心作應用，能夠即時監視及控制軟體內的封包，並使用協議獨立交換器架構(PISA)建構，這代表如需更新協議時能夠直接像是軟體升級一般的佈署，在軟體內調整網路協議並直接編譯到交換機中。

Tofino 晶片可以同時處理四條流水線並展現優異的6.5Tbps 的吞吐量並可以提供最高100GbE 的頻寬，二代與三代則分別可以處理12.8Tbps 以及25.6Tbps 並提供到400GbE 的頻寬，各國家的網路

實驗單位仍然利用 P4設備來達成各個網路架構的任務。

2.4 LACP（鏈路聚合控制協議）

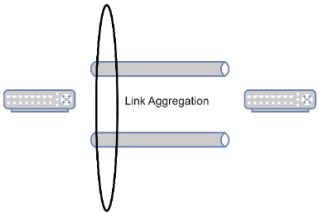

圖2. LACP 概念圖

如圖2. 所示，LACP（Link Aggregation Control Protocol 鏈路聚合控制協議）[6] 是基於IEEE802.3ad標準的一種實現鏈路動態匯聚的協議，LACP負責增加線路的可靠性以及負載平衡，並且可在斷線時動態調整群組聚合成員的狀態。如果該功能要實現的話，我們將以此協議所提供功能之概念運用在P4程式之中，在P4中有提供LAG的群組設置，並在成員的狀態設置為False時可以根據程式撰寫之P4語言中提供的動態調整演算法，使斷線之Port分配到其餘可用之Port，可以透過類似的方式來實作LACP之功能。

2.5 Action Selector

Action selector 是 P4語言所提供的一個函式，通常用於 ECMP 選擇 Next Hop 或是從 LAG 群組中多個實體 Port 進行選擇，通常是用於 Load Balance 的用途。

3. 基於 P4之對稱性 Weighted-LACP

本章節將會講解基於 P4之對稱性 Weighted-LACP 的概念實作。

3.1 概念拓樸圖

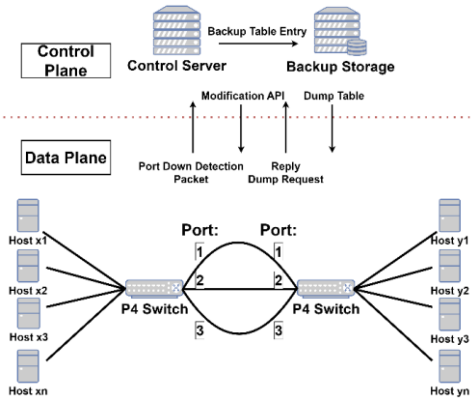

圖3. Weighted-LACP 拓樸圖

如圖3.所示，在此拓樸中將分為 Control Plane 以及 Data Plane，總共兩層，在 Data Plane 分別有 P4 Switch 兩台、Host 若干台，線路連接方式為 Host x1~xn 接上 P4 Switch 上以及將 Host y1~yn 連接上另一台 P4 Switch ，以對稱方式進行線路的連接，在 P4 Switch 之間的連接方式為將其中一台 P4 Switch 之 Port 1連接到另一台 P4 Switch 之 Port 1，其餘 Port 也是照著對稱方式進行連接，則其餘功能是由 P4 Switch 之斷線偵測功能，在當 Port 斷線時會發送警告封包至 Control Plane，這時由 Control Plane 透過 API 修改 LAG 群組之成員狀態，讓負載平衡不會因為斷線而造成封包傳送不出，另外的功能是提供定期使用 Dump 呼叫 P4 Switch 的狀態，用意是將 Table 資訊回傳至 Control Plane，再透過 Control Plane 的程式備份 Table 為災難還原後 P4 Switch 可以讀取的指令格式，將透過 Control Plane 所撰寫之應用程式搭配演算法去做控制，則該網路拓樸所使用之對稱式架構將會利用在封包之去回同路，藉由 Hash 值做 xor 計算，可將封包資訊條件相同但是 Src IP 以及 Dst IP 為互相交換之封包進行導往相同條件的分配 Port，例如 Src IP 為10.0.0.1，Dst IP 為10.0.0.2，如果藉由5-tuple 的條件進行 Hash 運算，當 Src IP 以及 Dst IP 欄位互換時，Hash 值也會有相當大之變動，這種變動將有可能將去回之封包導向不同之 Port，所以藉由 Src IP 以及 Dst IP 做 xor，就算欄位交換也將會獲得完全一樣的值，並透過5-tuple 之其他欄位也足以獲得足夠大的 Hash 變量，關於權重的部分將會採取利用 Hash 使用 mod 取餘數的方式，利用 Table 對應於餘數的方式來決定流量要往哪一個 Port 進行傳輸，本次實驗環境將採用 mod 8，餘數的數量將會大於 Port 的數目，利用該種方式可以利用本特性，讓開發者能夠自定義某些 Port 之權重，或是搭配 Control Plane 之應用可以利用演算法來控制權重的分配。

3.2 Action Selector 函數使用

Action Selector 函數所提供的功能以及使用方式通常會使用在 Load Balance 上，在使用 Action selector 的功能前，我們會先使用 Parser 解析封包的指定欄位，並將封包的欄位進行 Hash 的計算，例如常使用的5-tuple：Source IP、Source port、Destination IP、Destination port、Layer 4 protocol，透過這些值經過 Hash 之計算後將此 Hash 值使用 Selector 的方式進行 Match，要使用這個方式需要先設立一個 Action Profile，在這裡面需要先設置 Member Id 以及 Port 的對照 Entry，再針對 Action selector 下發群組成員總共有誰的 Entry，方可建立 LAG 群組。

如果需要使用非平衡輸出，利用不同 ID 去設置同一個 Port 是可以執行的，可以利用該機制去運作權重分配。

```
----- port_groups Dump Start -----
Entry 0:
Entry key:
    $ACTION_MEMBER_ID               : 0x00004E20
Entry data (action : Ingress.ctl_forward_packet.act_send):
    egress_port                     : 0x00

Entry 1:
Entry key:
    $ACTION_MEMBER_ID               : 0x00004E21
Entry data (action : Ingress.ctl_forward_packet.act_send):
    egress_port                     : 0x00

----- port_groups Dump End -----
```

圖4. Action Selector 非平衡輸出範例

3.3 測試

如圖5.，我們利用 Python Scapy 對 P4 負載平衡程式進行隨機 IP 封包之測試。

```
src_ip = ".".join(str(random.randint(0, 255)) for _ in range
dst_ip = ".".join(str(random.randint(0, 255)) for _ in range

p = (Ether(dst="00:11:22:33:44:55", src="00:aa:bb:cc:dd:ee")
     IP(src=src_ip, dst=dst_ip)/
     UDP(sport=7, dport=7)/
     "This is a test")

sendp(p, iface=iface)
```

圖5. Scapy 程式測試範例

如圖6.、圖7.、圖8.，為封包負載平衡測試，Veth1對應於 Tofino model 之 Port0，Veth3對應於 Tofino model Port1，Veth5對應於 Tofino model Port2，Veth7對應於 Tofino model Port3，測試結果為負載平衡程式正常運作。

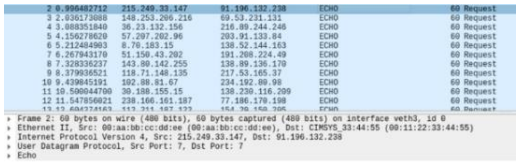

圖6. LACP 測試範例 Port 1

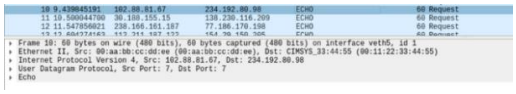

圖7. LACP 測試範例 Port 2

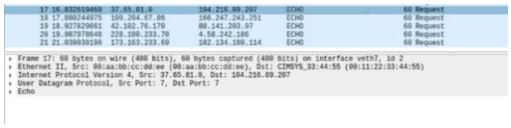

圖8. LACP 測試範例 Port 3

3.4 斷線復歸

藉由 Load Balance 可以有效分散網路流量到各線路，不過也有所限制，如果當某條線路斷線時，需要快速的將線路流量轉至其他正常之 Port，當線路回復時，則不能馬上切回正常線路，以 TCP 或其他對於 RTT 較為敏感的應用程式為例，如果馬上切回去原本線路，可能會因為封包去回不同路或是 Time out 等等問題導致封包無法正確傳遞，在 Action Selector 的機制底下，透過 Control Plane 下達 Flow 指令，將 LAG 群組中之無法連接的 Member 從 False 改為 True，Action Selector 將會立即將原先 Hash 計算之線路切換回去原本之狀態，因為有如此情況，所以我們需要在影響連線最小的情況下回復原本之 Port 設置。

本次舉的例子將設置一個 LAG 群組為 Port 1、Port 2、Port3，透過 Action Selector 之 Resilient 演算法進行示範。

	Port
Dst IP 1	1
Dst IP 2	2

Set Port 1 False

	Port
Dst IP 1	3
Dst IP 2	2

圖9. Action Selector Member 設置為 False

此時如圖9.，封包透過 Hash 進行計算後，Action selector 函式將送往第一個 IP 之封包導向 Port 1，以及送往第二個 IP 之封包送往 Port 2，Port 3為暫時閒置，這時候透過 Control Plane 下達 Port 1 為 False 之指令，此時將會透過演算法將原本 IP 1 所使用的 Port 1利用 Hash 導向至其他 Port，目前於範例被 Action Selector 函式導向 Port 3，而 IP 2所使用之 Port 2因為演算法將維持住 Port 2。

	Port
Dst IP 1	3
Dst IP 2	2

Set Port 1 True

	Port
Dst IP 1	1
Dst IP 2	2

圖10. Action Selector Member 設置為 True

但是如果線路修復完成，如圖10.，透過 Control Plane 進行將 Port 1 設置為 True，此演算法將會回復為原本所使用之 Port 1，封包有可能因為快速的變換 Port 導致 Time out 或是去回不同路而導致封包掉包的情況發生。

4. 客製方案

使用 Action Selector 方案雖然方便，但是因為部分機制，例如：Port 完全交由 Hash 機制分配，線路修復後 Port 自動回歸等等原因，如果我們需要確保連線的可用度不會因為這些原因導致 Time out 或是去回不同路等等問題，將會利用 Table 搭配 Control Plane 的功能去執行更客製化的流量操控。

4.1 使用 Table 實作動態化調整及權重配置

圖11. Table 機制實作負載平衡流程圖

如圖11.，根據 Action Selector 的原先機制發想，首先由封包傳入 Ingress 後，利用 Parser 定位各 Header 之 Data，並透過 Hash 函式，利用5-tuple 或額外想增加為變數的欄位算出該封包之 Hash 值，將 Hash 值經過 mod n 之計算，並利用 Match 指定之餘數，傳送至指定的 Port。

mod 8

餘數	Port
0	1
1	1
2	1
3	2
4	2
5	2
6	3
7	3

圖12. 負載平衡 Table 示範概念圖

在圖12.中，我們將以 Hash 值 mod 8為例，將 Table 預先寫入若 Match 到哪個餘數，則傳送至某 Port，將會像圖中所示大約將 Port 平均分配至 Port 1、Port 2、Port3。

餘數	Port
0	2
1	3
2	3
3	2
4	2
5	2
6	3
7	3

圖13. 動態分配概念圖

在圖13.中，則說明如斷線後利用 Control Plane 將原本流量導向 Port 1之 Table 進行 Port 的更動，則讓目前 Table 中使用之 Port 不會導向斷線之 Port。

餘數	Port
0	2
1	3
2	3
3	2
4	2
5	2
6	3
7	3

Set Port 1 True

餘數	Port
0	1
1	1
2	1
3	2
4	2
5	2
6	3
7	3

圖14. Action Selector 設置 Port 1為 True 示意圖

在圖14.中，利用 Action Selector 進行 Member 的調整，將會有立即性的 Port 變動。

餘數	Port
0	2
1	3
2	3
3	2
4	2
5	2
6	3
7	3

Set Port 1 True

餘數	Port
0	2
1	3
2	3
3	2
4	2
5	2
6	3
7	3

圖15. 暫緩已修復 Port1連線之更動概念圖

在圖15.中，如果我們利用 Control Plane 以及相關複合機制來判斷的方式運作，這將會使 Port 的更動暫時放緩，讓尚在運作的線路影響性降至最低，我們在下個章節將會講解到複合機制的判斷。

4.2 複合機制動態調整

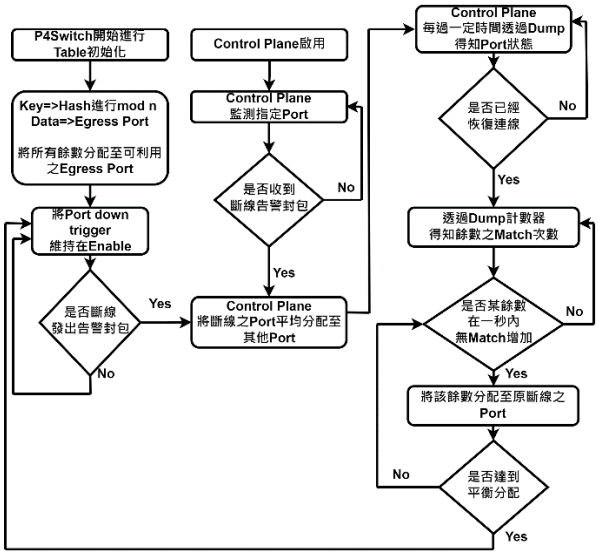

圖16. 動態化調整完整流程圖

在圖16.中，我們將會使用多種機制結合利用 Control Plane 去控制 Table 來讓負載平衡的連線可用度盡量不受影響，透過 Port down trigger 由 Data Plane 發出告警封包至 Control Plane，與 Control Plane 的配合下可以快速更正送往斷線 Port 之流量，透過監控通過之封包數量，在可允許範圍內且影響連線最小的情形下進行流量的變動，使用 Port down trigger 的優勢是從 Data Plane 可以迅速反應，第一時間將不依賴 Control Plane 之監控，則平時採用監控時間間隔較長的方式來減少 Control Plane 的負擔，將只在確定為斷線的情況下才加速監控間隔，確保第一時間能夠開始進行連線回復的機制。

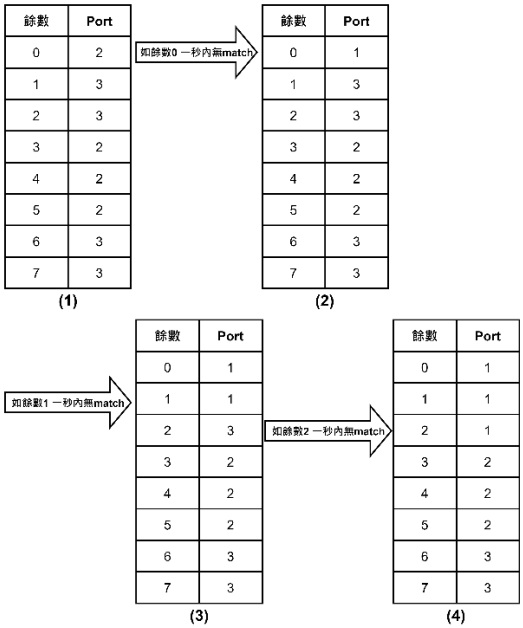

圖17. 動態化調整範例示意圖

如圖17.所示，我們將會採取階段式調整，將確定此 Table 的某些餘數目前沒有被 Match 到，方復歸回原使用之 Port，使連線影響性降至最小，在未來會在 Control Plane 增加演算法來計算每條線路之權重以及修正及調整判斷機制使 Port 分配之誤判降低。

4.3 權衡與優劣勢

利用我們提出的方式可以有效照開發者的想法利用 Control Plane 進行 Table 的更動而不用受限於 Action selector 的原先機制，不過這種利用 Table 控制的方法自由度雖高，但牽扯到複數 Table 的控制以及 Control Plane 需要進行額外監控，因此影響到效能之表現。倘若搭配 Tofino Switch 的專屬 Packet Generator 功能，可將部分監控機制分散於 Data Plane，除了可以比從 Control Plane 更快之反應外，也可搭配複合機制來讓 Control Plane 進行 Table 控制的判斷。

5. 結論

我們所提出的方法，將 Control Plane 從 Data Plane 中獲取資訊，去做斷線偵測以及演算法動態分配 Port，如果使用自定義 Table 進行實作，預計將可獲取更大自由的動態分配流量之演算法，並且倚靠斷線偵測之機制，可以不必時常從 Control Plane 下達 Dump 指令去獲取 Port 的狀態，僅在已知有 Port 斷線時，才會定時獲取 Port 的狀態，如果一旦斷線狀態回復後再利用 API 自動更改 Table 處理動態分配流量的機制。

本次提出的方法未來將會包含流量負載平衡、非平衡輸出之分配演算法、連線同路去回、版本控制、Table 備份等等之功能，透過開發非平衡輸出之分配演算法亦能增加本中心相關論文數量。

參考文獻

[1] P4_16 Portable Switch Architecture (PSA). [Online]. Available: https://p4.org/p4-spec/docs/PSA.html

[2] Intel® Tofino™ 系列可程式化乙太網路交換器 ASIC. [Online]. Available: https://www.intel.com.tw/content/www/tw/zh/products/network-io/programmable-ethernet-switch/tofino-series.html

[3] E. Haleplidis, K. Pentikousis, S. Denazis, H. Salim, D. Meyer, and O. Koufopavlou. Software-Dened Networking (SDN): Layers and Architecture Terminology. RFC 7426, Internet Engineering Task Force (IETF), January 2015.

[4] v1model Architecture Definition. [Online]. Available: https://github.com/p4lang/p4c/blob/main/p4include/v1model.p4

[5] Open Networking Foundation - Next-Gen SDN Tutorial - Session 1: P4 and P4Runtime Basics. [Online]. Available: https://opennetworking.org/wp-content/uploads/2019/10/NG-SDN-Tutorial-Session-1.pdf

[6] Link Aggregation Control Protocol [Online]. Available: https://www.ieee802.org/3/ad/public/mar99/seaman_1_0399.pdf