



**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**BÁO CÁO PROJECT III**

**GVHD : PGS.TS Nguyễn Khanh Văn**

**Bộ môn : Công nghệ phần mềm**

**Sinh viên : Đào Tuấn Dũng**

**MSSV : 20173050**

**Lớp : CNTT.09-K62**

*Hà Nội, 1/2021*

## **Lời cảm ơn**

Lời đầu tiên, em xin được gửi lời cảm ơn chân thành nhất đến thầy Nguyễn Khanh Văn đã luôn nhiệt tình hướng dẫn, chỉ bảo em trong quá trình làm đồ án Project 3. Để hoàn thành đồ án nghiên cứu sử dụng trình giả lập OMNeT++, ngoài sự nỗ lực của bản thân, em xin được gửi lời cảm ơn chân thành nhất đến thầy Nguyễn Tiến Thành đã hỗ trợ kỹ thuật và giành thời gian giúp đỡ để em hoàn thành đồ án.

Do thời gian làm đồ án trong một kỳ học, cũng như những hạn chế về trình độ của bản thân, bản báo cáo của em không tránh khỏi những thiếu sót, hạn chế. Em rất mong nhận được những ý kiến nhận xét, góp ý từ thầy cô để em rút kinh nghiệm và hoàn thành môn học tốt hơn.

Em xin chân thành cảm ơn!

Hà Nội, Tháng 1 năm 2021

Người thực hiện

**Đào Tuấn Dũng**

## Tóm tắt

OMNeT++ là viết tắt của cụm từ Objective Modular Network Tested in C++. OMNeT++ là một môi trường phát triển tích hợp – IDE (Integrated Development Environment) giả lập các sự kiện rời rạc, cung cấp môi trường để tiến hành mô phỏng hoạt động của mạng. Mục đích chính của ứng dụng là mô phỏng hoạt động mạng thông tin, bên cạnh đó do tính phổ cập và linh hoạt của nó, OMNeT++ còn được sử dụng trong nhiều lĩnh vực khác. Tuy vậy, OMNeT++ vẫn chưa được đưa vào giới thiệu rộng rãi cho sinh viên khối ngành kỹ thuật nói chung và sinh viên ngành công nghệ thông tin Bách khoa nói riêng. Để góp phần mở rộng cộng đồng sử dụng OMNeT++, em viết báo cáo này để giới thiệu về trình giả lập OMNeT++ và cách sử dụng OMNeT++ thông qua các ví dụ giả lập mạng cơ bản.

Báo cáo sẽ trình bày cách cài đặt, sử dụng OMNeT++ trên hệ điều hành Windows và cách viết mã NED, lập trình C++ trong OMNeT++. Sau đó là các ví dụ, cài đặt và chạy thử mã nguồn giả lập chương trình: “Mô phỏng quá trình truyền tin trong mạng trung tâm dữ liệu Fat-tree.”

Nội dung tiếp theo của báo cáo sẽ được tổ chức như sau:

Chương 1: Giới thiệu đề tài, mục tiêu và phạm vi đề tài, định hướng giải pháp

Chương 2, 3, 4, 5: Chương này trình bày các kiến thức cơ sở về OMNeT++, ngôn ngữ NED và C++, mạng trung tâm dữ liệu Fat-tree

Chương 6: Các ví dụ, chương trình demo và kết quả thực nghiệm

Chương 7: Những khó khăn gặp phải trong quá trình học

Chương 8: Chương này tổng kết lại quá trình học

Phụ lục: Hướng dẫn cách cài đặt OMNeT++ trên hệ điều hành Window, hướng dẫn chạy thử ví dụ và chương trình demo.

# Mục lục

Lời cảm ơn	2
Tóm tắt	4
Chương 1: Giới thiệu về tài	7
Chương 2: Giới thiệu OMNeT++	9
2.1. Định nghĩa	9
2.2. Các thành phần chính của OMNeT++	9
2.3. Ứng dụng	9
2.4. Sử dụng OMNeT++	10
2.4.1. Xây dựng và chạy thử các mô hình mô phỏng	10
2.4.2. Hệ thống file	12
3.1. Tổng quan về ngôn ngữ NED	14
3.1.1. Giới thiệu về ngôn ngữ NED	14
3.1.2. Các thành phần của ngôn ngữ mô tả NED	14
3.1.3. Các từ khoá	15
3.1.4. Nguyên tắc đặt tên	15
3.1.4. Chú thích	15
3.2. Ví dụ sử dụng ngôn ngữ NED	16
3.2.1. Network	16
3.2.2. Channel:	17
3.2.3. Cấu trúc của Module đơn	18
3.2.4 Cấu trúc của Node	19
3.3. Sử dụng ngôn ngữ NED	20
3.3.1. Các module đơn	20
3.3.2. Module phức hợp	21
3.3.3. Channels	22
3.3.4. Tham số	23
3.3.4. Gates	25
3.3.5. Submodules	25
3.3.6. Connections	26

<b>3.3.7. Multiple Connections</b>	<b>28</b>
<b>3.3.8. Mô hình tham số và kiểu kết nối</b>	<b>30</b>
<b>3.3.9. Metadata Annotations (properties)</b>	<b>31</b>
<b>3.4. Giới thiệu GNED</b>	<b>32</b>
<b>Chương 4: Lập trình C++ trong OMNeT++</b>	<b>33</b>
<b>4.1. Ngôn ngữ lập trình C++</b>	<b>33</b>
<b>4.2. Vai trò của C/C++ trong trình giả lập OMNeT++</b>	<b>33</b>
<b>4.3. Lập trình C++ trong OMNeT++</b>	<b>34</b>
<b>Chương 5: Kiến thức cơ sở mạng trung tâm dữ liệu Fat-tree</b>	<b>35</b>
<b>5.1. Mô hình mạng Fat-tree</b>	<b>35</b>
<b>5.2. Xây dựng bảng định tuyến</b>	<b>35</b>
<b>5.3. Công thức tính thông lượng trung bình</b>	<b>36</b>
<b>Chương 6: Kết quả thực nghiệm</b>	<b>37</b>
<b>6.1. Ví dụ</b>	<b>37</b>
<b>6.2. TicToc Tutorial</b>	<b>37</b>
<b>6.3. Mạng không dây (Wireless Sensor Networks – WSN)</b>	<b>38</b>
<b>6.4. Fat-tree</b>	<b>39</b>
<b>Chương 7: Những khó khăn gặp phải</b>	<b>41</b>
<b>Chương 8: Kết luận chung</b>	<b>42</b>
<b>Tài liệu tham khảo</b>	<b>43</b>
<b>Phụ lục</b>	<b>44</b>
<b>A Hướng dẫn cài đặt OMNET++ trên window.</b>	<b>44</b>
<b>B Hướng dẫn cài đặt và chạy ví dụ Demo</b>	<b>46</b>
<b>C Hướng dẫn cài đặt và chạy chương trình: “Tính thông lượng throughput theo thời gian của quá trình truyền tin trong mạng trung tâm dữ liệu (Fat-tree)</b>	<b>50</b>

# Chương 1: Giới thiệu đề tài

## 1.1. Đặt vấn đề

Ngày nay, cùng với sự phát triển của Internet thì việc kết nối mọi thứ thông qua Internet đang là xu hướng phát triển rất được quan tâm. Khi đó, mọi vật được kết nối với nhau thông qua mạng Internet. Vấn đề là làm cách nào để kết nối các thiết bị với nhau, sử dụng mô hình nào để kết nối, việc định tuyến trong quá trình gửi gói tin, thông tin gửi đi phải đảm bảo tính chính xác và bảo mật, đảm bảo hiệu quả của mô hình, ... Có rất nhiều vấn đề được đặt ra cần được giải quyết.

Trong đó, việc triển khai thực tế gặp nhiều rủi ro về chi phí cũng như hiệu quả thực tế của mô hình rất được quan tâm. Để giải quyết vấn đề này, việc giả lập lại hệ thống sẽ hoạt động như thế nào trong thực tế là rất cần thiết. Từ đó tối thiểu được các sai sót, rủi ro có thể gặp phải và tìm cách khắc phục trước khi triển khai. Do đó, trong đồ án môn học này em sử dụng trình giả lập OMNeT++. Đây là một trình giả lập mạnh mẽ, hiệu quả, dùng ngôn ngữ lập trình quen thuộc C/C++ để mô phỏng và tính toán. OMNeT++ là công cụ để mô phỏng mạng thông tin và được sử dụng rộng rãi trong nhiều lĩnh vực. OMNeT++ còn được sử dụng trong mục đích học tập và nghiên cứu tại nhiều trường đại học trên thế giới.

Tuy vậy, OMNeT++ vẫn chưa được đưa vào giới thiệu rộng rãi cho sinh viên khối ngành kỹ thuật nói chung và sinh viên ngành công nghệ thông tin Bách khoa nói riêng. Để góp phần mở rộng cộng đồng sử dụng OMNeT++, báo cáo này sẽ giới thiệu về trình giả lập OMNeT++ và cách sử dụng OMNeT++ thông qua các ví dụ giả lập mạng cơ bản.

## 1.2. Mục tiêu và phạm vi đề tài

Mục tiêu của đề tài là giới thiệu trình giả lập OMNeT++ và hướng dẫn cách sử dụng OMNeT++ trong giả lập mạng cho người mới bắt đầu học. Đề tài sẽ giới thiệu về cách cài đặt và sử dụng OMNeT++. Người đọc sẽ có kiến thức cơ sở về

giả lập mạng sử dụng OMNeT++, biết thêm về ngôn ngữ NED và sử dụng thành thạo trình giả lập OMNeT++. Ngoài ra, bạn sẽ được rèn luyện thêm về kỹ năng lập trình C/C++. Sau đó, sẽ tìm hiểu mô hình hoạt động của mạng trung tâm dữ liệu Fat-tree, mô phỏng quá trình truyền tin trong mạng trung tâm dữ liệu.

Phạm vi của đề tài:

- Giới thiệu trình giả lập OMNeT++ và hướng dẫn cách sử dụng OMNeT++ trong giả lập mạng.
- Mô phỏng lại hoạt động của mạng Fat-tree trong trung tâm dữ liệu.

### 1.3. Định hướng giải pháp

Để hoàn thành đề tài này, sinh viên cần thực hiện những công việc như sau:

- Cài đặt OMNeT++, tìm hiểu và thực hiện các ví dụ giả lập mạng Tictoc trên trang chủ của OMNeT++
- Thực hiện các bài hướng dẫn Tictoc và xây dựng một mạng đơn giản có thể truyền tin giữa các nút
- Xây dựng file NED cho mạng Fat-tree
- Tìm hiểu cách xây dựng bảng định tuyến cho mạng Fat-tree và sinh ngẫu nhiên cặp nguồn đích và đường đi định tuyến cho từng cặp
- Cài đặt và chạy thử chương trình mô phỏng quá trình truyền tin trong mạng trung tâm dữ liệu Fat-tree
- Viết báo cáo tổng kết quá trình học, giới thiệu về trình mô phỏng OMNeT++ và cách sử dụng nó.

## Chương 2: Giới thiệu OMNeT++

### 2.1. Định nghĩa

OMNeT++ (Objective Modular Network Tested in C++) là IDE giả lập các sự kiện rời rạc, chạy được trên cả Linux và Windows. OMNeT++ cung cấp sẵn các thành phần tương ứng với các mô hình thực tế. Các thành phần này (còn được gọi là các module) được lập trình theo ngôn ngữ C++, sau đó được tập hợp lại thành những thành phần hay những mô hình lớn hơn bằng một ngôn ngữ bậc cao (NED). OMNeT++ hỗ trợ giao diện đồ họa, tương ứng với các mô hình cấu trúc của nó đồng thời phần nhân mô phỏng (simulation kernel) và các module của OMNeT++ cũng rất dễ dàng nhúng vào trong các ứng dụng khác.

### 2.2. Các thành phần chính của OMNeT++

Các thành phần chính của OMNeT++ là:

- Thư viện phần nhân mô phỏng (simulation kernel)
- Ngôn ngữ mô tả cấu trúc liên kết NED
- Trình biên tập đồ họa (graphical network editor) cho các file NED (GNED)
- IDE mô phỏng dựa trên nền tảng Eclipse
- Giao diện đồ họa thời gian chạy mô phỏng tương tác (Qtenv)
- Giao diện dòng lệnh để thực hiện mô phỏng (Cmdenv)
- Các tiện ích (công cụ tạo makefile)
- Tài liệu hướng dẫn, ví dụ mô phỏng mẫu, ...

### 2.3. Ứng dụng

OMNeT++ là một công cụ mô phỏng các hoạt động mạng bằng các module được thiết kế hướng đối tượng. OMNeT++ thường được sử dụng trong các ứng dụng chủ yếu như:

- Mô phỏng lưu lượng của một mạng viễn thông

- Mô phỏng các giao thức
- Mô phỏng mạng đa xử lý và phân bổ rời rạc các hệ phần cứng
- Kiểm tra tính hợp lệ của cấu trúc phần cứng
- Đánh giá hoạt động của những hệ thống phần mềm phức tạp, ...

## 2.4. Sử dụng OMNeT++

### 2.4.1. Xây dựng và chạy thử các mô hình mô phỏng

Một mô hình OMNeT++ bao gồm những phần sau:

- Ngôn ngữ mô tả topology - NED (file có phần mở rộng .ned): mô tả cấu trúc của module với các tham số, các cổng, ... Các file .ned có thể được viết bằng bất kỳ bộ soạn thảo hoặc sử dụng chương trình GNED có trong OMNeT++.
- Định nghĩa cấu trúc của các message (các file có phần mở rộng .msg): Người sử dụng có thể định nghĩa rất nhiều kiểu message và thêm các trường dữ liệu cho chúng. OMNeT++ sẽ dịch những định nghĩa này sang các lớp C++ đầy đủ.
- Mã nguồn của các module đơn giản. Đây là các file C++ với phần mở rộng là .h hoặc .cc.

Hệ thống mô phỏng cung cấp cho ta các thành phần sau:

- Phần nhân mô phỏng. Phần này chứa code để quản lý quá trình mô phỏng và các thư viện lớp mô phỏng. Nó được viết bằng C++, được biên dịch và được đặt cùng dạng với các file thư viện (các file có phần mở rộng là .a hoặc .lib).
- Giao diện người sử dụng. Giao diện này được sử dụng khi thực hiện quá trình mô phỏng, tạo sự dễ dàng cho quá trình sửa lỗi, biểu diễn (demonstration) hoặc khi thực hiện mô phỏng theo từng khối (batch execution of simulations). Có một vài kiểu giao diện trong OMNeT++, tất cả đều được viết bằng C++, được biên dịch và đặt cùng nhau trong các thư viện (các file có phần mở rộng là .a hoặc .lib).

## **Thực hiện mô phỏng và phân tích kết quả**

Các chương trình thực hiện mô phỏng (the simulation executable) là các chương trình độc lập, tức là nó có thể chạy trên các máy khác không cài đặt OMNeT++ hay các file mô hình tương ứng. Khi chương trình khởi động, nó bắt đầu đọc file cấu hình (thông thường là file omnetpp.ini). File này chứa các thiết lập để điều khiển quá trình mô phỏng thực hiện, các biến cho các tham số của mô hình... File cấu hình cũng có thể được sử dụng để điều khiển nhiều quá trình mô phỏng, trong trường hợp đơn giản nhất là các quá trình mô phỏng này sẽ được thực hiện lần lượt bởi một chương trình mô phỏng (simulation program).

Đầu ra của quá trình mô phỏng là các file dữ liệu. Các file này có thể là các file vector, các file vô hướng hoặc các file của người sử dụng. OMNeT++ cung cấp một công cụ đồ họa Plove để xem và vẽ ra nội dung của các file vector. Tuy nhiên chúng ta cũng nên hiểu rằng khó mà có thể xử lý đầy đủ các file kết quả mà chỉ dùng riêng OMNeT++; các file này đều là các file có định dạng để có thể đọc được bởi các gói xử lý toán học của các chương trình như Matlab hay Octave, hoặc có thể được đưa vào bảng tính của các chương trình như OpenOffice Calc, Gnumeric hay Microsoft Excel. Tất cả các chương trình này đều có chức năng chuyên dụng trong việc phân tích số hóa, vẽ biểu diễn (visualization) vượt qua khả năng của OMNeT++. Các file vô hướng cũng có thể được biểu diễn bằng công cụ Scalar. Nó có thể vẽ được các biểu đồ, các đồ thị dựa vào tập hợp các tọa độ (x, y) và có thể xuất dữ liệu vào clipboard để có thể sử dụng trong các chương trình khác nhằm đưa những phân tích chi tiết hơn.

## **Giao diện người sử dụng**

Mục đích chính của giao diện người sử dụng là che những phần phức tạp bên trong cấu trúc của các mô hình đối với người sử dụng, dễ dàng điều khiển quá trình mô phỏng, và cho phép người sử dụng có khả năng thay đổi các biến hay các đối tượng

bên trong của mô hình. Điều này là rất quan trọng đối với pha phát triển và sửa lỗi trong dự án. Giao diện đồ họa cũng có thể được sử dụng để trình diễn hoạt động của mô hình.

Cùng một mô hình người sử dụng có thể trên nhiều giao diện khác nhau mà không cần phải thay đổi gì trong các file mô hình. Người sử dụng có thể kiểm thử và sửa lỗi rất dễ dàng qua giao diện đồ họa, cuối cùng có thể chạy nó dựa trên một giao diện đơn giản và nhanh chóng có hỗ trợ thực hiện theo khối (batch execution).

## **Các thư viện thành phần**

Các kiểu module có thể được lưu tại những vị trí độc lập với chỗ mà chúng thực sự được sử dụng. Đặc điểm này cung cấp cho người sử dụng khả năng các kiểu module lại với nhau và tạo ra các thư viện thành phần.

## **Các chương trình mô phỏng độc lập**

Các chương trình thực hiện quá trình mô phỏng có thể được lưu nhiều lần, không phụ thuộc vào các mô hình, sử dụng cùng một thiết lập cho các module đơn giản. Người sử dụng có thể chỉ ra trong file cấu hình mô hình nào sẽ được chạy. Điều này tạo khả năng cho người sử dụng có thể xây dựng những chương trình thực hiện lớn bao gồm nhiều quá trình mô phỏng, và phân phối nó như một công cụ mô phỏng độc lập. Khả năng linh hoạt của ngôn ngữ mô tả topology cũng hỗ trợ cho hướng tiếp cận này.

### **2.4.2. Hệ thống file**

Sau khi cài đặt OMNet++, thư mục omnetpp trên hệ thống máy của bạn nên chứa các thư mục con dưới đây.

Hệ thống mô phỏng:

omnetpp/	thư mục gốc của OMNeT++
bin/	các công cụ trong OMNeT++ (GNED, nedtool...)

include/	các file header cho mô hình mô phỏng
lib/	các file thư viện
bitmaps/	các biểu tượng đồ họa
doc/	các file hướng dẫn, readme...
manual/	file hướng dẫn dạng HTML
api/	API tham chiếu dạng HTML
nedxml-api/	API tham chiếu cho thư viện NEDXML
src/	mã nguồn của tài liệu
src/	mã nguồn của OMNeT++
nedc/	nedtool, trình biên dịch message
sim/	phần nhân mô phỏng
parsim/	các file dành cho việc thực hiện phân tán
netbuilder/	các file dành cho việc đọc động các file NED
envir/	mã nguồn cho giao diện người sử dụng
cmdenv/	giao diện người dùng dòng lệnh
tkenv/	giao diện người sử dụng dựa trên Tcl/tk
gned/	công cụ soạn thảo file NED
plove/	công cụ vẽ và phân tích đầu ra dạng vector
scalars/	công cụ vẽ và phân tích đầu ra dạng vô hướng
nedxml/	thư viện NEDXML
utils/	các tiện ích khác...
test/	bộ kiểm thử hồi quy
core/	kiểm tra thư viện mô phỏng
distrib/	kiểm tra cho các bản phân phối tích hợp

...

## Chương 3: Ngôn ngữ NED (Network Description)

### 3.1. Tổng quan về ngôn ngữ NED

#### 3.1.1. Giới thiệu về ngôn ngữ NED

NED (Network Description) là ngôn ngữ mô tả cấu trúc liên kết của OMNeT++. Nó có cú pháp đơn giản nhưng nó rất mạnh khi định nghĩa các cấu trúc liên kết thông thường như mạng dạng hình sao (Star Topology), mạng dạng vòng (Ring Topology), mạng dạng tuyến tính (Linear Bus Topology), mạng hình lưới (Mesh), hypercube, ...

NED được sử dụng để mô tả topology của một mô hình trong OMNeT++. NED sử dụng phương pháp mô tả module hoá. Điều này có nghĩa là một mạng có thể được mô tả như một tập hợp các mô tả thành phần (các kênh, các kiểu module đơn giản hay kết hợp). Các kênh, các kiểu module đơn giản và kết hợp được sử dụng để mô tả một mạng nào đó có thể được sử dụng lại khi mô tả một mạng khác.

Các file chứa mô tả mạng thường có phần mở rộng là .ned. Các file NED có thể được load động vào các chương trình mô phỏng, hay có thể được dịch sang C++ bằng bộ biên dịch của NED và được liên kết bên trong các chương trình thực hiện.

Vai trò của NED trong trình giả lập OMNeT++:

- Network Description (NED) cho phép người dùng khai báo các module đơn, các kết nối và cách các module đơn ghép lại tạo thành module phức tạp.
- Các khai báo ở NED giúp người dùng đặc tả được cấu trúc tĩnh của một mạng.
- NED cho phép các module và channel có các interface cũng như các quan hệ kế thừa.

#### 3.1.2. Các thành phần của ngôn ngữ mô tả NED

Một file NED bao gồm các phần như sau:

- Các chỉ dẫn import

- Khai báo các kênh
- Khai báo các module đơn giản và kết hợp
- Khai báo mạng

### **3.1.3. Các từ khoá**

Người sử dụng cần phải chú ý không sử dụng những từ khoá có sẵn của NED để đặt tên cho các đối tượng khác. Các từ khoá cơ bản của NED bao gồm: import, channel, endchannel, simple, endsimple, module, endmodule, error, delay, datarate, const, parameters, gates, submodules, connections, atesizes, if, for, do, endfor, network, endnetwork, nocheck, ref, ancestor, true, false, like, input, numeric, string, bool, char, xml, ...

### **3.1.4. Nguyên tắc đặt tên**

Trong NED, người sử dụng có thể đặt tên cho các module, các kênh, các module con, các tham số, các cổng, các thuộc tính và hàm chức năng của kênh ... Các tên này có thể bao gồm các chữ cái tiếng Anh, các chữ số và dấu gạch dưới “\_”. Tên luôn được đặt bắt đầu bằng chữ cái hoặc dấu gạch dưới. Trong trường hợp muốn đặt tên bắt đầu bằng chữ số, bạn có thể sử dụng thêm một dấu gạch dưới đặt ở đầu, ví dụ như \_3Com.

Nếu tên bao gồm nhiều từ nên viết hoa ở đầu mỗi từ hoặc có thể sử dụng dấu gạch dưới. Tên của các module, kênh và mạng nên bắt đầu bằng chữ cái in hoa còn tên của tham số, cổng và các module con nên bắt đầu bằng chữ cái thường.

NED là một ngôn ngữ có phân biệt ký tự viết hoa/thường.

### **3.1.4. Chú thích**

Các dòng chú thích có thể đặt ở bất kì vị trí nào trong file NED. Tương tự như cú pháp của C++, các dòng chú thích trong NED bắt đầu bằng dấu ‘//’.

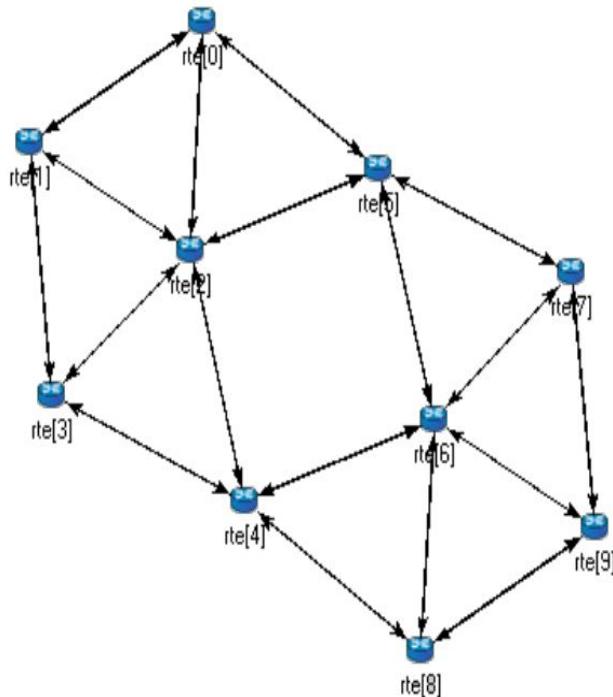
Chú thích trong NED có thể được sử dụng trong những công cụ tạo tài liệu (document generator) như JavaDoc, Doxygen.

### 3.2. Ví dụ sử dụng ngôn ngữ NED

Trong phần này, tác giả sẽ giới thiệu cách sử dụng ngôn ngữ NED thông qua ví dụ thực tế về một mạng liên lạc.

Mạng giả định bao gồm các node, mỗi node có các ứng dụng hoạt động trên đó để tạo ra gói tin. Các node có thể điều hướng cho các gói tin. Để đơn giản bỏ qua các tầng vận tải trong mạng.

#### 3.2.1. Network



Hình 3.1: Mạng minh họa

```

//  

// A network  

//  

network Network  

{  

    submodules:  

        node1: Node;  

        node2: Node;  

        node3: Node;  

        ...  

    connections:  

        node1.port++ <--> {datarate=100Mbps;} <--> node2.port++;  

        node2.port++ <--> {datarate=100Mbps;} <--> node4.port++;  

        node4.port++ <--> {datarate=100Mbps;} <--> node6.port++;  

        ...  

}

```

### 3.2.2. Channel:

Dùng để thiết lập thông số datarate

```

//  

// A Network  

//  

network Network  

{  

    types:  

        channel C extends ned.DatarateChannel {  

            datarate = 100Mbps;  

        }  

    submodules:  

        node1: Node;  

        node2: Node;  

        node3: Node;  

        ...  

    connections:  

        node1.port++ <--> C <--> node2.port++;  

        node2.port++ <--> C <--> node4.port++;  

        node4.port++ <--> C <--> node6.port++;  

        ...  

}

```

### 3.2.3. Cấu trúc của Module đơn

Module đơn là loại module không chứa các module khác ở bên trong.

Có ba loại module: App (để khai báo cách thức tạo gói tin), Routing (để khai báo cách thức định tuyến), Queue (để khai báo cách thức lưu trữ các gói tin)

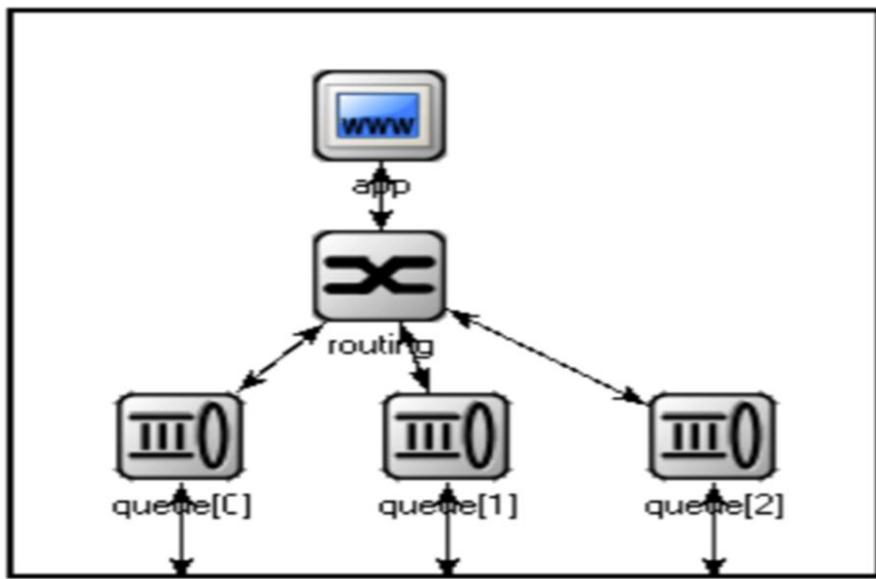
Chi tiết các cách thức đó sẽ được cài đặt bằng ngôn ngữ C++.

```
simple App
{
    parameters:
        int destAddress;
        ...
        @display("i=block/browser");
    gates:
        input in;
        output out;
}

simple Routing
{
    ...
}

simple Queue
{
    ...
}
```

### 3.2.4 Cấu trúc của Node



Hình 3.2: Module phức hợp

Một node là một module phức hợp, gồm các App, Routing và Queue. Node có tham số địa chỉ và @display để hiển thị trực quan trên giao diện người dùng.

Node có vector các gate, mang tên port[ ]

```

module Node
{
    parameters:
        int address;
        @display("i=misc/node_vs,gold");
    gates:
        inout port[];
    submodules:
        app: App;
        routing: Routing;
        queue[sizeof(port)]: Queue;
    connections:
        routing.localOut --> app.in;
        routing.localIn <-- app.out;
        for i=0..sizeof(port)-1 {
            routing.out[i] --> queue[i].in;
            routing.in[i] <-- queue[i].out;
            queue[i].line <--> port[i];
        }
}

```

### 3.3. Sử dụng ngôn ngữ NED

#### 3.3.1. Các module đơn

Nếu một số module có các lớp triển khai C++ nằm trong namespace → Có thể định nghĩa @namespace để thay thế.

```

@namespace(mylib);

simple App {
    ...
}

simple Router {
    ...
}

simple Queue {
    ...
}

```

Một module đơn cũng có thể kế thừa module đơn khác.

```

simple Queue
{
    int capacity;
    ...
}

simple BoundedQueue extends Queue
{
    capacity = 10;
}

```

### 3.3.2. Module phức hợp

Module phức hợp có gates và parameters giống như module đơn.

C++ cho module phức có thể được ghi đè ghi đè bằng thuộc tính `@class` → Không nên sử dụng

Đóng gói mã vào một module đơn và thêm nó dưới dạng mô hình con.

```

module Host
{
    types:
    ...
    parameters:
    ...
    gates:
    ...
    submodules:
    ...
    connections:
    ...
}

```

Module phức hợp có chứa các module đơn bên trong; có thể định nghĩa các gates và parameters nhưng không khai báo được mã nguồn C++ đi kèm với module phức hợp.

```

module WirelessHostBase
{
    gates:
        input radioIn;
    submodules:
        tcp: TCP;
        ip: IP;
        wlan: IEEE80211;
    connections:
        tcp.ipOut --> ip.tcpIn;
        tcp.ipIn <-- ip.tcpOut;
        ip.nicOut++ --> wlan.ipIn;
        ip.nicIn++ <-- wlan.ipOut;
        wlan.radioIn <-- radioIn;
}

module WirelessHost extends WirelessHostBase
{
    submodules:
        webAgent: WebAgent;
    connections:
        webAgent.tcpOut --> tcp.appIn++;
        webAgent.tcpIn <-- tcp.appOut++;
}

```

### 3.3.3. Channels

Đặc tả thông số thuộc tính và hành vi của một đường kết nối giữa các module.

Đặc tả thông số thông qua các tham số *delay*, *disabled* (của module DelayChannel) và tham số *datarate*, *ber*, *per* (của module DatarateChannel)

Thiết lập được mã nguồn C++ để đặc tả hành vi.

```

channel DatarateChannel2 extends ned.DatarateChannel
{
    double distance @unit(m);
    delay = this.distance / 200000km * 1s;
}

```

```

channel Backbone extends ned.DatarateChannel
{
    @backbone;
    double cost = default(1);
}

```

### 3.3.4. Tham số

Là các thông số của các module, kiểu double, int, bool, string, xml và có thể khai báo thêm volatile cũng như @unit

Gán giá trị thông qua NED file hoặc file cấu hình omnetpp.ini

```

simple App
{
    parameters:
        string protocol;           // protocol to use: "UDP" / "IP" / "ICMP" / ...
        int destAddress;          // destination address
        volatile double sendInterval @unit(s) = default(exponential(1s));
                                    // time between generating packets
        volatile int packetLength @unit(byte) = default(100B);
                                    // length of one packet
        volatile int timeToLive = default(32);
    gates:
        input in;
        output out;
}

```

### Assigning a Value-Gán giá trị

- Tham số có thể nhận các giá trị của chúng theo nhiều cách: từ mã NED, từ cấu hình (omnetpp.ini), tương tác người dùng.
- NED cho phép một người gán các tham số tại một số vị trí: trong các lớp con thông qua kế thừa; trong các mô hình con, trong các mạng và các module phức trực tiếp hoặc gián tiếp chứa mô hình con hoặc kết nối tương ứng.

```

simple PingApp extends App
{
    parameters:
        protocol = "ICMP/ECHO"
        sendInterval = default(1s);
        packetLength = default(64byte);
}

```

## Expressions

Biểu thức trong NED có cú pháp giống cú pháp trong lập trình C. Biểu thức trong NED có thể đề cập đến tham số module, vector công, kích thước vector module, chỉ mục module hiện tại trong vector module, đề cập đến các tham số module phức, module hiện tại.

Tham số module con được xác định với cú pháp:

```
submodule.parametername (or submodule[index].parametername).
```

## Volatile

- Volatile: Biểu thức giá trị của tham số được ước tính mỗi khi đọc tham số → giúp giá trị của tham số được tính toán lại mỗi khi tham số đó được truy cập.
- Non-volatile: Tham số chỉ được đánh giá một lần (chúng được đánh giá và thay thế bằng hằng số kết quả khi bắt đầu mô phỏng).

```
simple Queue
{
    parameters:
        volatile double serviceTime;
}
```

## Units

Dùng chỉ định @unit khi muốn khai báo một tham số để có đơn vị đo lường liên quan.

Các khai báo @unit(s) và @unit(byte) chỉ định đơn vị đo cho tham số.

```
simple App
{
    parameters:
        volatile double sendInterval @unit(s) = default(exponential(350ms));
        volatile int packetLength @unit(byte) = default(4KiB);
    ...
}
```

## XML Parameters

OMNeT ++ có hỗ trợ tích hợp cho các tệp XML, sử dụng trình phân tích cú pháp XML (LibXML2 hoặc Expat).

Truy cập thông qua hai loại tham số NED là xml và hàm xmldoc().

```
simple TrafGen {
    parameters:
        xml profile;
    gates:
        output out;
}

module Node {
    submodules:
        trafGen1 : TrafGen {
            profile = xmldoc("data.xml");
        }
        ...
}
```

### 3.3.4. Gates

Gate là điểm kết nối các module. Có 3 loại cổng: input, output, inout.

Một cổng chỉ có thể được kết nối với một cổng khác. Có thể tạo ra cổng đơn và danh sách các cổng. Kích thước của danh sách các cổng được truy vấn từ các biểu thức NED khác nhau.

```
simple Classifier {
    parameters:
        int numCategories;
    gates:
        input in;
        output out [numCategories];
}
```

### 3.3.5. Submodules

Submodules là module con của một module phức. Kiểu của module con được khai báo tĩnh và vẫn có thể khai báo động.

NED hỗ trợ các mảng module con (vector) và các mô hình con có điều kiện là tốt. Kích thước mảng các module con không giống kích thước mảng các cổng, phải luôn được chỉ định trước.

```
module Node
{
    submodules:
        routing: Routing;      // a submodule
        queue[sizeof(port)]: Queue; // submodule vector
```

### 3.3.6. Connections

Một số lưu ý:

- Các kết nối không thể trải dài tên các cấp thứ bậc.
- Có thể kết nối hai cổng module con, cổng module con với cổng bên trong của module mẹ.
- Không thể kết nối bất kỳ cổng nào bên ngoài module mẹ với một cổng của module con bên trong.

### Channel Specification

Đặc tả kỹ thuật của kênh (--> channel spec-->bên trong một kết nối)

Các kết nối sau sử dụng hai loại kênh do người dùng xác định: Ethernet 100 và Backbone.

```
a.g++ <--> Ethernet100 <--> b.g++;
a.g++ <--> Backbone {cost=100; length=52km; ber=1e-8;} <--> b.g++;
a.g++ <--> Backbone {@display("ls=green,2");} <--> b.g++;
```

Các tham số kết nối, tương tự như tham số submodule. Có thể sử dụng phép gán.

Một channel được xác định bằng tên của cổng nguồn cộng với tên channel.

```

module Queueing
{
    parameters:
        source.out.channel.delay = 10ms;
        queue.out.channel.delay = 20ms;
    submodules:
        source: Source;
        queue: Queue;
        sink: Sink;
    connections:
        source.out --> ned.DelayChannel --> queue.in;
        queue.out --> ned.DelayChannel <--> sink.in;
}

```

Sử dụng kết nối hai chiều sẽ phức tạp hơn, vì cả hai hướng phải được đề cập riêng. Cú pháp ++ không phải lúc nào cũng dễ dàng tìm ra cổng nào ánh xạ tới các kết nối mà người ta cần cấu hình ---> Đối tượng kết nối có thể được đặt tên để ghi đè tên mặc định “channel”.

```

network Network
{
    parameters:
        hostA.g$o[0].channel.datarate = 100Mbps; // the A -> B connection
        hostB.g$o[0].channel.datarate = 100Mbps; // the B -> A connection
        hostA.g$o[1].channel.datarate = 1Gbps; // the A -> C connection
        hostC.g$o[0].channel.datarate = 1Gbps; // the C -> A connection
    submodules:
        hostA: Host;
        hostB: Host;
        hostC: Host;
    connections:
        hostA.g++ <--> ned.DatarateChannel <--> hostB.g++;
        hostA.g++ <--> ned.DatarateChannel <--> hostC.g++;
}

```

### Reconnecting Gates (Cổng kết nối lại)

Lỗi kết nối NED là do cổng đã được kết nối → Có thể được ghi đè bằng thuộc tính @reconnect

@reconnect cho phép người dùng sửa đổi các kết nối trong module phức được kế thừa.

```

module Base {
    submodules:
        a: A;
        b: B;
    connections:
        a.out --> b.in;
}

module Derived extends Base {
    submodules:
        c: C; // inserted between a and b
    connections:
        a.out --> {@reconnect;} --> c.in;
        c.out --> {@reconnect;} --> b.in;
}

```

## Channel Names

Tên mặc định là “channel”, có thể ghi đè tên và có thể ghi đè tên mặc định cho mỗi loại channel. Channel names giúp cho việc đánh địa chỉ dễ dàng hơn khi các tham số kênh được gán từ tệp ini.

Trường hợp không có tên rõ ràng, tên kênh xuất phát từ thuộc tính `@defaultname` của loại kênh nếu tồn tại.

```

r1.pppg++ <--> eth1: EthernetChannel <--> r2.pppg++;
a.out --> foo: {delay=1ms;} --> b.in;
a.out --> bar: --> b.in;

```

```

channel Eth10G extends ned.DatarateChannel like IEth {
    @defaultname(eth10G);
}

```

### 3.3.7. Multiple Connections

Ví dụ về kết nối dạng Binary Tree

Lưu ý: Không phải mọi cổng của module sẽ được kết nối.

```

simple BinaryTreeNode {
    gates:
        inout left;
        inout right;
        inout parent;

module BinaryTree {
    parameters:
        int height;
    submodules:
        node[2^height-1]: BinaryTreeNode;
    connections allowunconnected:
        for i=0..2^(height-1)-2 {
            node[i].left <--> node[2*i+1].parent;
            node[i].right <--> node[2*i+2].parent;
        }
}

```

## Random Graph

Các kết nối có điều kiện có thể được sử dụng để tạo các cấu trúc liên kết ngẫu nhiên.

Đoạn mã sau tạo một đồ thị con kết nối ngẫu nhiên của đồ thị đầy đủ.

```

module RandomGraph {
    parameters:
        int count;
        double connectedness; // 0.0<x<1.0
    submodules:
        node[count]: Node {
            gates:
                in[count];
                out[count];
        }
    connections allowunconnected:
        for i=0..count-1, for j=0..count-1 {
            node[i].out[j] --> node[j].in[i]
            if i!=j && uniform(0,1)<connectedness;
        }
}

```

## Các mô hình kết nối

- Subgraph of a Full Graph

```
for i=0..N-1, for j=0..N-1 {  
    node[i].out[...] --> node[j].in[...] if condition(i, j);  
}
```

- Connections of Each Node

```
for i=0..Nnodes, for j=0..Nconns(i)-1 {  
    node[i].out[j] --> node[rightNodeIndex(i, j)].in[j];  
}
```

- Enumerate All Connections

```
for i=0..Nconnections-1 {  
    node[leftNodeIndex(i)].out[...] --> node[rightNodeIndex(i)].in[...];  
}
```

### 3.3.8. Mô hình tham số và kiểu kết nối

- Parametric Submodule Types

```
network Net6  
{  
    parameters:  
        string nodeType;  
    submodules:  
        node[6]: <nodeType> like INode {  
            address = index;  
        }  
    connections:  
        ...  
}
```

- Conditional Parametric Submodules

```

module Node
{
    parameters:
        string tcpType = default("Tcp");
    submodules:
        tcp: <tcpType> like ITcp if tcpType!="";
}

```

Parametric Connection Types: Hoạt động tương tự như mô hình tham số

```

a.g++ <--> <channelType> like IMyChannel <--> b.g++;
a.g++ <--> <channelType> like IMyChannel {@display("ls=red");} <--> b.g++;

```

### 3.3.9. Metadata Annotations (properties)

Các thuộc tính của NED là các metadata Annotations có thể được thêm vào các module, parameters, gates, connections, gói...

```

@namespace(foo); // file property

module Example
{
    parameters:
        @node; // module property
        @display("i=device/pc"); // module property
        int a @unit(s) = default(1); // parameter property
    gates:
        output out @loose @labels(pk); // gate properties
    submodules:
}

```

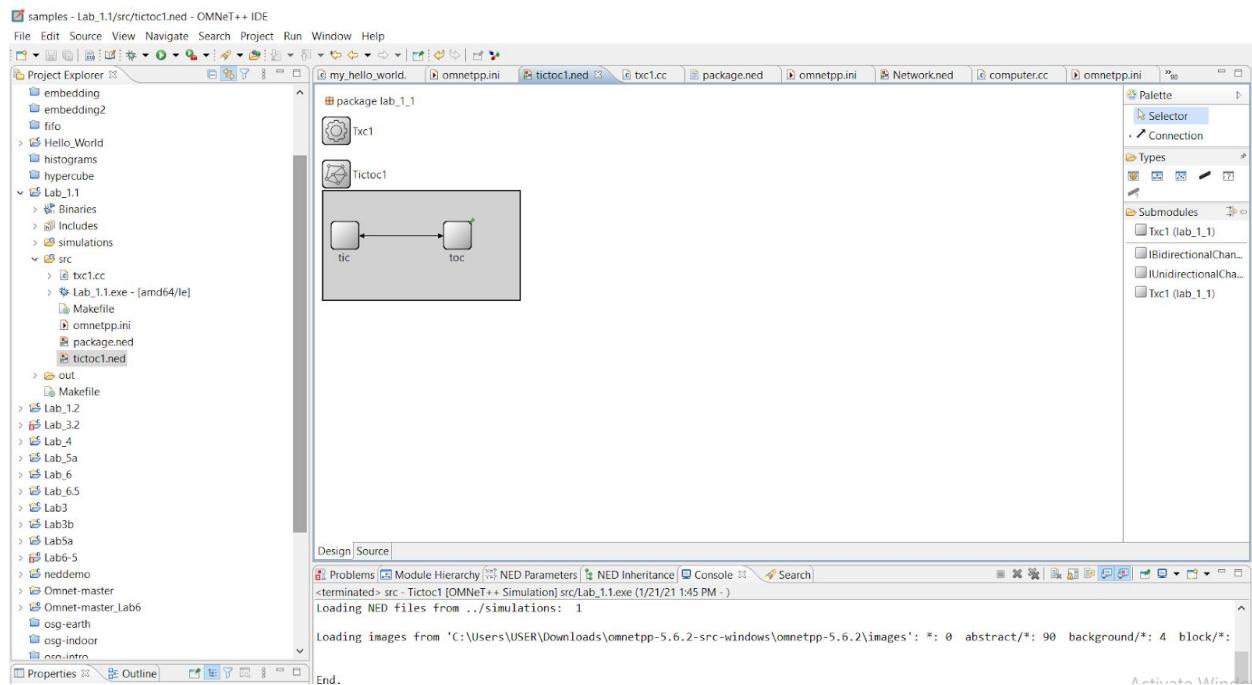
```

src: Source {
    parameters:
        @display("p=150,100"); // submodule property
        count @prompt("Enter count:"); // adding a property to a parameter
    gates:
        out[] @loose; // adding a property to a gate
    }
    ...
connections:
    src.out++ --> { @display("ls=green,2"); } --> sink1.in; // connection prop.
    src.out++ --> Channel { @display("ls=green,2"); } --> sink2.in;
}

```

### 3.4. Giới thiệu GNED

GNED (graphical network editor) là trình biên tập đồ họa cho các file NED



Giao diện của GNED hỗ trợ hai chế độ:

- Đồ họa (Design): là giao diện mặc định.
- Mã nguồn (NED Source): cho phép người sử dụng có thể sửa lại mã nguồn một cách trực tiếp.

## **Chương 4: Lập trình C++ trong OMNeT++**

### **4.1. Ngôn ngữ lập trình C++**

C++ là một loại ngôn ngữ lập trình bậc trung (middle-level). Đây là ngôn ngữ lập trình đa năng được tạo ra bởi Bjarne Stroustrup như một phần mở rộng của ngôn ngữ lập trình C, hoặc "C với các lớp Class". Ngôn ngữ đã được mở rộng đáng kể theo thời gian và C ++ hiện đại có các tính năng: lập trình tổng quát, lập trình hướng đối tượng, lập trình thủ tục, ngôn ngữ đa mẫu hình tự do có kiểu tĩnh, dữ liệu trừu tượng, và lập trình đa hình, ngoài ra còn có thêm các tính năng, công cụ để thao tác với bộ nhớ cấp thấp. Từ thập niên 1990, C++ đã trở thành một trong những ngôn ngữ thương mại ưa thích và phổ biến của lập trình viên.

C++ được thiết kế hướng tới lập trình hệ thống máy tính và phần mềm nhúng trên các mạch vi xử lý, bao gồm cả hệ thống có tài nguyên hạn chế và tài nguyên khổng lồ, với ưu điểm vượt trội về hiệu suất, hiệu quả và tính linh hoạt cao. C ++ có thể tìm thấy ở mọi nơi, với những điểm mạnh là cơ sở hạ tầng phần mềm và các ứng dụng bị hạn chế tài nguyên. Bao gồm: phần mềm ứng dụng máy tính cá nhân, trò chơi điện tử, các hệ thống máy chủ (ví dụ: phần mềm thương mại điện tử, cổ máy tìm kiếm trên web hoặc máy chủ SQL) và các ứng dụng ưu tiên về hiệu suất (ví dụ: tổng đài thông tin liên lạc hoặc thiết bị thăm dò không gian). C++ hầu hết được thực thi dưới dạng là một ngôn ngữ biên dịch, có thể chạy trên nhiều nền tảng khác nhau như Windows, Mac OS, Linux, Ubuntu và các phiên bản Unix. Nhiều nhà cung cấp cung cấp các trình biên dịch C ++, bao gồm Tổ chức Phần mềm Tự do, Microsoft, Intel và IBM.

### **4.2. Vai trò của C/C++ trong trình giả lập OMNeT++**

OMNeT ++ là một thư viện và khung mô phỏng C ++ dựa trên thành phần, có thể mở rộng, chủ yếu để xây dựng các trình mô phỏng mạng. Mạng bao gồm các mạng truyền thông có dây và không dây, mạng trên chip, mạng xếp hàng, ... Chức năng dành riêng cho tên miền như hỗ trợ mạng cảm biến, mạng ad-hoc, mạng không

dây, giao thức Internet, mô hình hiệu suất, ... được cung cấp bởi các khung mô hình, được phát triển như các dự án độc lập. OMNeT ++ cung cấp IDE dựa trên Eclipse, môi trường thời gian chạy đồ họa và một loạt các công cụ khác. Có các phần mở rộng cho mô phỏng thời gian thực, mô phỏng mạng, tích hợp cơ sở dữ liệu, tích hợp SystemC và một số chức năng khác. OMNeT ++ được phân phối theo Giấy phép Công cộng Học thuật (Academic Public License).

#### **4.3. Lập trình C++ trong OMNeT++**

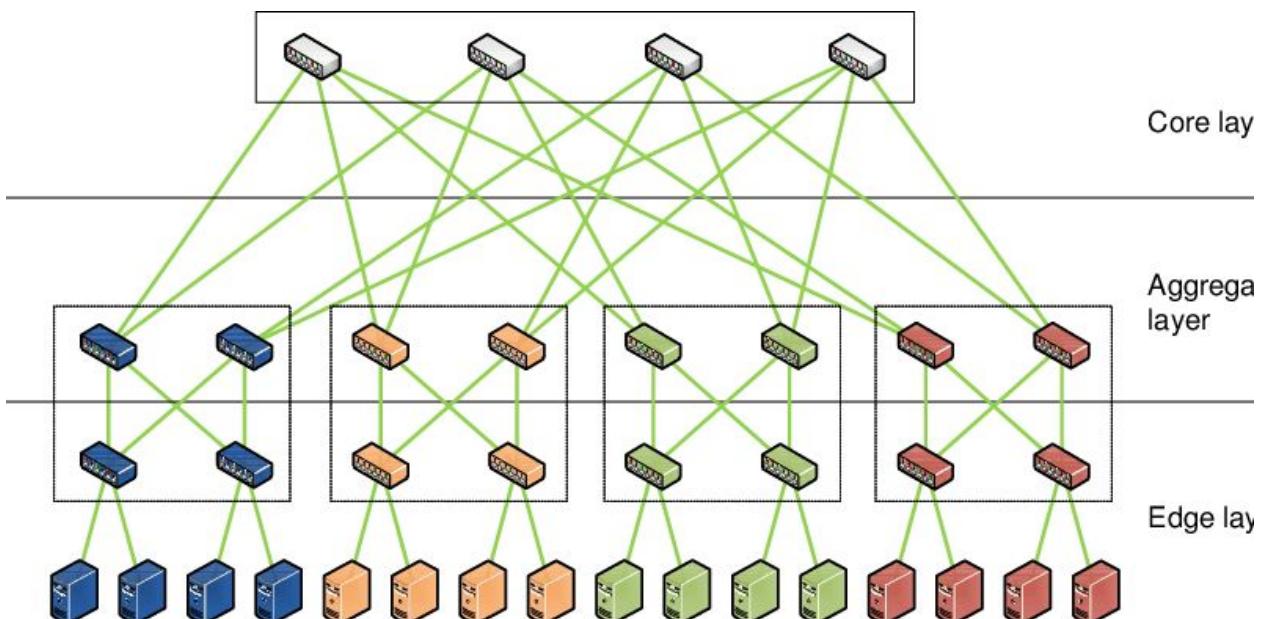
<b>STT/Nội dung</b>	<b>C++ trong OMNeT++</b>	<b>Lập trình C++</b>
1. Ép kiểu dữ liệu	double x; int (x);	double x; (int) x;
2. In ra màn hình	EV << “Print example”;	cout << “Print example”;
3. Khai báo kiểu dữ liệu số thực	float x = 0.5f;	float x = 0.5; hoặc float x = 0.5f;
4. Truy cập phần tử trong mảng. int out[10];	cGate *gate = gate("out", 3); // out[3]	out[3]; hoặc *(out+3); // out[3]

# Chương 5: Kiến trúc cơ sở mạng trung tâm dữ liệu Fat-tree

## 5.1. Mô hình mạng Fat-tree

Fat-tree là một cấu trúc mạng được sử dụng phổ biến trong các trung tâm dữ liệu, một mạng fat-tree được chia làm ba tầng khác nhau: Core, Aggregation và Edge. Hình dưới mô tả kiến trúc của mạng fat-tree với số cổng của một switch là  $k = 4$ .

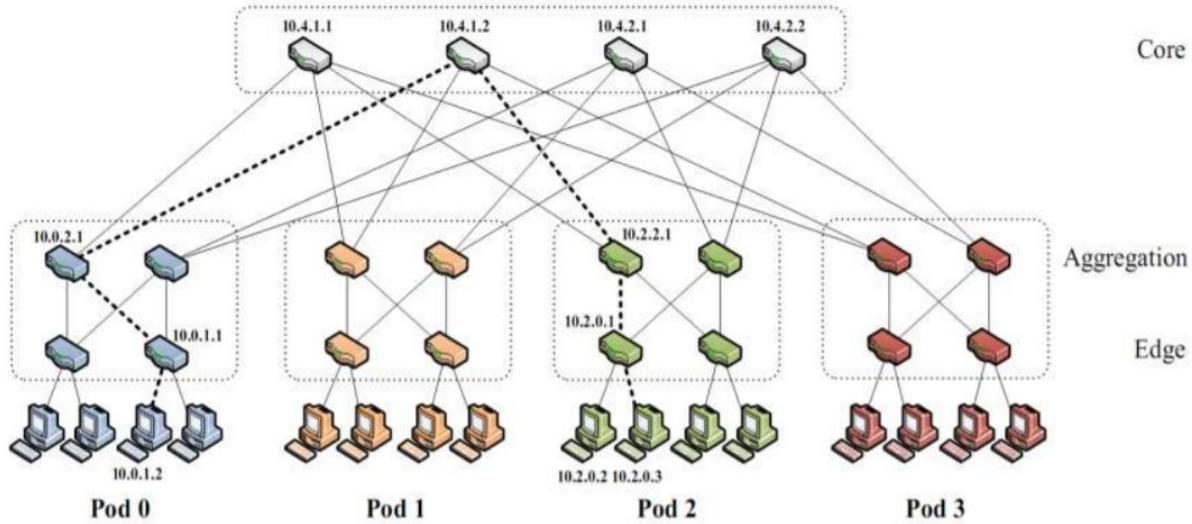
Ở kiến trúc Fat-tree, mỗi host (máy tính) kết nối với một switch. Mỗi switch ở tầng Core kết nối với  $k$  switch khác. Mỗi switch ở tầng Aggregation kết nối với  $k$  switch khác, còn mỗi switch ở tầng Edge thì kết nối với  $(k/2)$  máy tính và  $(k/2)$  switch khác. Số máy tính trong mạng Fat-tree là  $k*k*k/4$  và số switch sẽ là:  $5*k*k/4$ .



## 5.2. Xây dựng bảng định tuyến

Bảng định tuyến là bảng ghi thông tin về đường đi đến một nút đích hoặc nhóm nút đích nào đó. Ở từng switch khác nhau sẽ có một bảng định tuyến. Với Fat Tree thì các switch cùng loại (cùng là Edge Switch, Aggressive switch, hoặc core switch) sẽ có bảng định tuyến giống nhau.

Từng switch sẽ xây dựng bảng định tuyến riêng cho mình khi khởi động. Khi có gói tin đến switch đó thì gói tin được bóc tách lấy thông tin của nút nguồn và nút đích, dựa trên bảng định tuyến xây dựng sẵn thì switch sẽ tìm ra nút kế tiếp trong đường đi đã được định tuyến. Do các bảng định tuyến ở các switch là giống nhau nên tại từng nút sẽ định tuyến ra cùng một đường đi từ nút nguồn tới nút đích.



### 5.3. Công thức tính thông lượng trung bình

Công thức tính thông lượng trung bình (Throughput) của mạng theo từng khoảng thời gian (interval time) như sau:

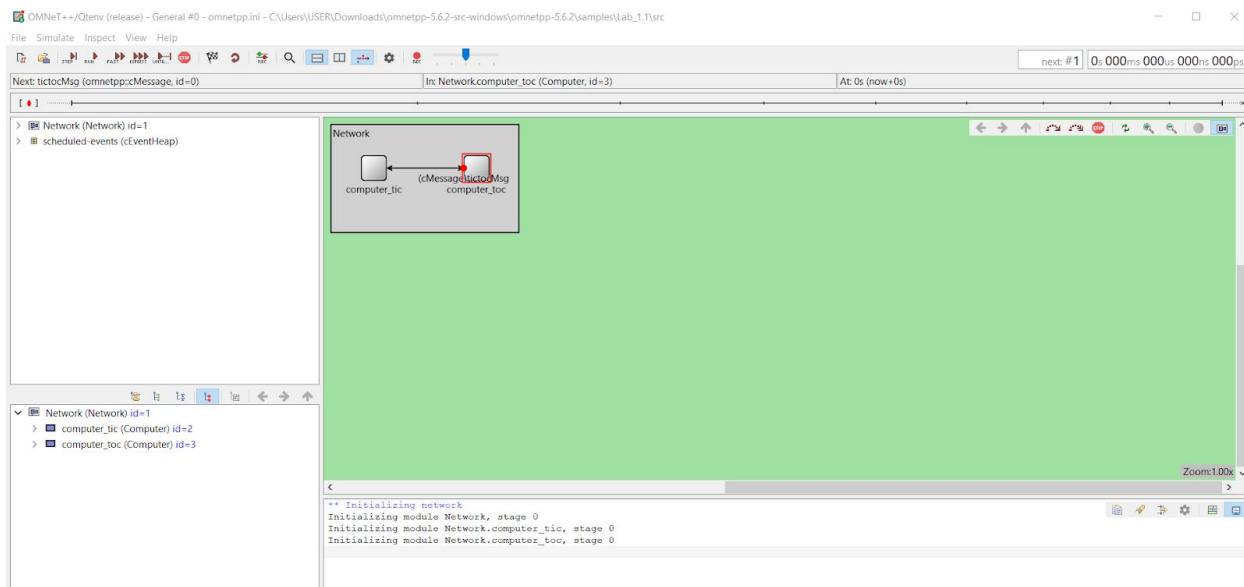
$$Throughput = \frac{Kích thước 1 gói tin * số gói tin nhận được}{Băng thông * số nút nhận * interval time} * 100\%$$

Thông lượng trung bình cho biết được hiệu quả truyền tin trong mạng.

# Chương 6: Kết quả thực nghiệm

## 6.1. Ví dụ

Đây là ví dụ cơ bản hướng dẫn sử dụng OMNeT++ để gửi gói tin giữa hai máy tính. Cụ thể chương trình sẽ được trình bày trong phần phụ lục.



## 6.2. TicToc Tutorial

“Tictoc Tutorial” là các ví dụ hướng dẫn chuẩn trên trang chủ của OMNeT++. Hướng dẫn này hướng dẫn bạn cách xây dựng và làm việc với một mô hình mô phỏng mạng, cho bạn thấy một số tính năng thường được sử dụng của OMNeT++.

Hướng dẫn dựa trên mô phỏng ví dụ Tictoc mà bạn có thể tìm thấy trong samples/tictoc của cài đặt OMNeT++, vì vậy bạn có thể thử ngay cách hoạt động của các ví dụ này. Tuy nhiên, bạn sẽ thấy hướng dẫn hữu ích hơn nhiều nếu bạn thực sự thực hiện các bước được mô tả ở đây. Bạn cần có kiến thức cơ sở về lập trình C ++ và đã quen với việc phát triển C/C ++ (chỉnh sửa tệp nguồn, biên dịch, gỡ lỗi).

Bạn có thể xem tại link sau:

<https://docs.omnetpp.org/tutorials/tictoc/>

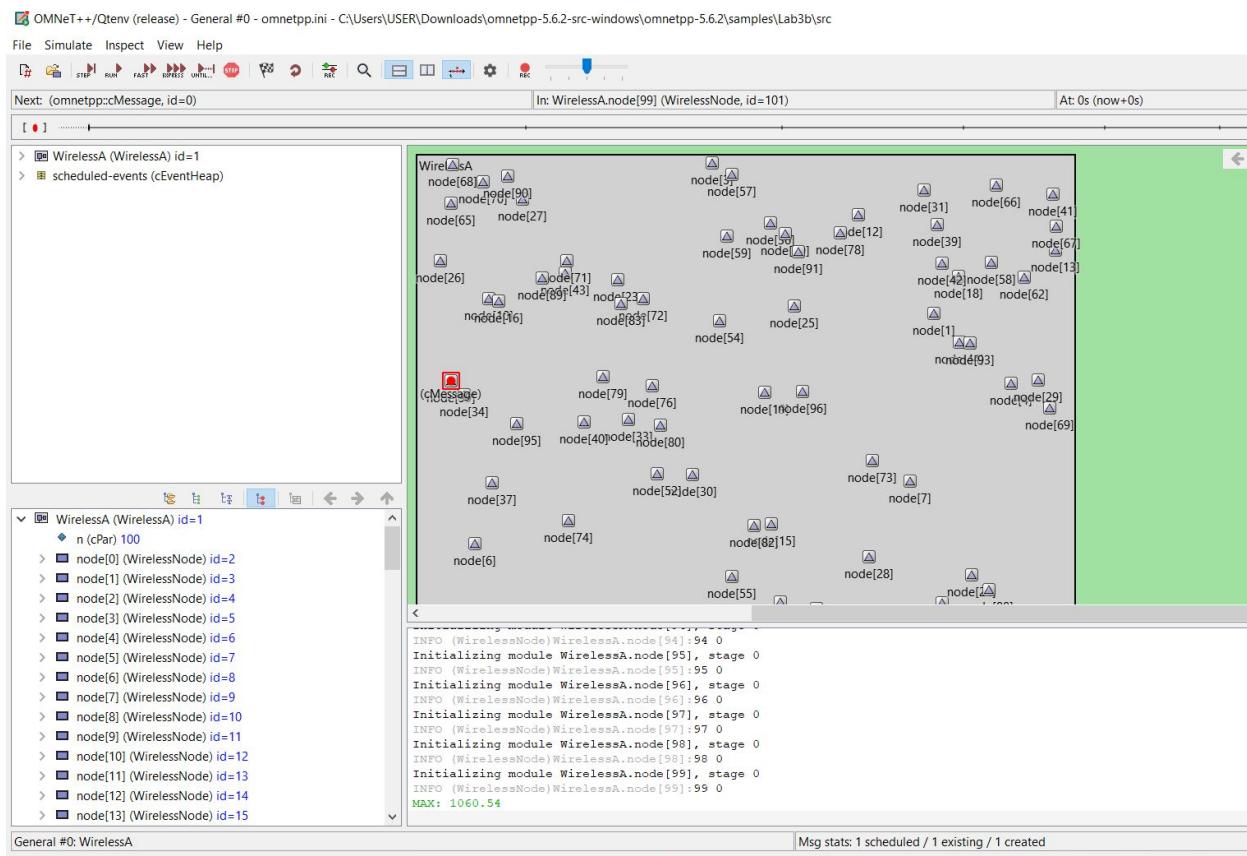
hoặc <https://doc.omnetpp.org/omnetpp4/tictoc-tutorial/>

Kết quả thực hiện: Cài đặt và chạy được tất cả các bài hướng dẫn Tictoc trong tài liệu trên. Tìm hiểu và nghiên cứu chương trình của các ví dụ để giúp hiểu sâu hơn về OMNeT++ và cách sử dụng khi áp dụng vào bài toán mới.

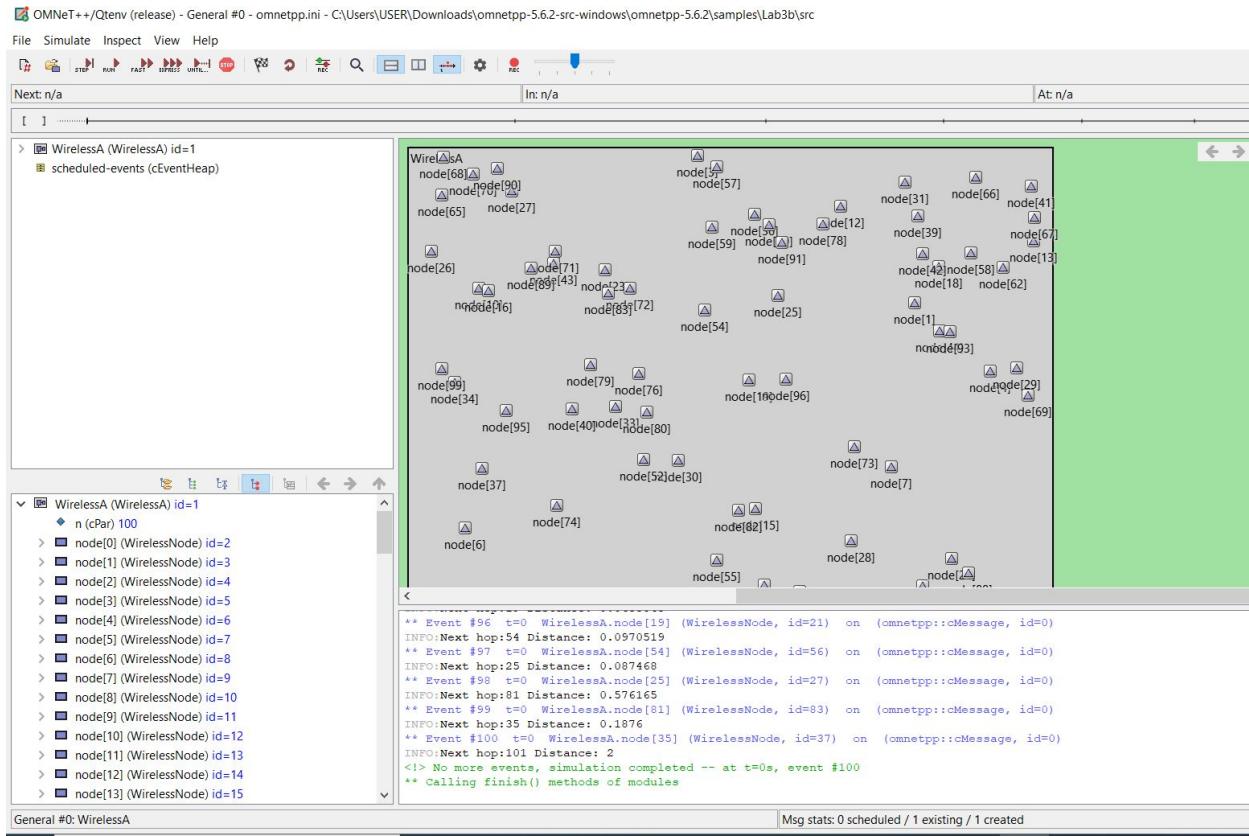
### 6.3. Mạng không dây (Wireless Sensor Networks – WSN)

Bài toán: Tạo ra 100 nút sensor trong không gian có kích thước  $8 \times 8$ , các nút mạng nằm ngẫu nhiên và trong mỗi nút ta tính toán được có khoảng cách giữa nó với một nút bất kỳ. Chương trình tìm đường đi cho gói tin, một khi gói tin đến được một nút thì nó sẽ được gửi đến nút gần nhất (mà chưa đi qua) với nút hiện tại.

Kết quả hiển thị:



## Kết quả chạy chương trình:

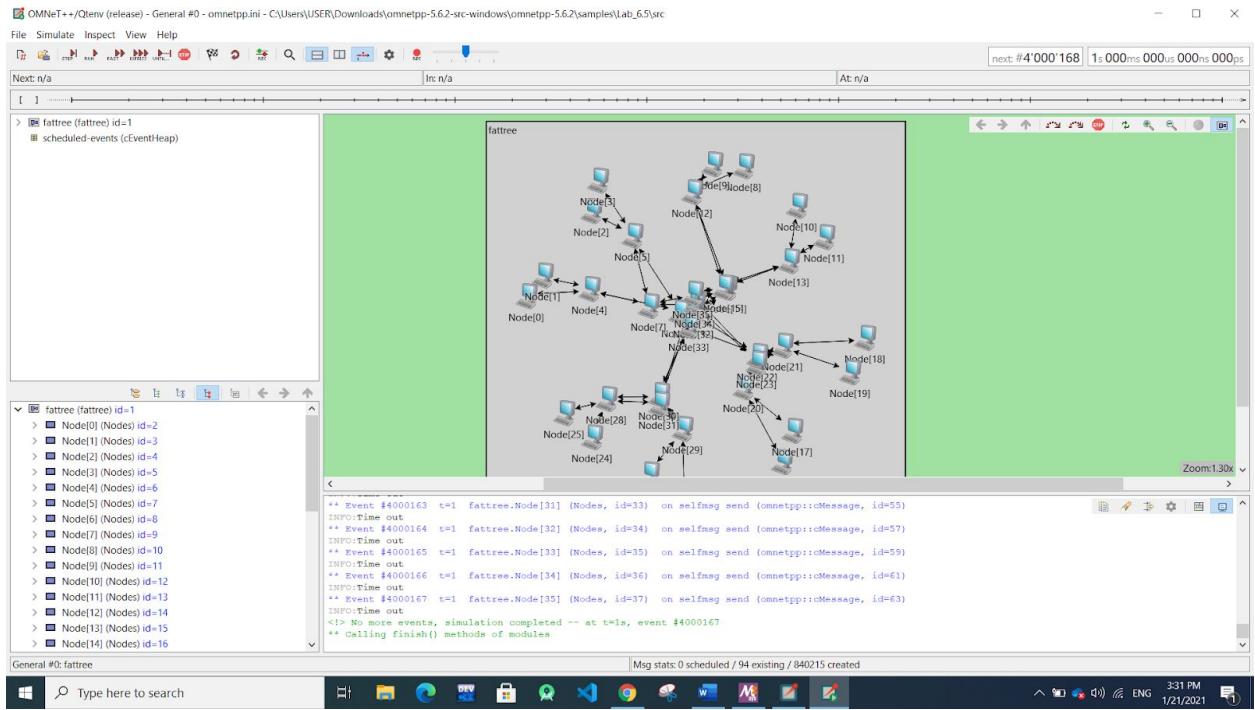


### 6.4. Fat-tree

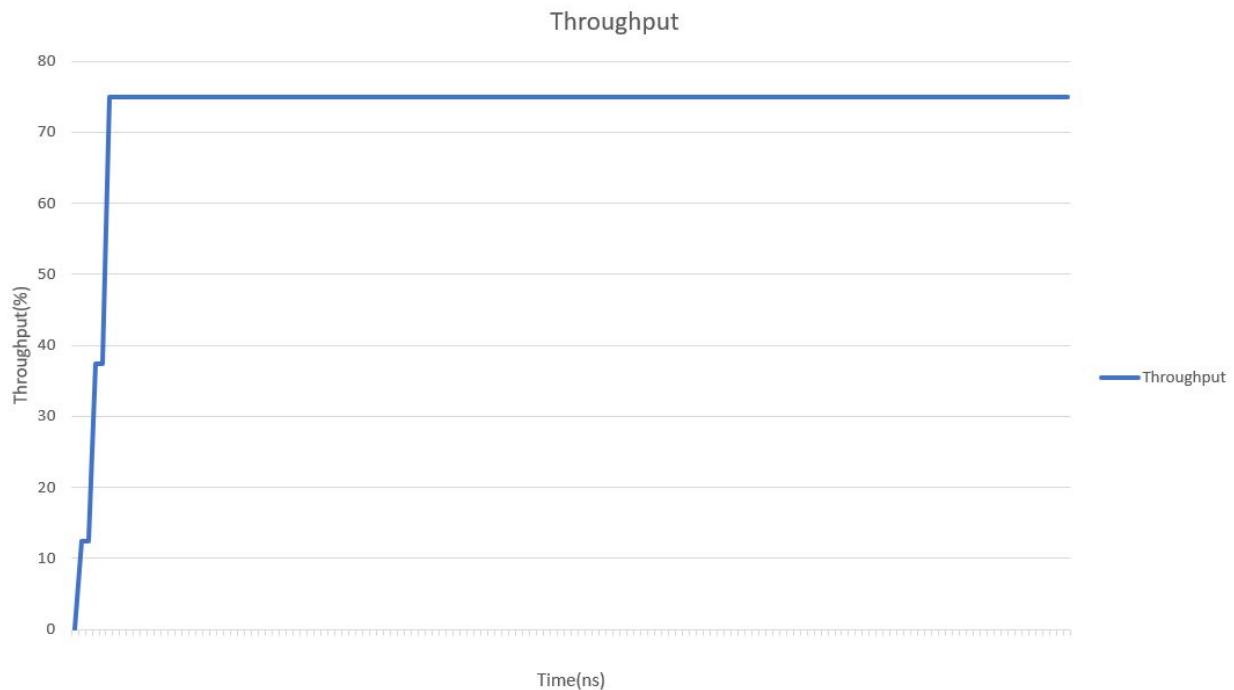
Cài đặt và chạy chương trình giả lập quá trình truyền tin trong mạng trung tâm dữ liệu Fat-tree. Hướng dẫn chi tiết được trình bày trong phần phụ lục.

Kết quả thực nghiệm:

- Hình ảnh kết quả chạy chương trình:



- Kết quả thông lượng throughput được ghi trong file output.txt, áp dụng công thức tính thông lượng trung bình ở chương 5 (chi tiết được trình bày trong phụ lục). Ta tiến hành vẽ đồ thị được kết quả như hình vẽ:



## **Chương 7: Những khó khăn gặp phải**

Trong quá trình học và chạy chương trình trên OMNeT++, em gặp phải một số khó khăn như: tiếp cận một công nghệ mới về mô phỏng mạng OMNeT++; tài liệu tiếng Việt ít, tài liệu tiếng Anh chủ yếu là các bài báo khoa học; cú pháp (syntax) ngôn ngữ NED, C++ trong OMNeT++ có nhiều điểm khác so với cú pháp C++ hay các ngôn ngữ lập trình khác; để tra cứu cú pháp hay cách sử dụng OMNeT++ thì có rất ít tài liệu tham khảo. Em đã trình bày một số điểm khác C++ trong OMNeT++ và C++ ở chương 5.

Để giải quyết vấn đề này, em được sự giải đáp từ thầy và các bạn, làm các bài Tictoc Tutorial để hiểu được cơ bản cú pháp, cách hoạt động các chương trình giả lập truyền tin trên OMNeT++. Ngoài ra, em đọc tài liệu hướng dẫn “Simulation Manual” trên trang chủ của OMNeT++, đây là tài liệu tổng hợp kiến thức khá đầy đủ về giả lập mạng trong OMNeT++. “Simulation Manual” tổng hợp lại chi tiết các nội dung lý thuyết, kiến thức nền tảng đến các ví dụ, cú pháp từng phần để xây dựng được ứng dụng mô phỏng mảng hoàn chỉnh.

## **Chương 8: Kết luận chung**

Trong quá trình làm Project 3, em được học và tiếp cận với phần mềm mô phỏng giả lập mạng OMNeT++, viết mã NED và mã nguồn C++ trong OMNeT++. Em đã tìm hiểu và chạy các chương trình ví dụ Tictoc Tutorial của OMNeT++, bài lab về quá trình gửi gói tin và tìm đường đi trong mạng không dây. Em đã cài đặt và chạy thử chương trình tính thông lượng throughput theo thời gian của quá trình truyền tin trong mạng mạng trung tâm dữ liệu Fat-tree (mã nguồn được lấy từ link Github <https://github.com/Hongbeubeu/OmnetProject2>), tìm hiểu cách ghép cặp nguồn-đích để xây dựng mạng Fat-tree được trình bày trong bài báo “A scalable commodity data center network architecture”.

Báo cáo đã trình bày sơ bộ kết quả làm việc ở trên và phần giới thiệu về OMNeT++, cách sử dụng OMNeT++ trong giả lập mạng.

## **Tài liệu tham khảo**

[1] OMNeT++ <https://omnetpp.org/>

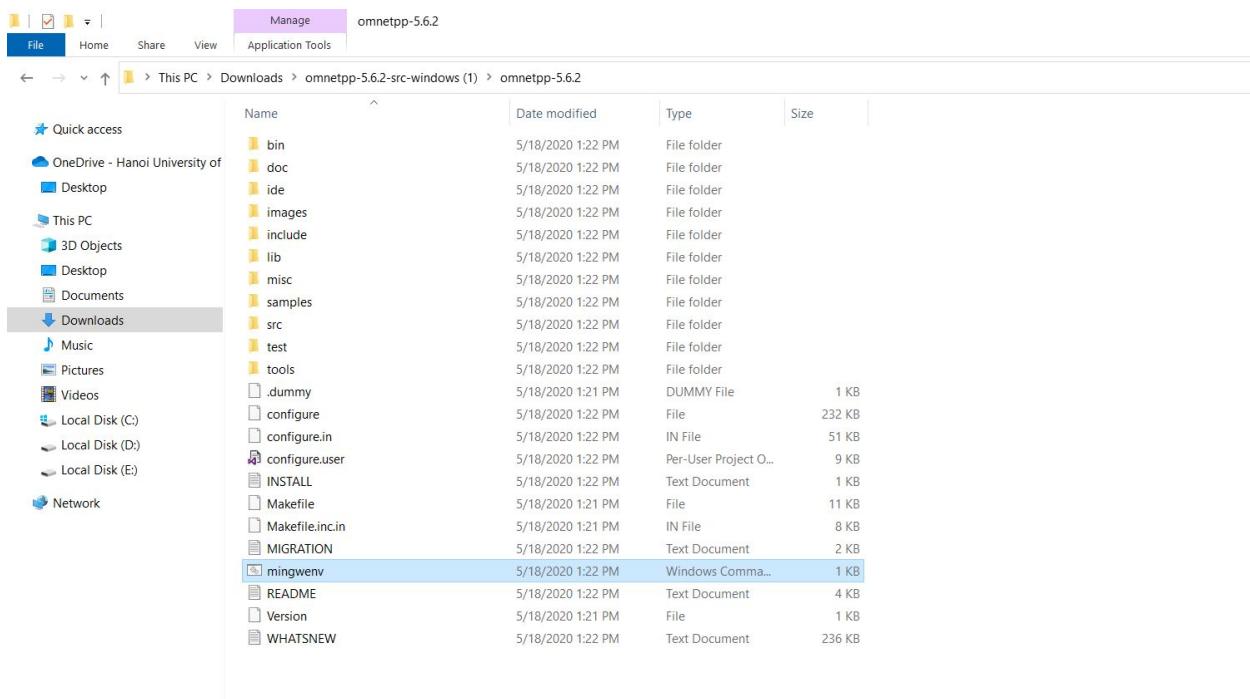
[2] Simulation Manual OMNeT++ version 5.6.1  
<https://doc.omnetpp.org/omnetpp/SimulationManual.pdf>

## Phụ lục

### A Hướng dẫn cài đặt OMNET++ trên window.

Trước tiên, bạn cần download bản OMNeT++ trên trang chủ của OMNeT++ là: <https://omnetpp.org/download/>. Sau khi download xong, ta giải nén và thực hiện các bước như sau:

Bước 1: OMNeT++ được cài đặt bằng dòng lệnh, và để sử dụng, ta chỉ cần kích đúp chuột vào file mingwenv.cmd, nhấn Enter trên bàn phím, sau đó tiếp tục thực hiện các lệnh dưới đây.



Bước 2: Kiểm tra sự toàn vẹn của tệp tin config.user bằng câu lệnh notepad config.user. Nếu như tệp tin không toàn vẹn, bạn download lại và làm từ đầu.

Bước 3: Cài đặt các thư viện và các tiến trình xử lý của chương trình. Ta chỉ cần gõ 2 lệnh sau để build

./configure

make

```

cat: /c/Users/USER/Downloads/omnetpp-5.6.2-src-windows (1)/omnetpp-5.6.2: Is a directory
cat: '(1)/omnetpp-5.6.2//Version': No such file or directory
Welcome to !

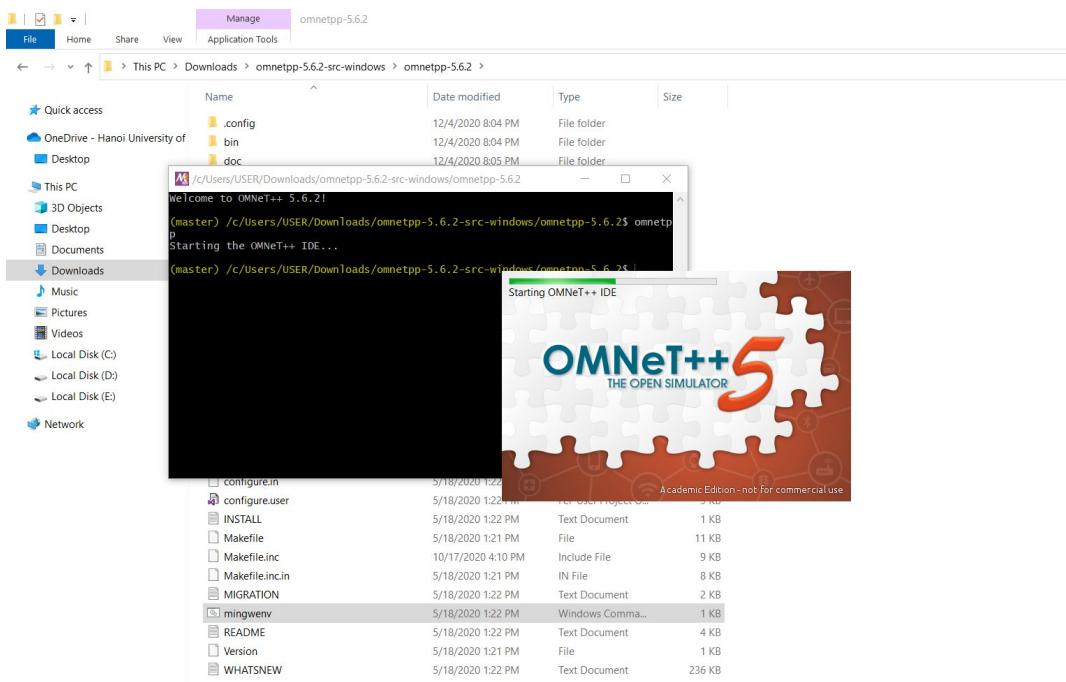
Type "./configure" and "make" to build the simulation libraries.

When done, type "omnetpp" to start the IDE.

(master) /c/Users/USER/Downloads/omnetpp-5.6.2-src-windows (1)/omnetpp-5.6.2$ ./configure
configure: loading site script /mingw64/etc/config.site
checking build system type... x86_64-w64-mingw32
checking host system type... x86_64-w64-mingw32
configure: -----
configure: reading configure.user for your custom settings
configure: -----
checking for clang... clang
checking whether the C compiler works... yes
checking for C compiler default output file name... a.exe
checking for suffix of executables... .exe
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether clang accepts -g... yes

```

Bước 4: Mở OMNeT++ IDE. Sau khi cài build các process và thư viện xong. Từ các lần sau, bạn chỉ cần vào thư mục OMNeT++, kích đúp chuột vào file mingwenv.cmd, rồi gõ omnetpp, chọn Launch để khởi động IDE.



## B Hướng dẫn cài đặt và chạy ví dụ Demo

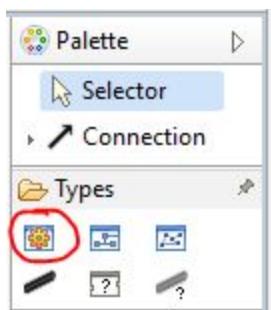
Để cài đặt demo tictoc này, ta cần cài đặt OMNeT++ và khởi động IDE của nó như trình bày ở trên, sau đó, thực hiện các bước như dưới đây.

Bước 1: Đầu tiên, ta chọn new project OMNET++ bằng cách chọn File > New > OMNeT++ Project.

Bước 2: Nhập tên, ví dụ Demo, sau đó, chọn next, chọn Empty project, sau đó chọn Finish.

Bước 3: Tạo file .ned bằng cách kích chuột phải vào project > New > Network Description File. Đổi tên, và giữ lại đuôi .ned. Bấm next, chọn empty NED file và finish.

Bước 4: Chọn simple module như hình được khoanh đỏ trong hình vẽ.



sau đó, đổi tên thành computer

Bước 5: Chọn network như hình được khoanh đỏ trong hình vẽ.



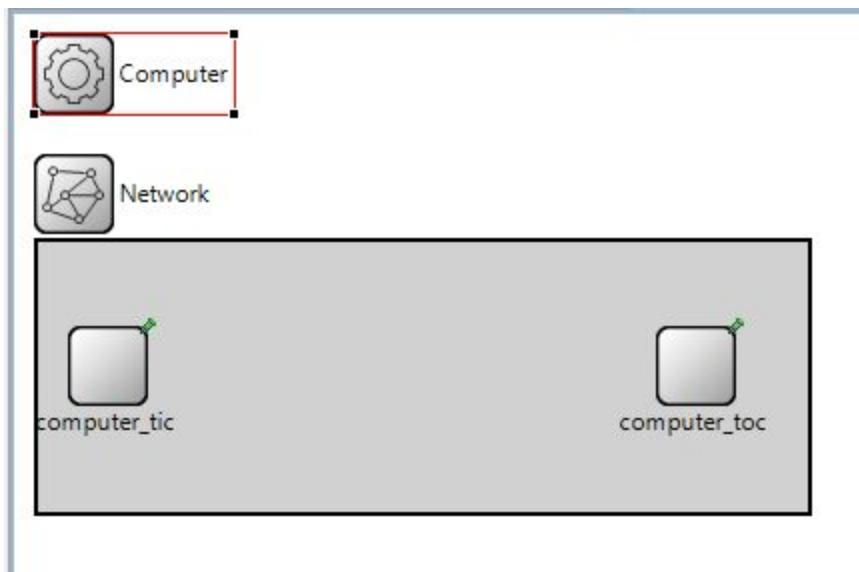
Cấu hình cho network. Chọn vào tab Source, ở trong block simple Computer, ta thêm dòng sau:

**gates:**

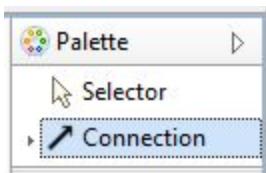
input in;

output out;

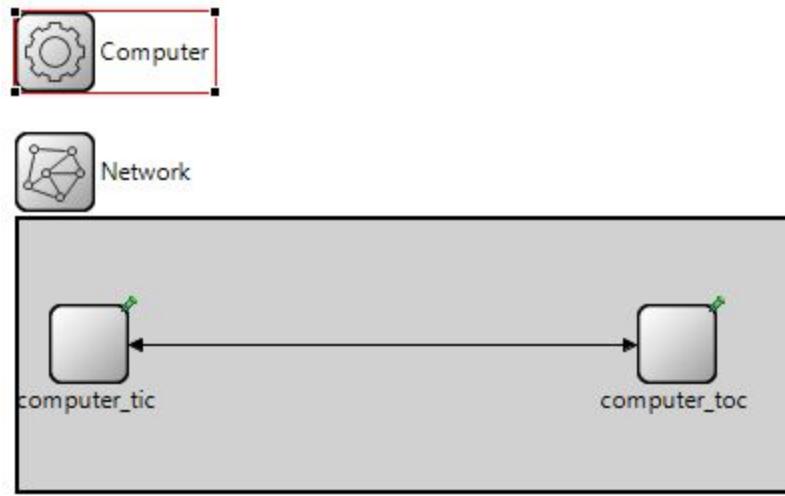
Bước 6: Ta quay lại bên tab design, sử dụng bằng cách kéo biểu computer ở ngoài vào trong network, và đổi tên, ta được như hình sau.



Sau đó, chọn connect được lựa chọn như trong hình,



với điểm bắt đầu từ  $\text{computer\_tic}$  sang  $\text{computer\_toc}$ . Và lựa chọn lại lại từ  $\text{computer\_toc}$  sang  $\text{computer\_tic}$ , ta được kết quả như hình vẽ.



Bước 7: Tạo file cc. Ta chọn project, kích chuột phải, chọn new > Source file. Nhập tên file với đuôi là .cc, ví dụ, computer.cc. Và finish. Tiếp theo, ta định nghĩa file này với hai hàm initialize() và handleMessage(cMessage \*msg) là hai hàm cơ bản của OMNeT++ với đoạn code như dưới đây.

```

include <string.h>
include <omnetpp.h>
using namespace omnetpp;

class Computer: public cSimpleModule {
protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};

Define_Module(Computer);

void Computer::initialize() {
    if (strcmp("computer_tic", getName()) == 0) {
        cMessage *msg = new cMessage("tictocMsg");
        send(msg, "out");
    }
}

```

```

    }
}

void Computer::handleMessage(cMessage *msg) {
    send(msg, "out");
}

```

Lưu ý, tên class phải giống như tên của module được định nghĩa như ở trên.

Bước 8: Tạo file init, tương tự các file trên, ta tạo ra file omnet.ini. Ta định nghĩa nội dung của file này như sau:

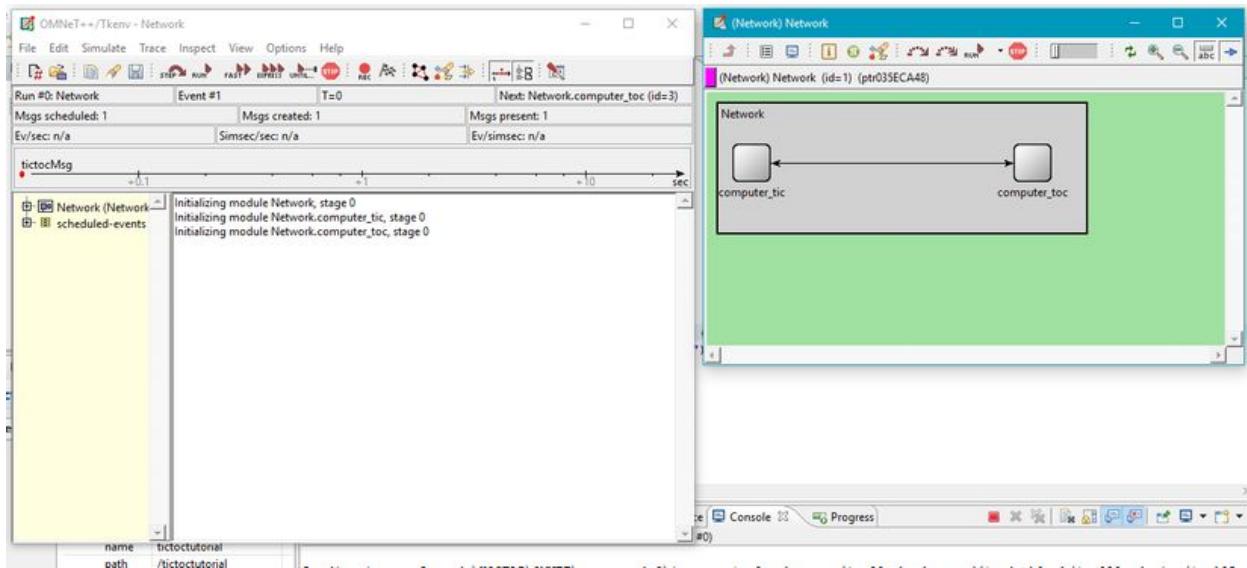
[General]

network = Network

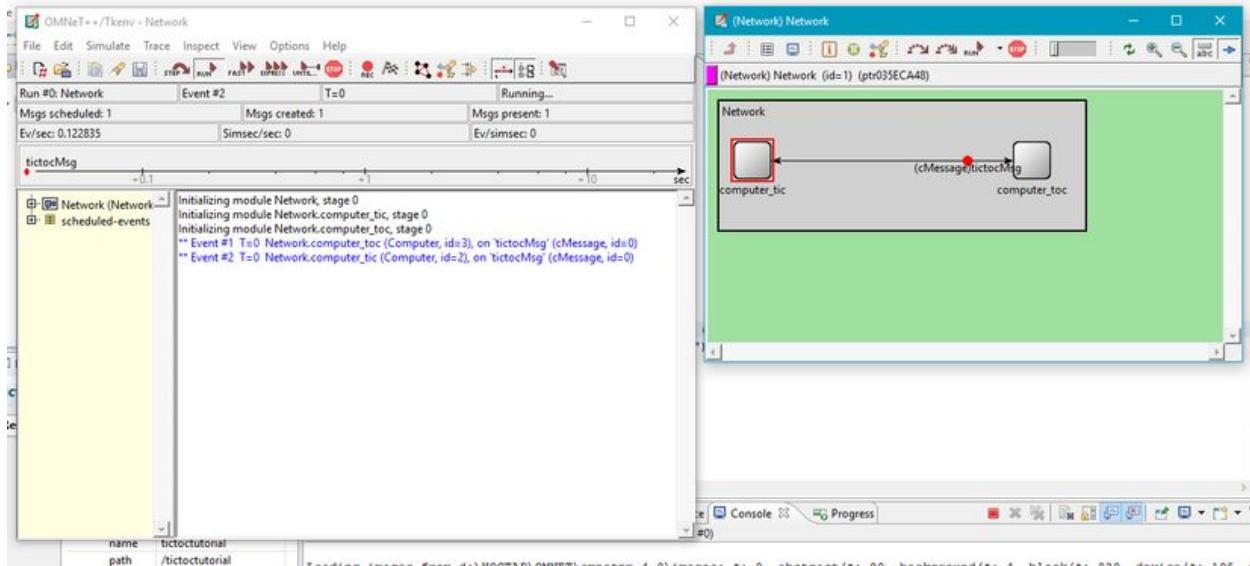
Trong đó, tên của của network được gán bằng tên của phần định nghĩa ở trên.

Bước 9: Build project. Ta chọn project, kích chuột phải, chọn build.

Bước 10: Run project. Ta chọn project, kích chuột phải, chọn run as, lựa chọn OMNeT++ Simulation. Ta có kết quả như hình dưới.



Bạn có thể thực hiện từng bước bằng cách chọn Run trên giao diện (hoặc nhấn tổ hợp Ctrl+F4) hoặc chạy liên tục bằng chọn Fast (hoặc nhấn F5). Và ta sẽ có được kết quả như sau:

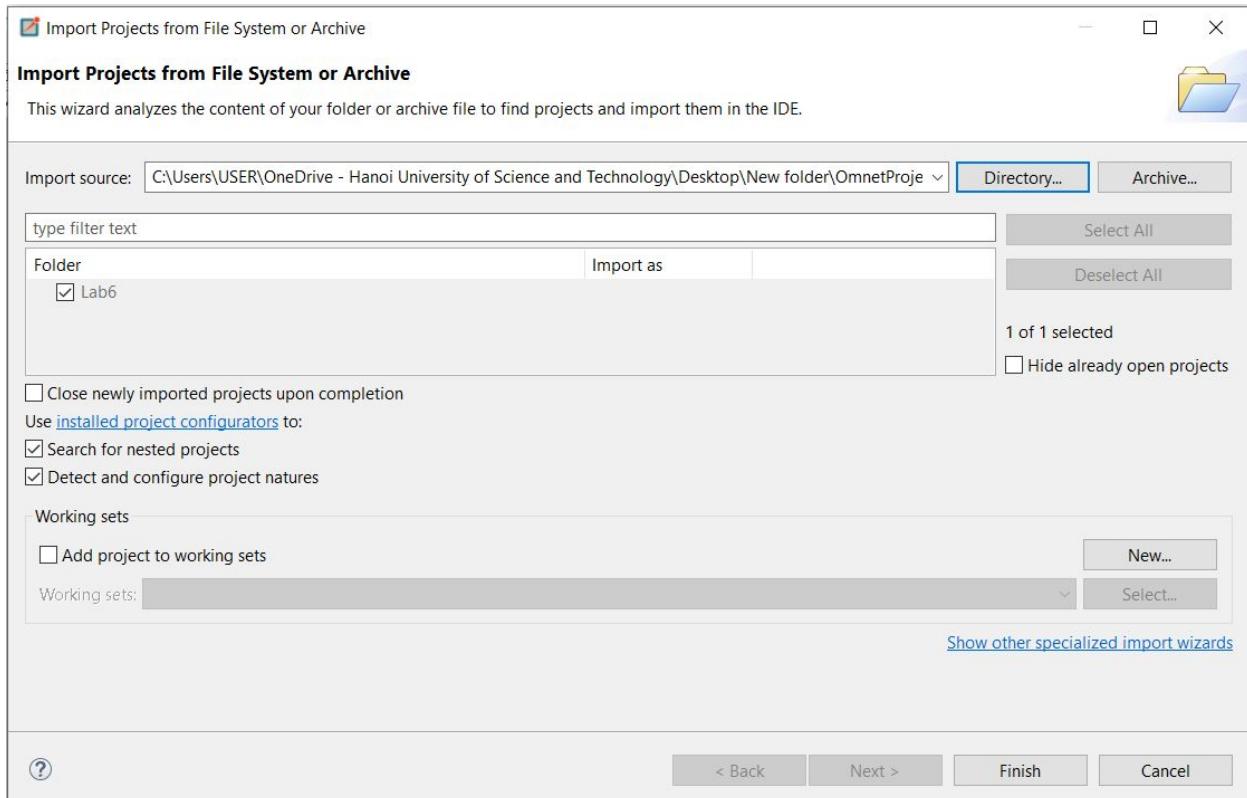


Trên đây là giới thiệu về OMNeT++ và xây dựng demo nho nhỏ bằng OMNeT++.

### **C Hướng dẫn cài đặt và chạy chương trình: “Tính thông lượng throughput theo thời gian của quá trình truyền tin trong mạng trung tâm dữ liệu (Fat-tree)**

Bước 1: Đầu tiên, bạn Download ZIP hoặc clone project tại link Github ://github.com/Hongbeubeu/OmnetProject2, sau đó tiến hành giải nén.

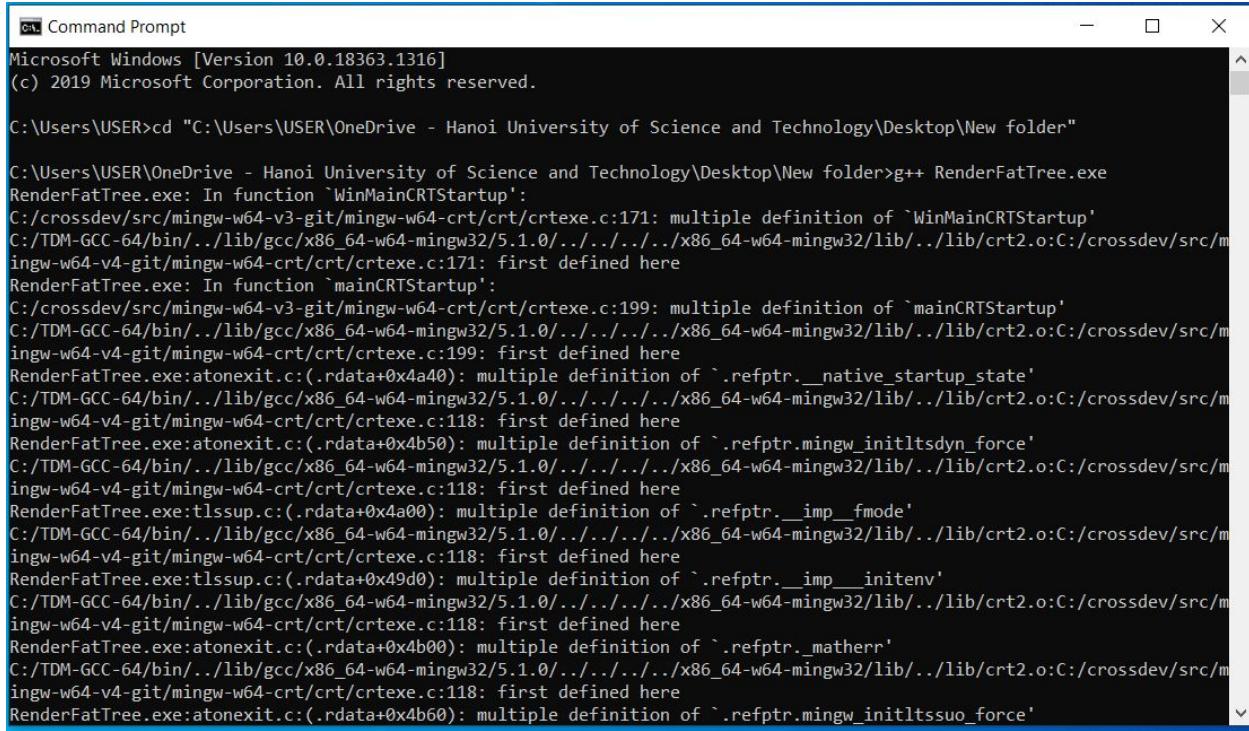
Bước 2: Import thư mục Lab6 vừa giải nén vào IDE OMNeT++ đã khởi động, File > Import Projects from File System or Archive, chọn đường dẫn thư mục vào Import source, rồi Finish.



Bước 3: Chạy file RenderFatTree để sinh cặp nguồn đích trong mạng Fat-tree (cần đổi đường dẫn file bạn muốn sinh ra mạng Fat-tree trong máy tính của bạn). Máy tính của bạn cần cài gcc hoặc g++, rồi thực hiện lệnh như sau:

- Chọn Command Prompt (mở bằng lệnh cmd trên taskbar), cd đến thư mục chứa file RenderFatTree, bạn có thể kéo thả file thư mục.
- Thực hiện lệnh: gcc RenderFatTree.exe hoặc g++ RenderFatTree.exe

Kết quả thực hiện như hình sau:



```
c:\ Command Prompt
Microsoft Windows [Version 10.0.18363.1316]
(c) 2019 Microsoft Corporation. All rights reserved.

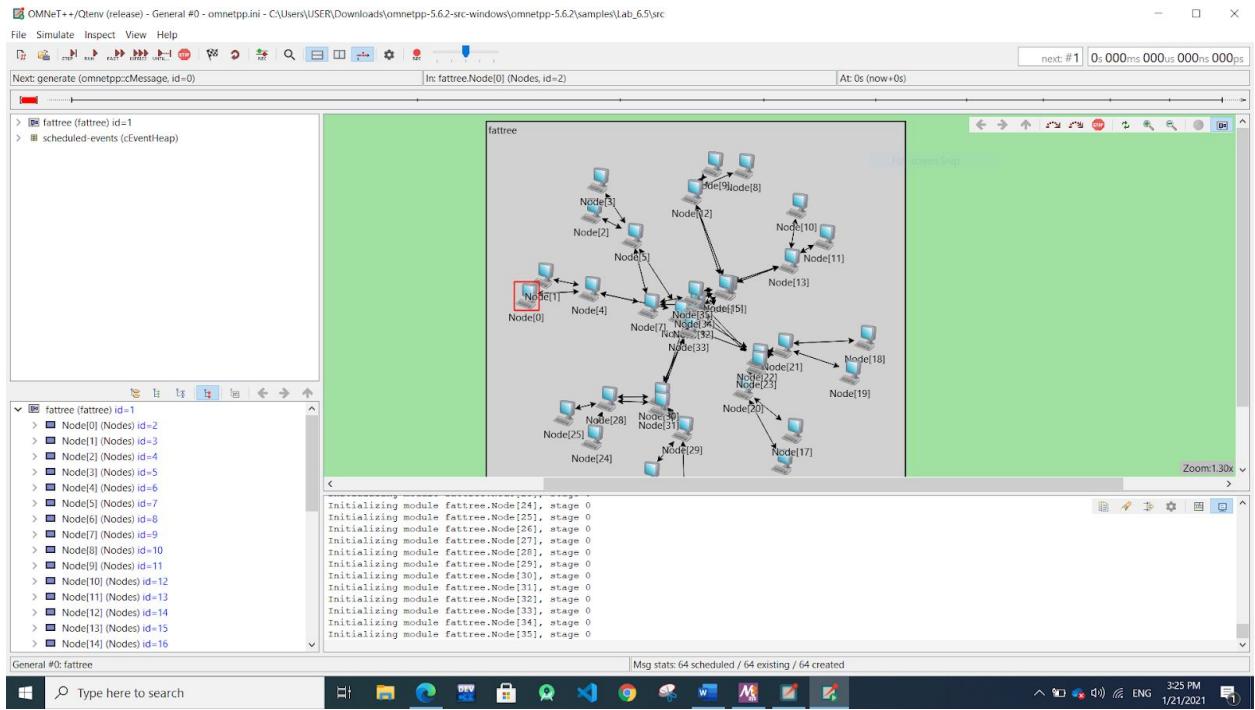
C:\Users\USER>cd "C:\Users\USER\OneDrive - Hanoi University of Science and Technology\Desktop\New folder"

C:\Users\USER\OneDrive - Hanoi University of Science and Technology\Desktop\New folder>g++ RenderFatTree.exe
RenderFatTree.exe: In function `WinMainCRTStartup':
C:/crossdev/src/mingw-w64-v3-git/mingw-w64-crt/crt/crtexe.c:171: multiple definition of `WinMainCRTStartup'
C:/TDM-GCC-64/bin/../../lib/gcc/x86_64-w64-mingw32/5.1.0/../../../../x86_64-w64-mingw32/lib/../../crt2.o:C:/crossdev/src/mingw-w64-v4-git/mingw-w64-crt/crt/crtexe.c:171: first defined here
RenderFatTree.exe: In function `mainCRTStartup':
C:/crossdev/src/mingw-w64-v3-git/mingw-w64-crt/crt/crtexe.c:199: multiple definition of `mainCRTStartup'
C:/TDM-GCC-64/bin/../../lib/gcc/x86_64-w64-mingw32/5.1.0/../../../../x86_64-w64-mingw32/lib/../../crt2.o:C:/crossdev/src/mingw-w64-v4-git/mingw-w64-crt/crt/crtexe.c:199: first defined here
RenderFatTree.exe:atонexit.c:(.rdata+0x4a0): multiple definition of `__refptr.__native_startup_state'
C:/TDM-GCC-64/bin/../../lib/gcc/x86_64-w64-mingw32/5.1.0/../../../../x86_64-w64-mingw32/lib/../../crt2.o:C:/crossdev/src/mingw-w64-v4-git/mingw-w64-crt/crt/crtexe.c:118: first defined here
RenderFatTree.exe:atонexit.c:(.rdata+0x4b50): multiple definition of `__refptr.mingw_initltsdyn_force'
C:/TDM-GCC-64/bin/../../lib/gcc/x86_64-w64-mingw32/5.1.0/../../../../x86_64-w64-mingw32/lib/../../crt2.o:C:/crossdev/src/mingw-w64-v4-git/mingw-w64-crt/crt/crtexe.c:118: first defined here
RenderFatTree.exe:tlssup.c:(.rdata+0x49d0): multiple definition of `__refptr.__imp__initenv'
C:/TDM-GCC-64/bin/../../lib/gcc/x86_64-w64-mingw32/5.1.0/../../../../x86_64-w64-mingw32/lib/../../crt2.o:C:/crossdev/src/mingw-w64-v4-git/mingw-w64-crt/crt/crtexe.c:118: first defined here
RenderFatTree.exe:atонexit.c:(.rdata+0x4b00): multiple definition of `__refptr._matherr'
C:/TDM-GCC-64/bin/../../lib/gcc/x86_64-w64-mingw32/5.1.0/../../../../x86_64-w64-mingw32/lib/../../crt2.o:C:/crossdev/src/mingw-w64-v4-git/mingw-w64-crt/crt/crtexe.c:118: first defined here
RenderFatTree.exe:atонexit.c:(.rdata+0x4b60): multiple definition of `__refptr.mingw_initltssuo_force'
```

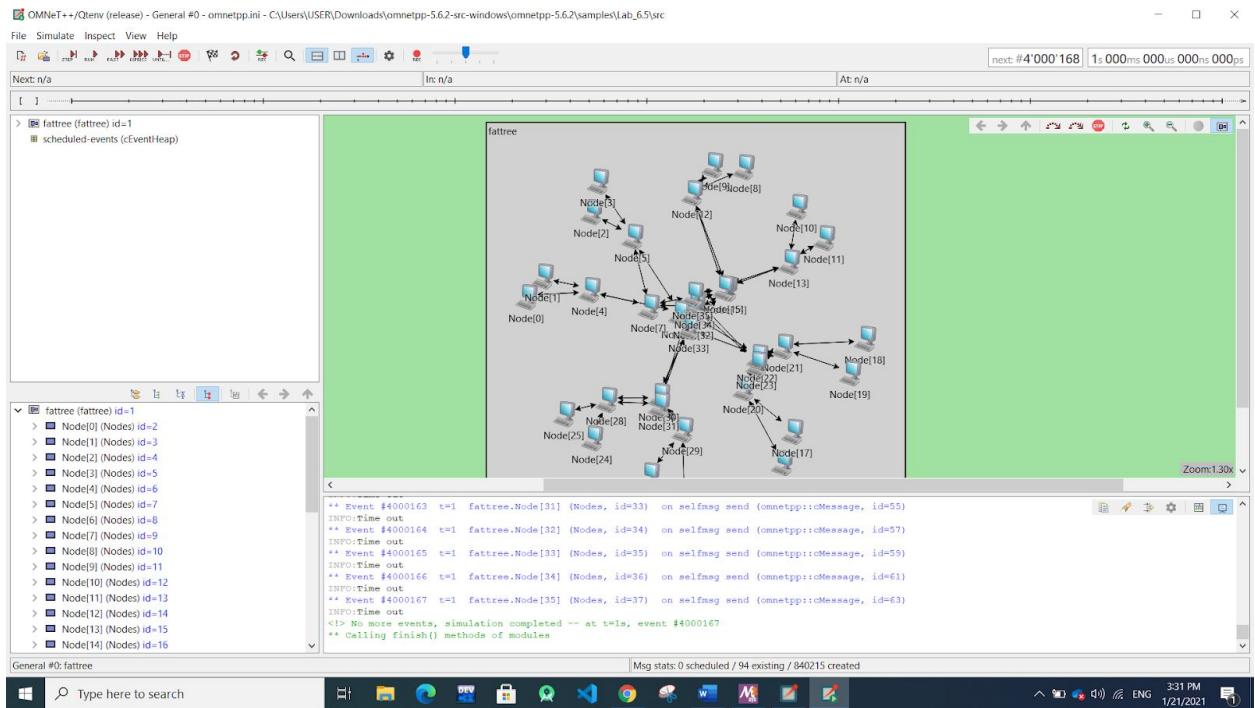
Bước 4: Đổi đường dẫn file output.txt bạn muốn ghi file kết quả output.txt trong file lab6.5.cc

Bước 5: Chạy Project. Kích chuột phải vào file omnetpp.ini > Run As > OMNeT++ Simulation.

Kết quả chạy được hiện ra màn hình:



Bước 6: Chọn Run để chạy Project và chờ khi kết thúc thì tắt giao diện hiển thị, kết quả thông lượng throughput đã được ghi trong file output.txt bạn đã chọn.



Kết quả trong file output.txt:

Theo chương 5, có công thức tính thông lượng trung bình (Throughput) của mạng theo từng khoảng thời gian (interval time) như sau:

$$Throughput = \frac{\text{Kích thước 1 gói tin} * \text{số gói tin nhận được}}{\text{Băng thông} * \text{số nút nhận} * \text{interval time}} * 100\%$$

Trong đó:

- + Kích thước 1 gói tin = 100 Kb ~ 100000 bit
  - + Số gói tin nhận được = n (kết quả trong file output.txt)
  - + Băng thông = 1 Gbps ~ 1 tỷ bit/s
  - + Số nút nhận = 8
  - + interval time = 0.1 ms

## Bảng kết quả:

n	0	1	3	6
Throughput (%)	0	12.5	37.5	75

Kết quả, tính thông lượng throughput theo thời gian (interval time) của quá trình truyền tin trong Fat-tree được biểu diễn đồ thị như hình vẽ.

