ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

TÌM ĐƯỜNG ĐI NGẮN NHẤT BẰNG THUẬT TOÁN DIJKSTRA TRONG RYU CONTROLLER

Học phần: Mạng máy tính nâng cao

Lớp: CQ2019/4

Họ và tên các thành viên:

1. **Bùi Lê Tuấn Anh – 19120163**
2. **Ngô Nhật Tân – 19120128**

Thành phố Hồ Chí Minh, tháng 12 năm 2021

Nội dung

1. GIỚI THIỆU VỀ RYU CONTROLLER

RYU Controller là một SDN controller, chuyên dùng để điều khiển và quản lý các đường truyền trong hệ thống mạng. RYU Controller hoạt động ở Control Plane và có một số tính chất sau:

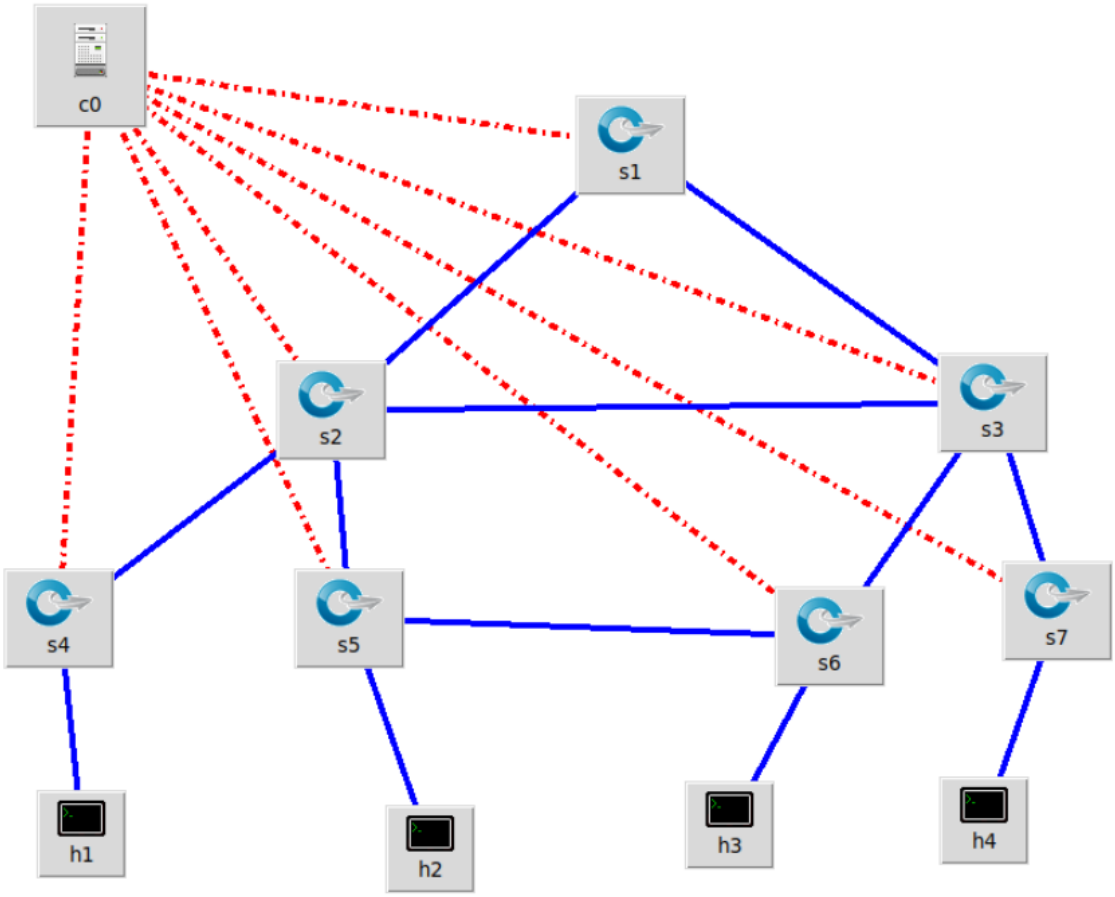
- Tính linh động: Toàn bộ các hoạt động của các thiết bị mạng đều được kiểm soát bởi Controller, giúp quá trình hoạt động của các thiết bị mạng trở nên thông suốt hơn.
- Tính tối giản: Với sự kiểm soát của Controller, các thiết bị mạng chỉ cần thực hiện theo cấu hình và quy trình có sẵn, giúp giảm thiểu sai sót và tăng tính hiệu quả cho toàn bộ hệ thống mạng.

RYU Controller tương thích với chuẩn OpenFlow, do đó thích hợp để cài đặt và cấu hình trên các hệ điều hành mạng khác nhau. Trong phần báo cáo này, nhóm sẽ sử dụng RYU Controller kết hợp với các thông số kỹ thuật sau để cài đặt thuật toán tìm đường đi ngắn nhất giữa các switch:

- Network OS – Hệ điều hành Mạng: Mininet
- Host OS – Hệ điều hành kết nối: Ubuntu 20.04 Focal Fossa.
- Ngôn ngữ lập trình chủ yếu: Python
- Chuẩn OpenFlow: OpenFlow 1.3
- Phần mềm xây dựng sơ đồ mạng (Topology): MiniEdit
- IDE cài đặt thuật toán: VIM

2. TÌM ĐƯỜNG ĐI NGẮN NHẤT VỚI RYU CONTROLLER

Nhóm tiến hành cài đặt thuật toán Dijkstra cho hệ thống mạng sau đây:



Phần cấu hình thuật toán:

```
def minimum_distance(distance, Q):
    min = float('Inf')
    tmp = list(Q)
    node = tmp[0]

    for v in Q:
        if distance[v] <= min:
            min = distance[v]
            node = v
    return node
```

```
def get_path(src, dst, first_port, final_port):

    # Dijkstra's algorithm
    print("get_path is called, src=", src, " dst=", dst,
          " first_port=", first_port, " final_port=", final_port)

    distance, previous = {}, {}

    for dpid in switches:
        distance[dpid] = float('Inf')
        previous[dpid] = None

    distance[src] = 0
    Q = set(switches)
    print("Q=", Q)

    while len(Q) > 0:
        u = minimum_distance(distance, Q)
        Q.remove(u)

        for p in switches:
            if adjacency[u][p] is not None:
                w = 1
                if distance[u] + w < distance[p]:
                    distance[p] = distance[u] + w
                    previous[p] = u
```

```
    r = []
    p = dst
    r.append(p)
    q = previous[p]
    while q is not None:
        if q == src:
            r.append(q)
            break       You, seconds ago • Uncommitted char
        p = q
        r.append(p)
        q = previous[p]

    r.reverse()
    if src == dst:
        path = [src]
    else:
        path = r

    # Now add the ports
    r = []
    in_port = first_port
    for s1, s2 in zip(path[:-1], path[1:]):
        out_port = adjacency[s1][s2]
        r.append((s1, in_port, out_port))
        in_port = adjacency[s2][s1]
    r.append((dst, in_port, final_port))
    return r
```

Giải thích thuật toán:

- Tại hàm này, đầu tiên ta tạo hai biến lưu dạng dict là **previous** (lưu giá trị node liền kề, cụ thể ở đây là **OpenFlow switch**) và **distance** (lưu khoảng cách giữa các node với nhau)
- Duyệt thuật toán Dijkstra, với khoảng cách giữa 2 node liền kề là 1. Ta tìm vị trí node có khoảng cách ngắn nhất so với node liền kề. Cộng khoảng cách rồi so sánh khoảng cách mới với khoảng cách trước đó, nếu nhỏ hơn thì xác định

khoảng cách mới. Lần lượt làm như vậy cho đến khi kết thúc thuật toán, tìm thấy được đường đi ngắn nhất giữa các node với nhau.

Đánh giá chung:

- *Thuật toán này có một tính chất là **hội tụ chậm hơn** so với một số thuật toán khắc và chỉ thích hợp với **đồ thị có trọng số dương**. Do đó khi cài đặt thuật toán này có thể sẽ xảy ra tình huống gói tin được gửi từ host lên OpenFlow switch nhưng switch không thể chọn được đường đi, buộc phải đẩy gói tin lên controller và mất một khoảng thời gian để xây dựng đồ thị chính xác.*
- *Như vậy, rất có thể gói tin sẽ bị đánh dấu là **Destination Host Unreachable (Không đến được điểm đích)** do timeout. Tuy nhiên, điểm yếu này có thể được khắc phục bằng cách khởi động lại Controller để xử lý. Trong thời gian tới, nhóm có thể xem xét cải tiến thuật toán để tăng tốc độ và độ tối ưu cho controller.*

Hình ảnh minh họa cho quá trình thực hiện:

- Chạy Controller và mô hình mạng. Tiến hành quá trình **h1 ping h4**

```
mininet> h1 ping h4 -c8
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=0.994 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.054 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.053 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.045 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=0.062 ms
64 bytes from 10.0.0.4: icmp_seq=6 ttl=64 time=0.056 ms
64 bytes from 10.0.0.4: icmp_seq=7 ttl=64 time=0.060 ms
64 bytes from 10.0.0.4: icmp_seq=8 ttl=64 time=0.048 ms

--- 10.0.0.4 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7148ms
rtt min/avg/max/mdev = 0.045/0.171/0.994/0.310 ms
mininet> S
```

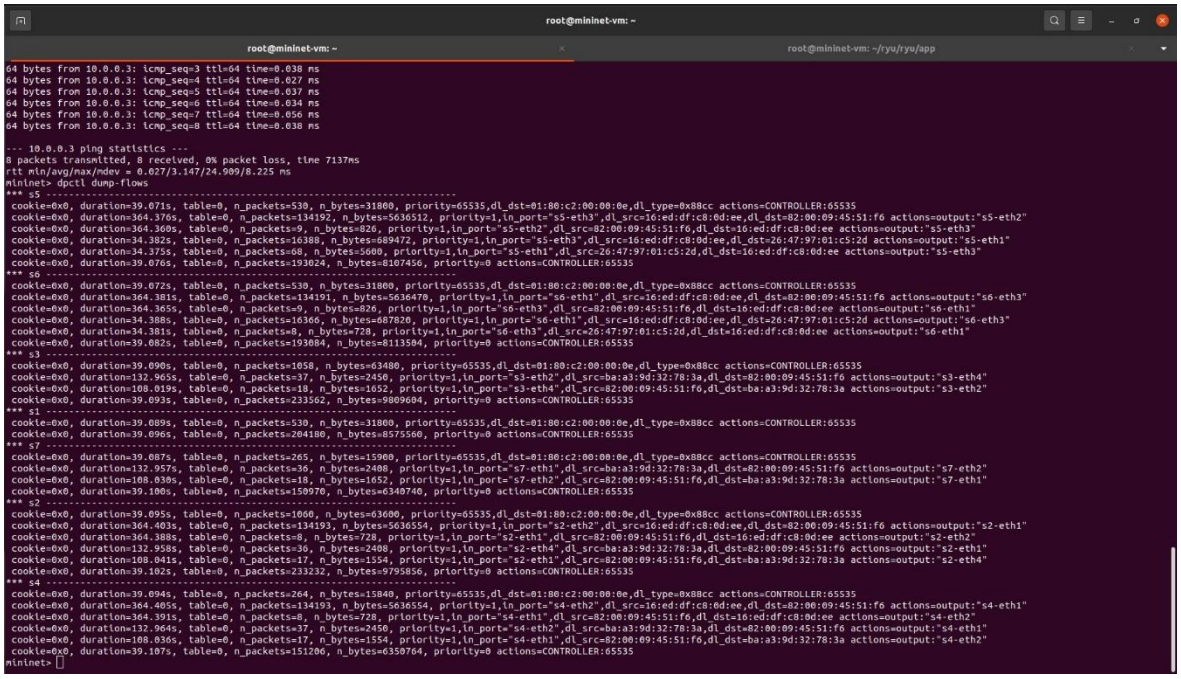
- **h1 ping h3**

```
anthony2708@anthony2708-Vostro-5581:~$ ssh mininet@127.0.0.1 -p 2222 -X
mininet@127.0.0.1's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-42-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Last login: Sun Dec 12 05:06:53 2021
mininet@mininet-vm:~$ sudo -s -E
root@mininet-vm:~# ls
mininet  oflops  oftest  openflow  pox  ryu  topo1.mn  topo1.py
root@mininet-vm:~# python2 topo1.py
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
h4 h3 h1 h2
*** Starting controllers
*** Starting switches
*** Post configure switches and hosts
*** Starting CLI:
mininet> h1 ping h3 -c8
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=55.9 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.112 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.043 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.055 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.047 ms
64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=0.049 ms
64 bytes from 10.0.0.3: icmp_seq=7 ttl=64 time=0.043 ms
64 bytes from 10.0.0.3: icmp_seq=8 ttl=64 time=0.051 ms

--- 10.0.0.3 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7128ms
rtt min/avg/max/mdev = 0.043/7.042/55.941/18.481 ms
mininet>
```

- **h2 ping h3**

```
mininet> h2 ping h3 -c8
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=24.9 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.038 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.038 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.027 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.037 ms
64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=0.034 ms
64 bytes from 10.0.0.3: icmp_seq=7 ttl=64 time=0.056 ms
64 bytes from 10.0.0.3: icmp_seq=8 ttl=64 time=0.038 ms

--- 10.0.0.3 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7137ms
rtt min/avg/max/mdev = 0.027/3.147/24.909/8.225 ms
mininet>
```

- **Các luồng được cài ở các OpenFlow switch**



Như vậy, toàn bộ quá trình thực hiện đã thành công theo yêu cầu.

TÀI LIỆU THAM KHẢO

1. Hướng dẫn code thuật toán Dijkstra cho RYU Controller: http://csie.nqu.edu.tw/smallko/sdn/dijkstra_ryu.htm
2. Github: https://github.com/amitsk1994/mininet-RYU-ShortestPath
3. Website Ryu, cung cấp hướng dẫn cho việc cấu hình chung: https://ryu.readthedocs.io/en/latest/writing_ryu_app.html