ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN TỬ - VIỄN THÔNG

BÁO CÁO
LẬP TRÌNH MẠNG

ĐỀ TÀI:
ROUTING PRESSURE – CHỈ SỐ ĐÁNH GIÁ THUẬT TOÁN ĐỊNH TUYẾN DỰA TRÊN KÊNH VÀ LƯU LƯỢNG DÙNG NS-3

Sinh viên thực hiện: Nguyễn Hữu Duy (106220215)
Hà Lê Minh Hiếu (106220216)
Nguyễn Việt Hoàng (106220217)
Lớp học phần: 22.44
GVHD: TS. Nguyễn Văn Hiếu

TP. Đà Nẵng, 2025

BẢNG PHÂN CÔNG CÔNG VIỆC TRONG NHÓM

Họ và tên sinh viên	Số thẻ sinh viên	Lớp sinh hoạt	Phân công nhiệm vụ	%
Nguyễn Hữu Duy	106220215	22KTMT1	Phân tích mô hình, các trường hợp mô phỏng, viết báo cáo phần giới thiệu, phương pháp, công cụ sử dụng	40%
Hà Lê Minh Hiếu	106220216		Tìm hiểu xây dựng code mô phỏng trên ns3, phân tích các metrics, viết báo cáo phần phân tích kết quả	40%
Nguyễn Việt Hoàng	106220217		Tìm hiểu đề tài, hỗ trợ viết báo cáo	20%
Tổng				100%

MỤC LỤC

1. Giới thiệu

1.1. Vấn đề nghiên cứu

Trong nghiên cứu về hiệu năng thuật toán định tuyến cho Network-on-Chip (NoC) với topo mesh 2D, "độ thích nghi" (degree of adaptiveness) – tức số đường đi ngắn nhất giữa mỗi cặp nguồn–đích – thường được dùng làm thước đo, nhưng bộc lộ nhiều hạn chế: các thuật toán có cùng độ thích nghi vẫn cho hiệu năng rất khác nhau, đôi khi thuật toán "thích nghi hơn" lại có độ trễ cao hơn; đồng thời không giải thích được vì sao, ở đâu và khi nào xảy ra nghẽn, và việc dùng giá trị trung bình trên toàn mạng làm mất nhiều thông tin chi tiết. Để khắc phục, nghiên cứu đề xuất một metric mới là **routing pressure (áp lực định tuyến)**, dựa trên lượng gói tin mà thuật toán dồn lên từng kênh truyền. Với mỗi cặp nguồn–đích, xác định các đường đi hợp lệ theo thuật toán định tuyến, giả sử gói phân bố đều trên các đường này, từ đó tính được áp lực mà mỗi đường đi gây lên từng kênh; tổng các giá trị này tạo thành **channel pressure**, và kênh có channel pressure lớn nhất quyết định routing pressure của thuật toán. Metric này cho phép nhận diện kênh nào sẽ nghẽn trước, tương quan tốt với độ trễ và thông lượng, và có thể dùng để ước lượng tốc độ bơm gói tối đa trước khi mạng bắt đầu tắc nghẽn mà không cần mô phỏng nặng. Trên cơ sở đó, nhiều thuật toán như XY, OE, NF,… được so sánh lại dưới góc nhìn routing pressure.

1.2. Routing pressure

Dựa trên tài liệu gốc, khái niệm **routing pressure** được xây dựng qua hai lớp: **channel pressure** (áp lực trên từng kênh) và **routing pressure của toàn bộ thuật toán**.

Trước hết, với mỗi **cặp giao tiếp** (source–destination) có tốc độ bơm gói PIR_i, giả sử thuật toán định tuyến cho phép p_i đường đi ngắn nhất khác nhau. Lượng gói từ nguồn được giả thiết phân bố đều trên các đường này, nên mỗi đường phải xử lý PIR_i/p_igói. Nếu một kênh nào đó nằm trên $k_i$trong số p_iđường đi, thì tổng phần tải từ cặp giao tiếp đó dồn lên kênh bằng k $\cdot$ PIR_i/p_i. **Channel pressure** của một kênh được định nghĩa là tổng tất cả các "phần tải" như vậy, cộng qua mọi cặp giao tiếp đang tồn tại trong traffic:

$$\rho = \sum_{i=1}^{COMMs} \frac{k}{p_i} . PIR_i$$

Trong đó:

- **COMMs :** tổng số phiên giao tiếp (communications) đang xét trong mạng
- **i :** Chạy từ 1 đến COMMS, tức là đang duyệt từng cặp nguồn–đích một
- $\boldsymbol{p_i}$ **:** số đường đi hợp lệ (thường là các đường đi ngắn nhất) mà thuật toán định tuyến cho phép cho cặp giao tiếp thứ i
- $\boldsymbol{k}$ **:** Với cặp giao tiếp thứ i, k là số đường trong số p_i đường *đi qua đúng cái kênh đang xét*

Khi mọi nút nguồn đều có cùng tốc độ bơm gói PIR, channel pressure của một kênh có thể viết dưới dạng

$$\rho = PIR \, . \sum_{i=1}^{COMMs} \frac{k}{p_i}$$

Routing pressure của một thuật toán định tuyến đối với một traffic cụ thể được định nghĩa là **giá trị channel pressure lớn nhất trong toàn mạng**. Nói cách khác, chỉ cần xét kênh "căng" nhất (bottleneck link); nếu lượng gói dồn lên kênh này vượt quá băng thông vật lý thì nghẽn sẽ xuất hiện tại đó, và nếu kênh này chưa nghẽn thì các kênh khác cũng chưa. Từ đây trở đi ta ký hiệu φ là routing pressure.

Từ định nghĩa trên có thể suy ra mối liên hệ trực tiếp giữa routing pressure và **ngưỡng tốc độ bơm gói tối đa** để mạng không bị nghẽn. Nếu mỗi kênh phục vụ được f_{rate} flit mỗi chu kỳ, mỗi gói có độ dài len flit, điều kiện không nghẽn tại kênh "căng" nhất cho ta bất đẳng thức

$$\varphi \cdot PIR \leq \frac{f_{\text{rate}}}{len} \Rightarrow PIR_{\max} \approx \frac{f_{\text{rate}}}{len \cdot \varphi} .$$

Như vậy, với cùng cấu hình phần cứng (cùng f_{rate}, len), thuật toán nào có ρ nhỏ hơn (tức routing pressure thấp hơn) sẽ cho phép **PIR cực đại cao hơn**, nghĩa là chịu tải tốt hơn trước khi mạng bắt đầu tắc nghẽn.

Có thể tóm lại: routing pressure là một **chỉ số gắn với kênh và có tính đến traffic**, đo mức độ tập trung lưu lượng lên các kênh. Chỉ số này vừa giúp dự đoán vị trí/nguyên nhân nghẽn (kênh có channel pressure lớn nhất), vừa cho phép ước lượng định lượng ngưỡng tải $PIR_{\max}$ của thuật toán định tuyến mà không cần mô phỏng chi tiết.

1.3. Độ trễ trung bình

Phần **độ trễ trung bình** trong bài được định nghĩa như sau:

- Gọi K là **tổng số gói tin đã đến được đích** trong thời gian đo.
- Gọi lat_i là **độ trễ** (tính theo số chu kỳ) của gói thứ i, đo từ lúc gói được chèn vào mạng đến lúc tới đích.

Khi đó, **độ trễ trung bình của gói** được tính bởi:

$$\text{Average packet latency} = \frac{1}{K}\sum_{i=1}^{K} lat_i$$

tức là lấy **trung bình cộng độ trễ của tất cả các gói đã nhận được**.

Trong cùng đoạn, thông lượng được định nghĩa là:

$$\text{throughput} = \frac{\text{total received flits}}{(\text{number of nodes}) \times (\text{total cycles})}$$

2. Phương pháp

2.1. Mô hình Noc (Network on chip)

Trong các hệ thống System-on-Chip (SoC) hiện đại tích hợp hàng chục đến hàng nghìn lõi xử lý và mô-đun ngoại vi, **Network-on-Chip (NoC)** được xem là kiến trúc truyền thông trên chip có khả năng mở rộng tốt hơn nhiều so với bus chia sẻ truyền thống. Thay vì cho tất cả các IP core kết nối chung vào một hoặc vài đường bus, NoC

tổ chức các lõi tính toán, bộ nhớ và bộ điều khiển I/O thành các **node** và liên kết chúng thông qua một **mạng chuyển mạch gói (packet-switched)** gồm các **router** và **liên kết (link)** giống như một mạng máy tính thu nhỏ trên cùng một đế silicon.

Mỗi node thường gắn với một router NoC; các router này kết nối với nhau theo một **topology** nhất định, phổ biến nhất là **mesh 2D**, ngoài ra còn có torus, tree, fat-tree, ring,… Giao tiếp giữa các lõi được thực hiện bằng cách đóng gói dữ liệu thành **packet**, chia nhỏ thành các **flit**, sau đó truyền qua nhiều router trung gian tới đích. Đường đi của gói tin được quyết định bởi **thuật toán định tuyến (routing algorithm)**; hiệu năng của NoC phụ thuộc mạnh vào thuật toán này cũng như cơ chế điều khiển luồng, chính sách ưu tiên, v.v.

Ưu điểm chính của NoC là khả năng **mở rộng băng thông tuyến tính theo số lượng kênh**, hỗ trợ **truyền song song giữa nhiều cặp nguồn–đích**, giảm tắc nghẽn so với bus, đồng thời cho phép thiết kế SoC theo kiểu **mô-đun, dễ tái sử dụng IP core**. Tuy nhiên, NoC cũng đặt ra nhiều thách thức như thiết kế thuật toán định tuyến tránh deadlock, giảm độ trễ và tiêu thụ năng lượng, phân tích và dự đoán tắc nghẽn trên các kênh, cũng như đảm bảo chất lượng dịch vụ (QoS) cho các luồng lưu lượng khác nhau.

2.2. Nhiệm vụ các node

Kịch bản mô phỏng xây dựng một mạng **Network-on-Chip (NoC)** dạng lưới 2D (mesh) trên nền mô phỏng ns-3. Mạng gồm các **node** được bố trí thành ma trận width × height (mặc định 7×7, tức 49 node). Mỗi node đồng thời đóng vai trò **nguồn phát** và **router**: node sẽ **phát gói tin** theo quá trình Poisson (tức thời gian giữa các lần phát tuân theo phân phối mũ âm, tạo lưu lượng ngẫu nhiên) với **tốc độ chèn gói** được cấu hình bởi tham số PIR (packet injection rate). Mỗi gói tin có kích thước cố định (32 byte, tương đương 8 flit) và được truyền trên các liên kết point-to-point có băng thông 16 Gbps, độ trễ 0 ns để mô phỏng mạng trên chip. Khi gửi đi, gói tin được gắn thẻ thời gian để đo **độ trễ**; header của gói mang ID nguồn, ID đích và bộ đếm số hop đã qua.

Hoạt động của node: Mỗi node chạy một ứng dụng **NoC router**. Ứng dụng này định

kỳ lên lịch phát gói mới theo PIR cấu hình. Khi đến thời điểm phát, node chọn ngẫu nhiên một đích trong tập cho trước (phụ thuộc **mẫu lưu lượng** đã chọn) và tạo một gói tin, gắn thẻ thời gian và header rồi truyền vào mạng. Gói tin được chuyển theo kiểu **hop-by-hop** qua các node trung gian: tại mỗi bước, node nhận gói sẽ kiểm tra nếu mình là **đích** thì dừng lại và ghi nhận thời gian nhận để tính toán độ trễ; nếu chưa đến đích, node sẽ **định tuyến gói** đến hàng xóm thích hợp dựa trên **thuật toán định tuyến** đã chọn. Các thuật toán định tuyến hỗ trợ gồm **XY (X-Y dimension order)**, **YX (Y-X dimension order)**, **Odd-Even (OE)** và **Negative-First (NF)** – đây đều là các thuật toán tránh nghẽn trên mesh 2D. Ví dụ, thuật toán XY buộc gói đi theo trục X trước rồi trục Y, còn OE và NF là các thuật toán bán thích ứng cấm một số kiểu rẽ nhất định để tránh nghẽn hoặc deadlock. Mỗi node duy trì các cổng kết nối với láng giềng (Đông/Tây/Nam/Bắc) và sử dụng **bộ chọn next hop** tuân theo logic thuật toán định tuyến để quyết định chuyển gói tới cổng nào. Nhờ đó, tất cả các node vừa tạo lưu lượng đầu vào vừa chuyển tiếp lưu lượng cho các node khác như một router NoC.

2.3. Kịch bản mô phỏng

Thời gian mô phỏng được chia làm hai giai đoạn: **khởi động (warm-up)** và **đo lường**. Mặc định, mô phỏng chạy tổng cộng 21000 chu kỳ (cycle) – tương ứng 21µs vì 1 chu kỳ = 1 ns – trong đó 1000 chu kỳ đầu để ổn định mạng (warm-up) và 20000 chu kỳ sau để thu thập số liệu (measurement). Trong suốt giai đoạn đo, chương trình theo dõi và tính toán **độ trễ trung bình của gói** (đơn vị: số chu kỳ mỗi gói) và **thông lượng** mạng (số flit trung bình mỗi node gửi được trên mỗi chu kỳ). Cuối mô phỏng, các giá trị này được xuất ra để đánh giá hiệu năng mạng. Bên cạnh đó, chương trình còn tính **"áp lực định tuyến" φ (routing pressure)** – một chỉ số kênh liên quan đến thuật toán định tuyến và mẫu lưu lượng. Để tính φ, mã nguồn sẽ liệt kê **toàn bộ các đường đi ngắn nhất** có thể từ mỗi nguồn tới mỗi đích theo luật của thuật toán định tuyến đã chọn, rồi xác định mức tải dồn cao nhất trên một kênh (liên kết) bất kỳ khi tất cả các node đều gửi lưu lượng. Nói cách khác, φ biểu thị mức **áp lực lớn nhất trên một kênh** trong mạng dưới thuật toán định tuyến và mẫu lưu lượng đang xét. **Đơn vị của φ** là phi-

dimensionless (không có đơn vị trực tiếp, mà có thể hiểu gián tiếp là "số gói trên mỗi chu kỳ qua kênh bận nhất khi mỗi node gửi 1 gói/chu kỳ"). Từ φ, chương trình tính được **ngưỡng PIR tối đa** lý thuyết (ký hiệu PIR_max) theo công thức:

$$\mathrm{PIR}_{\max} = \frac{\mathrm{frate}/\mathrm{len}}{\varphi},$$

trong đó *frate* là tốc độ gửi flit mỗi chu kỳ của kênh và *len* là số flit mỗi gói. Với cấu hình mô phỏng, *frate* = 0.5 flit/chu kỳ và *len* = 8 flit/gói, do đó một kênh có thể phục vụ tối đa (0.5/8) = **0,0625 gói/chu kỳ**. Giá trị φ cho biết kênh bận nhất gánh bao nhiêu phần lưu lượng; chẳng hạn, nếu φ = 6, điều đó nghĩa là kênh tắc nhất phải chuyển trung bình 6 gói mỗi chu kỳ khi mỗi node gửi 1 gói/chu kỳ. Khi đó **PIR_max = 0,0625 / 6 ≈ 0,0104167 gói/(node*chu kỳ)** – tức khoảng 1,04% tốc độ chèn gói tối đa lý thuyết. Nói cách khác, nếu mọi node gửi với tốc độ PIR_max thì kênh "nóng" nhất vừa đầy tải (đạt 100% dung lượng); vượt quá PIR_max sẽ gây tắc nghẽn tại kênh này. Chỉ số routing pressure φ vì vậy được dùng để **đánh giá hiệu năng thuật toán định tuyến một cách tĩnh** (không cần mô phỏng) và **dự báo điểm bão hòa của mạng**: thuật toán nào phân tải đều hơn (φ nhỏ) sẽ cho phép tăng lưu lượng cao hơn trước khi nghẽn (PIR_max lớn hơn), còn thuật toán có φ lớn sẽ dễ nghẽn tại một điểm nút cổ chai sớm hơn.

***** Tham số mô phỏng chính**

Bảng dưới đây liệt kê các tham số chính của kịch bản mô phỏng:

Tham số	**Giá trị (cấu hình)**
Kích thước mạng	Lưới 2D kích thước 7×7 (49 node). Mỗi node gán một ID duy nhất từ 0 đến 48 theo tọa độ (x,y).
Thuật toán định tuyến	Mặc định **XY** (định tuyến thứ tự trục X rồi Y). Hỗ trợ các tùy chọn khác: **YX**, **Odd-Even (OE)**, **Negative-First (NF)**.

Tham số	Giá trị (cấu hình)
Mẫu lưu lượng (traffic)	Mặc định **uniform** (chọn ngẫu nhiên đích bất kỳ với xác suất như nhau). Hỗ trợ **transpose1** và **transpose2** (các mẫu hoán vị đối xứng trên ma trận).
Tốc độ bơm gói (PIR)	Mặc định **0,01 gói/(node*chu kỳ)**. (Đơn vị này nghĩa là mỗi node phát trung bình 0,01 gói mỗi chu kỳ; tương đương 0,01/1ns ≈ 10^7 gói/giây). Lưu ý quá trình phát là Poisson ngẫu nhiên quanh giá trị trung bình này.
Độ dài gói	**32 byte mỗi gói** (tương ứng **8 flit** trong mô hình NoC). Mỗi flit được truyền trong 2 chu kỳ ở tốc độ kênh hiện tại, nên một gói cần 16 chu kỳ để đi hết một liên kết.
Đơn vị chu kỳ	**1 ns mỗi chu kỳ**. (Thời gian mô phỏng tính bằng đơn vị chu kỳ, ví dụ 1000 chu kỳ = 1000 ns).
Thời gian mô phỏng	**21000 chu kỳ** (tương đương 21 μs). Bao gồm **1000 chu kỳ warm-up** + **20000 chu kỳ đo** hiệu năng.

Chú thích:

- Trong quá trình đo lường, chương trình thu thập **độ trễ trung bình trên mỗi gói** (tính bằng số chu kỳ) và **thông lượng trung bình** (số flit chuyển được trên mỗi node mỗi chu kỳ) để đánh giá hiệu quả truyền dẫn. Đồng thời, chỉ số **routing pressure φ** được tính toán nhằm xác định **mức phân bổ tải** trên mạng cho thuật toán định tuyến đang dùng, qua đó suy ra **PIR_max** – tốc độ chèn gói tối đa lý thuyết trước khi mạng bắt đầu nghẽn do quá tải. Các thông số và kết quả này giúp so sánh hiệu năng giữa các thuật toán định tuyến khác nhau trong mạng NoC.
- Định nghĩa transpose 1 và transpose 2:

  Giả sử mesh là **N × N**, mỗi node có tọa độ

  (x, y)với $0 \leq x, y < N$.

Trong code:

Coord src = IdxToCoord(i, width); → lấy (x, y)của node nguồn.

Sau đó tính $(dstX, dstY)$khác nhau tùy kiểu traffic.

a. Transpose2 – "hoán vị theo đường chéo chính"

Định nghĩa:

$$(x, y) \longrightarrow (y, x)$$

Tức là **đổi chỗ** hai tọa độ, giống transpose ma trận bình thường.

Ví dụ với N = 4:

(0, 0) → (0, 0)

(0, 1) → (1, 0)

(1, 3) → (3, 1)

(2, 0) → (0, 2)

Trong code:

// transpose2

dstX = src.y;

dstY = src.x;

Ý nghĩa: các node "phía trên" gửi xuống dưới, bên trái gửi sang phải… tạo ra **các luồng chéo qua đường chéo chính**, làm nhiều đường đi cắt nhau ở giữa mạng.

b. Transpose1 – "hoán vị theo đường chéo phụ + đảo"

Định nghĩa trong code:

// N = width = height

dstX = (N - 1) - src.y;

dstY = (N - 1) - src.x;

Tức là:

$$(x, y) \longrightarrow (N - 1 - y,\ N - 1 - x)$$

Có thể hiểu là: **đổi chéo + lật qua tâm** (xoay đối xứng qua đường chéo phụ).

Ví dụ N = 4:

(0, 0) → (3, 3)

(0, 1) → (2, 3)

(1, 0) → (3, 2)

(1, 2) → (1, 2) (trùng thì code đổi sang đích khác cho khỏi trùng chính nó)

Các luồng lúc này thường đi **từ góc này sang góc đối diện**, rất dài và gặp nhau ở khu trung tâm.

3. Công cụ sử dụng

3.1. NS3

Trong dự án, ns-3 đóng vai trò là **khung mô phỏng** giúp xây dựng mạng NoC 2D mesh, sinh lưu lượng, thực thi thuật toán định tuyến và đo các chỉ số hiệu năng. Cụ thể, ns-3 cung cấp sẵn các lớp **Node, NetDevice, Channel, Application, RandomVariable, Mobility, Animation…**, còn phần logic NoC (header, tag, router, routing XY/OE/NF, tính routing pressure) được cài đặt thêm bởi người lập trình. Nhờ đó, thay vì tự xây dựng một simulator từ đầu, ta tận dụng bộ lập lịch sự kiện rời rạc, mô hình liên kết point-to-point, hàng đợi, hệ thống thời gian và API đo đạc của ns-3 để tập trung vào bài toán chính là **so sánh thuật toán định tuyến và routing pressure**.

3.2. Ngôn ngữ C++ và Python

C++ được dùng để cài đặt **toàn bộ lõi mô phỏng NoC**:

- **Mô hình NoC và định tuyến**
  - Định nghĩa các cấu trúc như Coord, SendTimeTag, NoCHeader, lớp ứng dụng NoCRouterApp…
  - Cài đặt các hàm trợ giúp cho topo mesh 2D, ánh xạ ID ↔ tọa độ, di chuyển theo hướng (E/W/N/S), kiểm tra biên.
  - Cài đặt **thuật toán định tuyến** XY, YX, OE, NF bằng các hàm C++ (kiểm tra turn hợp lệ, chọn next hop,…).
- **Sinh traffic và xử lý gói tin**
  - NoCRouterApp kế thừa ns3::Application, dùng C++ để:
    - Lên lịch phát gói tiếp theo với Simulator::Schedule.
    - Sinh thời gian giữa hai lần phát bằng ExponentialRandomVariable (Poisson).
    - Chọn đích ngẫu nhiên bằng UniformRandomVariable.
    - Gắn header, tag thời gian, gửi gói qua NetDevice::Send.
    - Khi nhận gói: kiểm tra ID đích, nếu là đích thì tính độ trễ; nếu không thì tính hướng trước đó và gọi hàm chọn next hop.
- **Tính toán routing pressure φ**
  - Dùng C++ duyệt toàn bộ cặp nguồn–đích, liệt kê các đường đi tối thiểu hợp lệ bằng DFS, đếm số lần mỗi link xuất hiện, từ đó tính được **φ** và **PIR_max**.
  - Phần này chạy hoàn toàn tĩnh (không cần mô phỏng thời gian), nên C++ cho hiệu năng tốt khi duyệt nhiều route.
- **Xuất kết quả dạng text**
  - Sau khi Simulator::Run() kết thúc, C++ tính:
    - Tổng số gói nhận được
    - Độ trễ trung bình (cycles)
    - Throughput (flits/(node·cycle))
    - φ và PIR_max
  - Các giá trị này được **in ra stdout**, sau đó Python sẽ đọc và xử lý.

Sau khi có chương trình C++/ns-3, Python được dùng như **công cụ hậu xử lý (post-processing)**:

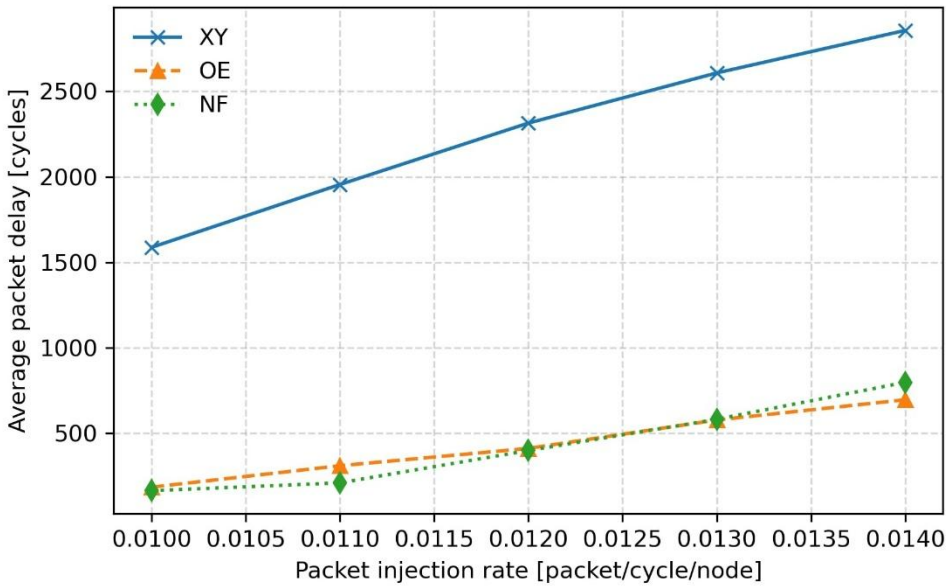
- **Tự động chạy nhiều kịch bản**
  - Python gọi chương trình mô phỏng C++ nhiều lần (ví dụ với các giá trị PIR: 0.0100, 0.0110, 0.0120, 0.0130, 0.0140; và với routing = XY, OE, NF).
  - Mỗi lần chạy sẽ thu lại dòng output chứa Avg packet latency, Throughput, phi, PIR_max,…
  - Python parse text đó (regex / split chuỗi) và lưu vào mảng hoặc DataFrame.
- **Tạo file dữ liệu / bảng kết quả**
  - Lưu các cặp (PIR, latency, throughput, phi, PIR_max, routing, traffic) vào .csv hoặc .xlsx, tiện cho báo cáo.
  - Cũng có thể dùng pandas để tính thêm các thống kê (tỉ lệ PIR/PIR_max, so sánh giữa thuật toán,…).
- **Vẽ đồ thị biểu diễn kết quả**
  - Dùng **matplotlib** (hoặc seaborn) để vẽ các đồ thị như hình bạn gửi:
    - Trục X: **PIR [packet/cycle/node]**
    - Trục Y: **Average packet delay [cycles]**
    - Mỗi đường là một thuật toán định tuyến (XY, OE, NF) với ký hiệu khác nhau.
  - Có thể vẽ thêm:
    - Throughput vs PIR
    - PIR/PIR_max vs latency
    - So sánh φ giữa các thuật toán.

4. Phân tích kết quả

4.1. Nhận xét chung

Kết quả mô phỏng thu được cho thấy độ trễ trung bình của gói tin đều tăng khi tăng lưu lượng đưa vào mạng (PIR), đúng với trực giác và xu hướng trong bài báo gốc. Tuy nhiên, giá trị tuyệt đối của độ trễ trong mô hình ns-3 cao hơn khá nhiều so với Noxim (hàng trăm đến hàng nghìn chu kỳ thay vì vài chục đến vài trăm chu kỳ). Nguyên nhân là do mô hình trong luận văn chỉ mô phỏng router ở mức gói tin trên liên kết point-to-point với hàng đợi DropTail lớn, chưa mô phỏng chi tiết wormhole, bộ đệm 4 flit và cơ chế phân xử vòng tròn như trong Noxim. Vì vậy, phần này chủ yếu tập trung so sánh xu hướng tương đối giữa các thuật toán định tuyến, hơn là so khớp tuyệt đối từng điểm số.

4.2. Kết quả dưới tình huống lưu lượng Transpose1:



Hình 1. Độ trễ gói tin trung bình theo PIR với lưu lượng transpose1 dưới các thuật toán định tuyến XY, OE và NF

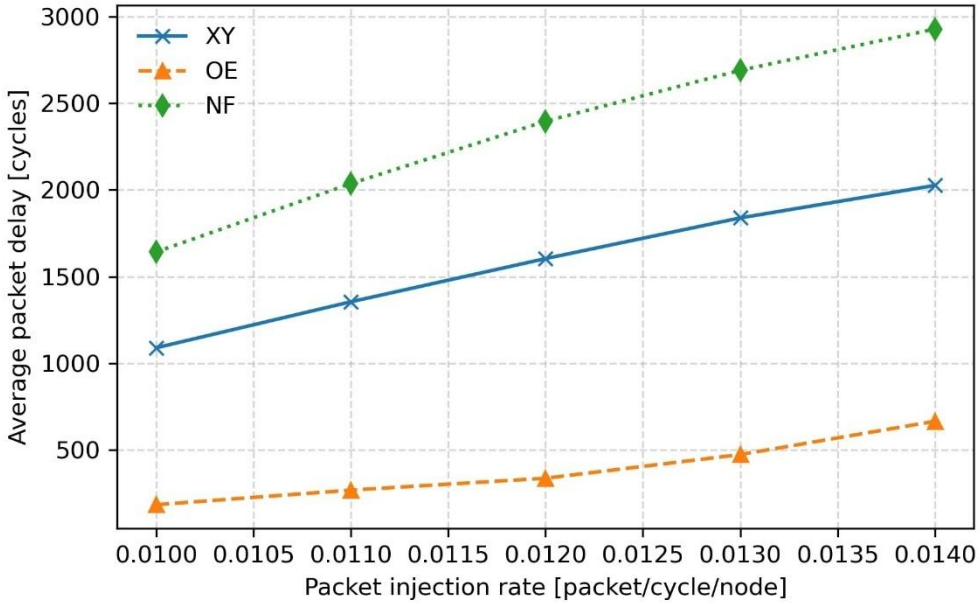
Với lưu lượng transpose1 trên lưới 7×7, Hình 1 thể hiện độ trễ trung bình theo PIR cho ba thuật toán XY, OE và NF. Kết quả cho thấy rõ ràng rằng thuật toán XY luôn có độ trễ lớn nhất. Khi PIR tăng từ 0,010 lên 0,014 packet/cycle/node, độ trễ trung bình của XY tăng từ khoảng 1 586 chu kỳ lên gần 2 857 chu kỳ. Trong cấu hình này, áp lực

định tuyến của XY là $\varphi \approx 6$, dẫn đến ngưỡng lý thuyết PIR_max ≈ 0,0104. Như vậy, ngay từ PIR = 0,011 trở lên ta đã có tỉ lệ PIR / PIR_max > 1, tức là đang bơm lưu lượng vượt quá cận trên do φ dự đoán, và điều này được phản ánh bằng việc độ trễ của XY tăng rất nhanh.

Đối với NF, giá trị φ cũng bằng 6 nên PIR_max lý thuyết giống XY. Tuy nhiên trong mô phỏng, NF cho độ trễ thấp hơn XY rất nhiều: độ trễ tăng từ khoảng 162 chu kỳ ở PIR = 0,010 lên khoảng 797 chu kỳ ở PIR = 0,014. Ngay cả khi PIR / PIR_max ≈ 1,34 (PIR = 0,014, đã vượt xa ngưỡng φ–bound), mạng dùng NF vẫn hoạt động tốt hơn XY rất rõ rệt. Điều này cho thấy cùng một tập đường cực tiểu nhưng cách chọn nhánh cụ thể của NF (ưu tiên hướng âm) giúp phân bố lưu lượng đều hơn lên các kênh, giảm nghẽn cổ chai so với XY.

Thuật toán OE có áp lực định tuyến nhỏ hơn, $\varphi \approx 4{,}36$, tương ứng PIR_max ≈ 0,0143. Ở dải PIR khảo sát 0,010–0,014, tỉ lệ PIR / PIR_max chỉ nằm trong khoảng 0,70–0,98 nên OE luôn hoạt động dưới ngưỡng φ–bound. Thực nghiệm cho thấy độ trễ của OE chỉ từ khoảng 183 chu kỳ lên khoảng 695 chu kỳ, thấp hơn XY rất nhiều và ngang ngửa, thậm chí tốt hơn NF trong một số điểm. Như vậy, với transpose1, OE là thuật toán cân bằng tốt nhất giữa khả năng chịu tải (PIR_max cao) và độ trễ.

4.3. Kết quả dưới tình huống lưu lượng Transpose2:



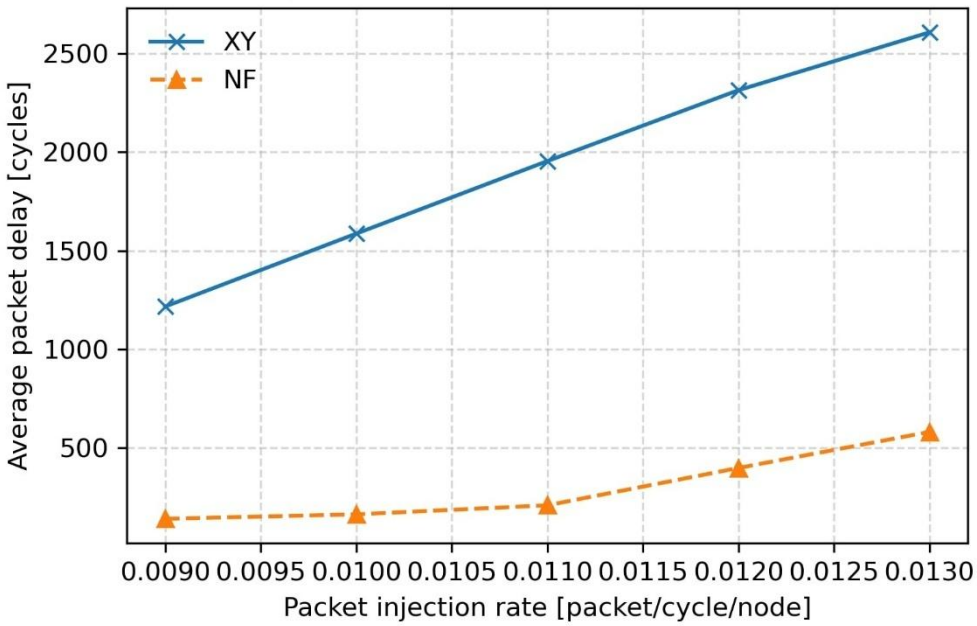
Hình 2. Độ trễ gói tin trung bình theo PIR với lưu lượng transpose2 dưới các thuật toán định tuyến XY, OE và NF

Hình 2 trình bày quan hệ giữa độ trễ và PIR dưới lưu lượng transpose2. Về mặt giá trị φ, XY vẫn có φ ≈ 6 (PIR_max ≈ 0,0104), OE có φ ≈ 4,36 (PIR_max ≈ 0,0143), còn NF có φ nhỏ nhất φ ≈ 2,60, cho PIR_max lý thuyết khoảng 0,024, tức là gần gấp đôi XY. Nếu chỉ nhìn vào φ thì NF được dự đoán là chịu tải tốt nhất.

Tuy nhiên, kết quả ns-3 lại cho thấy xu hướng ngược lại. Độ trễ của XY tăng từ khoảng 1 089 chu kỳ (PIR = 0,010) lên khoảng 2 026 chu kỳ (PIR = 0,014), và mức tăng nhanh rõ rệt khi PIR / PIR_max vượt 1. OE vẫn giữ độ trễ thấp và tăng tương đối từ tốn, khoảng 185 → 666 chu kỳ trong cùng dải PIR, phù hợp với việc toàn bộ điểm đo vẫn nằm trong vùng PIR < PIR_max của OE. Ngược lại, NF cho độ trễ rất cao dù PIR / PIR_max mới chỉ trong khoảng 0,42–0,58, tức là còn xa ngưỡng φ–bound: độ trễ tăng từ khoảng 1 642 chu kỳ lên gần 2 927 chu kỳ. Điều này cho thấy giả thiết của metric φ (gói được phân bố đều trên tất cả đường cực tiểu khả dĩ) không còn đúng với cách hiện thực Negative-First trong ns-3. Cách ưu tiên hướng âm và ngẫu nhiên hóa của NF có thể khiến nhiều đường đi bị dồn tải, làm hiệu năng thực tế kém hơn so với

cận trên lý thuyết.

4.4. Vùng tải gần cực đại dưới XY và NF với transpose1:



Hình 3. Độ trễ gói tin trung bình ở vùng PIR lớn với lưu lượng transpose1 cho các thuật toán XY và NF.

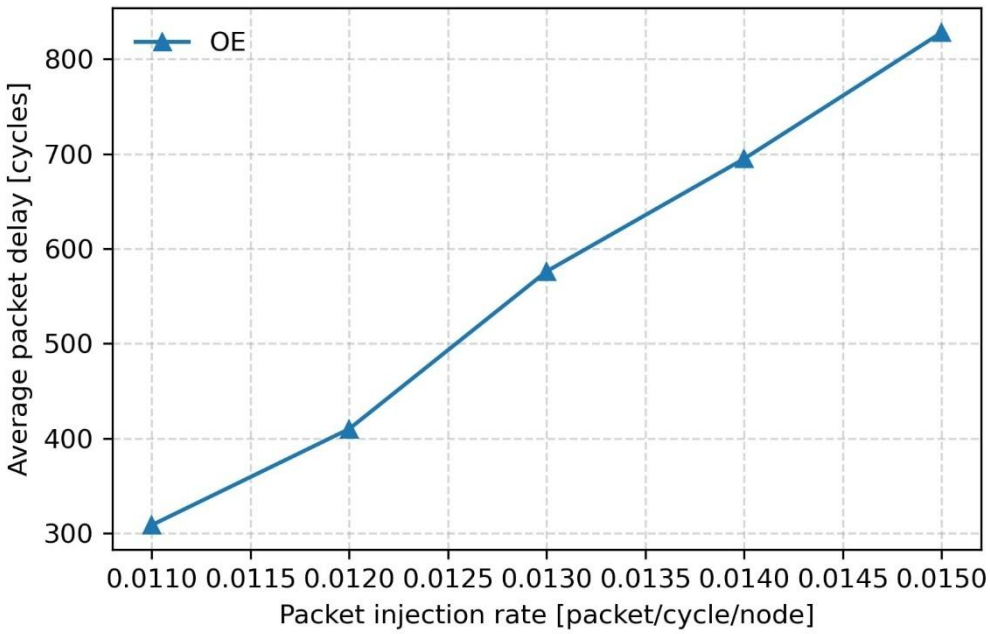
Hình 3 phóng to vùng PIR gần cực đại đối với lưu lượng transpose1, với các giá trị từ 0,009 đến 0,013 packet/cycle/node cho hai thuật toán XY và NF. Ở đây cả hai thuật toán đều có $\varphi = 6$, PIR_max ≈ 0,0104, nên tỉ lệ PIR / PIR_max tăng từ 0,864 (PIR = 0,009) lên 1,248 (PIR = 0,013).

Đối với XY, khi PIR tiến sát và vượt qua PIR_max, độ trễ tăng rất nhanh: từ khoảng 1 217 chu kỳ ở PIR = 0,009 lên khoảng 2 608 chu kỳ ở PIR = 0,013. Điều này minh hoạ khá rõ vai trò của φ: ngay khi kênh có áp lực lớn nhất bị "no" theo dự đoán của φ, toàn mạng bắt đầu bão hòa, độ trễ lan truyền tăng đột biến.

Trong khi đó, NF vẫn giữ được độ trễ thấp đáng kể trong cùng dải PIR. Độ trễ tăng từ khoảng 140 chu kỳ (PIR = 0,009) lên khoảng 581 chu kỳ (PIR = 0,013). Mặc dù từ PIR ≈ 0,011 trở đi PIR / PIR_max của NF đã lớn hơn 1, mạng vẫn hoạt động ổn định hơn XY rất nhiều. Kết quả này cho thấy φ là một chỉ báo khá "khắt khe" đối với NF:

về mặt lý thuyết, NF đã bước vào vùng quá tải, nhưng nhờ cách chọn đường "âm trước" mà cấu hình ns-3 vẫn khai thác được thêm một khoảng tải nữa trước khi độ trễ bùng nổ.

4.5. Vùng tải gần cực đại dưới OE với transpose1



Hình 4. Độ trễ gói tin trung bình ở vùng PIR lớn với lưu lượng transpose1 cho thuật toán định tuyến OE.

Hình 4 xét riêng thuật toán OE trên lưu lượng transpose1 với PIR trong khoảng 0,011–0,015. Với $\varphi \approx 4{,}36$, OE có cận trên PIR_max ≈ 0,0143. Trong ba điểm đầu tiên (0,011–0,013), tỉ lệ PIR / PIR_max nằm trong khoảng 0,77–0,91 và độ trễ tăng từ khoảng 309 lên 576 chu kỳ, vẫn ở mức khá thấp so với XY. Tại PIR = 0,014, PIR / PIR_max ≈ 0,98, độ trễ đạt khoảng 695 chu kỳ, cho thấy mạng đã tiến rất gần đến vùng bão hòa nhưng vẫn chưa có "gấp khúc" rõ rệt trên đường cong.

Khi PIR tăng lên 0,015, PIR / PIR_max ≈ 1,05, tức là đã vượt nhẹ qua ngưỡng φ–bound. Tại điểm này, độ trễ nhảy lên khoảng 828 chu kỳ, cao hơn đáng kể so với xu hướng tuyến tính trước đó. Điều này phù hợp với trực giác từ metric φ: OE chịu tải tốt

hơn XY và NF dưới transpose1 (PIR_max lớn hơn), nên mạng có thể vận hành ở mức PIR khoảng 0,014 mà độ trễ vẫn trong ngưỡng chấp nhận được. Chỉ khi PIR tiếp tục tăng vượt quá cận trên do φ dự đoán, độ trễ mới bắt đầu tăng mạnh.

5. Kết luận

Trong khuôn khổ đề tài, nhóm đã xây dựng được một mô hình mô phỏng mạng trên chip dạng lưới 7×7 trong ns-3 dựa trên các liên kết point-to-point, ánh xạ thời gian mô phỏng sang đơn vị "chu kỳ" và cài đặt cơ chế sinh gói Poisson đúng theo định nghĩa PIR (số gói trên nút trên chu kỳ). Trên nền tảng đó, đề tài đã hiện thực ba thuật toán định tuyến XY, OE và NF ở mức gói tin, đồng thời mô phỏng được ba mẫu hình lưu lượng đặc trưng là uniform, transpose1 và transpose2. Bên cạnh phần mô phỏng, nhóm cũng hiện thực thuật toán tính áp lực định tuyến φ của Tang bằng cách liệt kê toàn bộ đường đi cực tiểu dưới từng thuật toán định tuyến và từng mẫu hình lưu lượng, từ đó suy ra cận trên lý thuyết PIR_max = (frate/len)/φ cho mỗi kịch bản.

Từ hệ thống này, nhóm đã xây dựng tập script tự động quét PIR, thu thập các chỉ số độ trễ trung bình, thông lượng, φ, PIR_max và tỉ lệ PIR/PIR_max, sau đó vẽ lại được các đồ thị tương tự các Hình 12, 13, 16, 17 trong bài báo gốc cho trường hợp XY/OE/NF (không có APSRA). Các kết quả mô phỏng cho thấy xu hướng chung phù hợp với trực giác và với bài báo: độ trễ tăng nhanh khi PIR tiến gần hoặc vượt ngưỡng φ-bound; dưới lưu lượng transpose1, thuật toán XY có độ trễ lớn nhất còn OE và NF cho độ trễ nhỏ hơn đáng kể; OE chịu tải tốt hơn XY nhờ có áp lực định tuyến nhỏ hơn. Với lưu lượng transpose2, XY và OE vẫn duy trì được độ trễ ở mức chấp nhận được, trong khi NF cho độ trễ rất cao – đây là một điểm khác biệt thú vị so với dự đoán lý thuyết từ φ và cho thấy hành vi thực tế của thuật toán định tuyến còn phụ thuộc vào cách hiện thực cụ thể trong router.

Tuy nhiên, mô hình hiện tại vẫn tồn tại nhiều hạn chế khiến các giá trị tuyệt đối thu được chênh lệch khá lớn so với kết quả của Noxim. Router trong ns-3 mới chỉ được mô phỏng ở mức gói tin trên liên kết point-to-point với hàng đợi DropTail lớn, chưa mô phỏng chi tiết cơ chế wormhole switching với bộ đệm 4 flit mỗi cổng và phân xử

vòng tròn như trong bài báo. Cách hiện thực OE và NF dựa trên turn-model kết hợp chọn ngẫu nhiên cũng có thể khác với giả thiết khi tác giả bài báo tính φ, dẫn đến phân bố tải thực tế trên các kênh không trùng với phân bố dùng trong phân tích. Ngoài ra, do hạn chế về thời gian, đề tài chưa triển khai thuật toán APSRA, chưa khảo sát các kích thước mạng khác và chưa hiệu chỉnh kỹ các tham số warm-up, thời gian đo để xác định chính xác điểm bão hòa cho từng cấu hình.

Từ những kết quả và hạn chế trên có thể thấy rằng routing pressure φ vẫn là một thước đo hữu ích để ước lượng vùng tải an toàn của mạng NoC: với các thuật toán "đơn giản" như XY và OE, vị trí điểm gãy trên đường cong độ trễ bám khá sát ngưỡng PIR_max mà φ dự đoán. Đồng thời, các sai khác giữa lý thuyết và thực nghiệm – đặc biệt với thuật toán NF và với mô hình router ở mức gói tin – cũng là một kết quả có ý nghĩa, cho thấy khi kiến trúc router và cơ chế chọn đường thay đổi, mối quan hệ giữa φ và hiệu năng thực tế cũng thay đổi theo. Vì vậy, mặc dù đề tài chưa tái hiện được đầy đủ các giá trị định lượng của bài báo, phần mềm mô phỏng xây dựng trên ns-3 vẫn là một bước thử nghiệm quan trọng, tạo nền tảng để tiếp tục hoàn thiện mô hình wormhole chi tiết hơn, bổ sung APSRA và mở rộng sang các nghiên cứu NoC khác trong tương lai.

6. Tài liệu kham khảo

[1] "Routing Pressure: A Channel-Related and Traffic-Aware Metric of Routing Algorithm," *NS3 Simulation Projects*, Oct. 10, 2015. [Online]. Available: <https://ns3simulation.com/routing-pressure-a-channel-related-and-traffic-aware-metric-of-routing-algorithm/\>