313551137 官劉翔

**實作步驟&結果:**

1. 要先開啟mininet再開啟ryu (這樣topology discover才能正常運行)
2. 啟動ryu後，確定有number of switch and link (lab2的topology 應該為8個switch與26條link)，如果失敗就重啟ryu (--observe-links是必要的，否則controller無法得到topology

ryu-manager dual\_disjoint\_shortest\_path.py --observe-links

1. 結果  
   一張含有 螢幕擷取畫面 的圖片

   自動產生的描述

**實作流程:**

我在這裡的實作流程為下

1. 先用tree2,2 topology，確認get\_topology可以順利讓controller收到network topology
2. 使用lab2測試用的topology，並將一開始的ARP flood的broadcast storm解決掉
3. 實作出single\_shortest path
4. 實作出dual\_shortest path
5. 實作出dual & disjoint\_shortest



**實作細節 (程式碼講解):**

首先，我看程式的整體架構。這次主要目的是實現具有雙路徑功能的最短路徑轉發。在網路中，有時候使用單一路徑可能會造成網路擁塞或者在路徑故障時無法快速恢復，因此我實現了一個可以同時使用兩條最不相交路徑的控制器。

**在程式的初始化部分：**

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

我使用了NetworkX的DiGraph（有向圖）來儲存網路拓撲。這個選擇非常重要，因為在實際網路中，連接是有方向性的，每個方向可能有不同的端口號和特性。paths字典用來快取已經計算過的路徑，這是一個重要的優化，因為路徑計算是比較耗時的操作。switch\_mac\_matches用來追踪已經處理過的MAC地址配對，避免重複處理相同的流量，也避免廣播風暴。group\_ids則用於管理OpenFlow的群組表，這是實現多路徑轉發的關鍵。

**在Topology discover部分**，get\_topology函數扮演著關鍵角色：

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

每當有新的交換機加入網路時，這個函數就會被調用。它使用Ryu的API獲取所有交換機和它們之間的連接信息，然後更新我的NetworkX圖。這個函數不僅僅是單純地收集信息，它還需要正確處理每條連接的雙向性質，因為在SDN中，我需要知道每個方向的具體端口號。

在**雙路徑的實現上**，find\_disjoint\_paths\_from\_shortest函數：

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

這個函數的設計思路是，在所有最短路徑中找出重疊節點最少的兩條路徑。這樣做有幾個好處：首先，由於是在最短路徑中選擇，所以不會有路徑過長的問題；除此之外，選擇重疊最少的路徑可以提高網路的可靠性，因為如果一個節點故障，影響的路徑會更少。

**在實際的封包處理中，packet\_in\_handler函數扮演著核心角色：**

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

當交換機收到一個新的封包且不知道如何處理時，就會將它送到控制器，觸發packet\_in事件。這個函數首先解析封包，獲取源MAC地址和目的MAC地址。這些信息對於後續的路徑選擇非常重要。

特別值得注意的是對broadcast封包的處理：

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

這部分處理了ARP請求，這在網路中是非常重要的，因為它用於解析IP地址和MAC地址的對應關係。我使用switch\_mac\_matches來追踪已經處理過的ARP請求，避免產生廣播風暴。

最後，在add\_flow部分：

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

這部分代碼根據計算出的路徑數量決定使用add\_flow還是add\_dual\_path\_group。如果只有一條路徑，就直接add\_flow；如果有多條路徑，就創建group並讓流表指向這個group。這樣的設計既保證了單路徑的簡單高效，又支持了多路徑的負載均衡。

**在多路徑轉發的實現上 (add\_dual\_path\_group)**：

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

這個函數使用OpenFlow的群組表功能來實現多路徑轉發。每條路徑被放入一個bucket中，並被賦予相同的權重(weight=50)，這樣可以實現負載均衡。watch\_port參數允許交換機監控端口狀態，如果某個端口出現問題，可以自動切換到其他可用路徑。

**get\_out\_ports函數**則負責實際的路徑選擇：

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

這個函數首先檢查目的地是否在網路中，如果是的話就尋找所有可能的最短路徑，然後使用find\_disjoint\_paths\_from\_shortest選出最合適的路徑對。這個過程的結果會被快取在paths字典中，提高後續處理相同流量的效率。

我的設計不僅實現了基本的轉發功能，還加入了多路徑、負載均衡等進階特性，同時也考慮了效能優化和錯誤處理。