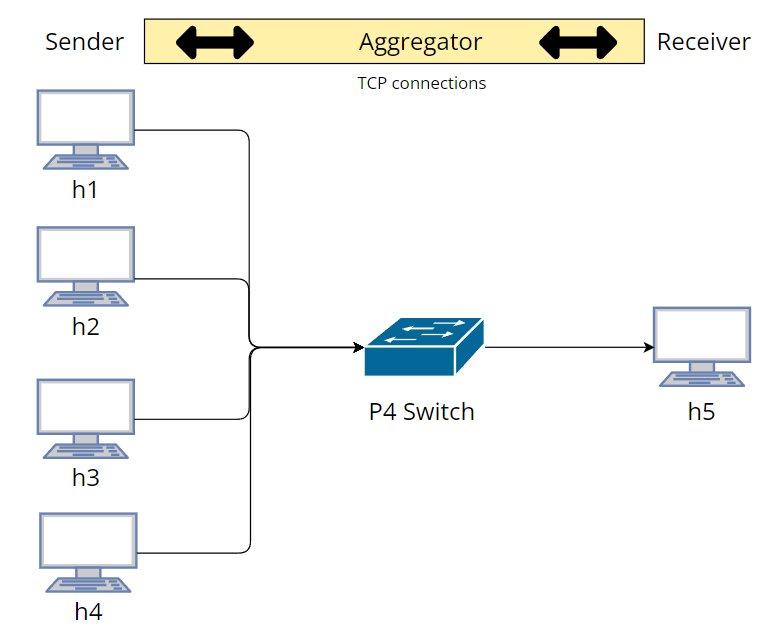
**專題摘要**

本專題探討了如何在軟體定義網路(Software-defined Networking, SDN)環境下，利用P4語言實作在TCP連線中運作的資料聚合機制。此機制由三個核心部分組成：Reliable、In-Network 以及 Aggregation。主要目的是讓資料透過可程式化交換機在網路內部處理資料而非常見的終端，從而提高資料處理的效率。

**實驗架構與方法**

實驗環境利用Mininet模擬網路拓樸，並使用P4語言運行於bmv2網路模擬交換機上，形成一個小型的獨立模擬網路。每個發送端與接收端之間均建立TCP連線，透過scapy或iperf傳送封包。封包解析後，交換機根據TCP header中的sequence number來辨識封包順序，並將封包的payload暫存於交換機內部的register中。當特定序號的資料收集完成後，將結果傳送給接收端，實現資料的網內處理。

實驗架構如圖所示：



**核心技術與實作機制**

TCP Cheater

為了維持TCP連線的穩定性，系統運用了TCP cheater機制，通過修改封包內容欺騙每個終端認為其TCP封包未被改動。此方法依賴TCP的SEQ number及ACK number機制，並在sender與receiver完成TCP handshake後將initial SEQ number與ACK number存入register中。透過控制封包的payload大小以及計算相對序號(relative\_seq\_num)，交換機可以處理每個封包的payload並將結果存入對應的封包中。最後，系統選擇一個sender作為leader，將資料通過該sender的TCP連線傳送至接收端，避免TCP錯誤重傳。

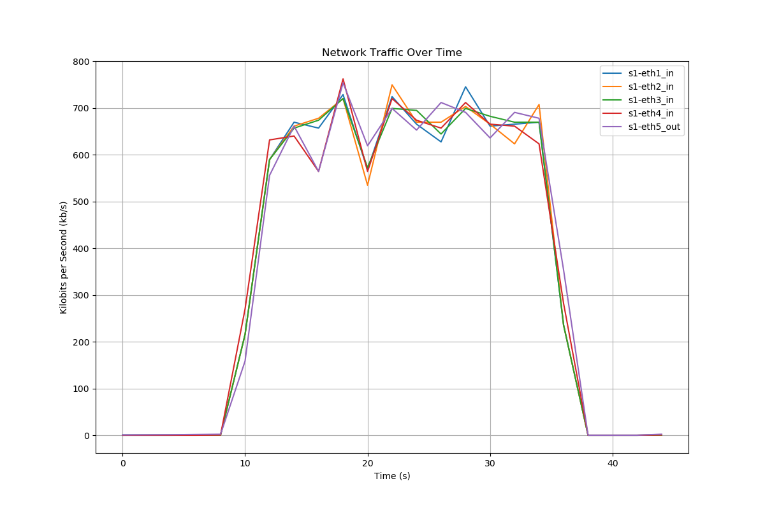
**Ring Buffer**

為了有效聚合發送端的資料，我們設計了一個環狀緩衝區(Ring Buffer)來儲存封包序號、payload與聚合進度。緩衝區以序號排列資料，並維護min\_index與max\_index指標以追蹤尚未聚合及最早未ack的封包。當新封包進入時，會根據序號與min\_index、max\_index比較後進行相應處理。當接收到的ack封包通過交換機時，會將其廣播至所有發送端，完成TCP連線的回復。

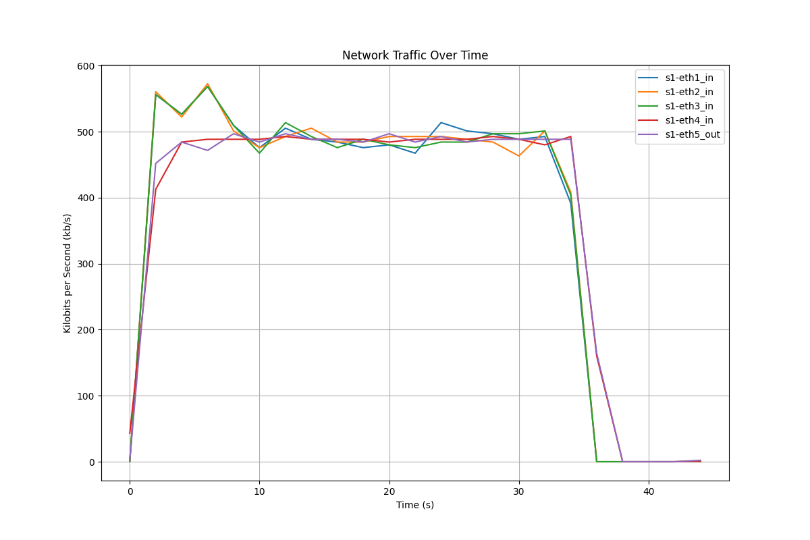
**實驗結果**

為了驗證機制的效能，我們設計了四種情境進行測試：

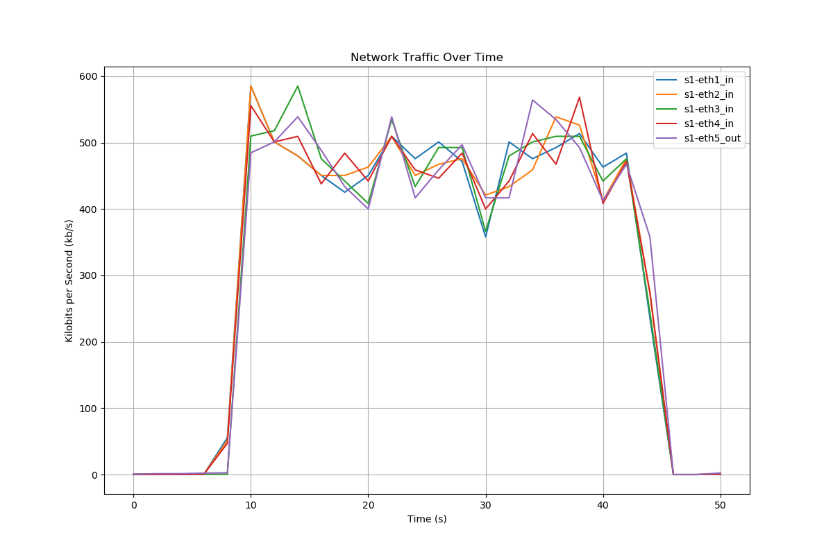
1. **Sufficient Bandwidth**：在所有link具有相同頻寬的情況下，測試結果顯示每條link速率相近，與預期一致。



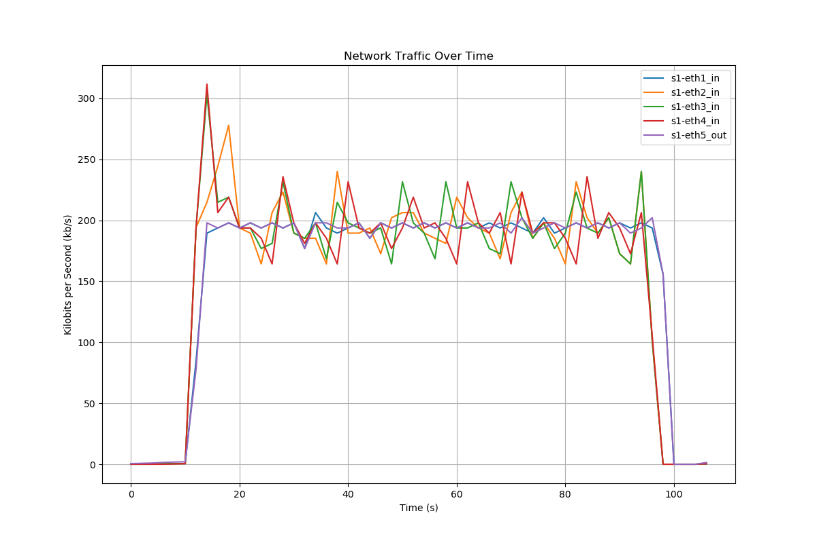
1. **Single Slower Sender**：當其中一個發送端頻寬較低時，速率會收斂至較慢的發送端頻寬，與TCP的機制相符。



1. **Slower Receiver**：接收端頻寬較低時，整體速率會收斂至接收端的頻寬，並顯示出瓶頸效應。



1. **All Different Bandwidth**：當發送端與接收端頻寬不同時，較高頻寬的發送端會一開始速度較快，最終收斂至最慢頻寬附近。



**結論**

通過本專題的實驗，我們證明了在SDN環境下運行的可程式化交換機可以成功實現資料的網內聚合，有效提高資料傳輸效率。未來的工作可以針對實體硬體的限制進行進一步的優化和改進。