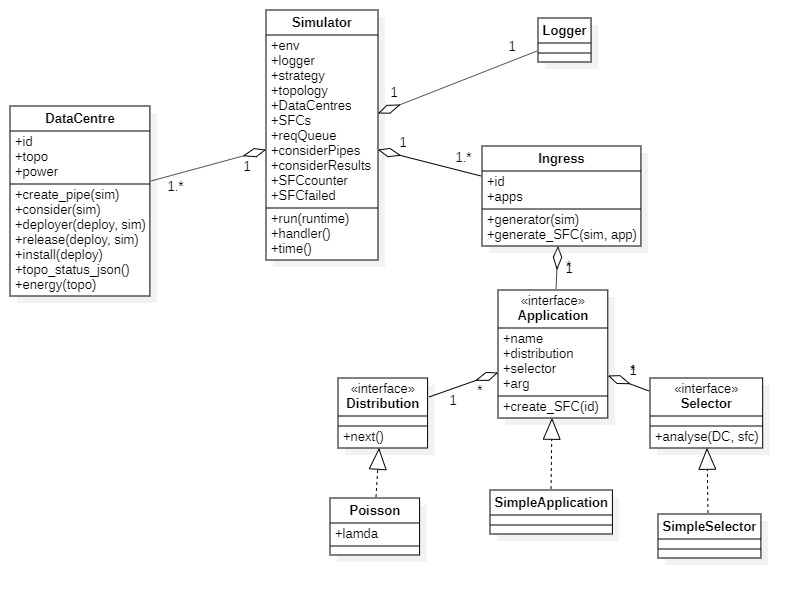
Class Diagram



Quick views:

+ các “topo”, bao gồm topo lớn, topo fat tree trong DC, topo của SFC, dùng networkx

+ application định hình kiểu SFC (nối chuỗi hay waxman, phân phối, thuật toán xếp SFC vào DC.

+ các node lớn gồm DataCentre và Ingress được tách ra class riêng, 1 Ingress có thể chạy nhiều app với nhiều phân phối khác nhau và thuật toán khác nhau. DataCentre không có nhiều tuỳ biến

+ công suất được tính trong quá trình code chạy chứ không phân tích dữ liệu sau mô phỏng

+ có thể thêm các độ trễ tại các khâu: deploy, remove, …, nếu cần thiết

Problem:

+ chưa giải quyết được strategy 2 trong bài báo

+ chưa tích hợp topo waxman

- log chưa hoàn chỉnh, cần cải thiện thêm sau mỗi buổi báo cáo

- các cấu trúc dữ liệu chưa được hoàn chỉnh, có thể là hơi rối và hơi thừa

- chưa xem xét đến khả năng có thể bỏ được tài nguyên simpy.Store hay không? (loại tài nguyên này cho vào code có cần thiết hay hữu ích không hay làm code rối hơn?)

giải thích hoạt động

khởi tạo như file main\_centralized.py hoặc main\_distributed.py

Simulator.run():

Text

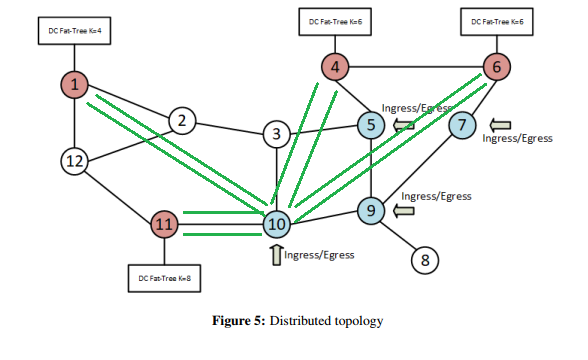
Description automatically generated

gọi hàm Ingress.generator(), hàm này sẽ tạo ra 4 tiến trình generate\_SFC tại mối Ingress

tạo tiến trình DataCentre.consider

1. Các SFC được tạo ra từ các Ingress sẽ được bỏ vào 1 hàng đợi là Simulator.reqQueue

2. Simulator.handler lần lượt lấy ra từng SFC một và gửi bản sao SFC này cho tất cả DC



đường ống màu xanh là DataCentre.considerPipe, đường ống kết nối mỗi DC tới Ingress tạo request, đường ống này được tạo ở:

Simulator.py

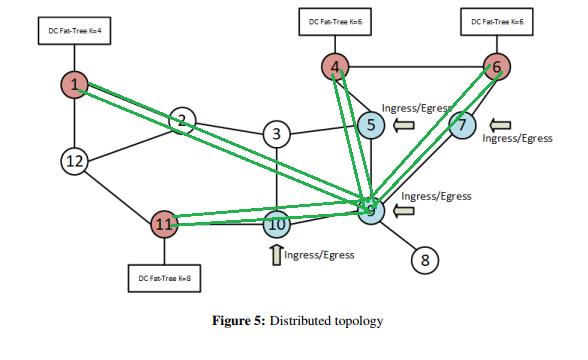
Text

Description automatically generated

và DataCentre.py



đường ống này luôn có 1 đầu ở DC, đầu còn lại ở Ingress có sfc đang được xử lí. nếu sfc từ Ingress-10 đang được xử lí, pipe có dạng như trên, nếu sfc từ Ingress-9 đang được xử lí, pipe như sau:



3. tại mối DC sẽ xử lí sfc nhận được, deploy “giả” sfc đó theo thuật toán tối ưu, và trả về thông tin có deploy được không, nếu có thì deploy như nào, năng lượng tăng thêm là bao nhiêu

tại Simulator.handler() có được những thông tin trên sẽ lựa chọn DC nào tối ưu nhất để đặt sfc lên đó, ở đây dùng thuật toán đơn giản từ dòng 59-68

Text

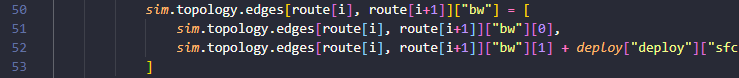
Description automatically generated

nếu chọn được DC để deploy, sẽ đưa (đoạn này không dùng simpy mà chỉ gọi hàm, ở dòng 77) sfc tới cho DC xử lí, DC tiến hành cập nhật thông số ở hàm deployer và gán tiến trình release sau thời gian TTL sẽ giải phóng sfc đó khỏi hệ thống

DC.install là deploy giả, topo của DC không bị đổi

DC.deployer gọi tới DC.install và lần này sẽ thay đổi thông số topo

khi cập nhật thông số, ở hàm DC.deployer và DC.release, chú ý đoạn code ngu này



…[“bw”] = [1000, 12], là một mảng có ý nghĩa [max bandwidth, usage], đoạn trên nhằm mục đích tăng usage lên một lượng bằng bw của link ảo nhúng vào, dùng += không được nên phải dùng như kia @@, có lẽ cái này là tại networkx

một số formdata được ghi trong file formdata.txt

file analyse.py là test thử phân tích dữ liệu log xem code chạy có đúng không nhưng ngại quá chưa test lại, tới bh đã lọc được kha khá case chạy sai của code và đã sửa

folder results chứa thông tin log, result\_event.csv log ra các sự kiện và trường tương ứng, các sự kiện create, deployed và remove sẽ kèm theo một thông tin là topo, và thông tin này được log ra file result\_topo.txt, cụ thể:

create: log ra topo của SFC

deployed, removed: log ra topo của DC liên quan tới sự kiện đó

file này dài lắm không xem được bằng mắt mà phải dùng code python chọc vào đọc @@, cấu trúc bên trong:

[

{“event”:1, “node”: …, “link”: …}, // sự kiện deployed, remove

{“event”:1, “vnf”: …, “vlink”: …} // sự kiện create

]

code có liên quan đến syntax trên bao gồm:

Ingress.py

Text

Description automatically generated

DataCentre.py

Text

Description automatically generated

Logger.py

