ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÀI TIỂU LUẬN KẾT THÚC HỌC PHẦN

MẠNG TRUYỀN THÔNG VÀ MÁY TÍNH 2

Nhóm 2:

Nguyễn Lâm Anh - 19021412

Phạm Minh Đức - 17021499

Phạm Xuân Thành - 19021515

Nguyễn Thanh Tùng - 19021534

Khoa: Điện Tử - Viễn Thông

Lớp học phần: ELT3214E_20

Hà Nội, tháng 1 năm 2022

ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

ĐỀ BÀI: XÂY DỰNG MÔ PHỎNG VÀ ĐÁNH GIÁ HÀNG ĐỢI ƯU TIÊN (PRIORITY QUEUING)

Giảng viên hướng dẫn: TS. Lâm Sinh Công

Khoa: Điện Tử - Viễn Thông

Lớp học phần: ELT3214E_20

Hà Nội, tháng 1 năm 2022

Phụ lục:

I. Tổng quan về hàng đợi.

1. Hàng đợi trong một hệ thống dịch vụ.
2. Hàng đợi một máy chủ.
3. Hàng đợi nhiều máy chủ.

II.Hàng đợi ưu tiên:

1.Nguyên tắc

2. Cơ chế

3. Các công thức.

III.Hàng đợi ưu tiên không có trước:

1.Nguyên tắc:

2. Các công thức.

IV. Mô phỏng:

1. Giới thiệu ns3
2. Kiến trúc mạng mô phỏng
3. Phân tích code mô phỏng kiến trúc (file queue-discs-benchmark.cc)
4. Giới thiệu phần mềm NetAnim

V. Kết Quả Và Nhận Xét:

1.Kết quả từ ns-3 và đánh giá

2.Kết quả từ NetAnim

3.Nhận xét:

Phụ chú: Tài liệu tham khảo:

BÁO CÁO MÔ PHỎNG VÀ NHẬN XÉT KẾT QUẢ HÀNG ĐỢI

I.Tổng quan về hàng đợi:

1. Hàng đợi trong một hệ thống dịch vụ:

Hàng đợi là một mô hình toán học của hệ thống dịch vụ. Nó được định nghĩa bởi:

1: quá trình đến

2: quá trình dịch vụ.

3: các yếu tố cấu trúc.

Yếu tố cấu trúc ở đây mang nghĩa là kích thước hàng chờ, quy luật hàng đợi (chính sách dịch vụ) và số lượng máy chủ. Giả định rằng các máy chủ có thể hoán đổi cho nhau và hoàn toàn có thể được truy cập, trừ khi được chỉ định. Hơn nữa, chúng ta luôn giả định rằng các hệ thống dịch vụ được coi là "kiệm việc", "một lần đến, một lần phục vụ" [1].

- "Kiệm việc" có nghĩa là không có yêu cầu nào được tạo ra hay bị phá hủy trong hệ thống, tức là một khách hàng vào hệ thống cuối cùng sẽ hoàn thành dịch vụ của mình với xác suất 1 và ngược lại, bất kỳ máy chủ nào không hoạt động sẽ ngay lập tức chọn một máy chủ khác khách hàng từ hàng chờ, nếu có ít nhất một [1].

- "Một lần đến, một lần phục vụ" có nghĩa là trường hợp hai hoặc nhiều khách hàng đến cùng lúc hay chấm dứt các dịch vụ tương ứng của họ cùng lúc, sẽ bị loại trừ.

Sự tiến hóa của hàng đợi được mô tả bằng các quy trình ngẫu nhiên đại diện cho số lượng công việc chưa hoàn thành trong hàng đợi tại một thời điểm nhất định, U (t), hoặc số lượng khách hàng trong hệ thống, Q (t), thời gian chờ đợi của một khách hàng đến được chấp nhận vào hệ thống tại thời điểm t, W (t), số lượng máy chủ tham gia vào dịch vụ, M (t), và những người khác tương tự [1].

Một ký hiệu để biểu thị các đặc điểm chính của hệ thống xếp hàng là được định nghĩa bởi David G. Kendall. Nó bao gồm năm bộ A/B/m/K/N.

- Các chữ cái A và B là viết tắt của từ viết tắt, biểu thị quá trình đến và phục vụ tương ứng. Các từ viết tắt được sử dụng là:

+ M cho quá trình Poisson, tức là, thời gian đến hoặc dịch vụ theo cấp số nhân âm, được phân phối giống hệt nhau và độc lập với một khác

+ D là viết tắt của định nghĩa

+ G (đôi khi GI) là một quá trình đổi mới chung

+ MMPP là Markov Modulated Poisson Process, tức là quy trình Poisson có tỷ lệ đến trung bình là chức năng của trạng thái của quá trình Markov trạng thái rời rạc, hữu hạn

…

- Đối với ba trường cuối cùng của ký hiệu:

+ m là một số nguyên ≥ 1 mà biểu thị số lượng máy chủ

+ K là kích thước của hàng chờ, tức là số lượng khách hàng tối đa có thể đáp ứng được trong hệ thống đang chờ dịch vụ

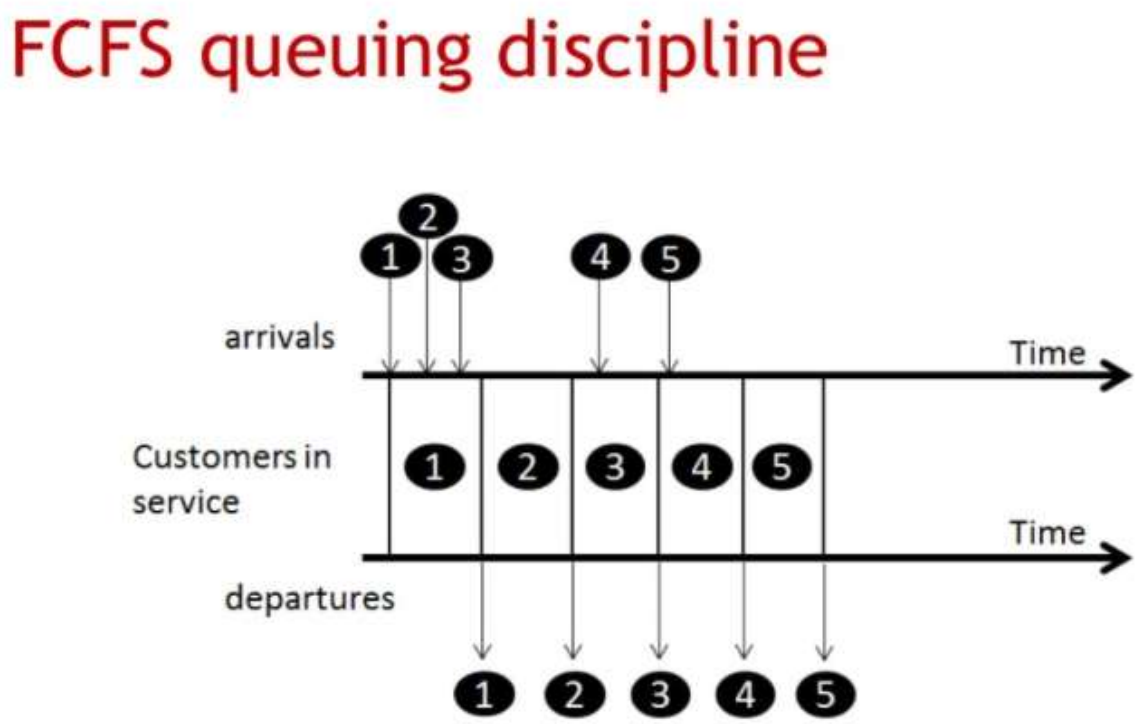
+ N biểu thị quy mô dân số khách hàng đưa ra yêu cầu dịch vụ cho hàng đợi

Một trong ba số nguyên này có thể là vô hạn. Nếu như chúng là vô hạn, các tham số K và N được bỏ qua trong ký hiệu. Ví dụ: M/M/1 biểu thị một máy chủ duy nhất, hệ thống xếp hàng kích thước vô hạn có lượt đến theo sau một quy trình Poisson và thời gian phục vụ theo cấp số nhân i.i.d. biến ngẫu nhiên. Mô hình này chỉ có thể trì hoãn các yêu cầu dịch vụ của khách hàng, khi không có yêu cầu nào bị từ chối vì thiếu chỗ trong hàng đợi [1].

2. Hàng đợi một máy chủ:

Mô hình chung tương ứng được ghi nhận là hàng đợi G/G/1 trong ký hiệu của Kendall. Trừ khi có quy định khác, chúng ta giả sử chính sách lập lịch của hàng đợi là ai đến trước, phục vụ trước (FCFS) và máy chủ đang "kiệm việc", tức là nó không thể ở chế độ nhàn rỗi nếu có khách hàng được phục vụ và không có khách hàng nào rời khỏi hàng đợi cho đến khi hoàn toàn nhận được số lượng dịch vụ mà nó yêu cầu. Đối với

hàng chờ, chúng ta biểu thị kích thước của nó bằng K: tối đa khách hàng K + 1 có thể được lưu trữ trong hệ thống xếp hàng, một khách hàng đang được phục vụ, những người khác đang chờ dịch vụ. K = ∞ là trường hợp đặc biệt (hàng đợi vô hạn). Cho một hữu hạn K có khả năng một khách hàng đến thấy hàng đợi đã đầy. Trong đó trường hợp, nó bị mất trong hàng đợi, tức là, yêu cầu dịch vụ của nó bị từ chối và biến mất [1].



Hình 1: FCFS queuing [5].

3. Hàng đợi nhiều máy chủ:

Không có giải pháp chung đơn giản nào cho hệ thống xếp hàng aG/G/m/K. Nhiều trường hợp đặc biệt có thể được phân tích. Trong số đó, M/M/m/K (cả với hữu hạn và vô hạn K), M/G/m/0, G/M/m, và nhiều trường hợp hàng đợi máy chủ vô hạn. Thay vì trình bày đầy đủ, chúng tôi hướng đến việc tìm hiểu tác động của việc có nhiều máy chủ đối với lưu lượng truy cập các mô hình [1].

Các hàng đợi một máy chủ mô hình hóa những hệ thống trong đó toàn bộ khả năng phục vụ được cấp cho khách hàng, miễn là hệ thống cần nó để hoàn thành nhu cầu dịch vụ. Chúng ta phạm qui luật First-come, first served (FCFS. Trong nhiều trường

hợp, không thể chia nhỏ lượng phục vụ. Bất cứ lúc nào máy chủ không hoạt động (chỉ khi không có khách hàng chờ đợi máy chủ "kiệm việc" [1]) hoặc nó được giao hoàn toàn cho một số khách hàng. Ví dụ, dung lượng của liên kết đầu ra của thẻ giao diện của bộ định tuyến IP được cho là toàn bộ gói head-of-the-line đang đợi trong bộ đệm giao diện. Trong khi gói đó đang tham gia vào dòng đầu ra, được gửi ở tốc độ liên kết, không gói nào khác có thể sử dụng cùng một liên kết (cùng chiều).

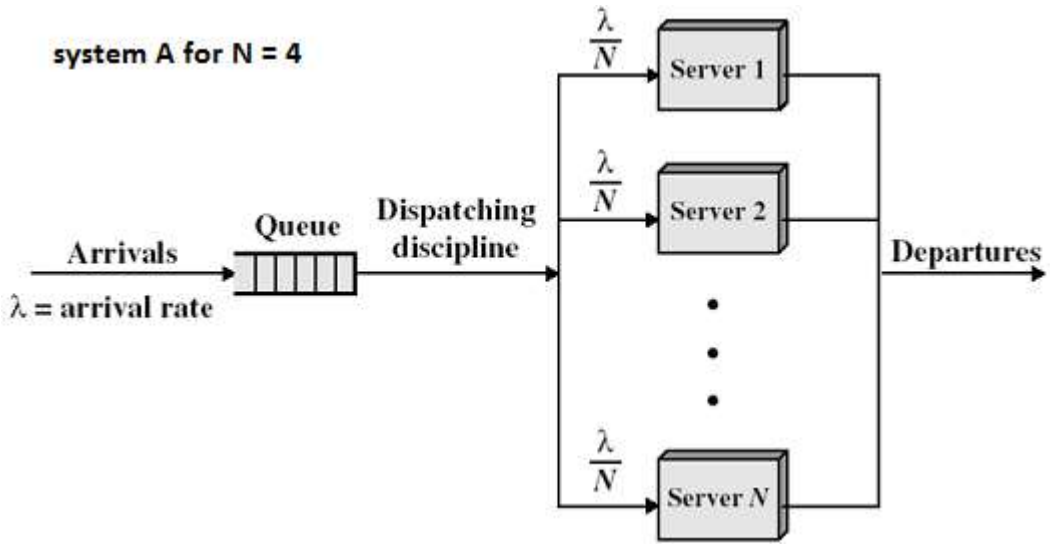
Điều này chắc chắn khác với tình huống một số thiết bị đầu cuối của người dùng chia sẻ dung lượng của một trạm cơ sở di động. Trong trường hợp cuối cùng này, việc truyền dữ liệu đồng thời có thể diễn ra trong hướng downlink, giữa các trạm gốc và các thiết bị đầu cuối của người dùng... Sau đó nhiều hơn một gói tin cùng lúc có thể được gửi qua kênh vô tuyến. Không thể mô hình hóa dung lượng kênh vô tuyến của trạm gốc như một máy chủ duy nhất. Trên thực tế, nó có thể phân phát song song một số gói nhất định [1].

Các ví dụ trên chỉ ra rằng những hệ thống có thể được cấu trúc để phục vụ nhiều khách hàng cùng lúc phải được mô hình hóa thành nhiều máy chủ hàng đợi. Một số đặc điểm cấu trúc bây giờ nên được chỉ định:

+ Khả năng thay thế cho nhau: Các máy chủ có thể tương đương, tức là mỗi máy chủ đều cung cấp cùng một loại dịch vụ, hoặc chúng có thể có các khả năng khác nhau.

+ Khả năng tiếp cận: Khách hàng chỉ có thể truy cập một số máy chủ, tùy thuộc vào lớp hoặc nhu cầu dịch vụ cụ thể của họ.

+ Khả năng phân biệt: Khách hàng có thể chọn một trong các máy chủ có sẵn với xác suất bằng nhau hoặc tham gia một máy chủ cụ thể theo một số tiêu chí. Điều này ngụ ý rằng các máy chủ cỏ thể được gắn nhãn theo một số cách riêng biệt [1].



Hình 2: Multi-server queue model [8].

II. Hàng đợi ưu tiên:

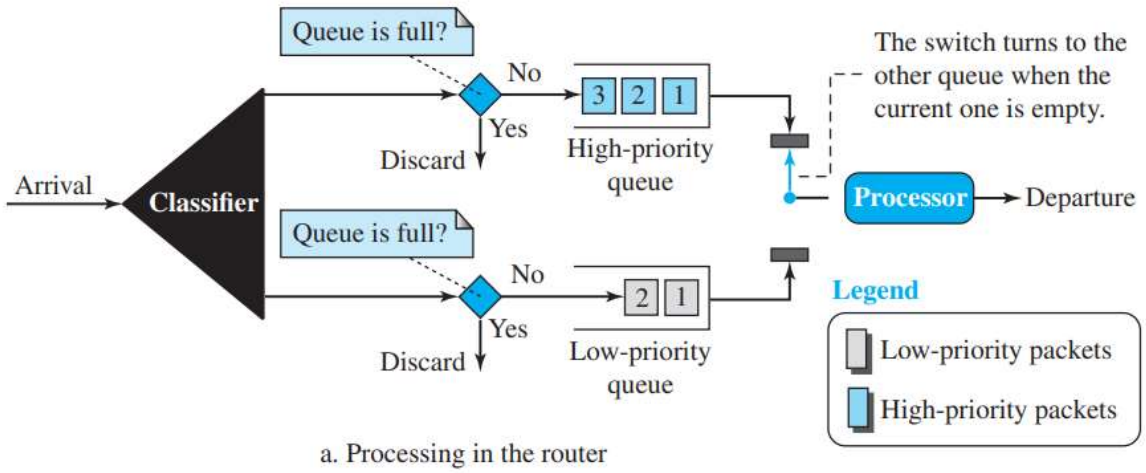
1.Nguyên Tắc

Sự delay trong hàng đợi thường làm giảm chất lượng dịch vụ của mạng và việc sử dụng hàng đợi ưu tiên đã giải quyết được vấn đề này [2]. Trong hàng đợi ưu tiên, trước tiên các gói được gán cho một lớp ưu tiên. Mỗi ưu tiên lớp có hàng đợi riêng của nó. Các gói trong hàng đợi có mức độ ưu tiên cao nhất được xử lý trước. Các gói trong hàng đợi có mức độ ưu tiên thấp nhất được xử lý sau cùng [2]. Các gói được phân loại dựa trên các tiêu chuẩn phân loại của người sử dụng, và được đặt ở một trong số các hàng đợi đầu ra với các độ ưu tiên: độ ưu tiên cao, trung bình, bình thường (không được ưu tiên), ưu tiên thấp.

Ưu tiên trước: Nếu khách hàng có mức độ ưu tiên j đang được sử dụng dịch vụ và khách hàng có mức độ ưu tiên i <j tham gia vào hệ thống, dịch vụ sẽ bị gián đoạn ngay lập tức và khách hàng mới, mức độ ưu tiên cao hơn được đưa vào sử dụng, giải phóng khách hàng đang được được phục vụ.

Ưu tiên trước – tiếp tục: Khi máy chủ hoạt động trở lại và khách hàng bị gián đoạn có quyền lấy nó, theo các quy tắc ưu tiên, dịch vụ bắt đầu ngay từ nơi nó đã bị dừng trong lần gián đoạn trước.

Ưu tiên trước – lặp lại: Bất cứ khi nào nối lại dịch vụ sau khi bị gián đoạn, khách hàng sẽ bắt đầu lại từ đầu; do đó công việc được thực hiện trong thời gian phục vụ từng phần trước đó bị mất

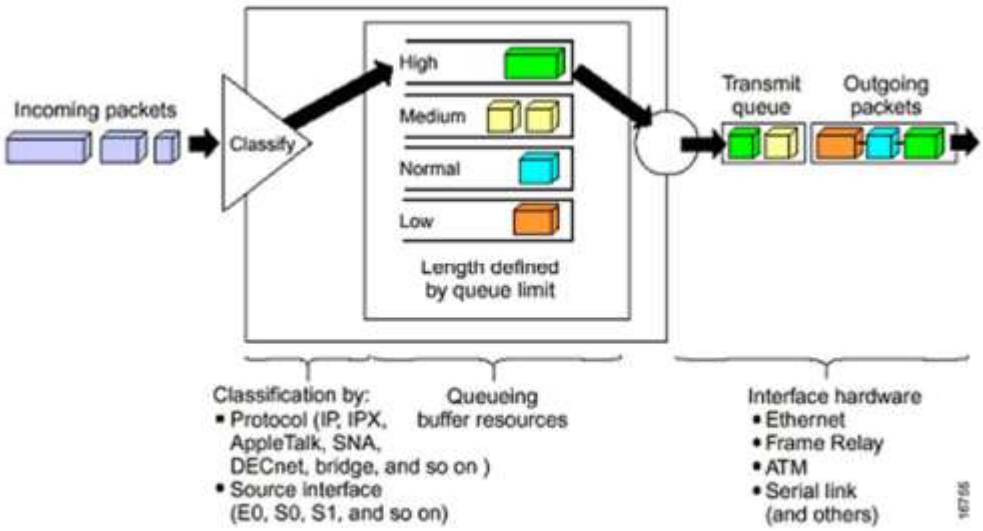


Hình 3: Mô hình xử lý trong bộ định tuyến (router) [2].

2. Cơ chế:

- Các gói được phân loại như thế nào trong kĩ thuật PQ
- Các gói được phân loại theo:

  - Loại giao thức hoặc giao thức con
  - Giao diện đầu vào
  - Kích thước các gói tin
  - Các Fragment
  - Danh sách truy nhập



Hình 3.4 : Cơ chế làm việc của PQ

- Sau đó cần các cơ chế sau:

- Một bộ phân loại có nhiệm vụ kiểm tra các phần header của gói tin để quyết định xem gói nào nên được đưa vào hàng đợi nào. Bộ lập lịch có nhiệm vụ làm rỗng các hàng đợi, bắt đầu bằng hàng đợi có độ ưu tiên cao nhất sau đó đến các hàng đợi trung bình, tiếp theo là các hàng đợi trung bình, cuối cùng là hàng đợi có độ ưu tiên thấp. Các hàng đợi được phục vụ cho tới khi nào không còn gói nào trong chúng thì lúc đó bộ lập lịch mới chuyển sang hàng đợi khác.

3.Các công thức

+ Công thức hàng đợi ưu tiên:

Average queue length:

$$E[N_1] = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu - \lambda}$$

Average system time:

$$E[S_1] = \frac{E[N]}{E[X]} = \frac{1}{\mu - \lambda}$$

Theorem (Total number of customers: Average queue length)

$$E[N_1] = \frac{\lambda_1 + \lambda_2}{\mu - \lambda_1 + \lambda_2}$$

Theorem (For Type 2: Average system time)

$$E[N_2] = \frac{\lambda_1 + \lambda_2}{\mu - \lambda_1 + \lambda_2} - E[N_1] = \frac{\rho_2}{(1-\rho_1)(1-\rho_1-\rho_2)}$$

$$E[S_2] = \frac{E[L_2]}{\lambda_2} = \frac{1/\mu}{(1-\rho_1)(1-\rho_1-\rho_2)}$$

+ Công thức hàng đợi:

$$E[N] = \lambda E[S]$$

Theorem (Average waitting time in Queue)

$$E[W] = E[S] - \frac{1}{\mu} = \frac{1}{\mu - \lambda} - \frac{1}{\mu}$$

Theorem (Number of frames in system)

$$N_Q = \lambda E[W] = \frac{\lambda}{\mu - \lambda} - \frac{\lambda}{\mu}$$

III.Hàng đợi ưu tiên không có trước:

1. Nguyên tắc:

- Ưu tiên không có trước: Sau khi khách hàng được cấp phát máy chủ, máy chủ sẽ ở lại với khách hàng đó cho đến khi nhu cầu của họ được đáp ứng đầy đủ, bất kể mức độ ưu tiên của khách hàng đến sau khi bắt đầu dịch vụ hiện tại; nói cách khác, dịch vụ không thể bị gián đoạn.

2. Các công thức:

- Một số công thức tính toán:

$$E[S_1] = \frac{1}{\mu}E[L_1] + \frac{1}{\mu} + \rho_2\frac{1}{\mu}$$
$$E[L_1] = \lambda_1 E[S_1]$$

IV. Mô phỏng:

1. Giới thiệu ns3:

a) Lớp Kiểm Soát Lưu Lượng (Traffic Control Layer):

- Lớp Kiểm Soát Lưu Lượng nhằm đưa một tính năng tương đương với Kiểm soát Lưu lượng của Linux vào ns-3. Lớp Kiểm soát lưu lượng nằm giữa thiết bị mạng (L2) và bất kỳ giao thức mạng nào (ví dụ: IP). Nó chịu trách nhiệm xử lý các gói tin và thực hiện các hành động trên chúng: lập lịch, bỏ, đánh dấu, khống chế dung lượng, ... Lớp Điều khiển lưu lượng chặn cả các gói đi theo hướng đi xuống từ lớp mạng tới thiết bị mạng và các gói đến theo hướng ngược lại. Hiện tại, chỉ các gói gửi đi mới được xử lý bởi lớp Điều khiển lưu lượng. Đặc biệt, các gói gửi đi được xếp vào hàng trong một quy tắc xếp hàng, có thể thực hiện nhiều hành động trên chúng.

b) Nguyên tắc Hàng đợi IP (theo NS-3):

- Hay còn gọi là lập lịch mạng hay thuật toán hàng đợi là 1 " trọng tài " trên một nút trong mạng truyền thông chuyển mạch gói. Nó quản lý chuỗi các gói mạng trong hàng đợi truyền và nhận của ngăn xếp giao thức và bộ điều khiển giao diện mạng. Các gói tin được nhận bởi lớp Điều khiển lưu lượng (Traffic control layer) để truyền tới một thiết bị mạng , các gói tin đó có thể được chuyển tới một nguyên tắc hàng đợi (queue disc) để thực hiện lập lịch và khống chế lưu lượng (policing) . Một thiết bị mạng có thể chỉ có một (root) nguyên tắc hàng đợi được cài đặt trên đó. Việc cài đặt nguyên tắc hàng đợi trên thiết bị mạng là không bắt buộc. Nếu một netdevice không có đĩa hàng đợi được cài đặt trên đó, thì lớp điều khiển lưu lượng sẽ gửi các gói trực tiếp đến thiết bị. Đối với Linux-NS3, nguyên tắc hàng đợi có thể biểu diễn 1 hàng đợi đơn giản hoặc có thể là 1 cấu trúc phức tạp, nhưng bắt buộc phải có những thành phần căn bản sau:

. hàng đợi (queue), lưu trữ các gói đang chờ truyền.

. các lớp (classes), cho phép định nghĩa các phương pháp xử lý khác nhau cho các phân đoạn giao thông khác nhau.

. các bộ lọc (filters), xác định hàng đợi hoặc lớp mà một gói được sử dụng.

*** Chú ý: Class này ko giống class trong C++ lập trình hướng đối tượng, vì linux sử dụng thuật ngữ classful queue disc và classless queue disc**

Mọi Nguyên Tắc thu thập số liệu thống kê về tổng số gói / byte nhận được từ các lớp trên (trong trường hợp root) hoặc từ " cha mẹ " (trong trường hợp nguyên tắc hàng đợi con), thêm vào hàng đợi (enqueued), xóa trong hàng đợi (dequeued), xếp lại (requeued), bỏ (dropped), bỏ trước thêm, bỏ sau xóa, được đánh dấu và được lưu trữ trong các nguyên tắc này và được gửi đến thiết bị hoặc đến bộ nguyên tắc mẹ. Lưu ý rằng các gói có thể bị xóa hoặc sắp lại, tức là được cơ sở hạ tầng kiểm soát lưu lượng giữ lại, nếu thiết bị chưa sẵn sàng nhận chúng. Các gói được xếp lại không phải là một phần của nguyên tắc xếp hàng. Các đặc điểm nhận dạng sau đây được lưu giữ:

- dropped = dropped before enqueue + dropped after dequeuer
- received = dropped before enqueue + enqueued
- queued = enqueued – dequeued
- sent = dequeued - dropped after dequeue (- 1 if there is a requeued packet)

c) Priority queue NS3:

PrioQueueDisc thực hiện một chính sách ưu tiên nghiêm ngặt, trong đó các gói chỉ được xóa từ một băng tần nếu các băng tần có mức ưu tiên cao hơn đều trống. PrioQueueDisc là một classful queue disc và có thể có số lượng băng tần tùy ý, mỗi dải được xử lý bởi bất kỳ loại queue disc nào. Dung lượng của PrioQueueDisc không giới hạn; các gói chỉ có thể được loại bỏ bởi các queue disc con (có thể có dung lượng hạn chế). Nếu không có bộ lọc gói nào được cài đặt hoặc có thể phân loại gói, thì gói đó sẽ được xếp vào hàng ưu tiên dựa trên mức ưu tiên của nó, và nó được sử dụng như một chỉ mục trong một mảng gọi là priomap.

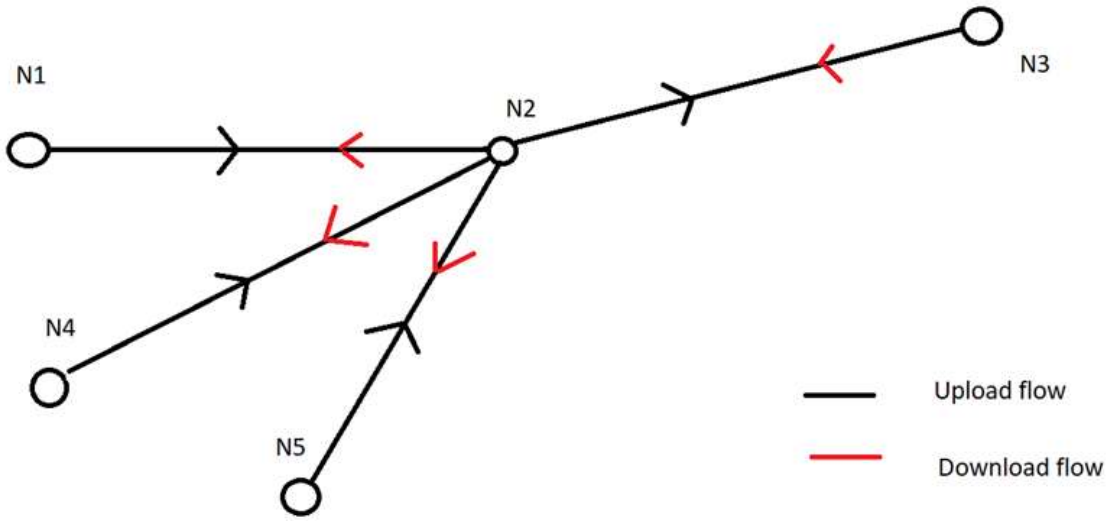
Nếu người dùng không thêm lớp queue disc nào trước khi queue disc được khởi tạo, ba đĩa hàng đợi con kiểu FifoQueueDisc sẽ tự động được thêm vào. Cần lưu ý rằng PrioQueueDisc cần ít nhất hai đĩa xếp hàng con.

d) FIFO (first in first out) - NS3:

Cấu trúc dữ liệu hàng đợi tuân theo phương pháp First-In-First-Out, tức là dữ liệu được nhập vào đầu tiên sẽ được truy cập đầu tiên.

Đối vs NS-3 - linux, Các gói được xếp trong hàng đợi nội bộ duy nhất, được triển khai dưới dạng hàng đợi DropTail. Dung lượng hàng đợi có thể được chỉ định dưới dạng gói hoặc byte, tùy thuộc vào giá trị của thuộc tính.

2. Kiến trúc mạng mô phỏng:



Mô hình mạng mô phỏng

3. Phân tích code mô phỏng kiến trúc (file queue-discs-benchmark.cc)

Các bước:

+) Tạo các node.

```
NodeContainer n1, n2, n3, n4 , n5;
  n1.Create (1);
  n2.Create (1);
  n3.Create (1);
  n4.Create (1);
  n5.Create (1);
```

+) Tạo và định cấu hình liên kết truy cập và liên kết tắc nghẽn.

```
PointToPointHelper accessLink;
         accessLink.SetDeviceAttribute ("DataRate", StringValue ("100Mbps"));
         accessLink.SetChannelAttribute ("Delay", StringValue ("0.1ms"));
         accessLink.SetQueue ("ns3::DropTailQueue", "MaxSize", StringValue
         ("100p"));

         PointToPointHelper bottleneckLink;
         bottleneckLink.SetDeviceAttribute ("DataRate", StringValue (bandwidth));
         bottleneckLink.SetChannelAttribute ("Delay", StringValue (delay));
        bottleneckLink.SetQueue   ("ns3::DropTailQueue",   "MaxSize",   StringValue
      (std::to_string (netdevicesQueueSize) + "p"));
```

Một vài thông số đầu vào như: Datarate, delay, maxsize.

+) Truy cập cấu hình kiểm soát lưu lượng liên kết.

```
         TrafficControlHelper tchPfifoFastAccess;
           tchPfifoFastAccess.SetRootQueueDisc ("ns3::PfifoFastQueueDisc", "MaxSize",
         StringValue ("1000p"));
```

+) Cấu hình kiểm soát lưu lượng liên kết nút cổ chai

```
         TrafficControlHelper tchBottleneck;
           TrafficControlHelper tchaccess;


        else if (queueDiscType.compare ("prio") == 0)
    {
      uint16_t handle = tchBottleneck.SetRootQueueDisc ("ns3::PrioQueueDisc",
"Priomap",
                                                        StringValue ("1 2 2
          2 1 2 0 0 1 1 1 1 1 1 1 1"));
      TrafficControlHelper::ClassIdList cid = tchBottleneck.AddQueueDiscClasses
           (handle,   2, "ns3::QueueDiscClass");
      tchBottleneck.AddChildQueueDisc (handle, cid[0], "ns3::FifoQueueDisc");
      tchBottleneck.AddChildQueueDisc (handle, cid[1], "ns3::RedQueueDisc");
           }
```

Trong trường hợp là hàng đợi ưu tiên, sử dụng câu lệnh trên để cấu hình các gói tin, cho thấy mức độ ưu tiên trong priomap.

+) Cấu hình dòng chảy, Luồng TCP hai chiều với ping như thử nghiệm flent tcp_bidirectional

```
       uint16_t port = 7;

         ApplicationContainer uploadApp, downloadApp, sourceApps;
```

+) Định cấu hình và cài đặt quy trình tải lên.

```
      Address addUp (InetSocketAddress (Ipv4Address::GetAny (), port));
```

```
  PacketSinkHelper sinkHelperUp ("ns3::TcpSocketFactory", addUp);
  sinkHelperUp.SetAttribute ("Protocol", TypeIdValue
(TcpSocketFactory::GetTypeId ()));
  uploadApp.Add (sinkHelperUp.Install (n3));

  InetSocketAddress socketAddressUp = InetSocketAddress (n3Interface.GetAddress
(0), port);
  OnOffHelper onOffHelperUp ("ns3::TcpSocketFactory", Address ());
  onOffHelperUp.SetAttribute ("Remote", AddressValue (socketAddressUp));
  onOffHelperUp.SetAttribute ("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1]"));
  onOffHelperUp.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));
  onOffHelperUp.SetAttribute ("PacketSize", UintegerValue (flowsPacketsSize));
  onOffHelperUp.SetAttribute ("DataRate", StringValue (flowsDatarate));
  sourceApps.Add (onOffHelperUp.Install (n1));

  InetSocketAddress socketAddressUp1 = InetSocketAddress (n4Interface.GetAddress
(0), port);
  OnOffHelper onOffHelperUp1 ("ns3::TcpSocketFactory", Address ());
  onOffHelperUp1.SetAttribute ("Remote", AddressValue (socketAddressUp1));
  onOffHelperUp1.SetAttribute ("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1]"));
  onOffHelperUp.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));
  onOffHelperUp1.SetAttribute ("PacketSize", UintegerValue (flowsPacketsSize));
  onOffHelperUp1.SetAttribute ("DataRate", StringValue (flowsDatarate));
  sourceApps.Add (onOffHelperUp1.Install (n4));

  InetSocketAddress socketAddressUp2 = InetSocketAddress (n5Interface.GetAddress
(0), port);
  OnOffHelper onOffHelperUp2 ("ns3::TcpSocketFactory", Address ());
  onOffHelperUp2.SetAttribute ("Remote", AddressValue (socketAddressUp2));
  onOffHelperUp2.SetAttribute ("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1]"));
  onOffHelperUp.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));
  onOffHelperUp2.SetAttribute ("PacketSize", UintegerValue (flowsPacketsSize));
  onOffHelperUp2.SetAttribute ("DataRate", StringValue (flowsDatarate));
  sourceApps.Add (onOffHelperUp2.Install (n5));

  port = 8;
```

N3 sẽ được tải lên giao thức để trở thành tram thu, n1 n4 và n5 là các trạm phát được tải lên các thông số đầu vào.

+) Định cấu hình và cài đặt quy trình tải xuống

```
Address addDown (InetSocketAddress (Ipv4Address::GetAny (), port));
  PacketSinkHelper sinkHelperDown ("ns3::TcpSocketFactory", addDown);
  sinkHelperDown.SetAttribute ("Protocol", TypeIdValue
(TcpSocketFactory::GetTypeId ()));
  downloadApp.Add (sinkHelperDown.Install (n1));

  Address addDown1 (InetSocketAddress (Ipv4Address::GetAny (), port));
  PacketSinkHelper sinkHelperDown1 ("ns3::TcpSocketFactory", addDown1);
```

```
  sinkHelperDown1.SetAttribute ("Protocol", TypeIdValue
(TcpSocketFactory::GetTypeId ()));
  downloadApp.Add (sinkHelperDown1.Install (n4));

  Address addDown2 (InetSocketAddress (Ipv4Address::GetAny (), port));
  PacketSinkHelper sinkHelperDown2 ("ns3::TcpSocketFactory", addDown2);
  sinkHelperDown2.SetAttribute ("Protocol", TypeIdValue
(TcpSocketFactory::GetTypeId ()));
  downloadApp.Add (sinkHelperDown2.Install (n5));

  InetSocketAddress socketAddressDown = InetSocketAddress
(n1Interface.GetAddress (0), port);
  OnOffHelper onOffHelperDown ("ns3::TcpSocketFactory", Address ());
  onOffHelperDown.SetAttribute ("Remote", AddressValue (socketAddressDown));
  onOffHelperDown.SetAttribute ("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1]"));
  onOffHelperDown.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));
  onOffHelperDown.SetAttribute ("PacketSize", UintegerValue (flowsPacketsSize));
  onOffHelperDown.SetAttribute ("DataRate", StringValue (flowsDatarate));
sourceApps.Add (onOffHelperDown.Install (n3));
```

+) Định cấu hình và cài đặt ping.

```
V4PingHelper ping = V4PingHelper (n3Interface.GetAddress (0));
  ping.Install (n1);
  ping.Install (n4);
  ping.Install (n5);
  Config::Connect ("/NodeList/*/ApplicationList/*/$ns3::V4Ping/Rtt", MakeCallback
   (&PingRtt));

  uploadApp.Start (Seconds (0));
  uploadApp.Stop (Seconds (stopTime));
  downloadApp.Start (Seconds (0));
  downloadApp.Stop (Seconds (stopTime));

  sourceApps.Start (Seconds (0 + 0.1));
  sourceApps.Stop (Seconds (stopTime - 0.1));

  Ptr<OutputStreamWrapper> uploadGoodputStream = ascii.CreateFileStream
   (queueDiscType + "-upGoodput.txt");
  Simulator::Schedule (Seconds (samplingPeriod), &GoodputSampling, queueDiscType
    + "-upGoodput.txt", uploadApp,
                         uploadGoodputStream, samplingPeriod);
  Ptr<OutputStreamWrapper> downloadGoodputStream = ascii.CreateFileStream
   (queueDiscType + "-downGoodput.txt");
  Simulator::Schedule (Seconds (samplingPeriod), &GoodputSampling, queueDiscType
    + "-downGoodput.txt", downloadApp,
                         downloadGoodputStream, samplingPeriod);
```

+) Giám sát đóng lưu lượng

```
  Ptr<FlowMonitor> flowMonitor;
  FlowMonitorHelper flowHelper;
  flowMonitor = flowHelper.InstallAll();

  AnimationInterface anim("queue-discs-benchmark.xml");
  Simulator::Stop (Seconds (stopTime));
  Simulator::Run ();
```

```
  flowMonitor->SerializeToXmlFile(queueDiscType + "-flowMonitor.xml", true,
true);

  Simulator::Destroy ();
  return 0;
```

Định tuyến các dòng data, chạy mô phỏng và hiện thị kết quả.

4.Giới thiệu phần mềm NetAnim

- NetAnim là một chương trình độc lập sử dụng các tệp theo dõi tùy chỉnh được tạo bởi giao diện hoạt ảnh để hiển thị mô phỏng bằng đồ họa. NetAnim dựa trên bộ công cụ GUI Qt4 đa nền tảng.

NetAnim GUI cung cấp các nút phát, tạm dừng và ghi âm. Chơi và tạm dừng bắt đầu và dừng mô phỏng. Nút ghi bắt đầu một loạt ảnh chụp màn hình của trình tạo hoạt ảnh, được ghi vào thư mục chứa tệp theo dõi đã được chạy. Hai thanh trượt cũng tồn tại. Thanh trượt trên cùng cung cấp chức năng "tìm kiếm", cho phép người dùng bỏ qua bất kỳ thời điểm nào trong mô phỏng. Thanh trượt dưới cùng thay đổi mức độ chi tiết của bước thời gian cho hoạt ảnh. Cuối cùng, có một nút thoát để dừng mô phỏng và thoát khỏi trình hoạt họa.

Một số tính năng nổi bật:

- Tạo hiệu ứng các gói tin qua các liên kết có dây và liên kết không dây (Hỗ trợ giới hạn cho các dấu vết LTE. Không hỗ trợ Ipv6)
- Dòng thời gian gói với bộ lọc regex trên siêu dữ liệu gói.
- Thống kê vị trí nút với biểu đồ quỹ đạo nút (đường đi của nút di động).
- In dữ liệu meta gói ngắn gọn trên các gói
- Sử dụng các biểu tượng tùy chỉnh cho các nút
- Phân tích cú pháp các tệp XML theo dõi luồng và hiển thị số liệu thống kê cho từng luồng.
- Hiển thị thông tin IP và MAC, bao gồm IP và MAC ngang hàng cho các liên kết điểm-điểm.

• Hiển thị các bộ đếm có giá trị kép hoặc uint32 so với thời gian cho nhiều nút trong biểu đồ hoặc bảng.

• Bước qua mô phỏng một sự kiện tại một thời điểm và tạm dừng mô phỏng tại một thời điểm nhất định

• In bảng định tuyến tại các nút tại các thời điểm khác nhau

*) Một số câu lệnh và thư viện cần thêm để thực hiện mô phỏng

- #include "ns3/netanim-module.h" (câu lệnh cài đặt thư viện animation)
- AnimationInterface anim("queue-discs-benchmark.xml"); (câu lệnh xuất file xml)

V. Kết Quả Và Nhận Xét:

1. Kết quả từ ns-3 và đánh giá

Từ các thông số đầu vào:

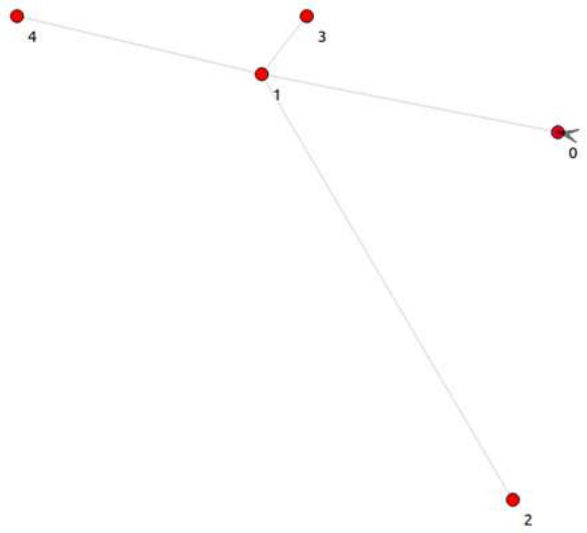
- bandwith = 10Mbps;
- Delay = 5ms;
- Queue discs size = 1000;
- Net device queue size = 10;
- Datarate = 100Mbps;
- Packetsize = 1000;

```
/NodeList/0/ApplicationList/2/$ns3::V4Ping/Rtt=10 ms
/NodeList/0/ApplicationList/2/$ns3::V4Ping/Rtt=34 ms
/NodeList/0/ApplicationList/2/$ns3::V4Ping/Rtt=33 ms
/NodeList/0/ApplicationList/2/$ns3::V4Ping/Rtt=37 ms
/NodeList/0/ApplicationList/2/$ns3::V4Ping/Rtt=39 ms
/NodeList/0/ApplicationList/2/$ns3::V4Ping/Rtt=31 ms
/NodeList/0/ApplicationList/2/$ns3::V4Ping/Rtt=36 ms
/NodeList/0/ApplicationList/2/$ns3::V4Ping/Rtt=39 ms
/NodeList/0/ApplicationList/2/$ns3::V4Ping/Rtt=33 ms
/NodeList/0/ApplicationList/2/$ns3::V4Ping/Rtt=35 ms
/NodeList/0/ApplicationList/2/$ns3::V4Ping/Rtt=32 ms
/NodeList/0/ApplicationList/2/$ns3::V4Ping/Rtt=35 ms
/NodeList/0/ApplicationList/2/$ns3::V4Ping/Rtt=36 ms
/NodeList/0/ApplicationList/2/$ns3::V4Ping/Rtt=32 ms
/NodeList/0/ApplicationList/2/$ns3::V4Ping/Rtt=38 ms
/NodeList/0/ApplicationList/2/$ns3::V4Ping/Rtt=38 ms
/NodeList/0/ApplicationList/2/$ns3::V4Ping/Rtt=29 ms
```

(Một đoạn quá trình ping Rtt của n1)

2. Kết quả từ NetAnim

- Cấu trúc mạng NetAnim:



(nhóm 2 đã chuẩn bị video báo cáo mô phỏng có đính kèm file với báo cáo word)

3. Nhận xét:

Mô phỏng đã hoạt động đúng với ý tưởng ban đầu. Kết quả mô phỏng đúng như lý thuyết, gói tin từ 3 node phát đã được đưa tới node nhận (n3) thành công và được gửi về

Về mặt ưu điểm của hệ thống mô phỏng, hệ thống đã mô phỏng tương đối chính xác về việc các gói tin được đưa đến các node và trở về. Sự đo đạc Round –trip time gần với kết quả đã tính toán. Nhưng về mặt nhược điểm của mô phỏng, hệ thống mô phỏng đã mô phỏng quá ít node phát, thậm chí kể cả khi đưa ra những thời điểm khắc nghiệt nhất nhưng dường như các gói tin vẫn đi cùng nhau điều này gây khó khăn trong việc phân tích gói tin ưu tiên.

Đối với ưu và nhược điểm của mô hình hàng đợi, từ prio queue, chúng ta có thể xây dựng được nhiều hàng đợi phục vụ cho các mục đích khác nhau ví dụ như: stric prio queue, weight fair queue, weighted round robin. Nhưng nếu có nhiều gói tin có prio cao, các gói tin prio thấp sẽ bị gửi đi rất trễ và gây đến sự khó chịu cho người dùng.

Phụ chú: Tài liệu tham khảo:

[1]: Network Traffic Engineering: Models and Applications – Andrea Baiocchi, Wiley; 1st edition (September 1, 2020)

[2]: Data Communication and Networking, 5th Edition, Behrouz A. Forouzan.

[3]: https://www.nsnam.org/doxygen/classns3_1_1_prio_queue_disc.html

[4]: https://www.nsnam.org/docs/models/html/prio.html

[5]: Priority Queuing – Shahad H. Zwayen

[6]: https://www.nsnam.org/wiki/NetAnim_3.108

[7]: Differentiated Service Queuing Disciplines in NS-3 - Robert Chang, Mahdi Rahimi, and Vahab Pournaghshband, Advanced Network and Security Research LaboratoryCalifornia State University, Northridge, California, USA

[8]: Analysis of Single Queue – Single Server and Single Queue - Multi Server Systems using Simulation: Case Studies of ECO and First Banks - Adamu Muhammad, January-2015

[9]:https://tailieu.vn/docview/tailieu/2011/20110829/kimku1/hang_doi_trong_mang_ip_3_1_5556.pdf?rand=390556