基於 P4之帶內網路遙測數據時間戳同步之研討

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
2503134, 1203053, 0303118, tlliu@niar.org.tw

摘要

因應軟體定義網路（SDN）[1]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，而 P4 交換器在進行時，無論是以 BMv2或是 Tofino 等等 P4交換器所自行擁有的設備時間戳是各自獨立的，所以如果需要進行多台設備之時間戳運算而非僅限於使用機器本身內部的時間戳，此時就需要為各設備進行時間戳同步的動作。

在本論文將會提出基於 P4之帶內網路遙測技術之中對於時間戳同步的可行做法，透過與 Control Plane 間之封包延遲進行計算，將可讓 P4交換器在不更改其設定的情況下算出與其他 P4交換機間之時間戳差異值，此外亦針對 Inter-domain 和 Intra-domain 之情境分別討論實作之細節，可讓各地之 P4交換機在 Control Plane 中保持時戳同步。

關鍵詞：P4、INT、In-band Network Telemetry

Abstract

Following the evolution of Software-Defined Networking (SDN), developers have expanded their research from OpenFlow in the control layer to the P4 language targeting the data plane. During the operation of P4 switches, whether using BMv2 or Tofino, the device timestamps are independent. Therefore, when timestamp calculations across multiple devices, it is necessary to synchronize timestamps across these devices rather than relying solely on the internal timestamps of individual machines.

This paper proposes a practical approach for timestamp synchronization based on P4 in-band network telemetry. By calculating packet delays within the control plane, P4 switches can compare timestamp differences with other P4 switches without modifying their configurations. Additionally, the paper discusses implementation details for both inter-domain and intra-domain, enabling P4 switches across different locations to maintain timestamp synchronization through the control plane.

Keywords: P4, INT, In-band Network Telemetry

1. 前言

Programmable Protocol- Independent Packet Processors(簡稱為 P4)[2]是一種與協定無關、可程式化的開放式交換器高階語言。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進行開發，相較於傳統網路，可程式化交換網路擁有更多的彈性及客製化。

在網管面中有一種 In-band Network Telemetry(INT)[3]的技術，利用 P4語言並透過 INT 的框架，可以讓開發者在 Data Plane 中收集資訊，並回傳給 Control Plane 透過這些資訊來針對 Data Plane 進行類似於流量政策的行為。

為了解決不同設備之間的時間戳問題，在本次基於P4之帶內網路遙測數據時間戳同步之研討中，將講解整體架構以及製作理念，未來會將此觀念引入由NCHC與NICT所共築之跨域P4 INT平台進行實作。本次論文將在章節2講解P4相關的背景知識，章節3講解時間戳同步可行機制之研討，並將結論陳述於章節4。

2. 背景知識

本章節將會介紹本篇論文所使用到之背景知識包含 P4程式語言、INT 以及跨域 INT 測試平台的介紹,分述如下。

2.1 P4(Programming Protocol-independent Packet Processors)

P4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語言，開發者可以解析各封包之架構，並針對各欄位進行所需之操作，P4的特色為可支援任何通訊協議以及任何平台並可隨時更改交換規則。

2.2 基於 P4之 In-band Network Telemetry (INT)技術

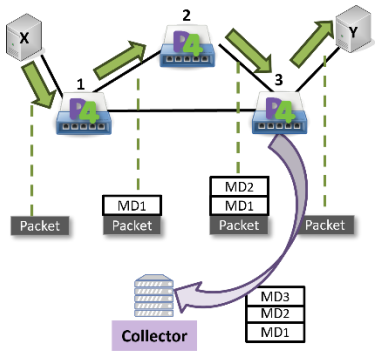

圖 1 INT 工作流程示意圖

In-band Network Telemetry 目前在 P4環境中已有可實作之規範，該架構如圖1.，可以允許開發者在 Data Plane 中收集資訊以及回報網路的狀態，而不需要 Control Plane 去介入收集資訊的行為，過程中可將 INT Header 嵌入至一般的資料封包、複製封包、或是特殊的探測封包，透過 Header 中的指

令及開發者撰寫之 P4程式，可以收集指定之資訊，最終這些資訊將傳送至監控系統以彙整整體網路之狀況。

2.3 NCHC、NICT 跨域 INT 測試平台

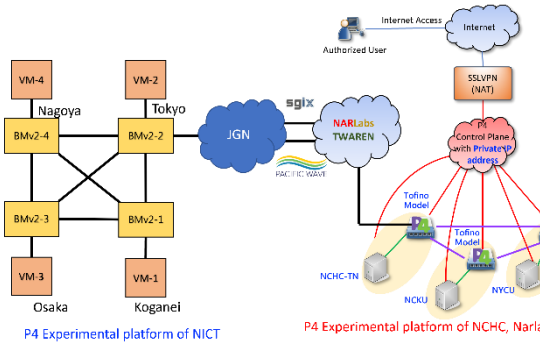

圖 2 NCHC、NICT 跨域 INT 測試平台

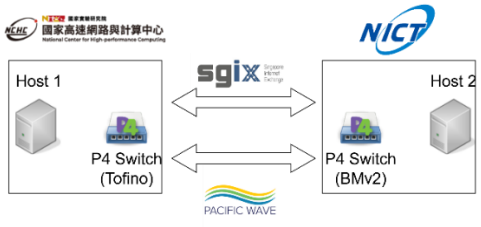

圖 3 跨域 INT 簡化實驗拓樸

如圖2.所示，我們的實驗整體架構為 NCHC、NICT 跨域 INT 測試平台[4]，左半邊為日本 NICT 的 P4實驗平台，透過 JGN 網路對外連接。右半邊為本中心之 P4實驗平台，透過 TWAREN 網路對外連線。現階段而言，雙邊實驗平台已經透過 LA 以及新加坡介接。由本中心提供 Tofino 交換器設備，而 NICT 提供 BMv2之軟體交換器，如此結合將可在此平台測量不同 P4設備間程式碼的兼容性。圖3.為簡化之實驗拓樸，未來將在兩種不同的平台之間進行連線測試，並使用不同之方法實作 INT。

2.4 INT 實驗方式不足之考量

大多數情況下之 P4 INT 實驗，通常用於融合不同間協議，估算機器內部之處理時間以及封包隊列擁擠程度來計算流量政策，或是減低INT本身的資源消耗或是壓縮資訊，如[5]是屬於 SRv6協議融合 INT 框架的論文， [6]則是提出在 INT 上減低資源消耗之方案，因針對跨域跨設備之INT實作方式較少，並為了可以讓更多的遙測資訊可供使用，我們可以針對時間戳的面向進行研討。

3. 時間戳同步之研討

本章節將會進行分別為 Intra-domain INT 以及 Inter-domain INT 實行方式之探討。

3.1 Intra-domain INT

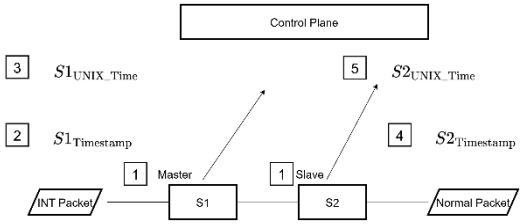

圖 4 Intra-domain INT 測試拓樸

關於 Intra-domain 方面，目前正在研究實作方式將如圖4.所示。

第一步將會先決定要於 Domain 內的其中一個節點當作基準點，這時可考慮採用 INT Source 節點作為 Master，其餘皆為 Slave，這時將採用 S1當作 Master，第二步將如一般的 INT 模式將封包經過 INT Source 節點，此時會將帶有 S1 Timestamp 的封包 Mirror 一份至 Control Plane，第三步 Control Plane 將會記錄收到該封包之 UNIX 時間，第四步 INT 封包將會繼續記錄 S2之 Timestamp 並 Mirror 一份封包至 Control Plane，第五步 Control Plane 將會記錄收到該封包之 UNIX 時間，此時暫定 INT 流程完整結束，第六步將會計算兩個 Mirror 封包之 UNIX 時間差異，如方程式(1)。

$$S1_{\text{UNIX\_Time}} - S2_{\text{UNIX\_Time}} = S1S2_{\text{diff}} \quad (1)$$

第七步將會計算 Master 對於 Slave 之 Timestamp 差異，如方程式(2)。

$$S1_{\text{Timestamp}} + S1S2_{\text{diff}} = S2_{\text{Slave}} \quad (2)$$

第八步，例如: S1至 S2之間的 UNIX 時間差為 0.913200048秒，此時 S1之 Timestamp 為 2631.370082073秒，對於 S1來說，正確之 S2 Timestamp 應為2,632.283282121秒，這個值暫定為 S2_slave，藉由這個數值，Control Plane 在一定時間內可以針對 S2之原機器帶有之 Timestamp 進行校正，關於校正數值之計算，如方程式(3)，可利用 S2_slave 減去原本 S2之 Timestamp，此時會得到校正數值，將 2,632.283282121 減去 4920.024457574 將會獲得 -2,287.741175453，此時這個值稱為 S2_modify，

$$S2_{\text{slave}} - S2_{\text{Timestamp}} = S2_{\text{modify}} \quad (3)$$

在一定時間內將會透過 S2_modify 在 Control Plane 針對 S2的 Timestamp 進行時間上的校正，因為流量管理政策都是經由 Control Plane 進行下發 Entry，相關數據也是經由 Control Plane 計算，所以此方法並不會實際的對 Data Plane 設備進行校正。

前面所提到的一定時間此講法，將是針對 Tofino Switch 內部之 Timestamp Clock 的計算上限，因為該值超過上限將會重置一次 Clock，所以目前

預估一天進行校正一次，若偵測到內部 Timestamp 快抵達上限，將延後一定時間進行校正，透過這些校正將能正確測量出 Hop 之間的 RTT。

3.2 Inter-domain INT

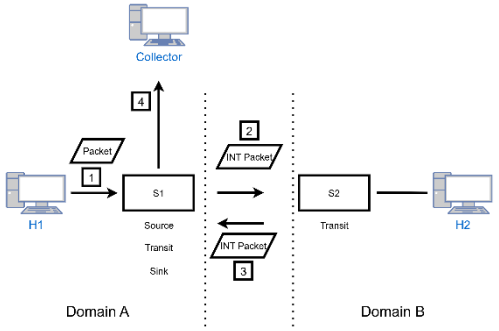

圖 5 Inter-domain INT 測試拓樸

關於 Inter-domain INT，我們將採用圖5.所示之拓樸以及步驟，可以有效利用 P4 INT 原有之功能，知道每個節點之間的RTT，步驟一，我們會先傳送一般封包至 S1 (Source 節點)，使一般封包攜帶 INT Header，並且記錄第一個節點的資訊，步驟二，傳送該封包至 S2 (Transit 節點)，這時該封包將會疊加第二個節點的 INT 資訊，步驟三，我們透過下發 Flow 給 Table 進行接下來的動作，如果條件符合來自 Domain 1之指定量測封包，則將 INGRESS PORT 寫入 EGRESS PORT，這些動作將會把封包回傳至原本的 S1節點，步驟四，當 S1偵測到 S2所送回之封包則啟用 SINK 節點的功能，進行 INT 封包的拔除以及將 Metadata 資料送往 Collector，步驟完全結束後將身為 INT 第一個節點的 S1 Egress Timestamp 扣除身為第三個節點的 S1 Ingress Timestamp 即可知道節點間之 RTT。

4. 結論

從我們的 Inter-domain 以及 Intra-domain 的 INT 實作方式了解到，除了時間戳的同步是一個挑戰，Inter-domain 所採用之方式因為是在 Data Plane 進行實作，然後才將資料彙集於 Control Plane，這對於 Control Plane 對於 Data Plane 的 RTT 時間影響較小，但若使用 Intra-domain 之測量方式，雖然能夠準確校正每一台的時間，但因為計算方式分布於 Control Plane 以及 Data Plane，所以如果需要近一步的校準時間，將要多加考慮 Control Plane 對於 Data Plane 的 RTT 時間影響，未來將針對本次提到之跨域 INT 實驗平台進行時間戳同步之實驗及測試。

參考文獻

[1] 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.

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

[3] In-band Network Telemetry (INT) Dataplane Specification. [Online]. Available: https://p4.org/p4-spec/docs/INT_v2_1.pdf

[4] 周大源, 胡乃元, 曾惠敏, 劉德隆, “台日間跨國多路徑帶內遙測實驗平台建置與測試規劃,” TANet2024研討會, 台北, 2024 年10月

[5] Liu, Ying, et al. "SFANT: A SRv6-based flexible and active network telemetry scheme in programming data plane." IEEE Transactions on Network Science and Engineering 11.3 (2023): 2415-2425.

[6] Yang, Kaicheng, et al. "SketchINT: Empowering INT with TowerSketch for per-flow per-switch measurement." *IEEE Transactions on Parallel and Distributed Systems* 34.11 (2023): 2876-2894.