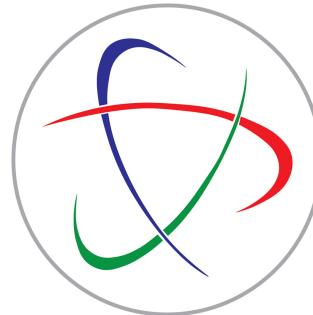


TRƯỜNG ĐẠI HỌC BÁCH KHOA - ĐẠI HỌC ĐÀ NẴNG
KHOA ĐIỆN TỬ - VIỄN THÔNG

o0o



LẬP TRÌNH MẠNG
BÁO CÁO CUỐI KỲ

Đề tài:

PROPAGATION LOSS MODEL

Giảng viên hướng dẫn : TS. Nguyễn Văn Hiếu

Sinh viên thực hiện : Nguyễn Ngọc Trung

21KTMT2

Lê Dương Khang

21KTMT2

Hoàng Bảo Long

21KTMT1

Lớp học phần : 21.44

Đà Nẵng, tháng 11 năm 2024

LỜI NÓI ĐẦU

Trong thời đại hiện nay, nhu cầu về nghiên cứu và phát triển các mô hình truyền sóng nhằm dự đoán và cải thiện chất lượng tín hiệu đang trở nên quan trọng hơn bao giờ hết. Đặc biệt, với sự phát triển nhanh chóng của các hệ thống mạng không dây và viễn thông, các yếu tố ảnh hưởng đến suy hao tín hiệu như khoảng cách, chướng ngại vật và môi trường truyền sóng đã đặt ra những thách thức lớn cho các kỹ sư và nhà nghiên cứu. Để đảm bảo chất lượng truyền thông và tối ưu hóa hệ thống, việc mô phỏng và đánh giá các mô hình truyền sóng trở thành một yêu cầu thiết yếu.

Đề tài của chúng em tập trung vào việc cài đặt và thực thi các mô hình suy hao truyền sóng trong công cụ mô phỏng mạng NS3. Với công cụ này, nhóm chúng em sẽ mô phỏng nhiều mô hình truyền sóng khác nhau, bao gồm các mô hình suy hao theo khoảng cách, mô hình ngẫu nhiên và các mô hình fading để kiểm tra khả năng truyền tín hiệu trong các điều kiện môi trường khác nhau. Dự án không chỉ giúp chúng em hiểu rõ hơn về bản chất suy hao tín hiệu mà còn giúp cung cấp cơ sở để tối ưu hóa và nâng cao hiệu quả truyền thông không dây.

Nhóm chúng em gồm ba thành viên: **Nguyễn Ngọc Trung, Lê Dương Khang, và Hoàng Bảo Long**. Báo cáo này ghi lại toàn bộ quá trình nghiên cứu, cài đặt và đánh giá các mô hình suy hao truyền sóng. Chúng em hy vọng rằng những kiến thức thu được qua đề tài này sẽ hữu ích cho các ứng dụng trong lĩnh vực truyền thông, đồng thời góp phần hỗ trợ cho các nghiên cứu và phát triển hệ thống mạng không dây trong tương lai.

Cuối cùng, chúng em xin chân thành cảm ơn thầy **Nguyễn Văn Hiếu** đã dành thời gian quan tâm và đóng góp ý kiến để giúp chúng em hoàn thiện báo cáo này. Xin đặc biệt gửi lời cảm ơn đến thầy hướng dẫn đã luôn tận tình hỗ trợ, hướng dẫn chúng em trong suốt quá trình thực hiện đề tài.

Một lần nữa, xin chân thành cảm ơn!

Sinh viên thực hiện,

Nguyễn Ngọc Trung
Lê Dương Khang
Hoàng Bảo Long

MỤC LỤC

I. Cài đặt và thực thi mô hình suy hao truyền sóng sử dụng NS3	4
1. Giới thiệu	4
1.1 Giới thiệu về NS3.....	4
1.2 Khái niệm về mô hình suy hao truyền sóng.....	4
1.3 Mục tiêu của báo cáo	5
2. Cài đặt và thiết lập hệ thống	6
2.1 Cài đặt NS3 và các công cụ hỗ trợ.....	6
2.2 Cấu hình mô phỏng.....	9
2.3 Lưu ý về cấu hình.....	11
3. Mô hình suy hao truyền sóng.....	11
3.1 Friis Propagation Loss Model.....	11
3.2 Log-Distance Propagation Loss Model	12
3.3 Random Propagation Loss Model với phân phối Exponential	13
3.4 Jakes Propagation Loss Model	13
3.5 Three-Log-Distance Propagation Loss Model.....	13
3.6 Nakagami Propagation Loss Model	14
3.7 Kết hợp Three-Log-Distance Propagation Loss Model và Nakagami Propagation Loss Model.....	15
4. Phân tích mã nguồn	15
4.1 Hàm TestDeterministic.....	15
4.2 Hàm TestProbabilistic	17
4.3 Hàm TestDeterministicByTime	19
4.4 Hàm main	21
4.5 Quy trình hoạt động của mã nguồn.....	25
4.6 Các tham số và thiết lập quan trọng.....	26
5. Kết quả và phân tích.....	26
5.1 Kết quả của Friis Propagation Loss Model	27
5.2 Kết quả của Log-Distance Propagation Loss Model.....	27
5.3 Kết quả của Random Propagation Loss Model với phân phối Exponential.....	28
5.4 Kết quả của Jakes Propagation Loss Model.....	29
5.5 Kết quả của Three-Log-Distance Propagation Loss Model	30
5.6 Kết quả của Nakagami Propagation Loss Model.....	31
5.7 Kết quả kết hợp Three-Log-Distance Loss Model và Nakagami Propagation Loss Model	31
6. Kết luận.....	32

II. Sử dụng NetAnim để mô phỏng một kiến trúc mạng (Network Topology).....	34
1. Giới thiệu	34
1.1. Giới thiệu về NetAnim	34
1.2. Giới thiệu về dự án mô phỏng.....	34
2. Cơ sở lý thuyết.....	35
2.1. Giới thiệu về P2P.....	35
2.2. Giới thiệu về mạng LAN.....	35
2.3. Giới thiệu về CSMA	35
2.4. Quy trình khởi tạo topology trong NS3	35
3. Các bước cài đặt và tiến hành bài mô phỏng.....	37
3.1. Các bước cài đặt NetAnim	37
3.2. Giải thích chương trình second.cc	39
3.3. Các bước tiến hành chạy mô phỏng	48
4. Kết quả mô phỏng và nhận xét	51
4.1. Kết quả mô phỏng.....	51
4.2. Nhận xét	53
5. Kết luận.....	56
III. Phát triển tính năng mới cho dự án: Mô phỏng Nhiều Gaussian và Fading Rayleigh trong mô hình truyền sóng bị suy hao do khoảng cách	58
1. Giới thiệu	58
2. Mục tiêu tính năng mới	58
3. Phương pháp triển khai.....	59
4. Kết quả.....	60
5. Đánh giá và nhận xét	64
5.1 Ưu điểm của tính năng mới.....	64
5.2 Hạn chế	65
6. Kết luận.....	66

I. Cài đặt và thực thi mô hình suy hao truyền sóng sử dụng NS3

1. Giới thiệu

Trong lĩnh vực truyền thông và mạng máy tính, một trong những thách thức lớn nhất là mô phỏng và dự đoán chính xác các hiện tượng ảnh hưởng đến tín hiệu khi nó truyền qua các môi trường khác nhau. Tín hiệu khi được phát từ một nguồn đến đích sẽ chịu ảnh hưởng bởi nhiều yếu tố môi trường như khoảng cách, chướng ngại vật, và tần số sóng. Hiểu và dự đoán sự suy hao tín hiệu là rất quan trọng trong việc thiết kế và tối ưu hóa các hệ thống truyền thông, đặc biệt là mạng không dây.

1.1 Giới thiệu về NS3

NS3 (Network Simulator 3) là một công cụ mô phỏng mạng mã nguồn mở mạnh mẽ, được sử dụng phổ biến trong nghiên cứu và giảng dạy về mạng máy tính. NS3 cho phép mô phỏng các giao thức và mô hình mạng từ cấp độ ứng dụng đến các mô hình truyền sóng, giúp người dùng kiểm thử và đánh giá hiệu suất của mạng trong các điều kiện khác nhau mà không cần triển khai thực tế. Một trong những tính năng quan trọng của NS3 là khả năng mô phỏng sự suy hao tín hiệu qua các mô hình suy hao truyền sóng khác nhau, cung cấp cho người dùng các công cụ để nghiên cứu và tối ưu hóa hệ thống mạng không dây.

1.2 Khái niệm về mô hình suy hao truyền sóng

Suy hao truyền sóng (Propagation Loss) là hiện tượng tín hiệu bị suy hao khi truyền qua không gian, đặc biệt là trong môi trường không dây. Để dự đoán mức độ suy hao này, các mô hình suy hao truyền sóng được phát triển nhằm tính toán ảnh hưởng của khoảng cách và các yếu tố môi trường đến công suất tín hiệu nhận được. Trong dự án này, các mô hình suy hao truyền sóng được mô phỏng bao gồm:

- **Friis Propagation Loss Model:**

Mô hình Friis sử dụng công thức Friis để tính toán suy hao tín hiệu trong môi trường không gian tự do, nơi tín hiệu không chịu ảnh hưởng của các yếu tố môi trường. Suy hao tín hiệu phụ thuộc vào khoảng cách giữa máy phát và máy thu theo quy luật nghịch đảo bình phương. Mô hình này phù hợp cho các môi trường lý tưởng hoặc các trường hợp trong phòng thí nghiệm.

- **Log-Distance Propagation Loss Model:**

Mô hình này phù hợp cho môi trường thực tế, khi suy hao tín hiệu không chỉ phụ thuộc vào khoảng cách mà còn chịu ảnh hưởng của các chướng ngại vật và môi trường xung quanh. Mô hình Log-Distance sử dụng một hàm mũ để tính toán mức độ suy hao, với hệ số suy hao có thể điều chỉnh để phù hợp với môi trường. Trong dự án, hệ số suy hao được đặt là 2.5.

- **Random Propagation Loss Model với phân phối Exponential:**

Mô hình Random sử dụng biến ngẫu nhiên để mô phỏng sự biến động của tín hiệu trong môi trường theo phân phối mũ, giúp mô phỏng các tình huống mà tín hiệu chịu nhiều yếu tố ngẫu nhiên, chẳng hạn như trong các môi trường có nhiều nguồn gây nhiễu hoặc các điều kiện thay đổi liên tục.

- **Jakes Propagation Loss Model:**

Mô hình Jakes là một mô hình phức tạp hơn, mô phỏng hiệu ứng Doppler, đặc biệt là trong các kênh di động. Hiệu ứng Doppler xảy ra khi nguồn phát và nguồn thu di chuyển tương đối với nhau, dẫn đến sự thay đổi về tần số và cường độ của tín hiệu nhận được. Trong dự án, tần số Doppler được thiết lập cho tín hiệu có tần số 5.15 GHz với tốc độ di chuyển 100 km/h, và mô hình được mô phỏng với hai độ phân giải thời gian khác nhau: 1 ms và 0.1 ms.

- **Three-Log-Distance Propagation Loss Model:**

Đây là một biến thể của mô hình Log-Distance, trong đó khoảng cách được chia thành ba vùng khác nhau, mỗi vùng có một hệ số suy hao riêng. Điều này cho phép mô phỏng các điều kiện môi trường thay đổi theo khoảng cách. Trong dự án, các giá trị hệ số suy hao mặc định được sử dụng, cùng với một cấu hình khác với các hệ số suy hao là 1.0, 3.0, và 10.0.

- **Nakagami Propagation Loss Model:**

Mô hình Nakagami là một mô hình xác suất, thường được sử dụng để mô phỏng các kênh fading trong các môi trường đa đường, nơi mà tín hiệu có thể đến nguồn thu từ nhiều hướng và có thể bị cản trở bởi các vật thể khác nhau. Mô hình này giúp mô phỏng hiện tượng fading của tín hiệu, đặc biệt hữu ích trong các môi trường đô thị hoặc các khu vực có nhiều chướng ngại vật.

- **Kết hợp Three-Log-Distance Loss Model và Nakagami Propagation Loss Model:**

Đây là mô hình kết hợp giữa Three-Log-Distance và Nakagami, cho phép mô phỏng đồng thời cả sự suy hao tín hiệu theo khoảng cách và hiện tượng fading. Mô hình này tạo ra một môi trường mô phỏng thực tế hơn, đặc biệt hữu ích cho các ứng dụng trong mạng không dây phức tạp, nơi tín hiệu phải vượt qua nhiều chướng ngại vật và chịu ảnh hưởng của fading.

1.3 Mục tiêu của báo cáo

Mục tiêu của báo cáo này là trình bày quá trình cài đặt, mô phỏng và đánh giá các mô hình suy hao truyền sóng trong NS3, bao gồm các mô hình Friis, Log-Distance, Random với phân phối Exponential, Jakes, Three-Log-Distance, Nakagami, và mô hình kết hợp Three-Log-Distance và Nakagami. Các mục tiêu chính của báo cáo bao gồm:

1. Cài đặt và cấu hình môi trường NS3 để chạy các mô hình truyền sóng.
2. Mô phỏng từng mô hình và phân tích kết quả.

3. So sánh mức độ suy hao tín hiệu của từng mô hình theo các khoảng cách và điều kiện môi trường khác nhau.
4. Đưa ra nhận xét về ứng dụng của từng mô hình trong thực tế và điều kiện môi trường phù hợp.

Báo cáo sẽ cung cấp cái nhìn sâu sắc về sự suy hao tín hiệu trong các môi trường khác nhau, giúp đưa ra các phương án thiết kế và tối ưu hóa hệ thống mạng không dây hiệu quả hơn.

2. Cài đặt và thiết lập hệ thống

Để thực hiện mô phỏng các mô hình suy hao truyền sóng trong NS3, ta cần thiết lập môi trường mô phỏng và cài đặt các phần mềm cần thiết. Phần này sẽ mô tả quá trình cài đặt NS3, các công cụ hỗ trợ và cấu hình ban đầu để chuẩn bị cho việc mô phỏng.

2.1 Cài đặt NS3 và các công cụ hỗ trợ

NS3 (Network Simulator 3) yêu cầu một số công cụ và thư viện hỗ trợ để hoạt động tốt. Các bước cài đặt cụ thể như sau:

Cập nhật hệ thống: Trước khi cài đặt các phần mềm cần thiết, ta tiến hành cập nhật hệ thống để đảm bảo rằng các gói phần mềm mới nhất được cài đặt:

sudo apt update

```
duongkhang@duongkhang-VirtualBox:~$ sudo apt update
[sudo] password for duongkhang:
Hit:1 http://vn.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://vn.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://vn.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
16 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Hình 1.1 Tiến hành cập nhật hệ thống

Cài đặt các công cụ và thư viện hỗ trợ: NS3 yêu cầu một số công cụ như *g++* (trình biên dịch C++), *python3* (phiên bản Python 3), *cmake* (công cụ quản lý xây dựng), *bzip2* (công cụ giải nén), và *gnuplot* (dùng để vẽ đồ thị kết quả mô phỏng):

sudo apt install g++ python3 cmake bzip2 gnuplot

```

duongkhang@duongkhang-VirtualBox:~$ sudo apt install g++ python3 cmake bzip2 gnuplot
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.12.3-0ubuntu2).
python3 set to manually installed.
bzip2 is already the newest version (1.0.8-5.1build0.1).
The following additional packages will be installed:
  aglfn cmake-data g++-13 g++-13-x86-64-linux-gnu g++-x86-64-linux-gnu
  gnuplot-data gnuplot-qt libdouble-conversion3 libjsoncpp25 libmd4c0
  libpcre2-16-0 libqt5core5t64 libqt5dbus5t64 libqt5gui5t64 libqt5network5t64
  libqt5printsupport5t64 libqt5qml5 libqt5qmlmodels5 libqt5quick5 libqt5svg5
  libqt5waylandclient5 libqt5waylandcompositor5 libqt5widgets5t64 librhash0
  libstdc++-13-dev libwxbase3.2-1t64 libwxgtk3.2-1t64 libxcb-xinerama0
  libxcb-xinput0 qt5-gtk-platformtheme qttranslations5-l10n qtwayland5
Suggested packages:
  cmake-doc cmake-format elpa-cmake-mode ninja-build g++-multilib
  g++-13-multilib gcc-13-doc gnuplot-doc qgnameplatform-qt5
  qt5-image-formats-plugins qt5-qmltooling-plugins libstdc++-13-doc
The following NEW packages will be installed:
  aglfn cmake cmake-data g++ g++-13 g++-13-x86-64-linux-gnu
  g++-x86-64-linux-gnu gnuplot gnuplot-data gnuplot-qt libdouble-conversion3
  libjsoncpp25 libmd4c0 libpcre2-16-0 libqt5core5t64 libqt5dbus5t64
  libqt5gui5t64 libqt5network5t64 libqt5printsupport5t64 libqt5qml5
  libqt5qmlmodels5 libqt5quick5 libqt5svg5 libqt5waylandclient5
  libqt5waylandcompositor5 libqt5widgets5t64 librhash0 libstdc++-13-dev
  libwxbase3.2-1t64 libwxgtk3.2-1t64 libxcb-xinerama0 libxcb-xinput0
  qt5-gtk-platformtheme qttranslations5-l10n qtwayland5
0 upgraded, 35 newly installed, 0 to remove and 16 not upgraded.
Need to get 51.8 MB of archives.
After this operation, 199 MB of additional disk space will be used.
Do you want to continue? [Y/n] y

```

Hình 1.2 Tiến hành cài đặt các công cụ và thư viện hỗ trợ

Tải về và giải nén NS3: Sau khi cài đặt các công cụ hỗ trợ, ta tải về bộ NS3 phiên bản 3.43 từ trang chủ và tiến hành giải nén:

wget <https://www.nsnam.org/releases/ns-allinone-3.43.tar.bz2>

tar -xjf ns-allinone-3.43.tar.bz2

```

duongkhang@duongkhang-VirtualBox:~$ wget https://www.nsnam.org/releases/ns-allinone-3.43
.tar.bz2
--2024-11-08 16:09:28-- https://www.nsnam.org/releases/ns-allinone-3.43.tar.bz2
Resolving www.nsnam.org (www.nsnam.org)... 143.215.76.161
Connecting to www.nsnam.org (www.nsnam.org)|143.215.76.161|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 40962361 (39M) [application/x-bzip2]
Saving to: 'ns-allinone-3.43.tar.bz2'

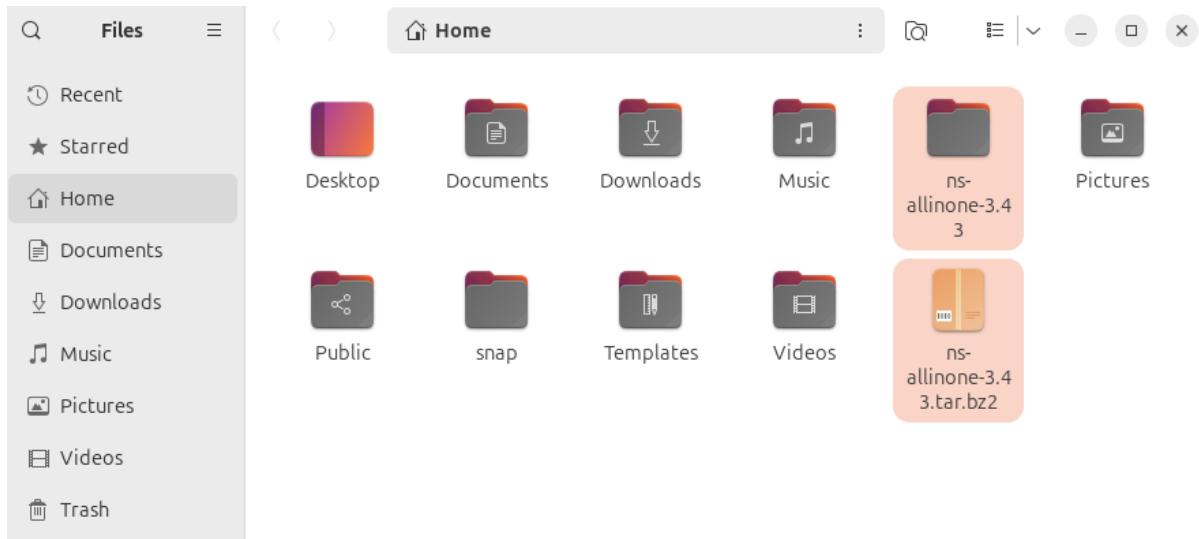
ns-allinone-3.43.tar. 100%[=====] 39.06M  134KB/s   in 7m 26s

2024-11-08 16:16:55 (89.7 KB/s) - 'ns-allinone-3.43.tar.bz2' saved [40962361/40962361]

duongkhang@duongkhang-VirtualBox:~$ tar -xjf ns-allinone-3.43.tar.bz2

```

Hình 1.3 Tiến hành tải về và giải nén NS3



Hình 1.4 Kết quả sau khi tải về và giải nén NS3

Xây dựng NS3: Ta chuyển đến thư mục chứa mã nguồn NS3 và tiến hành biên dịch, kích hoạt các ví dụ và kiểm tra để đảm bảo NS3 hoạt động chính xác:

`cd ~/ns-allinone-3.43`

`./build.py --enable-examples --enable-tests`

```
duongkhang@duongkhang-VirtualBox:~$ cd ~/ns-allinone-3.43
duongkhang@duongkhang-VirtualBox:~/ns-allinone-3.43$ ./build.py --enable-examples --enable-tests
# Build NetAnim
Entering directory `netanim-3.109'
=> qmake -v
Could not find qmake in the default path
=> qmake-qt5 -v
Could not find qmake-qt5 in the default path
=> qmake NetAnim.pro
Error building NetAnim. Ensure the path to qmake is correct.
Could not find qmake or qmake-qt5 in the default PATH.
Use ./build.py --qmake-path <Path-to-qmake>, if qmake is installed in a non-standard location
Note: Some systems use qmake-qt5 instead of qmake
Skipping NetAnim ....
Leaving directory `netanim-3.109'
# Building examples (by user request)
# Building tests (by user request)
# Build NS-3
Entering directory `/home/duongkhang/ns-allinone-3.43/.ns-3.43'
=> /usr/bin/python3 ns3 configure --enable-examples --enable-tests
```

Hình 1.5 Tiến hành xây dựng NS3

Kiểm tra cài đặt: Sau khi hoàn tất quá trình xây dựng, ta kiểm tra lại bằng cách chạy một ví dụ mặc định để đảm bảo NS3 đã được cài đặt thành công:

`cd ~/ns-allinone-3.43/ns-3.43`

`./ns3 run hello-simulator`

```
duongkhang@duongkhang-VirtualBox:~/ns-allinone-3.43$ cd ~/ns-allinone-3.43/ns-3.43
duongkhang@duongkhang-VirtualBox:~/ns-allinone-3.43/ns-3.43$ ./ns3 run hello-simulator
Hello Simulator
```

Hình 1.6 Tiến hành kiểm tra cài đặt

Nếu NS3 chạy thành công mà không gặp lỗi, quá trình cài đặt và thiết lập môi trường được coi là hoàn tất.

2.2 Cấu hình mô phỏng

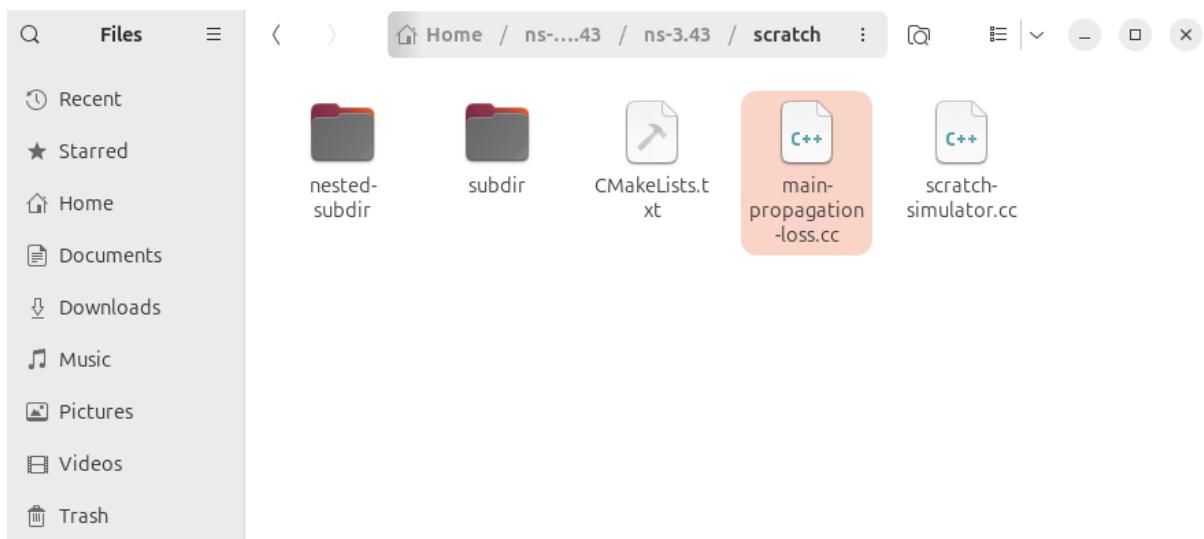
Sau khi cài đặt NS3, bước tiếp theo là cấu hình các mô hình suy hao truyền sóng để tiến hành mô phỏng. Trong dự án này, ta sử dụng mã nguồn đã được cung cấp sẵn từ NS3 (*main-propagation-loss.cc*) để thực hiện mô phỏng các mô hình khác nhau. Mỗi mô hình suy hao truyền sóng sẽ được thiết lập với các tham số cụ thể để phù hợp với các điều kiện mô phỏng khác nhau.

Sao chép mã nguồn có sẵn: Để thuận tiện cho việc chạy mô phỏng, ta sao chép tập tin *main-propagation-loss.cc* từ thư mục *src/propagation/examples/* vào thư mục *scratch* trong NS3:

```
cp ~/ns-allinone-3.43/ns-3.43/src/propagation/examples/main-propagation-loss.cc ~/ns-allinone-3.43/ns-3.43/scratch
```

```
duongkhang@duongkhang-VirtualBox:~/ns-allinone-3.43/ns-3.43$ cp ~/ns-allinone-3.43/ns-3.43/src/propagation/examples/main-propagation-loss.cc ~/ns-allinone-3.43/ns-3.43/scratch
```

Hình 1.7 Tiến hành sao chép mã nguồn có sẵn



Hình 1.8 Tập tin *main-propagation-loss.cc* đã được sao chép đến thư mục *scratch*

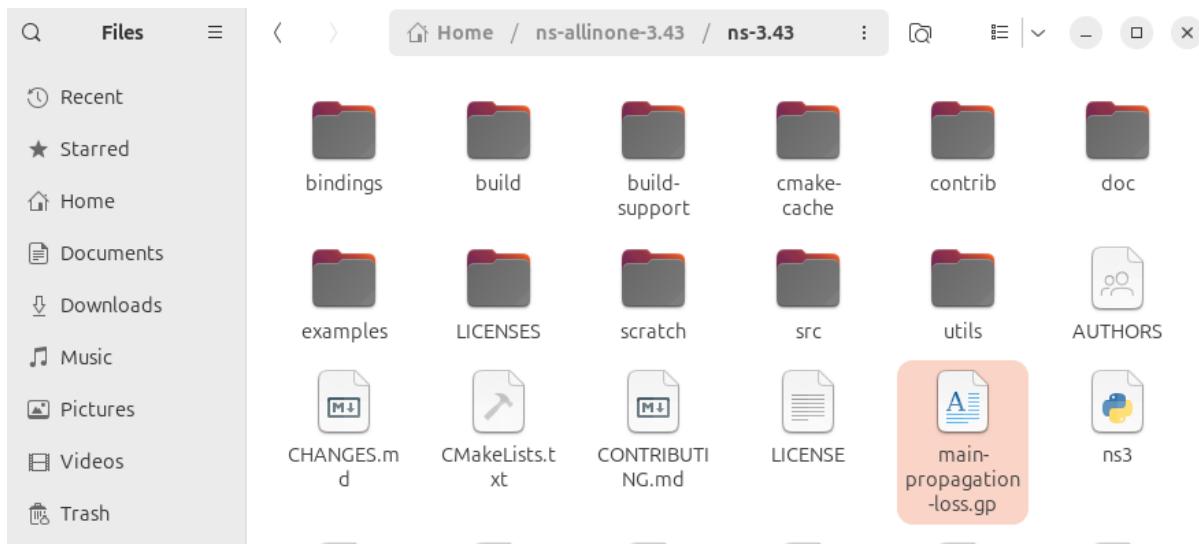
Chạy mô phỏng: Để thực hiện mô phỏng và lưu lại kết quả, ta tiến hành biên dịch và chạy tập tin *main-propagation-loss.cc* đã sao chép trong thư mục *scratch*. Kết quả mô phỏng sẽ được lưu trong tập tin *main-propagation-loss.gp*:

```
/ns3 build scratch/main-propagation-loss.cc
```

```
./ns3 run scratch/main-propagation-loss.cc > main-propagation-loss.gp
```

```
duongkhang@duongkhang-VirtualBox:~/ns-allinone-3.43/ns-3.43$ ./ns3 build scratch/main-propagation-loss.cc
[ 0%] Building CXX object scratch/CMakeFiles/scratch_main-propagation-loss.dir/main-propagation-loss.cc.o
[ 0%] Linking CXX executable /home/duongkhang/ns-allinone-3.43/ns-3.43/build/scratch/ns-3.43-main-propagation-loss-default
Finished executing the following commands:
/usr/bin/cmake --build /home/duongkhang/ns-allinone-3.43/ns-3.43/cmake-cache -j 3 --target scratch_main-propagation-loss
duongkhang@duongkhang-VirtualBox:~/ns-allinone-3.43/ns-3.43$ ./ns3 run scratch/main-propagation-loss.cc> main-propagation-loss.gp
```

Hình 1.9 Tiến hành chạy mô phỏng



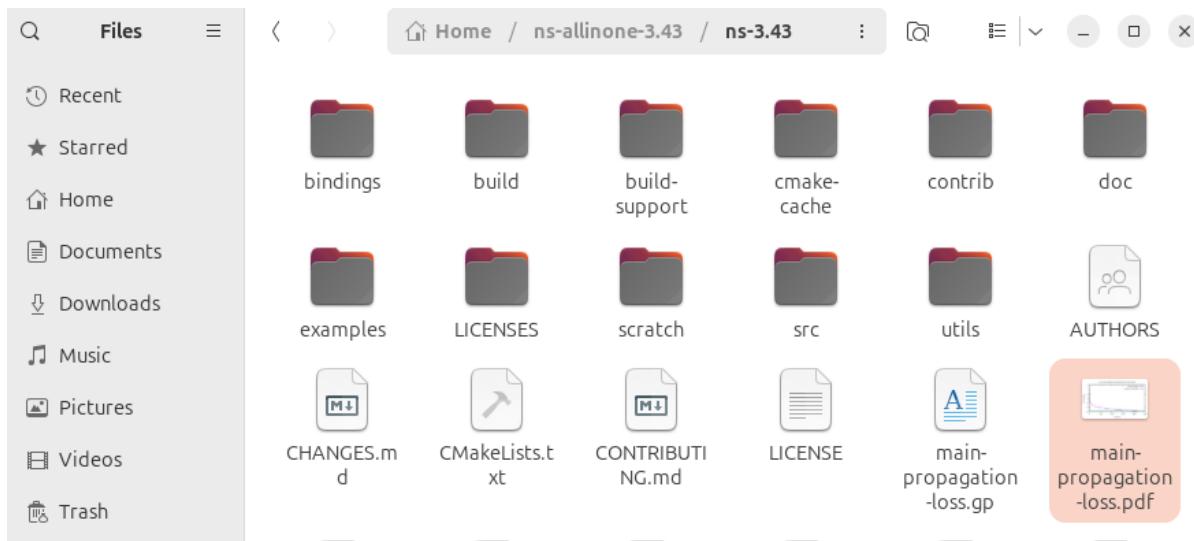
Hình 1.10 Kết quả mô phỏng được lưu trong tập tin main-propagation-loss.gp

Vẽ đồ thị kết quả: Sau khi có kết quả mô phỏng, ta sử dụng *gnuplot* để tạo đồ thị trực quan từ tập tin *main-propagation-loss.gp*:

```
gnuplot main-propagation-loss.gp
```

```
duongkhang@duongkhang-VirtualBox:~/ns-allinone-3.43/ns-3.43$ gnuplot main-propagation-loss.gp
```

Hình 1.11 Tiến hành vẽ đồ thị kết quả



Hình 1.12 Đồ thị được vẽ và lưu trong tập tin *main-propagation-loss.pdf*

2.3 Lưu ý về cấu hình

Tham số mô hình: Mỗi mô hình suy hao truyền sóng có thể được thiết lập với các tham số khác nhau để phù hợp với điều kiện thực tế của môi trường mô phỏng. Trong *main-propagation-loss.cc*, các tham số này đã được cấu hình sẵn. Ví dụ, với mô hình Log-Distance, hệ số suy hao được đặt là 2.5, và với mô hình Jakes, tần số Doppler được đặt cho tần số 5.15 GHz với tốc độ 100 km/h.

Kiểm tra và gỡ lỗi: Trong quá trình biên dịch và chạy mô phỏng, ta tiến hành kiểm tra từng bước để đảm bảo mã nguồn không gặp lỗi và các mô hình được thiết lập đúng cách. Điều này bao gồm kiểm tra lại các thư viện đã cài đặt và đảm bảo rằng NS3 chạy đúng phiên bản.

3. Mô hình suy hao truyền sóng

Trong dự án này, ta sử dụng *main-propagation-loss.cc* để mô phỏng các mô hình suy hao truyền sóng khác nhau trong NS3, mỗi mô hình có cách thức tính toán suy hao tín hiệu khác nhau tùy thuộc vào điều kiện môi trường và các yếu tố truyền sóng. Dưới đây là các mô hình đã được mô phỏng:

3.1 Friis Propagation Loss Model

Mô tả mô hình: Mô hình Friis được sử dụng để mô phỏng suy hao tín hiệu trong môi trường không gian tự do, nơi tín hiệu không bị ảnh hưởng bởi các yếu tố môi trường như chướng ngại vật hay nhiễu. Công thức Friis tính toán mức độ suy hao tín hiệu dựa trên khoảng cách giữa máy phát và máy thu theo quy luật nghịch đảo bình phương. Mô hình này phù hợp cho các môi trường lý tưởng, chẳng hạn như trong phòng thí nghiệm hoặc trong không gian mở mà không có vật cản.

Công thức:

$$P_r = \frac{P_t \cdot G_t \cdot G_r \cdot \lambda^2}{(4\pi d)^2 \cdot L} \quad (1)$$

Trong đó:

- P_r : Công suất tín hiệu nhận được (W)
- P_t : Công suất tín hiệu phát (W)
- G_t : Tăng cường của anten phát (không có đơn vị)
- G_r : Tăng cường của anten thu (không có đơn vị)
- λ : Bước sóng (m)
- d : Khoảng cách giữa máy phát và máy thu (m)
- L : Suy hao hệ thống (không có đơn vị)

3.2 Log-Distance Propagation Loss Model

Mô tả mô hình: Mô hình Log-Distance phù hợp cho các môi trường thực tế, khi tín hiệu không chỉ suy hao theo khoảng cách mà còn chịu ảnh hưởng bởi các chướng ngại vật. Công thức Log-Distance sử dụng một hàm mũ để tính toán mức độ suy hao tín hiệu, trong đó hệ số suy hao có thể điều chỉnh tùy thuộc vào môi trường. Trong dự án này, hệ số suy hao được đặt là 2.5, mô phỏng một môi trường có mức suy hao tín hiệu tương đối lớn do chướng ngại vật hoặc môi trường không gian đô thị.

Công thức:

$$L = L_0 + 10n \cdot \log_{10} \left(\frac{d}{d_0} \right) \quad (2)$$

Trong đó:

- L : Suy hao đường truyền (dB)
- L_0 : Suy hao tại khoảng cách tham chiếu (dB)
- n : Hệ số suy hao theo khoảng cách
- d : Khoảng cách thực tế giữa máy phát và máy thu (m)
- d_0 : Khoảng cách tham chiếu (m)

3.3 Random Propagation Loss Model với phân phối Exponential

Mô tả mô hình: Mô hình Random Propagation Loss với phân phối Exponential sử dụng biến ngẫu nhiên để mô phỏng sự biến động của tín hiệu trong các môi trường có nhiều yếu tố ngẫu nhiên, chẳng hạn như các nguồn gây nhiễu không đồng nhất hoặc môi trường không ổn định. Phân phối Exponential giúp mô phỏng sự thay đổi bất thường của tín hiệu, đặc biệt trong các điều kiện mà tín hiệu có thể bị suy hao mạnh tại một số điểm do tác động của các yếu tố không dự đoán trước.

Công thức:

$$P_r = P_t + X \quad (3)$$

Trong đó:

- P_r : Công suất tín hiệu nhận được (dB)
- P_t : Công suất tín hiệu phát (dB)
- X : Một biến ngẫu nhiên với phân phối Exponential, được tạo ra để mô phỏng sự biến động của tín hiệu.

3.4 Jakes Propagation Loss Model

Mô tả mô hình: Mô hình Jakes được sử dụng để mô phỏng hiệu ứng Doppler, xảy ra khi nguồn phát và nguồn thu di chuyển tương đối với nhau, dẫn đến sự thay đổi tần số của tín hiệu nhận được. Hiệu ứng Doppler rất quan trọng trong các kênh di động, nơi mà thiết bị di chuyển có thể gây ra sự thay đổi đáng kể về cường độ tín hiệu. Trong mô phỏng này, tần số Doppler được thiết lập cho tần số 5.15 GHz với tốc độ 100 km/h, và được thực hiện với hai độ phân giải thời gian khác nhau: 1 ms và 0.1 ms.

Công thức:

$$P_r = P_t + G_c \quad (4)$$

Trong đó:

- P_r : Công suất tín hiệu nhận được (dB)
- P_t : Công suất tín hiệu phát (dB)
- G_c : Hệ số tăng cường kênh (dB)

3.5 Three-Log-Distance Propagation Loss Model

Mô tả mô hình: Three-Log-Distance là một biến thể của Log-Distance, trong đó khoảng cách được chia thành ba vùng khác nhau, mỗi vùng có một hệ số suy hao riêng biệt. Điều này cho phép mô phỏng sự suy hao tín hiệu trong các điều kiện môi trường

thay đổi theo khoảng cách. Trong mô phỏng này, các hệ số suy hao mặc định được sử dụng, cùng với một cấu hình khác với các hệ số suy hao là 1.0, 3.0, và 10.0.

Công thức:

$$L = \begin{cases} 0 & \text{nếu } d < d_0 \\ L_0 + 10n_0 \cdot \log_{10}\left(\frac{d}{d_0}\right) & \text{nếu } d_0 \leq d < d_1 \\ L_0 + 10n_0 \cdot \log_{10}\left(\frac{d_1}{d_0}\right) + 10n_1 \cdot \log_{10}\left(\frac{d}{d_1}\right) & \text{nếu } d_1 \leq d < d_2 \\ L_0 + 10n_0 \cdot \log_{10}\left(\frac{d_1}{d_0}\right) + 10n_1 \cdot \log_{10}\left(\frac{d_2}{d_1}\right) + 10n_2 \cdot \log_{10}\left(\frac{d}{d_2}\right) & \text{nếu } d \geq d_2 \end{cases} \quad (5)$$

Trong đó:

- L : Suy hao đường truyền (dB)
- L_0 : Suy hao tại khoảng cách tham chiếu (dB)
- d : Khoảng cách thực tế giữa máy phát và máy thu (m)
- d_0, d_1, d_2 : Các khoảng cách xác định ba vùng (m).
- n_0, n_1, n_2 : Hệ số suy hao cho từng vùng.

3.6 Nakagami Propagation Loss Model

Mô tả mô hình: Nakagami là một mô hình xác suất thường được sử dụng để mô phỏng hiện tượng fading đa đường trong các môi trường mà tín hiệu có thể đến nguồn thu từ nhiều hướng khác nhau. Mô hình Nakagami cho phép mô phỏng sự phân tán của tín hiệu khi truyền qua các môi trường có nhiều chướng ngại vật hoặc có hiện tượng phản xạ mạnh.

Công thức xác suất:

$$p(x; m, \omega) = \frac{2m^m}{\Gamma(m)\omega^m} \cdot x^{2m-1} \cdot e^{-\frac{mx^2}{\omega}} \quad (6)$$

Trong đó:

- $p(x; m, \omega)$: Hàm mật độ xác suất của tín hiệu tại giá trị x
- x : Biến ngẫu nhiên đại diện cho cường độ tín hiệu nhận được
- m : Tham số hình dạng của phân phối Nakagami
- ω : Công suất trung bình nhận được.

3.7 Kết hợp Three-Log-Distance Propagation Loss Model và Nakagami Propagation Loss Model

Mô tả mô hình: Mô hình kết hợp giữa Three-Log-Distance và Nakagami cho phép mô phỏng đồng thời cả sự suy hao tín hiệu theo khoảng cách và hiện tượng fading. Việc kết hợp này tạo ra một mô hình truyền sóng thực tế hơn, đặc biệt hữu ích cho các ứng dụng trong mạng không dây phức tạp, nơi tín hiệu phải vượt qua nhiều chướng ngại vật và chịu ảnh hưởng của fading.

4. Phân tích mã nguồn

Trong phần này, ta sẽ phân tích các thành phần chính trong *main-propagation-loss.cc*, bao gồm các hàm và cấu trúc quan trọng thực hiện mô phỏng từng mô hình suy hao truyền sóng. Mã nguồn được thiết kế để mô phỏng và vẽ đồ thị kết quả của nhiều mô hình khác nhau trong NS3, bao gồm các mô hình như Friis, Log-Distance, Random với phân phối Exponential, Jakes, Three-Log-Distance, Nakagami, và sự kết hợp giữa Three-Log-Distance và Nakagami.

4.1 Hàm TestDeterministic

```
static Gnuplot
TestDeterministic(Ptr<PropagationLossModel> model, double targetDistance,
double step)
{
    Ptr<ConstantPositionMobilityModel> a =
CreateObject<ConstantPositionMobilityModel>();
    Ptr<ConstantPositionMobilityModel> b =
CreateObject<ConstantPositionMobilityModel>();

    Gnuplot plot;

    plot.AppendExtra("set xlabel 'Distance'");
    plot.AppendExtra("set ylabel 'rxPower (dBm)'");
    plot.AppendExtra("set key top right");

    double txPowerDbm = +20; // dBm

    Gnuplot2dDataset dataset;

    dataset.setStyle(Gnuplot2dDataset::LINES);

    {
```

```

a->SetPosition(Vector(0.0, 0.0, 0.0));

for (double distance = 0.0; distance < targetDistance; distance += step)
{
    b->SetPosition(Vector(distance, 0.0, 0.0));

    // CalcRxPower() returns dBm.
    double rxPowerDbm = model->CalcRxPower(txPowerDbm, a, b);

    dataset.Add(distance, rxPowerDbm);

    Simulator::Stop(Seconds(1.0));
    Simulator::Run();
}
}

std::ostringstream os;
os << "txPower " << txPowerDbm << "dBm";
datasetSetTitle(os.str());

plot.AddDataset(dataset);

plot.AddDataset(Gnuplot2dFunction("-94 dBm CSThreshold", "-94.0"));

return plot;
}

```

Hàm *TestDeterministic* được sử dụng để mô phỏng các mô hình suy hao truyền sóng có tính chất xác định, chẳng hạn như Friis và Log-Distance. Hàm này lấy đầu vào là một đối tượng *PropagationLossModel*, một khoảng cách đích (*targetDistance*), và một bước nhảy (*step*) để tính toán sự suy hao tín hiệu theo từng khoảng cách.

Cấu trúc chính của hàm như sau:

- Đối tượng Mobility:** Tạo hai đối tượng *ConstantPositionMobilityModel* (a và b) để thiết lập vị trí cố định cho nguồn phát và nguồn thu.
- Vòng lặp khoảng cách:** Sử dụng vòng lặp để di chuyển nguồn thu (b) đến các khoảng cách khác nhau so với nguồn phát (a). Tại mỗi khoảng cách, hàm *CalcRxPower* được gọi để tính công suất tín hiệu nhận được (*rxPowerDbm*) dựa trên mô hình truyền sóng đã chọn.

- **Lưu dữ liệu và vẽ đồ thị:** Dữ liệu công suất tín hiệu nhận được theo khoảng cách được lưu vào *Gnuplot2dDataset* và dùng để tạo đồ thị hiển thị mối quan hệ giữa công suất nhận được và khoảng cách.

4.2 Hàm TestProbabilistic

```

static Gnuplot
TestProbabilistic(Ptr<PropagationLossModel> model,
                  double targetDistance,
                  double step,
                  unsigned int samples)
{
    Ptr<ConstantPositionMobilityModel> a =
CreateObject<ConstantPositionMobilityModel>();
    Ptr<ConstantPositionMobilityModel> b =
CreateObject<ConstantPositionMobilityModel>();

    Gnuplot plot;

    plot.AppendExtra("set xlabel 'Distance'");
    plot.AppendExtra("set ylabel 'rxPower (dBm)'");
    plot.AppendExtra("set zlabel 'Probability' offset 0,+10");
    plot.AppendExtra("set view 50, 120, 1.0, 1.0");
    plot.AppendExtra("set key top right");

    plot.AppendExtra("set ticslevel 0");
    plot.AppendExtra("set xtics offset -0.5,0");
    plot.AppendExtra("set ytics offset 0,-0.5");
    plot.AppendExtra("set xrange [100:]");

    double txPowerDbm = +20; // dBm

    Gnuplot3dDataset dataset;

    dataset.setStyle("with linespoints");
    dataset.setExtra("pointtype 3 pointsize 0.5");

    typedef std::map<double, unsigned int> rxPowerMapType;

    // Take given number of samples from CalcRxPower() and show probability
    // density for discrete distances.

```

```

{
    a->SetPosition(Vector(0.0, 0.0, 0.0));

    for (double distance = 100.0; distance < targetDistance; distance += step)
    {
        b->SetPosition(Vector(distance, 0.0, 0.0));

        rxPowerMapType rxPowerMap;

        for (unsigned int samp = 0; samp < samples; ++samp)
        {
            // CalcRxPower() returns dBm.
            double rxPowerDbm = model->CalcRxPower(txPowerDbm, a, b);
            rxPowerDbm = dround(rxPowerDbm, 1.0);

            rxPowerMap[rxPowerDbm]++;
        }

        Simulator::Stop(Seconds(0.01));
        Simulator::Run();
    }

    for (auto i = rxPowerMap.begin(); i != rxPowerMap.end(); ++i)
    {
        dataset.Add(distance, i->first, (double)i->second / (double)samples);
    }
    dataset.AddEmptyLine();
}
}

std::ostringstream os;
os << "txPower " << txPowerDbm << "dBm";
datasetSetTitle(os.str());

plot.AddDataset(dataset);

return plot;
}

```

Hàm *TestProbabilistic* được thiết kế để mô phỏng các mô hình suy hao truyền sóng có tính chất xác suất, chẳng hạn như Nakagami. Hàm này nhận đầu vào gồm một đối

tượng *PropagationLossModel*, khoảng cách đích (*targetDistance*), bước nhảy (*step*), và số lượng mẫu (*samples*) để tính toán xác suất công suất tín hiệu tại từng khoảng cách.

Cấu trúc chính của hàm như sau:

- **Đối tượng Mobility:** Tương tự như *TestDeterministic*, hàm này cũng sử dụng hai đối tượng *ConstantPositionMobilityModel* để mô phỏng vị trí của nguồn phát và nguồn thu.
- **Vòng lặp và lấy mẫu:** Tại mỗi khoảng cách, hàm này lấy *samples* từ *CalcRxPower* và lưu số lượng xuất hiện của từng mức công suất tín hiệu nhận được vào *rxPowerMap*.
- **Tính xác suất:** Dữ liệu xác suất được tính toán bằng cách chia số lần xuất hiện của mỗi mức công suất tín hiệu cho tổng số mẫu, rồi thêm vào *Gnuplot3dDataset* để hiển thị đồ thị xác suất của công suất tín hiệu theo khoảng cách.

4.3 Hàm TestDeterministicByTime

```
static Gnuplot
TestDeterministicByTime(Ptr<PropagationLossModel> model,
                        Time timeStep,
                        Time timeTotal,
                        double distance)
{
    Ptr<ConstantPositionMobilityModel> a =
CreateObject<ConstantPositionMobilityModel>();
    Ptr<ConstantPositionMobilityModel> b =
CreateObject<ConstantPositionMobilityModel>();

    Gnuplot plot;

    plot.AppendExtra("set xlabel 'Time (s)'");
    plot.AppendExtra("set ylabel 'rxPower (dBm)'");
    plot.AppendExtra("set key center right");

    double txPowerDbm = +20; // dBm

    Gnuplot2dDataset dataset;

    dataset.SetStyle(Gnuplot2dDataset::LINES);

    {
        a->SetPosition(Vector(0.0, 0.0, 0.0));
        b->SetPosition(Vector(distance, 0.0, 0.0));
    }
}
```

```

Time start = Simulator::Now();
while (Simulator::Now() < start + timeTotal)
{
    // CalcRxPower() returns dBm.
    double rxPowerDbm = model->CalcRxPower(txPowerDbm, a, b);

    Time elapsed = Simulator::Now() - start;
    dataset.Add(elapsed.GetSeconds(), rxPowerDbm);

    Simulator::Stop(timeStep);
    Simulator::Run();
}

std::ostringstream os;
os << "txPower " << txPowerDbm << "dBm";
datasetSetTitle(os.str());

plot.AddDataset(dataset);

plot.AddDataset(Gnuplot2dFunction("-94 dBm CSThreshold", "-94.0"));

return plot;
}

```

Hàm *TestDeterministicByTime* được sử dụng cho mô phỏng các mô hình có sự thay đổi công suất tín hiệu theo thời gian, chẳng hạn như mô hình Jakes. Hàm này nhận đầu vào là một đối tượng *PropagationLossModel*, bước thời gian (*timeStep*), tổng thời gian mô phỏng (*timeTotal*), và khoảng cách giữa hai thiết bị (*distance*).

Cấu trúc chính của hàm như sau:

- **Đối tượng Mobility:** Thiết lập khoảng cách cố định giữa nguồn phát và nguồn thu bằng cách sử dụng *ConstantPositionMobilityModel*.
- **Vòng lặp thời gian:** Sử dụng vòng lặp để lấy mẫu công suất tín hiệu sau mỗi *timeStep* cho đến khi đạt *timeTotal*. Công suất tín hiệu được tính bằng *CalcRxPower* và được lưu vào *Gnuplot2dDataset*.
- **Lưu dữ liệu và vẽ đồ thị:** Đồ thị thể hiện mối quan hệ giữa công suất tín hiệu nhận được và thời gian, giúp hiển thị hiệu ứng Doppler khi tín hiệu thay đổi liên tục trong các kênh di động.

4.4 Hàm main

```
int
main(int argc, char* argv[])
{
    bool test = false;
    CommandLine cmd(__FILE__);
    cmd.AddValue("test", "Run as a test, sample the models only once", test);
    cmd.Parse(argc, argv);

    double testDeterministicDistance = 2500.0;
    double testProbabilisticDistance = 2500.0;
    unsigned int testProbabilisticSamples = 100000;
    Time testJakesTimeOneMsRes = Seconds(1.0);
    Time testJakesTimeZeroDotOneMsRes = Seconds(0.1);

    if (test)
    {
        testDeterministicDistance = 10;
        testProbabilisticDistance = 200;
        testProbabilisticSamples = 1;
        testJakesTimeOneMsRes = Seconds(0.001);
        testJakesTimeZeroDotOneMsRes = Seconds(0.0001);
    }

    GnuplotCollection gnuplots("main-propagation-loss.pdf");

    {
        Ptr<FriisPropagationLossModel> friis =
CreateObject<FriisPropagationLossModel>();

        Gnuplot plot = TestDeterministic(friis, testDeterministicDistance, 10.0);
        plot.SetTitle("NS3::FriisPropagationLossModel (Default Parameters)");
        gnuplots.AddPlot(plot);
    }

    {
        Ptr<LogDistancePropagationLossModel> log =
CreateObject<LogDistancePropagationLossModel>();
        log->SetAttribute("Exponent", DoubleValue(2.5));
    }
}
```

```

Gnuplot plot = TestDeterministic(log, testDeterministicDistance, 10.0);
plot.SetTitle("NS3::LogDistancePropagationLossModel (Exponent = 2.5)");
gnuplots.AddPlot(plot);
}

{
    Ptr<RandomPropagationLossModel> random =
CreateObject<RandomPropagationLossModel>();
    Ptr<ExponentialRandomVariable> expVar =
        CreateObjectWithAttributes<ExponentialRandomVariable>("Mean",
DoubleValue(50.0));
    random->SetAttribute("Variable", PointerValue(expVar));

    Gnuplot plot = TestDeterministic(random, testDeterministicDistance, 10.0);
    plot.SetTitle("NS3::RandomPropagationLossModel with Exponential
Distribution");
    gnuplots.AddPlot(plot);
}

{
    Ptr<JakesPropagationLossModel> jakes =
CreateObject<JakesPropagationLossModel>();

    // doppler frequency shift for 5.15 GHz at 100 km/h
    Config::SetDefault("NS3::JakesProcess::DopplerFrequencyHz",
DoubleValue(477.9));

    Gnuplot plot = TestDeterministicByTime(jakes, Seconds(0.001),
testJakesTimeOneMsRes, 100.0);
    plotSetTitle(
        "NS3::JakesPropagationLossModel (with 477.9 Hz shift and 1 millisec
resolution)");
    gnuplots.AddPlot(plot);
    // Usually objects are aggregated either to a Node or a Channel, and this
aggregation
    // ensures a proper call to Dispose. Here we must call it manually, since the
    // PropagationLossModel is not aggregated to anything.
    jakes->Dispose();
}

```

```

{
    Ptr<JakesPropagationLossModel> jakes =
CreateObject<JakesPropagationLossModel>();

    // doppler frequency shift for 5.15 GHz at 100 km/h
    Config::SetDefault("NS3::JakesProcess::DopplerFrequencyHz",
DoubleValue(477.9));

Gnuplot plot =
    TestDeterministicByTime(jakes, Seconds(0.0001),
testJakesTimeZeroDotOneMsRes, 100.0);
    plotSetTitle(
        "NS3::JakesPropagationLossModel (with 477.9 Hz shift and 0.1 millisec
resolution)");
    gnuplots.AddPlot(plot);
    // Usually objects are aggregated either to a Node or a Channel, and this
aggregation
    // ensures a proper call to Dispose. Here we must call it manually, since the
    // PropagationLossModel is not aggregated to anything.
    jakes->Dispose();
}

{

    Ptr<ThreeLogDistancePropagationLossModel> log3 =
CreateObject<ThreeLogDistancePropagationLossModel>();

    Gnuplot plot = TestDeterministic(log3, testDeterministicDistance, 10.0);
    plotSetTitle("NS3::ThreeLogDistancePropagationLossModel (Defaults)");
    gnuplots.AddPlot(plot);
}

{

    Ptr<ThreeLogDistancePropagationLossModel> log3 =
CreateObject<ThreeLogDistancePropagationLossModel>();
    // more prominent example values:
    log3->SetAttribute("Exponent0", DoubleValue(1.0));
    log3->SetAttribute("Exponent1", DoubleValue(3.0));
    log3->SetAttribute("Exponent2", DoubleValue(10.0));

    Gnuplot plot = TestDeterministic(log3, testDeterministicDistance, 10.0);
}

```

```

    plot.SetTitle("NS3::ThreeLogDistancePropagationLossModel (Exponents
1.0, 3.0 and 10.0)");
    gnuplots.AddPlot(plot);
}

{
    Ptr<NakagamiPropagationLossModel> nak =
CreateObject<NakagamiPropagationLossModel>();

Gnuplot plot =
    TestProbabilistic(nak, testProbabilisticDistance, 100.0,
testProbabilisticSamples);
    plotSetTitle("NS3::NakagamiPropagationLossModel (Default Parameters)");
    gnuplots.AddPlot(plot);
}

{
    Ptr<ThreeLogDistancePropagationLossModel> log3 =
CreateObject<ThreeLogDistancePropagationLossModel>();

Ptr<NakagamiPropagationLossModel> nak =
CreateObject<NakagamiPropagationLossModel>();
log3->SetNext(nak);

Gnuplot plot =
    TestProbabilistic(log3, testProbabilisticDistance, 100.0,
testProbabilisticSamples);
    plotSetTitle("NS3::ThreeLogDistancePropagationLossModel and "
                "NS3::NakagamiPropagationLossModel (Default Parameters)");
    gnuplots.AddPlot(plot);
}

gnuplots.GenerateOutput(std::cout);

// produce clean valgrind
Simulator::Destroy();
return 0;
}

```

Hàm *main* là điểm khởi đầu của chương trình, nơi tất cả các mô hình truyền sóng được cấu hình và chạy. Các bước chính trong *main* bao gồm:

1. **Xử lý tham số dòng lệnh:** Sử dụng *CommandLine* để nhận tham số *test* nhằm xác định số lượng mẫu cần lấy.
2. **Thiết lập mô hình và chạy mô phỏng:** Từng mô hình suy hao truyền sóng (như Friis, Log-Distance, Random với phân phối Exponential, Jakes, Three-Log-Distance, và Nakagami) được khởi tạo bằng cách sử dụng các đối tượng *PropagationLossModel* cụ thể. Mỗi mô hình được cấu hình với các tham số riêng biệt, và sau đó được truyền vào các hàm *TestDeterministic*, *TestProbabilistic*, hoặc *TestDeterministicByTime* để thực hiện mô phỏng.
3. **Lưu kết quả vào Gnuplot:** Kết quả của từng mô hình được lưu vào một đối tượng *GnuplotCollection* và xuất ra dưới dạng tập tin PDF (*main-propagation-loss.pdf*), hiển thị đồ thị của từng mô hình suy hao truyền sóng theo các điều kiện đã thiết lập.

4.5 Quy trình hoạt động của mã nguồn

Để giúp người đọc hiểu rõ hơn về cách *main-propagation-loss.cc* hoạt động, dưới đây là sơ đồ mô tả quy trình chính trong quá trình mô phỏng các mô hình suy hao truyền sóng. Sơ đồ này thể hiện các bước từ việc nhập tham số dòng lệnh, khởi tạo NS3 và cài đặt môi trường, cho đến việc chọn mô hình truyền sóng, chạy mô phỏng, lưu kết quả, và xuất tập tin PDF chứa các đồ thị kết quả.



Hình 1.13 Sơ đồ quy trình hoạt động của *main-propagation-loss.cc*

Sơ đồ này giúp minh họa rõ ràng các bước trong quy trình mô phỏng, từ khởi đầu đến kết thúc, và cách mà từng mô hình được xử lý trong mã nguồn. Các mô hình khác nhau như Friis, Log-Distance, Nakagami, và các mô hình kết hợp được gọi tương ứng theo từng bước, cho phép người dùng theo dõi luồng công việc dễ dàng.

4.6 Các tham số và thiết lập quan trọng

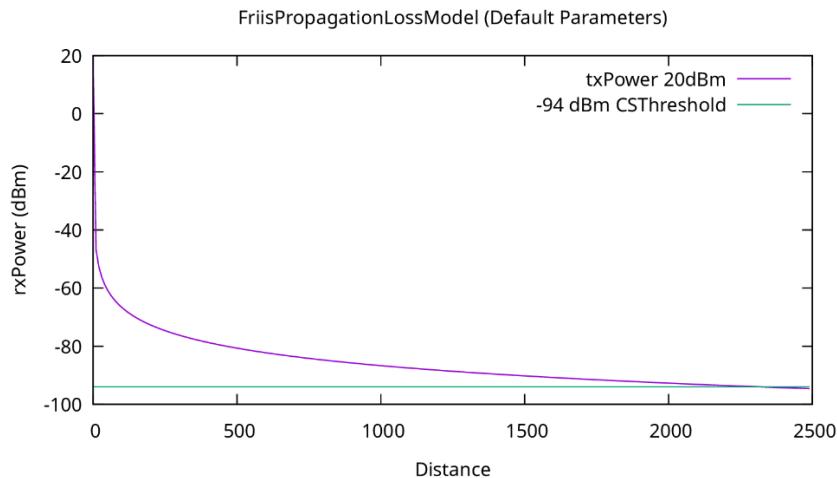
Dưới đây là một số tham số quan trọng được thiết lập trong *main-propagation-loss.cc* cho từng mô hình:

- **Friis Propagation Loss Model:** Sử dụng tham số mặc định.
- **Log-Distance Propagation Loss Model:** Đặt hệ số suy hao (Exponent) là 2.5 để mô phỏng một môi trường thực tế với độ suy hao tín hiệu lớn.
- **Random Propagation Loss Model:** Sử dụng phân phối Exponential với giá trị trung bình là 50.0 để mô phỏng sự biến động ngẫu nhiên của tín hiệu.
- **Jakes Propagation Loss Model:** Đặt tần số Doppler là 477.9 Hz để mô phỏng hiệu ứng Doppler cho tín hiệu có tần số 5.15 GHz ở tốc độ di chuyển 100 km/h.
- **Three-Log-Distance Propagation Loss Model:** Thiết lập với các hệ số suy hao mặc định, và một cấu hình khác với các hệ số suy hao là 1.0, 3.0, và 10.0 để mô phỏng sự suy hao tín hiệu trong ba vùng khoảng cách.
- **Nakagami Propagation Loss Model:** Sử dụng các tham số mặc định để mô phỏng hiện tượng fading trong môi trường đa đường.
- **Kết hợp Three-Log-Distance và Nakagami:** Thiết lập bằng cách kết hợp hai mô hình này thông qua phương thức *SetNext*, cho phép tín hiệu trải qua cả hai mô hình. Mô hình Three-Log-Distance mô phỏng sự suy hao tín hiệu theo ba vùng khoảng cách, sau đó tín hiệu tiếp tục bị ảnh hưởng bởi hiện tượng fading theo mô hình Nakagami. Kết hợp này cung cấp một cái nhìn toàn diện về sự suy hao tín hiệu trong môi trường phức tạp, nơi cả khoảng cách và fading đều ảnh hưởng đến chất lượng tín hiệu.

5. Kết quả và phân tích

Sau khi thực hiện mô phỏng, ta đã thu được các kết quả dưới dạng đồ thị, thể hiện mối quan hệ giữa khoảng cách (*Distance*) và công suất tín hiệu nhận được (*rxPower*), hoặc mối quan hệ giữa thời gian (*Time*) và công suất tín hiệu nhận được trong trường hợp của mô hình Jakes. Phần này sẽ phân tích kết quả của từng mô hình suy hao truyền sóng và so sánh hiệu quả của chúng trong các điều kiện môi trường khác nhau.

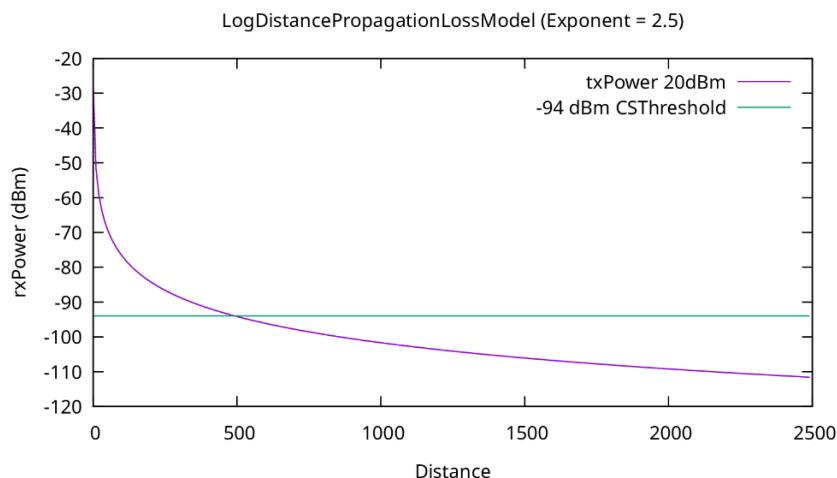
5.1 Kết quả của Friis Propagation Loss Model



Hình 1.14 Đồ thị kết quả mô phỏng của Friis Propagation Loss Model
với tham số mặc định

- Đồ thị:** Kết quả mô phỏng cho thấy công suất tín hiệu giảm dần theo khoảng cách theo quy luật nghịch đảo bình phương, đúng như lý thuyết của mô hình Friis trong không gian tự do.
- Phân tích:** Đối với môi trường lý tưởng không có chướng ngại vật, mô hình Friis cho kết quả chính xác và nhất quán. Tuy nhiên, mô hình này không thể hiện sự ảnh hưởng của các yếu tố môi trường phức tạp như vật cản hoặc nhiễu, do đó chỉ phù hợp với các kịch bản không gian tự do.
- Ứng dụng thực tế:** Mô hình này thích hợp cho các hệ thống truyền thông ngoài trời, như liên lạc vệ tinh hoặc truyền sóng qua không gian mở, nơi không có vật cản.

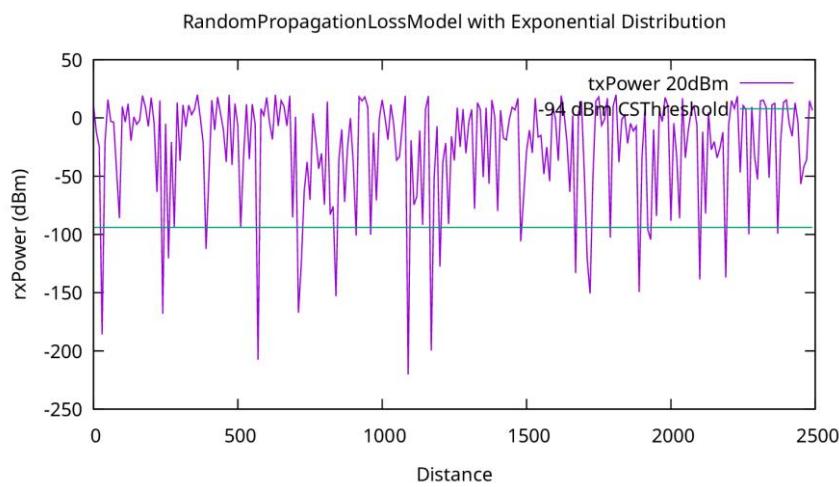
5.2 Kết quả của Log-Distance Propagation Loss Model



Hình 1.15 Đồ thị kết quả mô phỏng của Log-Distance Propagation Loss Model
với hệ số suy hao là 2.5

- **Đồ thị:** Công suất tín hiệu suy hao theo khoảng cách với tốc độ nhanh hơn so với mô hình Friis, do hệ số suy hao được đặt là 2.5, mô phỏng một môi trường có độ suy hao tín hiệu cao.
- **Phân tích:** Kết quả này phản ánh khả năng mô phỏng của mô hình Log-Distance trong các môi trường có chướng ngại vật, chẳng hạn như trong đô thị. Tín hiệu bị suy hao mạnh hơn so với không gian tự do do ảnh hưởng của các yếu tố môi trường.
- **Ứng dụng thực tế:** Mô hình này phù hợp cho các hệ thống truyền thông trong môi trường có nhiều vật cản, chẳng hạn như mạng di động trong đô thị, nơi mà các tòa nhà và kết cấu hạ tầng có ảnh hưởng lớn đến suy hao tín hiệu.

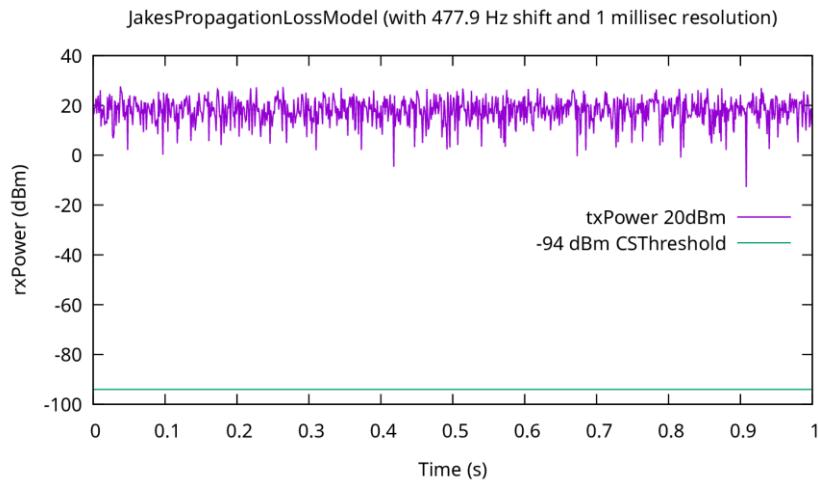
5.3 Kết quả của Random Propagation Loss Model với phân phối Exponential



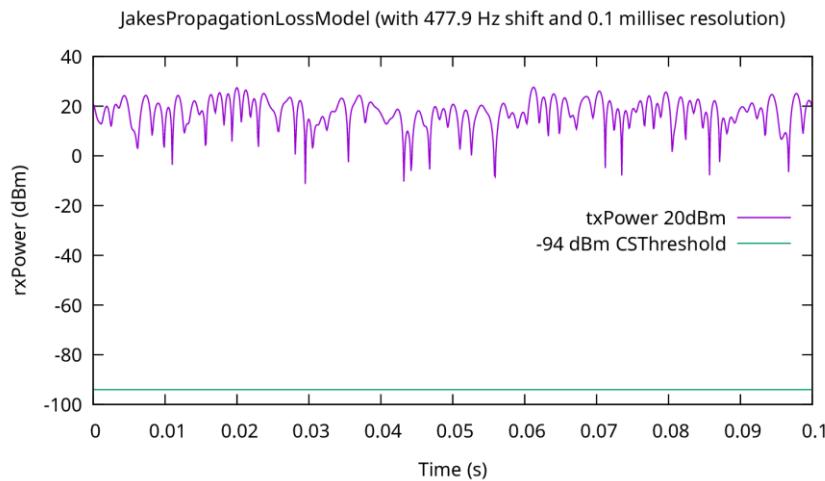
Hình 1.16 Đồ thị kết quả mô phỏng của Random Propagation Loss Model với phân phối Exponential

- **Đồ thị:** Công suất tín hiệu có xu hướng biến động ngẫu nhiên theo khoảng cách, thể hiện rõ sự thay đổi thất thường của tín hiệu trong các môi trường không ổn định hoặc có nhiều yếu tố ngẫu nhiên.
- **Phân tích:** Mô hình Random với phân phối Exponential cho thấy sự dao động của tín hiệu, phù hợp cho các môi trường có nhiều động không xác định. Tuy nhiên, mô hình này không cung cấp một xu hướng rõ ràng về sự suy hao tín hiệu, mà thay vào đó là sự thay đổi không dự đoán trước.
- **Ứng dụng thực tế:** Mô hình này phù hợp cho các môi trường mà tín hiệu chịu nhiều yếu tố ngẫu nhiên hoặc không xác định, chẳng hạn như môi trường trong các khu công nghiệp hoặc các khu vực có nguồn nhiễu không đồng nhất.

5.4 Kết quả của Jakes Propagation Loss Model



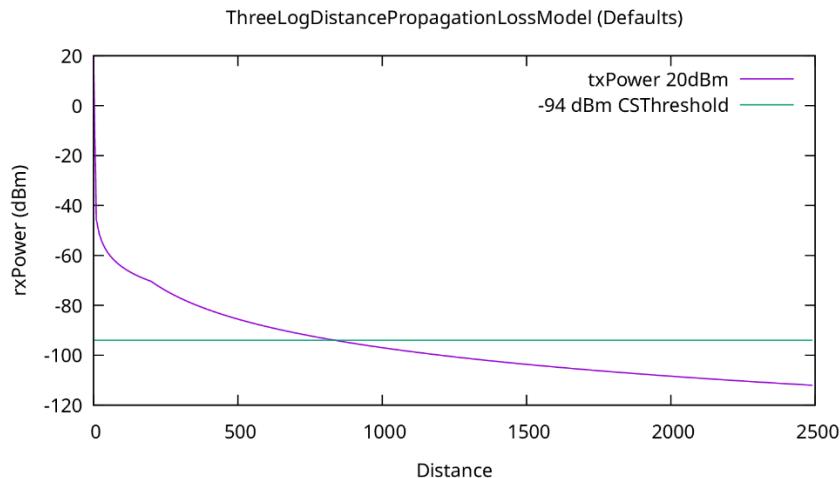
Hình 1.17 Đồ thị kết quả mô phỏng của Jakes Propagation Loss Model với tần số Doppler là 477.9 Hz và độ phân giải 1ms



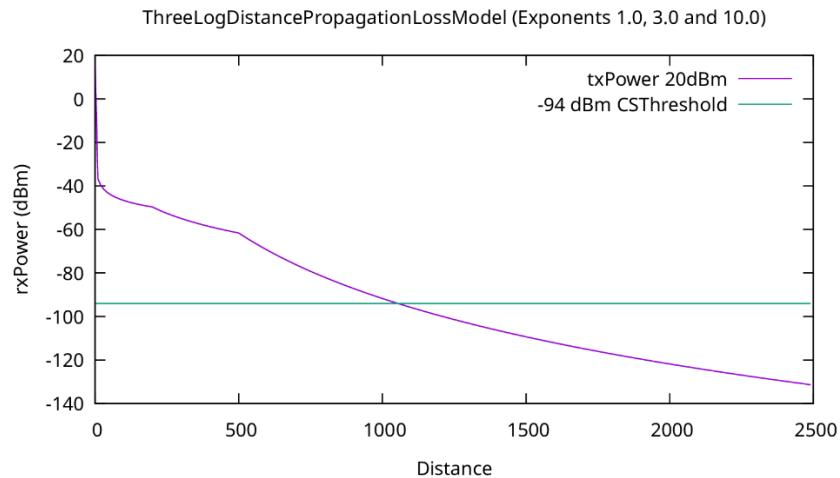
Hình 1.18 Đồ thị kết quả mô phỏng của Jakes Propagation Loss Model với tần số Doppler là 477.9 Hz độ phân giải 0.1ms

- **Đồ thị:** Đồ thị thể hiện sự thay đổi công suất tín hiệu theo thời gian do hiệu ứng Doppler, đặc biệt trong các kênh di động. Khi độ phân giải thời gian nhỏ hơn (0.1 ms), đồ thị cho thấy biến động nhanh hơn so với độ phân giải lớn hơn (1 ms).
- **Phân tích:** Mô hình Jakes cho thấy rõ hiệu ứng Doppler trong các mạng di động, nơi tín hiệu thay đổi liên tục khi có sự di chuyển giữa nguồn phát và nguồn thu. Hiệu ứng Doppler rõ rệt hơn khi độ phân giải thời gian giảm, thể hiện tần số của sự thay đổi tín hiệu cao hơn.
- **Ứng dụng thực tế:** Mô hình Jakes phù hợp cho các mạng di động như 4G, 5G, nơi các thiết bị thường xuyên di chuyển và hiệu ứng Doppler ảnh hưởng lớn đến chất lượng tín hiệu.

5.5 Kết quả của Three-Log-Distance Propagation Loss Model



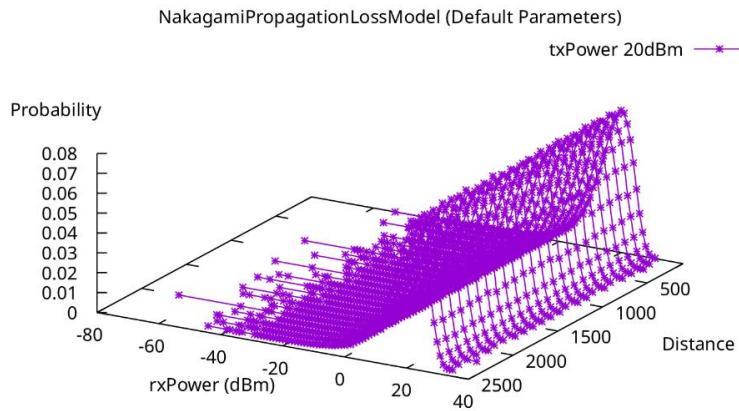
Hình 1.19 Đồ thị kết quả mô phỏng của Three-Log-Distance Propagation Loss Model với các hệ số suy hao mặc định



Hình 1.20 Đồ thị kết quả mô phỏng của Three-Log-Distance Propagation Loss Model với các hệ số suy hao là 1.0, 3.0, và 10.0

- Đồ thị:** Đồ thị cho thấy mức độ suy hao tín hiệu khác nhau ở ba vùng khoảng cách, với các hệ số suy hao khác nhau. Khi khoảng cách tăng, công suất tín hiệu giảm mạnh hơn, đặc biệt ở các vùng có hệ số suy hao lớn.
- Phân tích:** Mô hình này cung cấp một cái nhìn chi tiết về sự thay đổi suy hao tín hiệu theo khoảng cách trong các môi trường mà điều kiện truyền sóng thay đổi đáng kể. Kết quả cho thấy tín hiệu giảm nhanh hơn khi ở các vùng có hệ số suy hao lớn.
- Ứng dụng thực tế:** Three-Log-Distance phù hợp cho các môi trường hỗn hợp, như ở khu vực ngoại ô, nơi có sự chuyển đổi giữa các môi trường như không gian mở và khu vực có nhiều tòa nhà.

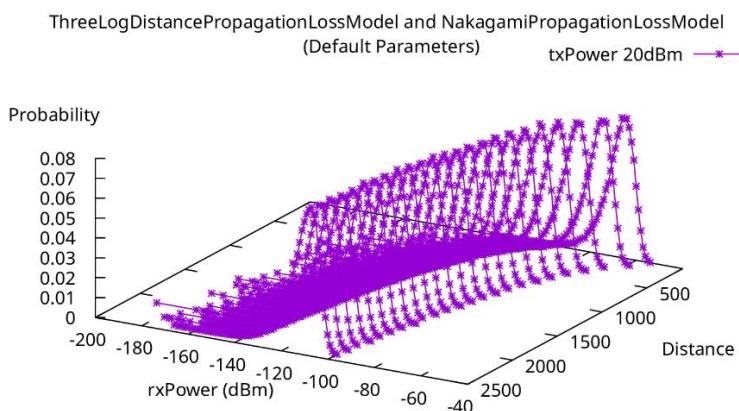
5.6 Kết quả của Nakagami Propagation Loss Model



Hình 1.21 Đồ thị kết quả mô phỏng của Nakagami Propagation Loss Model với các tham số mặc định

- **Đồ thị:** Đồ thị của Nakagami cho thấy sự phân bố xác suất của công suất tín hiệu tại các khoảng cách khác nhau, mô phỏng hiện tượng fading trong môi trường đa đường.
- **Phân tích:** Mô hình Nakagami mô phỏng chính xác hiện tượng fading, trong đó tín hiệu có thể dao động do ảnh hưởng của các chướng ngại vật hoặc phản xạ. Điều này thể hiện đặc điểm của các kênh fading đa đường, phù hợp cho các khu vực có nhiều chướng ngại vật.
- **Ứng dụng thực tế:** Mô hình Nakagami được ứng dụng rộng rãi trong các môi trường đô thị hoặc các khu vực có nhiều chướng ngại vật, nơi hiện tượng fading tác động lớn đến tín hiệu.

5.7 Kết quả kết hợp Three-Log-Distance Loss Model và Nakagami Propagation Loss Model



Hình 1.22 Đồ thị kết quả mô phỏng của kết hợp Three-Log-Distance Loss Model và Nakagami Propagation Loss Model với các tham số mặc định

- **Đồ thị:** Đồ thị thể hiện sự suy hao tín hiệu theo khoảng cách kết hợp với sự phân bố xác suất do hiện tượng fading, cho thấy mức độ ảnh hưởng đồng thời của khoảng cách và fading.
- **Phân tích:** Mô hình kết hợp này cho thấy cả hai yếu tố khoảng cách và fading đều có tác động mạnh đến tín hiệu. Sự suy hao tín hiệu do khoảng cách được điều chỉnh theo các hệ số trong Three-Log-Distance, trong khi fading được mô phỏng bởi phân phối xác suất của Nakagami.
- **Ứng dụng thực tế:** Mô hình này phù hợp cho các mạng không dây phức tạp, chẳng hạn như mạng di động trong đô thị với nhiều tòa nhà cao tầng và vật cản, nơi cả khoảng cách và fading đều ảnh hưởng đến chất lượng tín hiệu.

6. Kết luận

Qua dự án này, ta đã thành công trong việc cài đặt và mô phỏng các mô hình suy hao truyền sóng khác nhau trong NS3, bao gồm Friis, Log-Distance, Random với phân phối Exponential, Jakes, Three-Log-Distance, Nakagami, và mô hình kết hợp giữa Three-Log-Distance và Nakagami. Các kết quả mô phỏng đã cho thấy những đặc điểm riêng biệt của từng mô hình và cung cấp cái nhìn rõ ràng về sự suy hao tín hiệu trong các điều kiện môi trường khác nhau.

Mô hình Friis: Phù hợp cho môi trường không gian tự do, không có vật cản. Kết quả cho thấy suy hao tín hiệu theo khoảng cách với quy luật nghịch đảo bình phương. Mô hình này hữu ích trong các ứng dụng lý thuyết hoặc trong điều kiện phòng thí nghiệm.

Mô hình Log-Distance: Mô phỏng sự suy hao tín hiệu trong môi trường có nhiều chướng ngại vật, chẳng hạn như đô thị. Với hệ số suy hao điều chỉnh, mô hình này giúp mô phỏng các môi trường thực tế hơn, đặc biệt là trong các mạng di động trong thành phố.

Mô hình Random với phân phối Exponential: Thể hiện sự thay đổi ngẫu nhiên của tín hiệu trong các môi trường không ổn định hoặc có nhiều nguồn gây nhiễu. Mô hình này phù hợp cho các khu vực có sự thay đổi liên tục trong các điều kiện truyền sóng.

Mô hình Jakes: Mô phỏng hiệu ứng Doppler, thích hợp cho các kênh di động, nơi có sự di chuyển giữa các thiết bị. Đây là mô hình cần thiết cho các mạng di động như 4G và 5G.

Mô hình Three-Log-Distance: Mô hình này cho phép chia khoảng cách thành ba vùng khác nhau với các hệ số suy hao riêng, thích hợp cho các môi trường có sự thay đổi điều kiện theo khoảng cách, như ở khu vực chuyển đổi giữa đô thị và ngoại ô.

Mô hình Nakagami: Mô phỏng hiện tượng fading trong các môi trường đa đường, chẳng hạn như các khu vực đô thị có nhiều tòa nhà cao tầng và vật cản. Đây là mô hình hữu ích cho việc đánh giá ảnh hưởng của fading đến chất lượng tín hiệu.

Mô hình kết hợp giữa Three-Log-Distance và Nakagami: Cho phép mô phỏng đồng thời cả sự suy hao tín hiệu theo khoảng cách và hiện tượng fading, tạo ra một mô hình thực tế cho các ứng dụng mạng không dây phức tạp, đặc biệt là trong môi trường đô thị với nhiều tòa nhà và vật cản.

II. Sử dụng NetAnim để mô phỏng một kiến trúc mạng (Network Topology)

1. Giới thiệu

1.1. Giới thiệu về NetAnim

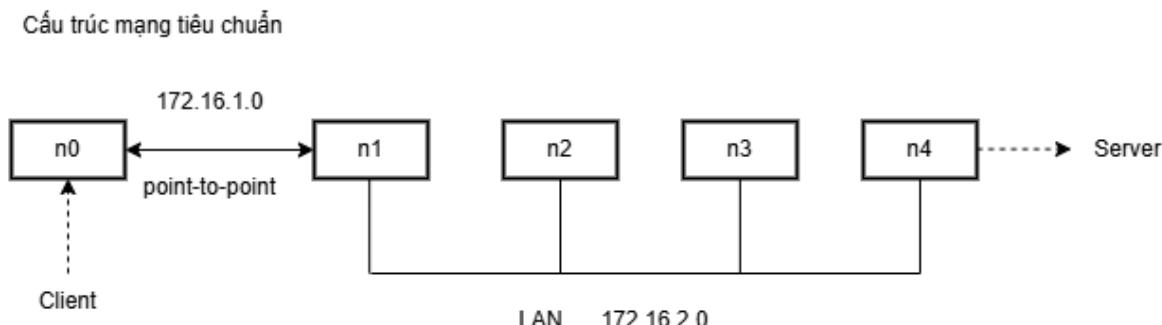
NetAnim là trình tạo hoạt ảnh ngoại tuyến dựa trên bộ công cụ Qt. Nó tạo hoạt ảnh mô phỏng được thực hiện trước đó bằng cách sử dụng tệp XML được tạo trong quá trình mô phỏng.

1.2. Giới thiệu về dự án mô phỏng

Ở bài mô phỏng này chúng ta sẽ tiến hành mô phỏng một mạng Network Topology sử dụng thư viện NS3 và tool NetAnim với kết nối điểm-điểm (P2P) và một mạng LAN CSMA (Ethernet).

Nút nguồn sẽ gửi một gói đến nút đích và các gói sẽ được đưa lên bus để mọi nút trong mạng đều nhận được gói đó và bất kỳ nút nào và chỉ dành cho nút đó chấp nhận gói.

Cấu trúc mạng của bài thí nghiệm mô phỏng này như sau:



Hình 2.1 Cấu trúc mạng P2P kết hợp với CSMA LAN (Network Topology)

- **Mô hình mạng:** Node 0 là Client, kết nối với Node 1 qua liên kết P2P. Node 1 kết nối với các node khác (2, 3, và 4) trong một mạng LAN CSMA. Node 4 đóng vai trò là Server.
- **Giao tiếp:** Client (Node 0) gửi các gói tin UDP echo đến Server (Node 4) qua mạng.
- **Hoạt ảnh và truy vết gói tin:** NetAnim dùng để hiển thị quá trình truyền gói tin, ghi nhận chi tiết như thời gian, nguồn và đích đến của các gói tin, nhằm xác minh kết nối mạng theo kịch bản đã cấu hình trong mã lệnh.
- **Địa chỉ IP cho đường truyền P2P:** 172.16.1.0; **địa chỉ IP cho đường truyền mạng LAN CSMA:** 172.16.2.0

2. Cơ sở lý thuyết

2.1. Giới thiệu về P2P

Đây là một loại kết nối trực tiếp giữa hai thiết bị mạng. Trong mô hình này, hai thiết bị (chẳng hạn như hai máy tính, hoặc một máy tính và một thiết bị mạng khác) được kết nối với nhau mà không cần thông qua bất kỳ thiết bị trung gian nào, như bộ định tuyến hoặc switch.

2.2. Giới thiệu về mạng LAN

Mạng LAN (Local Area Network) là một mạng máy tính cục bộ, cho phép các thiết bị kết nối và trao đổi dữ liệu với nhau trong một khu vực địa lý nhỏ, như trong một tòa nhà, văn phòng, trường học, hoặc nhà ở. Mạng LAN được sử dụng để kết nối các thiết bị như máy tính, máy in, điện thoại IP, và các thiết bị khác để chia sẻ tài nguyên và dữ liệu một cách nhanh chóng và hiệu quả.

2.3. Giới thiệu về CSMA

Carrier Sense Multiple Access (CSMA) là một phương pháp được sử dụng trong mạng máy tính để quản lý cách các thiết bị chia sẻ một kênh truyền thông để truyền dữ liệu giữa hai thiết bị.

Trong giao thức này, mỗi thiết bị trước đó sẽ nhận được kênh trước khi gửi dữ liệu. Nếu kênh đang bận, thiết bị sẽ được chờ đợi khi kênh đó miễn phí. Điều này giúp giảm thiểu, khi hai thiết bị gửi dữ liệu cùng một lúc, đảm bảo truyền thông mượt mà hơn trên mạng.

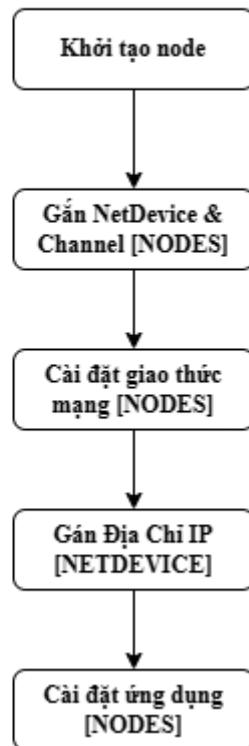
CSMA thường được sử dụng trong các công nghệ như Ethernet và Wi-Fi.

Thiết bị CSMA NS3 mô hình hóa một mạng đơn giản theo chuẩn Ethernet. Ethernet thực tế sử dụng sơ đồ **CSMA/CD (Carrier Sense Multiple Access with Collision Detection)** với độ trễ tăng theo cấp số nhân để chia sẻ môi trường truyền dẫn dùng chung. Các mô hình kênh và thiết bị CSMA NS3 chỉ là một tập hợp con của mô hình này.

2.4. Quy trình khởi tạo topology trong NS3

Việc khởi tạo một cấu trúc mạng (Network Topology) được xem là bước đầu trong quá trình mô phỏng mạng. Và tất cả các cấu trúc mạng (Network Topology) đều tuân theo các bước khởi tạo sau: Gồm có 5 bước khởi tạo cấu trúc mạng (Network Topology):

Quy trình khởi tạo kiến trúc mạng



Hình 2.2 Quy trình tạo kiến trúc mạng trong NS3

Bước 1: Khởi tạo node: Bắt đầu bằng cách tạo các node, đại diện cho các thiết bị trong mạng (ví dụ: máy tính, bộ định tuyến). Trong NS3, các node là các thành phần cơ bản, đóng vai trò là điểm cuối hoặc thiết bị trung gian trong mạng.

Bước 2: Gắn NetDevice & Channel [NODES]: Đặt các thuộc tính trên đó ngay bây giờ tốc độ dữ liệu tốc độ khung hình là gì và một số thứ khác nữa và bạn phải xác định kênh cũng như các thuộc tính của kênh ngay sau khi bước thứ hai hoàn tất.

Bước 3: Cài đặt giao thức mạng [NODES]: Cài đặt giao thức mạng cần thiết trên các node, thường là giao thức Internet Stack (TCP/IP) trong NS3, bao gồm các giao thức cần thiết cho việc giao tiếp (IP, TCP/UDP, v.v.).

Bước 4: Gán Địa Chỉ IP [NETDEVICE]: Gán địa chỉ IP cho mỗi thiết bị mạng. Điều này cho phép các node giao tiếp trong mạng bằng các giao thức dựa trên IP.

Bước 5: Cài đặt ứng dụng [NODES]: Cài đặt các ứng dụng trên các node để tạo và nhận lưu lượng. Các ứng dụng này có thể là các chương trình tạo dữ liệu, như gửi gói tin hoặc mô phỏng các tương tác máy chủ-khách.

3. Các bước cài đặt và tiến hành bài mô phỏng

3.1. Các bước cài đặt NetAnim

Bước 1: Cài đặt thư viện bắt buộc cho NetAnim

Đầu tiên chúng ta sẽ cập nhật danh sách các gói phần mềm từ các kho lưu trữ trên Ubuntu, đảm bảo rằng khi cài đặt phần mềm thì các gói:

sudo apt-get update

```
Get:8 http://vn.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [305 kB]
Get:9 http://vn.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:10 http://vn.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:11 http://vn.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [212 B]
Get:12 http://vn.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [21.0 kB]
Get:13 http://vn.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:14 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:15 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [7,192 B]
Get:16 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:17 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [51.9 kB]
Get:18 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Fetched 2,200 kB in 3s (817 kB/s)
Reading package lists... Done
hoangbaolong@hoangbaolong:~$
```

Hình 2.3 Kết quả sau khi chạy lệnh sudo apt-get update

Lệnh sau cài đặt các công cụ phát triển và thư viện Qt4/Qt5 nhằm tạo ra GUI:

sudo apt-get install qt4-dev-tools qt4-qmake, qt4-qmake qt4-default

hoặc **sudo apt-get install qt5-default**

Trong đó:

- **qt4-dev-tools:** Các công cụ phát triển cho Qt4.
- **qt4-qmake:** Các công cụ phát triển cho Qt4.
- **qt4-default:** Cấu hình mặc định cho các ứng dụng dựa trên Qt4, đảm bảo qmake biết sử dụng thư viện Qt4.

Bước 2: Tải NetAnim: Đã nằm sẵn trong gói NS3, nếu như chưa có NetAnim thì tải ở link sau: <https://www.nsnam.org/releases/ns-3-24/download/>

Bước 3: Cài đặt NetAnim: Sau khi tải xong, chúng ta sẽ chuyển đến thư mục netanim-3.109 trong ns-allinone-3.43:

cd ns-allinone-3.43/netanim-3.109/

Tiếp theo chúng ta sẽ xóa các lệnh tạm thời của các bản dựng trước đó để tránh xung đột trong quá trình biên dịch mới:

make clean

```
hoangbaolong@hoangbaolong:~/ns-allinone-3.43/netanim-3.109$ make clean
rm -f qrc_resources.cpp qrc_qtpropertybrowser.cpp
rm -f moc_prelude.h
rm -f moc_animatorscene.cpp moc_animpacket.cpp moc_netanim.cpp moc_animatormode.cpp moc_statsmode.cpp moc_qtvariantproperty.cpp moc_qtreetropertybrowser.cpp moc_qtpropertymanager.cpp moc_qtpropertybrowserutils_p.cpp moc_qtpropertybrowser.cpp moc_qtgroupboxpropertybrowser.cpp moc_qteditorfactory.cpp moc_qtbuttonpropertybrowser.cpp moc_animpropertybrowser.cpp moc_filepathmanager.cpp moc_fileeditfactory.cpp moc_fileedit.cpp moc_packetsmode.cpp moc_table.cpp moc_qcUSTOMplot.cpp
rm -f qtreetropertybrowser.moc qtpropertymanager.moc qteditorfactory.moc
rm -f main.o log.o fatal-error.o fatal-impl.o logqt.o resizableitem.o animnode.o animatorscene.o animpacket.o netanim.o animatormode.o mode.o animxmlparser.o animatorview.o animlink.o animresource.o statsview.o statsmode.o routingxmlparser.o routingstatsscene.o interfacestatsscene.o flowmonxmlparser.o flowmonstatsscene.o textbubble.o qtvariantproperty.o qtreetropertybrowser.o qtpropertymanager.o qtpropertybrowserutils.o qtpropertybrowser.o qtgroupboxpropertybrowser.o qteditorfactory.o qtbuttonpropertybrowser.o animpropertybrowser.o filepathmanager.o fileeditfactory.o packetsmode.o packetsview.o packetsscene.o graphpacket.o table.o countertablesscene.o qcUSTOMplot.o qrc_resources.o qrc_qtpropertybrowser.o moc_animatorscene.o moc_animpacket.o moc_netanim.o moc_animatormode.o moc_statsmode.o moc_qtpropertybrowserutils_p.o moc_animpropertybrowser.o moc_filepathmanager.o moc_fileeditfactory.o moc_packetsmode.o moc_table.o m
```

Hình 2.4 Kết quả sau khi chạy lệnh make clean

Thực hiện biên dịch mã nguồn NetAnim từ các tệp mã nguồn, lệnh này sẽ tạo các tệp thực thi từ mã nguồn:

make

```
hoangbaolong@hoangbaolong:~/ns-allinone-3.43/netanim-3.109$ make
g++ -c -pipe -O2 -Wall -Wextra -D_REENTRANT -fPIC -DNS3_LOG_ENABLE -DQT_NO_DEBUG -DQT_PRINTSUPPORT_LIB -DQT_WIDGETS_LIB -DQT_GUI_LIB -DQT_CORE_LIB -I . -I qtpropertybrowser/src -I /usr/include/x86_64-linux-gnu/qt5/QtPrintSupport -I /usr/include/x86_64-linux-gnu/qt5/QtWidgets -I /usr/include/x86_64-linux-gnu/qt5/QtGui -I /usr/include/x86_64-linux-gnu/qt5/QtCore -I . -I /usr/lib/x86_64-linux-gnu/qt5/mkspecs/linux-g++ -o main.o main.cpp
g++ -c -pipe -O2 -Wall -Wextra -D_REENTRANT -fPIC -DNS3_LOG_ENABLE -DQT_NO_DEBUG -DQT_PRINTSUPPORT_LIB -DQT_WIDGETS_LIB -DQT_GUI_LIB -DQT_CORE_LIB -I . -I qtpropertybrowser/src -I /usr/include/x86_64-linux-gnu/qt5/QtPrintSupport -I /usr/include/x86_64-linux-gnu/qt5/QtWidgets -I /usr/include/x86_64-linux-gnu/qt5/QtGui -I /usr/include/x86_64-linux-gnu/qt5/QtCore -I . -I /usr/lib/x86_64-linux-gnu/qt5/mkspecs/linux-g++ -o log.o log.cpp
g++ -c -pipe -O2 -Wall -Wextra -D_REENTRANT -fPIC -DNS3_LOG_ENABLE -DQT_NO_DEBUG -DQT_PRINTSUPPORT_LIB -DQT_WIDGETS_LIB -DQT_GUI_LIB -DQT_CORE_LIB -I . -I qtpropertybrowser/src -I /usr/include/x86_64-linux-gnu/qt5/QtPrintSupport -I /usr/include/x86_64-linux-gnu/qt5/QtWidgets -I /usr/include/x86_64-linux-gnu/qt5/QtGui -I /usr/include/x86_64-linux-gnu/qt5/QtCore -I . -I /usr/lib/x86_64-linux-gnu/qt5/mkspecs/linux-g++ -o fatal-error.o fatal-error.cpp
g++ -c -pipe -O2 -Wall -Wextra -D_REENTRANT -fPIC -DNS3_LOG_ENABLE -DQT_NO_DEBUG -DQT_PRINTSUPPORT_LIB -DQT_WIDGETS_LIB -DQT_GUI_LIB -DQT_CORE_LIB -I . -I qtpropertybrowser/src -I /usr/include/x86_64-linux-gnu/qt5/QtPrintSupport -I /usr/include/x86_64-linux-gnu/qt5/QtWidgets -I /usr/include/x86_64-linux-gnu/qt5/QtGui -I /usr/include/x86_64-linux-gnu/qt5/QtCore -I . -I /usr/lib/x86_64-linux-gnu/qt5/mkspecs/linux-g++ -o fatal-impl.o fatal-impl.cpp
g++ -c -pipe -O2 -Wall -Wextra -D_REENTRANT -fPIC -DNS3_LOG_ENABLE -DQT_NO_DEBUG -DQT_PRINTSUPPORT_LIB -DQT_WIDGETS_LIB -DQT_GUI_LIB -DQT_CORE_LIB -I . -I qtpropertybrowser/src -I /usr/include/x86_64-linux-gnu/qt5/QtPrintSupport -I /usr/include/x86_64-linux-gnu/qt5/QtWidgets -I /usr/include/x86_64-linux-gnu/qt5/QtGui -I /usr/include/x86_64-linux-gnu/qt5/QtCore -I . -I /usr/lib/x86_64-linux-gnu/qt5/mkspecs/linux-g++ -o logqt.o logqt.cpp
g++ -c -pipe -O2 -Wall -Wextra -D_REENTRANT -fPIC -DNS3_LOG_ENABLE -DQT_NO_DEBUG -DQT_PRINTSUPPORT_LIB -DQT_WIDGETS_LIB -DQT_GUI_LIB -DQT_CORE_LIB -I . -I qtpropertybrowser/src -I /usr/include/x86_64-linux-gnu/qt5/QtPrintSupport -I /usr/include/x86_64-linux-gnu/qt5/QtWidgets -I /usr/include/x86_64-linux-gnu/qt5/QtGui -I /usr/include/x86_64-linux-gnu/qt5/QtCore -I . -I /usr/lib/x86_64-linux-gnu/qt5/mkspecs
```

Hình 2.5 Kết quả sau khi chạy lệnh make

Tạo file Makefile từ tệp cấu hình NetAnim.pro (đây là tệp cấu hình qmake cho dự án NetAnim:

qmake NetAnim.pro

```
hoangbaolong@hoangbaolong:~/ns-allinone-3.43/netanim-3.109$ qmake NetAnim.pro  
Show Apps hoangbaolong@hoangbaolong:~/ns-allinone-3.43/netanim-3.109$
```

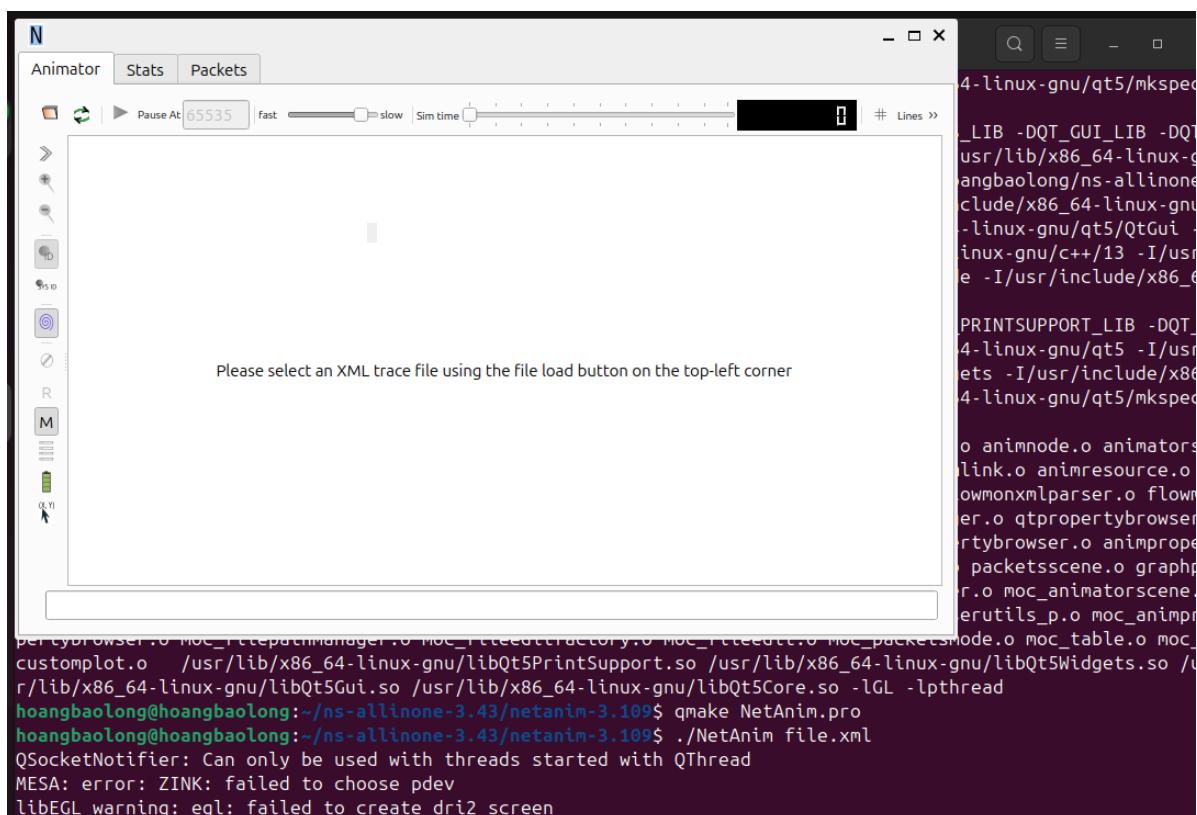
Hình 2.6 Kết quả sau khi chạy lệnh qmake NetAnim.pro

Tiếp theo chúng ta sẽ chạy chương trình NetAnim với tệp đầu vào là file.xml (chúng ta có thể thay đổi tên file nếu muốn chạy file khác), và đây là các tệp kết quả từ các mô phỏng mạng trong ns-3, tệp này sẽ chứa các thông tin về mô phỏng mà NetAnim sẽ hiển thị:

./NetAnim file.xml

```
hoangbaolong@hoangbaolong:~/ns-allinone-3.43/netanim-3.109$ ./NetAnim file.xml
```

Hình 2.7 Chạy lệnh mở cửa sổ NetAnim



Hình 2.8 Cửa sổ của NetAnim sau khi chạy lệnh ./NetAnim file.xml

3.2. Giải thích chương trình second.cc

Dưới đây là chương trình của bài mô phỏng second.cc:

```
#include "NS3/core-module.h"  
#include "NS3/network-module.h"
```

```

#include "NS3/csma-module.h"
#include "NS3/internet-module.h"
#include "NS3/point-to-point-module.h"
#include "NS3/applications-module.h"
#include "NS3/ipv4-global-routing-helper.h"
#include "NS3/netanim-module.h"

// Default Network Topology
//
//      172.16.1.0
// n0 ----- n1  n2  n3  n4 -> Server
//   point-to-point |  |  |  |
//                   =====
//                   LAN 172.16.2.0
//n0 is the client

using namespace NS3;

NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");

int main (int argc, char *argv[])
{
    bool verbose = true;
    uint32_t nCsma = 3;
    // unsigned integer 32 bits
    CommandLine cmd;
    cmd.AddValue ("nCsma", "Number of \\\"extra\\\" CSMA nodes/devices", nCsma);
    cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);

    cmd.Parse (argc, argv);

    if (verbose)
    {
        LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
        LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    }

    nCsma = nCsma == 0 ? 1 : nCsma;

    NodeContainer p2pNodes;

```

```

p2pNodes.Create (2);

NodeContainer csmaNodes;
csmaNodes.Add (p2pNodes.Get (1)); //make the first node
csmaNodes.Create (nCsma); //nCsma=3 already declared

PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install (p2pNodes);

CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);

InternetStackHelper stack;
stack.Install (p2pNodes.Get (0));
//stack.Install(p2pNodes.Get(1)); there is no need to install
stack.Install (csmaNodes);
// p2pNodes.Get(1) = csmaNodes.Get(0)

Ipv4AddressHelper address;
address.SetBase ("172.16.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign (p2pDevices);

address.SetBase ("172.16.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);

UdpEchoServerHelper echoServer (1234);

ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get
(nCsma));
//The Server is csmaNodes.Get(3) - n4 is the server

```

```

serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 1234);
echoClient.SetAttribute ("MaxPackets", UintegerValue (3));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (2048));

ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get (0));
//n0 is the client in the topology
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

pointToPoint.EnablePcapAll ("p2p");
//pcap will be opened using either wireshark or tcpdump
csma.EnablePcap ("csma1", csmaDevices.Get (1), true);
csma.EnablePcap ("csma3", csmaDevices.Get (3), true);
//Network Animation here.
AnimationInterface anim("second.xml");
anim.SetConstantPosition(p2pNodes.Get(0),10.0,10.0);
anim.SetConstantPosition(p2pNodes.Get(1),20.0,20.0);
anim.SetConstantPosition(csmaNodes.Get(1),30.0,30.0);
anim.SetConstantPosition(csmaNodes.Get(2),40.0,40.0);
anim.SetConstantPosition(csmaNodes.Get(3),50.0,50.0);
//Ascii Trace Metrics.
AsciiTraceHelper ascii;
pointToPoint.EnableAsciiAll(ascii.CreateFileStream("point2point.tr"));
csma.EnableAsciiAll(ascii.CreateFileStream("csma2.tr"));

Simulator::Run ();
Simulator::Destroy ();
return 0;
}

```

Giải thích chi tiết:

Khai báo các thư viện/ các module:

```

#include "NS3/core-module.h"
#include "NS3/network-module.h"
#include "NS3/csma-module.h"

```

```
#include "NS3/internet-module.h"
#include "NS3/point-to-point-module.h"
#include "NS3/applications-module.h"
#include "NS3/ipv4-global-routing-helper.h"
#include "NS3/netanim-module.h"
```

Trong trường hợp này, chúng ta có thể thấy rằng chúng tôi sẽ mở rộng ví dụ điểm-điểm (liên kết giữa các nút n0 và n1 bên dưới) bằng cách treo mạng BUS ở phía bên phải. Lưu ý rằng đây là cấu trúc liên kết mạng mặc định vì bạn thực sự có thể thay đổi số lượng nút được tạo trên mạng LAN. Nếu bạn đặt nCsma thành một, sẽ có tổng cộng hai nút trên mạng LAN (kênh CSMA) - một nút bắt buộc và một nút "phụ". Theo mặc định có ba nút "phụ" như bên dưới:

```
// Default Network Topology
//
//      172.16.1.0
// n0 ----- n1  n2  n3  n4 -> Server
//   point-to-point |  |  |  |
//                   =====
//                   LAN 172.16.2.0
//n0 is the client
```

```
using namespace NS3;
```

```
NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");
```

Chúng ta sử dụng cờ verbose để xác định xem các thành phần logging *UdpEchoClientApplication* và *UdpEchoServerApplication* có được bật hay không. Cờ này mặc định là true (các thành phần ghi nhật ký được bật) nhưng cho phép chúng tôi tắt logging trong quá trình kiểm tra hồi quy của ví dụ này sau đây:

```
int main (int argc, char *argv[])
{
    bool verbose = true;
    uint32_t nCsma = 3;
    // unsigned integer 32 bits
    CommandLine cmd;
    cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
    cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);

    cmd.Parse (argc, argv);

    if (verbose)
```

```

{
    LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
    LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
}

```

Ta có 2 dòng code dưới đây sử dụng để kích hoạt tính năng ghi log và nó cụ thể như sau:

```

LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

```

Trong đó:

- **LogComponentEnable** được sử dụng để kích hoạt tính năng ghi log cho các thành phần cụ thể của ứng dụng. Cụ thể, hàm này cho phép bạn kích hoạt mức độ log của một thành phần (component) nào đó trong quá trình mô phỏng, giúp bạn theo dõi các thông tin về hoạt động của ứng dụng.
- **UdpEchoClientApplication**: Ứng dụng client gửi các gói dữ liệu UDP echo
- **UdpEchoServerApplication**: Ứng dụng server nhận và phản hồi các gói dữ liệu UDP echo.
- **LOG_LEVEL_INFO**: Là mức độ LOG, là mức độ thông tin mà bạn muốn ghi lại.

Đoạn code dưới đây đảm bảo rằng phải có ít nhất 1 node phụ trong CSMA:

```
nCsma = nCsma == 0 ? 1 : nCsma;
```

Bước tiếp theo là tạo hai nút mà chúng ta sẽ kết nối thông qua liên kết điểm-điểm:

```
NodeContainer p2pNodes;
p2pNodes.Create (2);
```

Tiếp theo, chúng ta tiến hành khai báo một *NodeContainer* khác để giữ các nút sẽ là một phần của mạng bus (CSMA). Đầu tiên, chúng ta chỉ khởi tạo đối tượng *container*.

Dòng mã tiếp theo *Get* nút đầu tiên (như có chỉ mục là một) từ vùng chứa nút điểm-điểm và thêm nó vào vùng chứa các nút sẽ nhận các thiết bị CSMA. Nút được đề cập sẽ kết thúc bằng thiết bị điểm-điểm và thiết bị CSMA. Sau đó, chúng tôi tạo một số nút "bổ sung" tạo nên phần còn lại của mạng CSMA. Vì chúng ta đã có một nút trong mạng CSMA – nút sẽ có cả thiết bị mạng điểm-điểm và CSMA, nên số nút "phụ" có nghĩa là số nút bạn mong muốn trong phần CSMA trừ đi một:

```
NodeContainer csmaNodes;
csmaNodes.Add (p2pNodes.Get (1)); //make the first node
csmaNodes.Create (nCsma); //nCsma=3 already declared
```

Đoạn mã tiếp theo sẽ khởi tạo *PointToPointHelper* để cài đặt các thuộc tính mạng point-to-point giữa hai nút mặc định liên quan để tạo bộ phát 5 megabit/giây trên các thiết bị được tạo bằng trình trợ giúp và độ trễ 2ms trên các kênh do trình trợ giúp tạo:

```
PointToPointHelper pointToPoint;  
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));  
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
```

Sau đó, chúng ta khởi tạo *NetDeviceContainer* để theo dõi các thiết bị mạng điểm-điểm và chúng tôi cài đặt thiết bị trên các nút điểm-điểm:

```
NetDeviceContainer p2pDevices;  
p2pDevices = pointToPoint.Install (p2pNodes);
```

Dòng lệnh dưới đây sẽ giới thiệu về *CsmaHelper*, *CsmaHelper* hoạt động giống như *PointToPointHelper* nhưng nó tạo và kết nối các thiết bị và kênh CSMA, chúng ta sẽ cấu hình mạng CSMA với băng thông 100Mbps và độ trễ 6560 nanoseconds:

```
CsmaHelper csma;  
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));  
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));
```

Tiếp theo chúng ta tạo *NetDeviceContainer* để chứa các thiết bị được tạo bởi *PointToPointHelper* chúng ta tạo *NetDeviceContainer* để chứa các thiết bị được tạo bởi *CsmaHelper* của chúng tôi. Chúng tôi gọi phương thức cài đặt của *CsmaHelper* để cài đặt thiết bị vào các nút của *csmaNodes NodeContainer*:

```
NetDeviceContainer csmaDevices;  
csmaDevices = csma.Install (csmaNodes);
```

Bây giờ chúng ta đã tạo các nút, thiết bị và kênh nhưng chúng ta không có ngăn xếp giao thức nào. Giống như trong tập file *first.cc*, chúng ta sẽ sử dụng *InternetStackHelper* để cài đặt các ngăn xếp này:

```
InternetStackHelper stack;  
stack.Install (p2pNodes.Get (0));  
//stack.Install(p2pNodes.Get(1)); there is no need to install  
stack.Install (csmaNodes);  
// p2pNodes.Get(1) = csmaNodes.Get(0)
```

Chúng ta sẽ sử dụng *Ipv4AddressHelper* để gán địa chỉ IP cho giao diện thiết bị của mình. Đầu tiên, chúng tôi sử dụng mạng 172.16.1.0 để tạo hai địa chỉ cần thiết cho hai thiết bị điểm-điểm của mình:

```
Ipv4AddressHelper address;  
address.SetBase ("172.16.1.0", "255.255.255.0");  
Ipv4InterfaceContainer p2pInterfaces;  
p2pInterfaces = address.Assign (p2pDevices);
```

Bây giờ chúng ta cần gán địa chỉ IP cho mạng CSMA của mình. Thao tác này hoạt động giống như đối với trường hợp điểm-điểm, ngoại trừ hiện tại chúng tôi đang thực hiện thao tác trên vùng chứa có số lượng thiết bị CSMA thay đổi — hãy nhớ rằng chúng tôi đã thực hiện số lượng thiết bị CSMA có thể thay đổi bằng đổi số dòng lệnh. Trong trường hợp này, các thiết bị CSMA sẽ được liên kết với các địa chỉ IP từ số mạng 172.16.2.0, như được thấy bên dưới:

```
address.SetBase ("172.16.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);
```

Bây giờ chúng ta đã xây dựng cấu trúc liên kết nhưng chúng tôi cần các ứng dụng. Phần này chúng ta sẽ khởi tạo máy chủ trên một trong các nút có thiết bị CSMA và máy khách trên nút chỉ có thiết bị điểm-điểm.

Đầu tiên, chúng tôi thiết lập máy chủ echo cho nút “phụ” cuối cùng của mạng CSMA (nút n4). Chúng tôi tạo *UdpEchoServerHelper* với cổng đích là 1234 (sau này khi tạo máy khách cũng sẽ phải trùng cổng đích này) và cung cấp giá trị thuộc tính bắt buộc cho hàm tạo là số cổng máy chủ. Hãy nhớ rằng cổng này có thể được thay đổi sau bằng cách sử dụng phương thức *SetAttribution* nếu muốn, nhưng chúng tôi yêu cầu nó phải được cung cấp cho hàm tạo:

```
UdpEchoServerHelper echoServer (1234);
ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));
//The Server is csmaNodes.Get(3) - n4 is the server
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
```

Trong hàm *echoServer.Install()*, *csmaNodes NodeContainer* chứa một trong các nút được tạo cho mạng điểm-điểm và các nút “phụ” nCsma. Những gì chúng tôi muốn đạt được là nút cuối cùng trong số các nút “phụ”. Mục nhập thứ 0 của vùng chứa *csmaNodes* sẽ là nút điểm-điểm. Khi đó, cách dễ dàng để nghĩ về điều này là nếu chúng ta tạo một nút CSMA “bổ sung” thì nút đó sẽ nằm ở chỉ mục một trong các vùng chứa *csmaNodes*. Bằng quy nạp, nếu chúng ta tạo các nút “phụ” nCsma thì nút cuối cùng sẽ có chỉ số nCsma. Điều này được thể hiện trong hàm *Get* của dòng mã đầu tiên.

Ở hai dòng lệnh cuối cùng:

```
serverApps.Start (Seconds (1.0));
```

Lệnh này khởi động ứng dụng máy chủ sau 1 giây từ khi mô phỏng bắt đầu. Điều này có nghĩa là ứng dụng máy chủ sẽ bắt đầu lắng nghe và chờ nhận các gói tin từ máy khách sau 1 giây.

```
serverApps.Stop (Seconds (10.0));
```

Lệnh này dừng ứng dụng máy chủ sau 10 giây kể từ khi mô phỏng bắt đầu. Sau thời điểm này, ứng dụng máy chủ sẽ không còn hoạt động và sẽ không tiếp nhận bất kỳ gói tin nào nữa.

Tiếp theo chúng ta khởi tạo ứng dụng máy khách (Client). *UdpEchoClientHelper* tạo một máy khách UDP và gửi gói tin đến địa chỉ IP của máy chủ trong mạng CSMA, được lấy từ *csmaInterfaces.GetAddress (nCsma)* và gửi đến cổng đích mà máy khách gửi đến là 1234, trùng với cổng của *UdpEchoServer* trên máy chủ. Chúng ta cài đặt máy khách trên nút điểm-điểm ngoài cùng bên trái (nút n0) được thấy trong hình minh họa cấu trúc liên kết:

```
UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 1234);
```

Tiếp theo chúng ta cung cấp các thuộc tính cho máy khách (Client):

```
echoClient.SetAttribute ("MaxPackets", UIntegerValue (3));  
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));  
echoClient.SetAttribute ("PacketSize", UIntegerValue (2048));
```

Trong đó:

- **MaxPackets** được đặt là 3, nghĩa là máy khách sẽ gửi tổng cộng 3 gói tin.
- **Interval** được đặt là 1.0 giây, tức là máy khách sẽ gửi một gói tin mỗi giây.
- **PacketSize** được đặt là 2048 byte, quy định kích thước của mỗi gói tin gửi đi.

Máy khách được cài đặt trên nút *p2pNodes.Get(0)*, là nút n0 trong cấu trúc mạng, đóng vai trò là máy khách trong quá trình mô phỏng:

```
ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get (0));  
//n0 is the client in the topology
```

Thiết lập thời gian mô phỏng cho máy khách:

```
clientApps.Start (Seconds (2.0));  
clientApps.Stop (Seconds (10.0));
```

Về cơ bản, điều xảy ra là mỗi nút hoạt động như thể nó là một bộ định tuyến OSPF, có thể giao tiếp ngay lập tức và kỳ diệu với tất cả các bộ định tuyến khác ở phía sau. Mỗi nút tạo quảng cáo liên kết và truyền chúng trực tiếp đến trình quản lý tuyến toàn cầu sử dụng thông tin chung này để xây dựng bảng định tuyến cho mỗi nút. Việc thiết lập hình thức định tuyến này chỉ là một bước:

```
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

Kích hoạt NetAnim để lưu trữ hoạt động của các nút vào tệp second.xml nhằm hiển thị trong NetAnim ở dòng đầu tiên, ngoài ra ở những dòng tiếp theo nhằm khởi tạo vị trí của các nodes mạng trong kiến trúc mạng ở trên chúng ta (node n0,n1,n2,n3,n4) ở phần mềm NetAnim:

```
AnimationInterface anim("second.xml");
anim.SetConstantPosition(p2pNodes.Get(0),10.0,10.0);
anim.SetConstantPosition(p2pNodes.Get(1),20.0,20.0);
anim.SetConstantPosition(csmaNodes.Get(1),30.0,30.0);
anim.SetConstantPosition(csmaNodes.Get(2),40.0,40.0);
anim.SetConstantPosition(csmaNodes.Get(3),50.0,50.0);
```

3.3. Các bước tiến hành chạy mô phỏng

Bước 1: Copy file *second.cc* sang folder *srcatch*, rồi sau đó đổi tên thành *mySecond.cc*

Bước 2: Chạy file *mySecond.cc*:

```
cd ns-allinone-3.43/ns-3.43
```

```
./ns3 run scratch/mySecond.cc
```

```
hoangbaolong@hoangbaolong:~$ cd ns-allinone-3.43/ns-3.43/
hoangbaolong@hoangbaolong:~/ns-allinone-3.43/ns-3.43$ ./ns3 run scratch/mySecond
.cc
```

Hình 2.9 Hiển thị chạy *./ns3 run scratch/mySecond.cc*

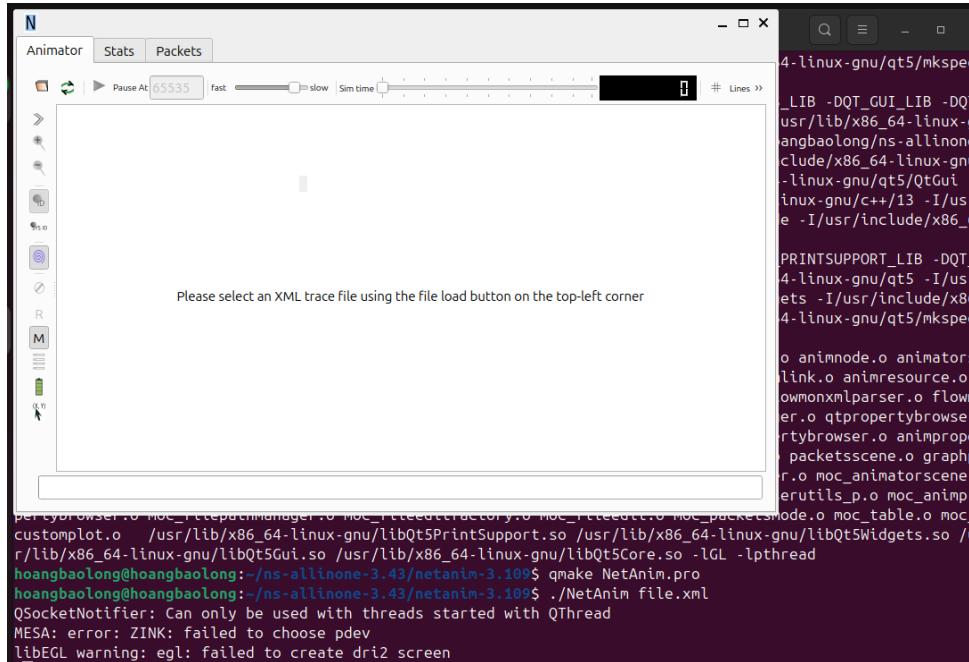
```
ice instead
Deprecation warning for name ns3::LrWpanNetDevice; use ns3::lrwpan::LrWpanNetDev
ice instead
Deprecation warning for name ns3::LrWpanNetDevice; use ns3::lrwpan::LrWpanNetDev
ice instead
Deprecation warning for name ns3::LrWpanNetDevice; use ns3::lrwpan::LrWpanNetDev
ice instead
Deprecation warning for name ns3::LrWpanNetDevice; use ns3::lrwpan::LrWpanNetDev
ice instead
Deprecation warning for name ns3::LrWpanNetDevice; use ns3::lrwpan::LrWpanNetDev
ice instead
Deprecation warning for name ns3::LrWpanNetDevice; use ns3::lrwpan::LrWpanNetDev
ice instead
At time +2s client sent 2048 bytes to 172.16.2.4 port 1234
At time +2.01262s server received 2048 bytes from 172.16.1.1 port 49153
At time +2.01262s server sent 2048 bytes to 172.16.1.1 port 49153
At time +2.02413s client received 2048 bytes from 172.16.2.4 port 1234
At time +3s client sent 2048 bytes to 172.16.2.4 port 1234
At time +3.00542s server received 2048 bytes from 172.16.1.1 port 49153
At time +3.00542s server sent 2048 bytes to 172.16.1.1 port 49153
At time +3.0109s client received 2048 bytes from 172.16.2.4 port 1234
At time +4s client sent 2048 bytes to 172.16.2.4 port 1234
At time +4.00542s server received 2048 bytes from 172.16.1.1 port 49153
At time +4.00542s server sent 2048 bytes to 172.16.1.1 port 49153
At time +4.0109s client received 2048 bytes from 172.16.2.4 port 1234
hoangbaolong@hoangbaolong:~/ns-allinone-3.43/ns-3.43$
```

Hình 2.10 Kết quả hiển thị sau khi chạy lệnh *./NS3 run scratch/mySecond.cc*

Bước 3: Mở terminal mở app NetAnim bằng lệnh:

`cd ns-allinone-3.43/netanim-3.109/`

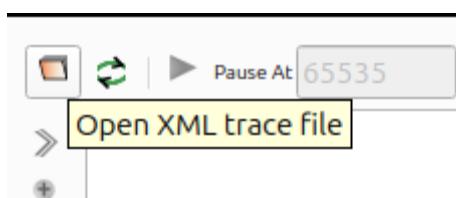
`./NetAnim`



Hình 2.11 Mở cửa sổ NetAnim

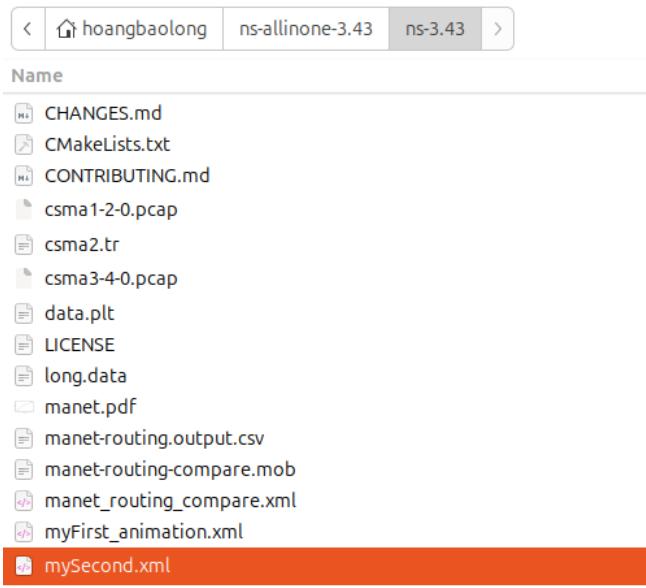
Bước 4: Chọn file tệp .xml để mô phỏng:

Đầu tiên chọn Open XML trace file:



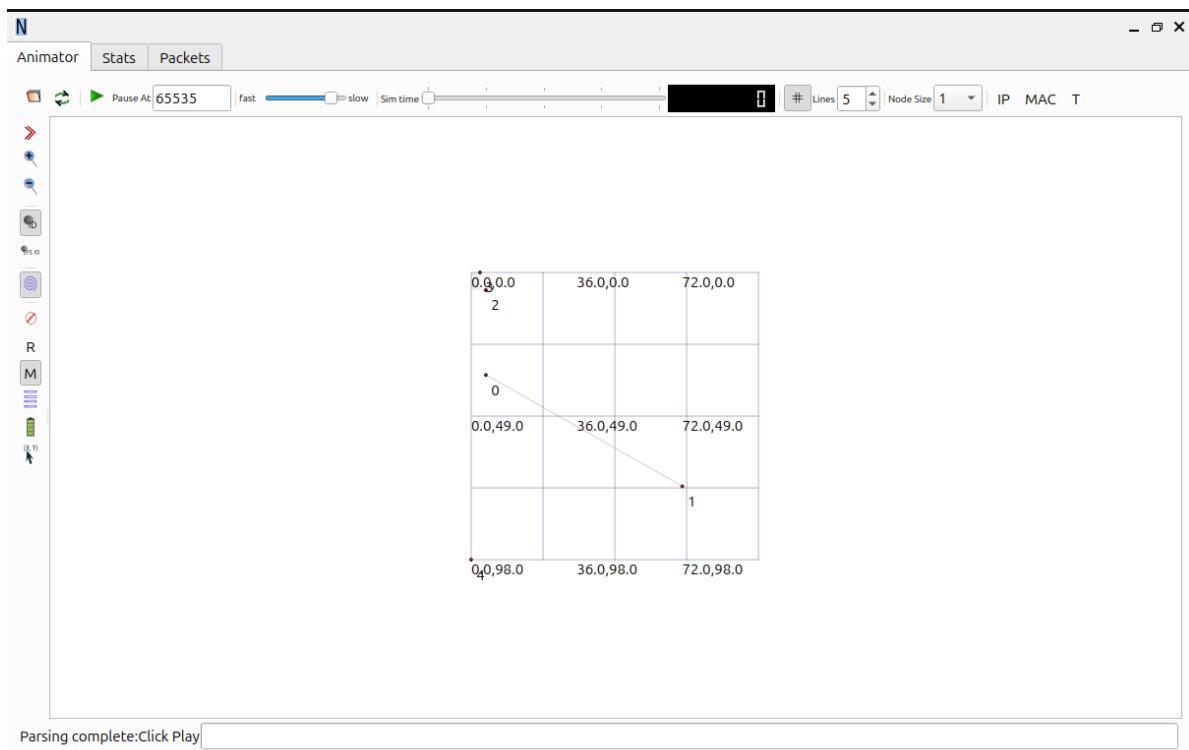
Hình 2.12 Mở file trace XML

Tiếp theo chọn file mySecond.xml:



Hình 2.13 Chọn file mySecond.xml để mô phỏng

Kết quả:



Hình 2.14 Sau khi chọn file mySecond.cc sẽ hiển thị như này

Bước 4: Chạy mô phỏng:

Chọn biểu tượng Play Animation:

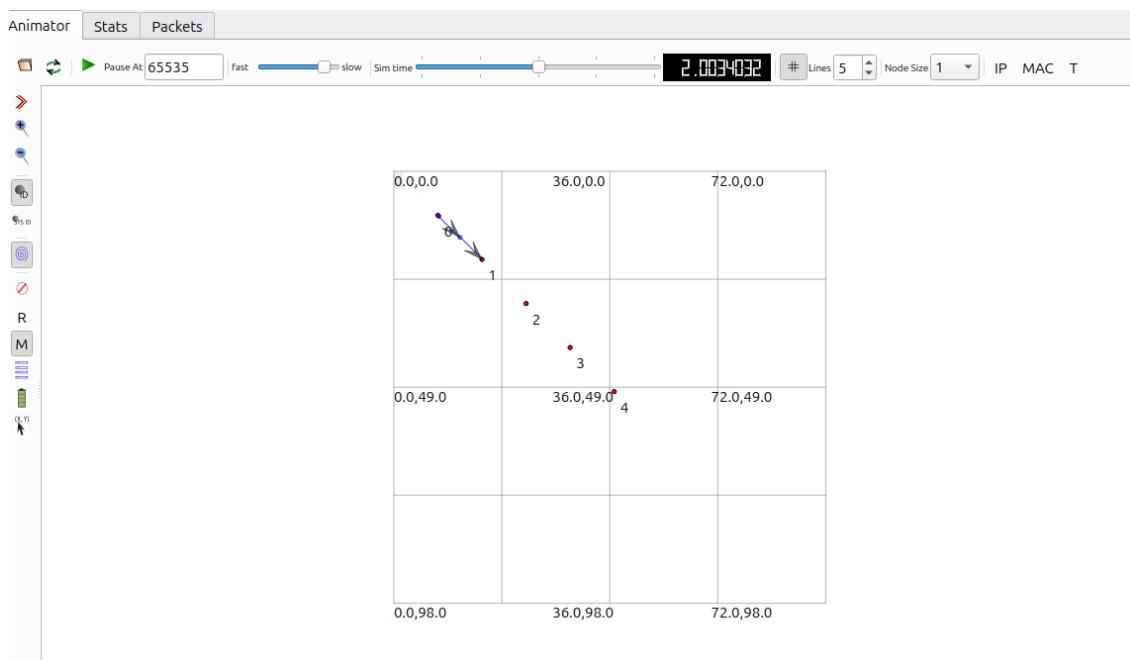


Hình 2.15 Nhấn chọn Play Animation để bắt đầu mô phỏng

4. Kết quả mô phỏng và nhận xét

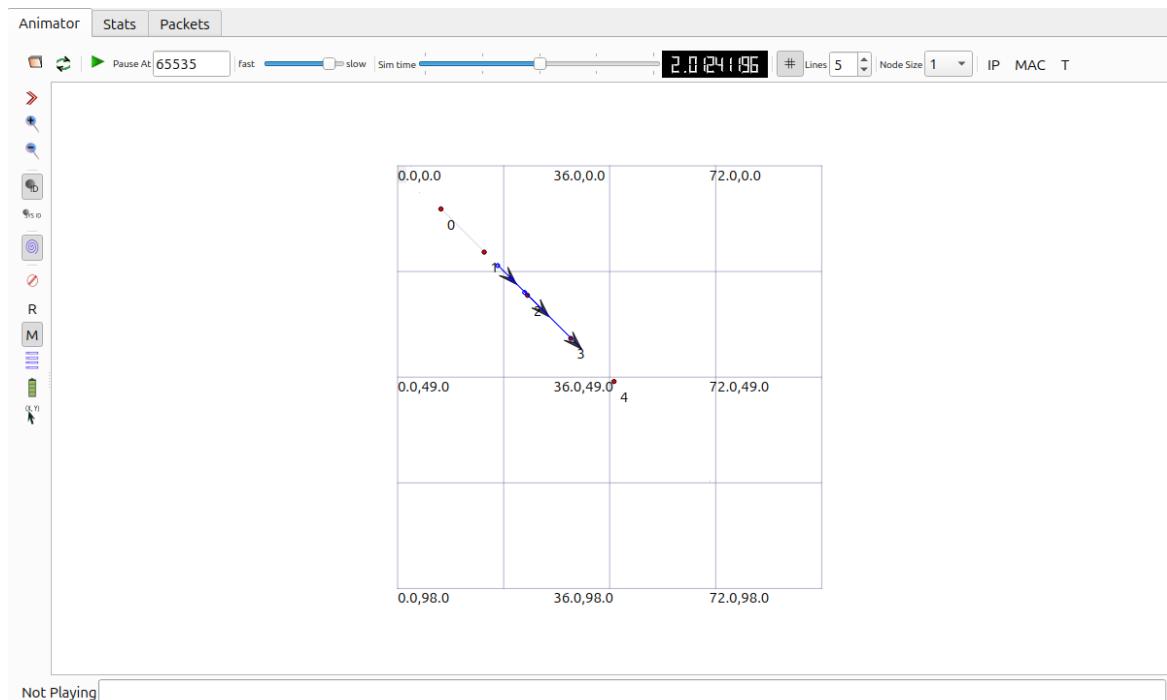
4.1. Kết quả mô phỏng

Đầu tiên sẽ có 3 gói tin được gửi từ node 0 (Client) đến node 1 theo kiểu truyền P2P:



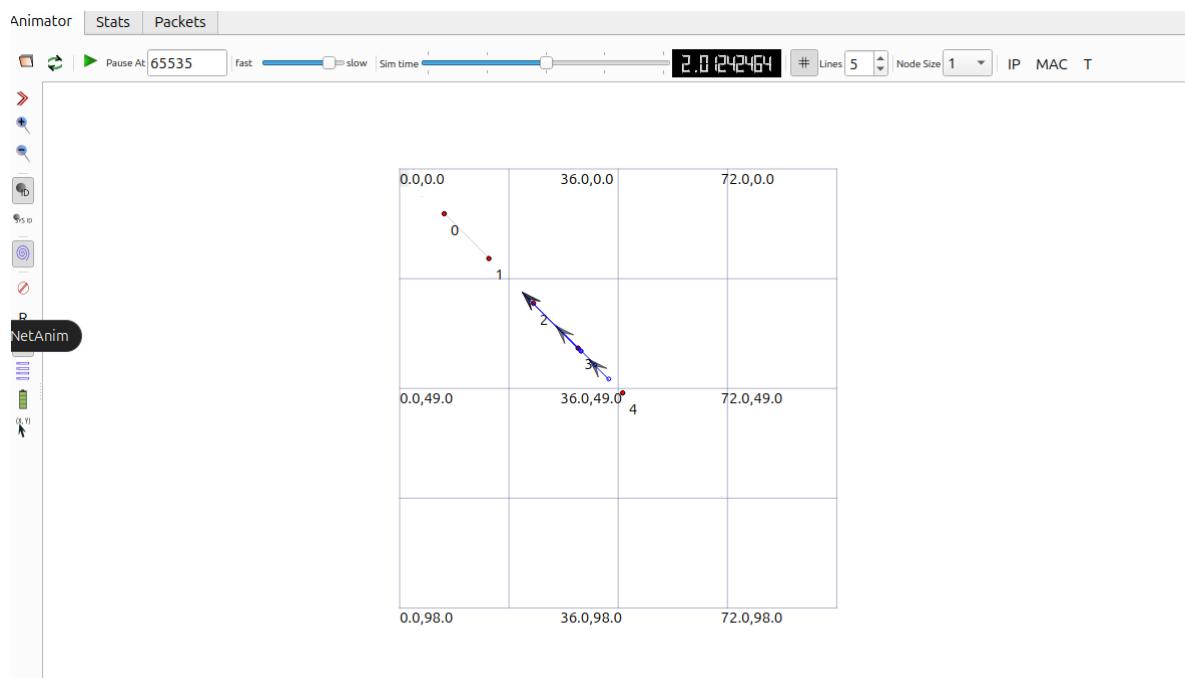
Hình 2.16 Gói tin gửi từ node 0 (Client) đến node 1

Sau khi node 1 nhận xong tất cả các gói dữ liệu thì sẽ tiến hành truyền 3 gói tin đó qua các node 2,3,4 bằng mạng CSMA LAN:



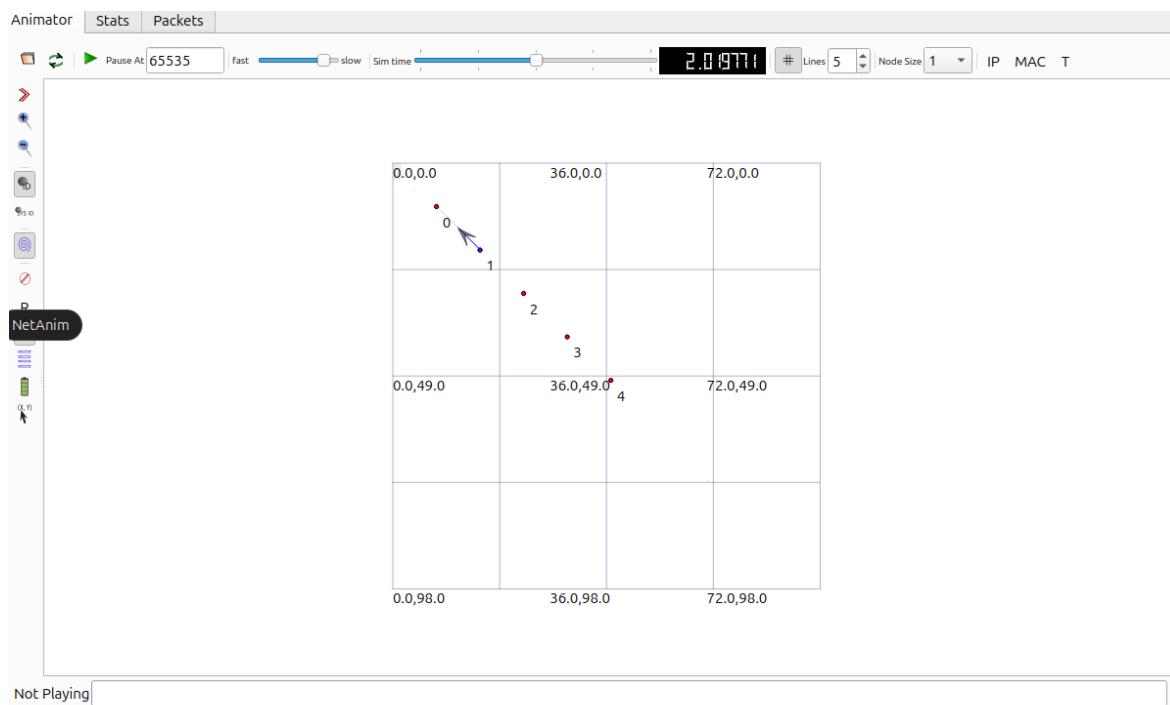
Hình 2.17 Sau khi node 1 nhận gói tin từ node 0 (Client), node 1 sẽ gửi các gói tin đến node 4 (Server)

Sau khi node cuối cùng của CSMA, node 4 (Server) thì node 4 sẽ gửi lại các gói dữ liệu qua lại các node 3,2,1 bằng mạng CSMA LAN:



Hình 2.18 Sau khi node 4 (Server) nhận được gói dữ liệu từ node 1, node 4 (Server) sẽ gửi lại các gói tin đến node 1

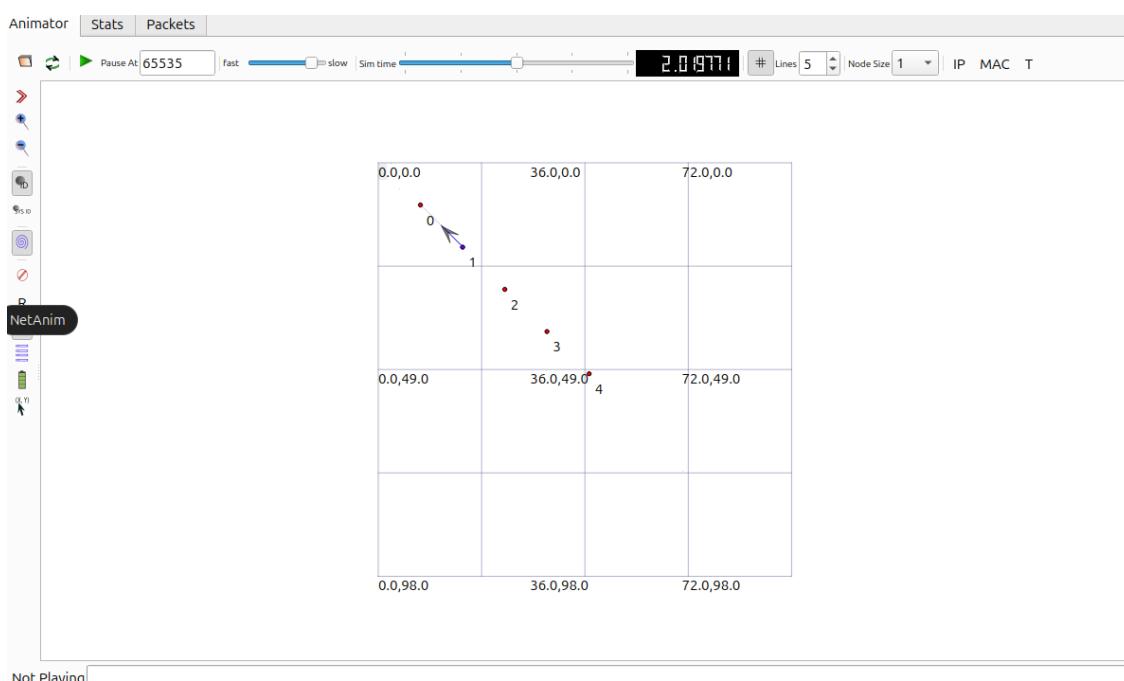
Sau khi node 1 nhận xong các packet thì sẽ truyền ngược lại cho node 0 (Client):



Hình 2.19 Sau khi node 1 nhận xong dữ liệu từ node 4 (Server), node 1 sẽ gửi gói tin đến node 0 (Client)

4.2. Nhận xét

4.2.1. Ở mục cửa sổ Animator



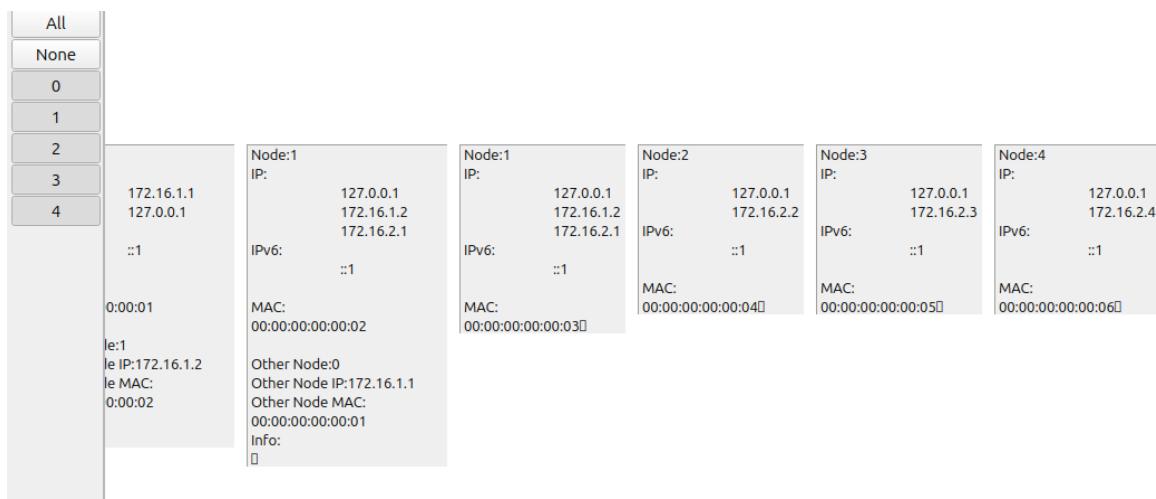
Hình 2.20 Cửa sổ Animator của NetAnim để xem chiêu gửi của các gói tin

Các thông số trong mô phỏng cần chú ý:

- Các nút:** Các điểm được đánh số (0, 1, 2, ...) đại diện cho các node trong mạng.
- Mũi tên:** Biểu thị đường truyền gói tin giữa các nút, với hướng và tốc độ truyền được thể hiện qua chiều và kích thước mũi tên.
- Tọa độ:** Tọa độ của các ô như "0.0, 0.0" xác định vị trí các nút trong không gian mô phỏng.
- Thời gian mô phỏng:** Đồng hồ và thanh điều chỉnh "Sim time" ở phía trên cho phép kiểm soát thời gian chạy của mô phỏng, cho phép người dùng theo dõi trạng thái mạng theo thời gian.

Ở cửa sổ Animator, chúng ta theo dõi được các gói tin truyền từ node này sang node khác. Ở bài mô phỏng này, ban đầu chúng ta thấy các gói tin sẽ truyền từ node 0 (Client) đến node 1 thông qua giao tiếp Point-to-Point, tiếp đó gói tin sẽ được truyền qua các node 2,3,4 từ node 1 thông qua mạng CSMA LAN.

4.2.2 Ở mục cửa sổ Stats:



Hình 2.21 Cửa sổ Stats để xem các thông tin của node mạng(IPv4, IPv6,...)

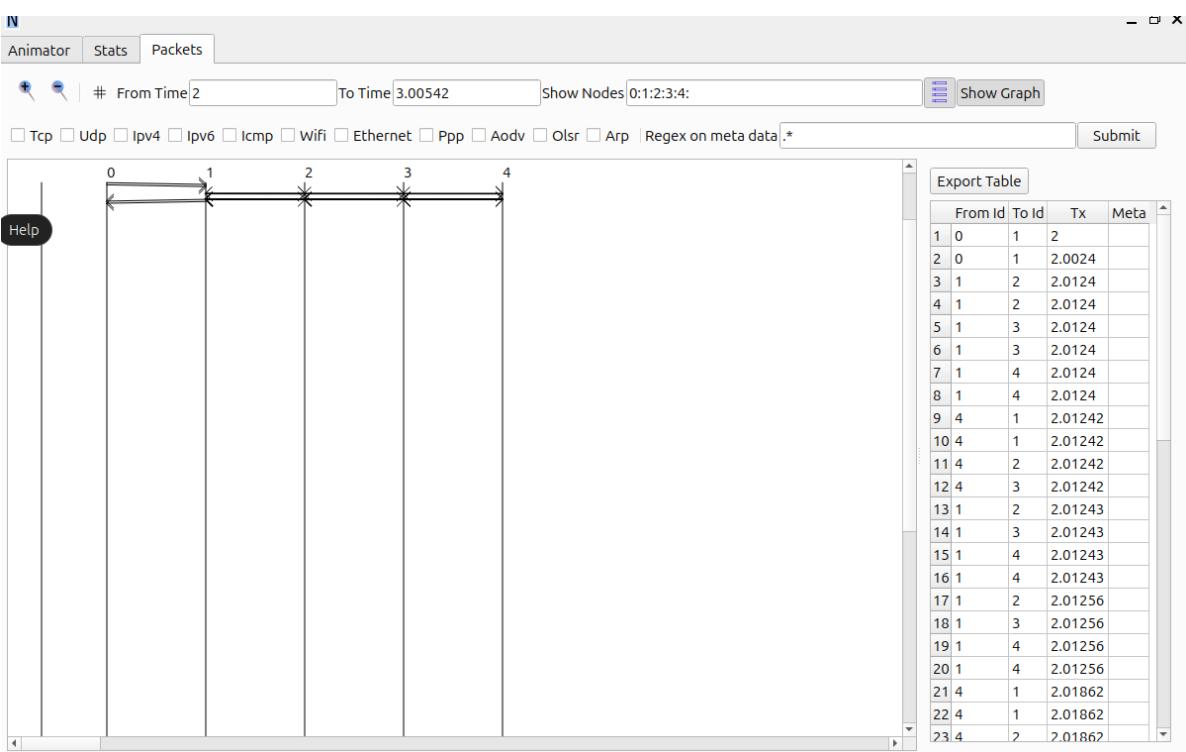
Các thông số này bao gồm:

- Node ID:** Mỗi ô đại diện cho một nút trong mạng, với nhãn "Node:0", "Node:1", v.v., để phân biệt các nút.
- IP:** Địa chỉ IP của mỗi nút, ví dụ:
- 127.0.0.1: Đây là địa chỉ loopback (localhost) mặc định, thường dùng để kiểm tra kết nối nội bộ trong mỗi nút.
- 172.16.x.x: Đây là địa chỉ IP trong mạng LAN của các nút, nơi x.x thay đổi để đại diện cho các địa chỉ IP duy nhất của từng nút. Các địa chỉ này giúp các nút xác định và truyền dữ liệu qua mạng (ví dụ: 172.16.1.2, 172.16.2.1,...).
- IPv6: ::1 là địa chỉ loopback IPv6, tương đương với 127.0.0.1 trong IPv4. Điều này thể hiện rằng mô phỏng hỗ trợ cả giao thức IPv4 và IPv6.

- **MAC:** Địa chỉ MAC của từng nút, như 00:00:00:00:00:01, 00:00:00:00:00:02, v.v. Địa chỉ MAC là một định danh duy nhất cho mỗi thiết bị trên mạng và giúp xác định nút trong tầng liên kết dữ liệu.
- **Other Node:** Phần này liệt kê thông tin về các nút khác mà nút hiện tại có kết nối trực tiếp hoặc tương tác, bao gồm địa chỉ IP và địa chỉ MAC của nút kia. Điều này hữu ích để hiểu rõ các kết nối giữa các nút trong mô phỏng.
- **Info:** Đây là phần trống, có thể dùng để hiển thị thêm thông tin hoặc trạng thái tùy chỉnh của từng nút (nếu có).

Ở cửa sổ Stats, ta có thể nắm thông tin về địa chỉ IP, địa chỉ MAC và thông tin trạng thái của các node trong kiến trúc mạng (Network Topology).

4.2.3. Ở mục cửa sổ Packets

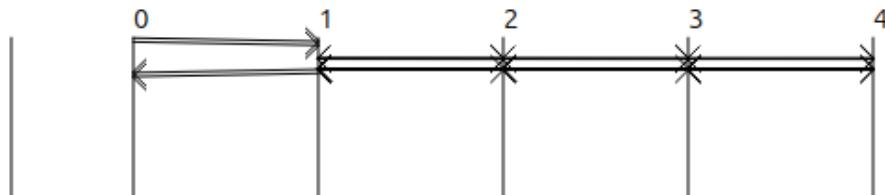


Hình 2.22 Cửa sổ Packets trong NetAnim cho ta biết thông tin truyền giữa các node và thời gian truyền giữa các node

Trong Hình 2.22, đây là giao diện của tab "Packets" trong NetAnim, nơi hiển thị lưu lượng gói tin giữa các nút trong mạng theo thời gian. Dưới đây là giải thích chi tiết:

- **Timeline Gói Tin:** Mỗi hàng ngang đại diện cho một nút (0, 1, 2, 3, 4) và các dòng mũi tên giữa các hàng là các gói tin được truyền từ nút này sang nút khác.
 - Các mũi tên chỉ ra hướng truyền của gói tin, cho thấy nút nào gửi gói tin và nút nào nhận.
 - Mỗi gói tin di chuyển theo chiều dọc, biểu diễn thời gian trong quá trình truyền giữa các nút.

- **Lọc Giao Thức:** Phần trên của màn hình có các hộp kiểm cho các giao thức như TCP, UDP, IPv4, IPv6, ICMP, Ethernet, v.v. Điều này cho phép bạn lọc các gói tin để xem các loại giao thức nhất định. Trong trường hợp này, bạn có thể thấy trạng thái của từng giao thức để xem chi tiết giao tiếp nào diễn ra trên mạng.
- **Thời gian:** Có các trường "From Time" và "To Time" để chọn khoảng thời gian của gói tin. Trong trường hợp này, thời gian bắt đầu từ 2 giây đến 3.00542 giây, giới hạn các gói tin hiển thị trong khoảng thời gian này.
- **Bảng Gói Tin:** Ở phía bên phải, bảng hiển thị chi tiết về các gói tin được truyền, bao gồm:
 - **From Id** và **To Id:** Các cột này chỉ ra ID của nút gửi và nhận gói tin.
 - **Tx:** Thời gian truyền của gói tin (thời gian khi gói tin bắt đầu được truyền).
 - **Meta:** Thông tin meta dữ liệu bổ sung (có thể bao gồm loại gói tin hoặc các thông tin khác liên quan đến giao thức).



Hình 2.23 Hướng truyền các gói tin giữa các node

Ở Hình 2.23 cho biết chiều truyền dữ liệu của các gói thông tin giữa các node 0,1,2,3,4.

Ở cửa sổ Packets của NetAnim, chúng ta có thể xác định được các gói tin được truyền từ node nào đến node nào tại thời gian nào. Ví dụ tại bài thí nghiệm này, ta có gói tin được truyền từ node 0 (Client) đến node 1 với 2 gói tin (mỗi gói tin đến node 1 với thời gian bắt được gói tin lần lượt là 2s và 2.0024s) sau đó node 1 sẽ truyền tiếp các gói tin đến các node 2,3,4 qua mạng CSMA LAN tại cùng thời gian là 2.0124s, đảm bảo tuân thủ tính chất truyền dữ liệu cho nhiều node của mạng CSMA LAN).

5. Kết luận

Việc sử dụng phần mềm ứng dụng NetAnim nhằm mô phỏng truyền dữ liệu mạng của các kiến trúc mạng (Network Topology) nhằm đánh giá hiệu quả, hiệu suất truyền của các kiến trúc mạng máy tính. Bảng gói tin trong NetAnim này cho phép bạn xem xét chi tiết từng gói tin, bao gồm thời gian truyền và nút nguồn - đích, giúp chúng ta phân tích sự hoạt động của các nút và xác định liệu mạng có xảy ra tắc nghẽn hoặc mất mát gói tin trong khoảng thời gian mô phỏng hay không. Ở bài mô phỏng này chúng ta đã mô phỏng được kiến trúc mạng Point-to-Point và mạng CSMA LAN và từ đó chúng

ta dựa vào các bảng gói tín cũng như đường truyền các gói tin mà có thể đánh giá được kiến trúc mạng (Network Topology) mà chúng ta muốn thử nghiệm.

III. Phát triển tính năng mới cho dự án: Mô phỏng Nhiễu Gaussian và Fading Rayleigh trong mô hình truyền sóng bị suy hao do khoảng cách

1. Giới thiệu

Trong lĩnh vực truyền thông không dây, việc mô phỏng tín hiệu là một bước quan trọng giúp các kỹ sư và nhà nghiên cứu đánh giá hiệu năng của hệ thống trong các điều kiện môi trường khác nhau. Tuy nhiên, tín hiệu vô tuyến khi truyền trong không gian thường không ổn định do chịu ảnh hưởng từ các yếu tố môi trường như nhiễu, phản xạ, khúc xạ, và hấp thụ. Những yếu tố này gây ra các hiện tượng như **nhiễu Gaussian** và **fading Rayleigh**, làm suy hao và biến động công suất nhận được của tín hiệu.

Để mô phỏng chính xác hơn, ngoài mô hình suy hao đã có trong *main-propagation-loss.cc (phần 1)*, nhóm đã phát triển thêm tính năng mô phỏng nhiễu Gaussian và fading Rayleigh. Những cải tiến này cho phép mô hình hóa tốt hơn các điều kiện truyền dẫn thực tế, tạo ra các dữ liệu có độ chính xác cao hơn về công suất tín hiệu nhận được. Cụ thể:

- **Nhiễu Gaussian:** Đại diện cho các loại nhiễu nền liên tục, chẳng hạn như nhiễu nhiệt từ môi trường hoặc từ các nguồn phát không mong muốn. Mô phỏng này được thực hiện qua biến ngẫu nhiên Gaussian (Normal Random Variable).
- **Fading Rayleigh:** Mô tả sự thay đổi cường độ tín hiệu do phản xạ đa đường, phổ biến trong các môi trường đô thị nơi tín hiệu phải vượt qua các vật cản như tòa nhà hoặc các cấu trúc lớn. Hiện tượng fading Rayleigh xảy ra khi không có đường truyền trực tiếp rõ ràng giữa bộ phát và bộ nhận, và chỉ có các thành phần đa đường.

Nhờ bổ sung hai yếu tố này, mô phỏng giờ đây không chỉ phản ánh suy hao cơ bản dựa trên khoảng cách mà còn phản ánh các yếu tố môi trường tác động lên tín hiệu, giúp nâng cao độ chính xác trong các nghiên cứu và thử nghiệm mạng không dây.

2. Mục tiêu tính năng mới

Mục tiêu của việc phát triển tính năng này bao gồm:

- **Tăng tính thực tế của mô phỏng truyền sóng:** Mô phỏng nhiễu Gaussian và fading Rayleigh giúp mô hình hóa các yếu tố ảnh hưởng phổ biến trong truyền thông không dây thực tế.
- **Tạo dữ liệu chi tiết để phân tích:** Lưu kết quả mô phỏng vào file CSV giúp dễ dàng so sánh công suất tín hiệu nhận trong các trường hợp khác nhau (không nhiễu, có nhiễu Gaussian, và có fading Rayleigh).
- **Hỗ trợ nghiên cứu các kịch bản khác nhau:** Cung cấp nền tảng cho các thử nghiệm sâu hơn, bao gồm việc đánh giá hiệu năng truyền dẫn trong các điều kiện môi trường và khoảng cách khác nhau.

3. Phương pháp triển khai

Việc triển khai tính năng mới dựa trên các bước sau:

Bước 1: Khởi tạo các mô hình vị trí và mô hình suy hao Friis:

Tạo hai thiết bị phát và nhận (a và b) sử dụng mô hình Constant Position Mobility Model để giữ chúng ở vị trí cố định. Thiết bị phát (a) được đặt tại tọa độ (0, 0, 0) và thiết bị nhận (b) được đặt tại nhiều khoảng cách khác nhau từ thiết bị phát.

CODE:

```
// Khởi tạo các mô hình vị trí cho hai thiết bị  
Ptr<ConstantPositionMobilityModel>  
    a = CreateObject<ConstantPositionMobilityModel>();  
Ptr<ConstantPositionMobilityModel>  
    b = CreateObject<ConstantPositionMobilityModel>();  
a->SetPosition(Vector(0.0, 0.0, 0.0));
```

Mô hình suy hao Friis (Friis Propagation Loss Model) được sử dụng để tính công suất tín hiệu nhận được dựa trên khoảng cách giữa hai thiết bị.

CODE:

```
// Khởi tạo mô hình suy hao Friis  
Ptr<FriisPropagationLossModel>  
    friis = CreateObject<FriisPropagationLossModel>();
```

Bước 2: Mô phỏng nhiễu Gaussian:

Để mô phỏng nhiễu Gaussian, nhóm tạo một biến ngẫu nhiên Gaussian (Normal Random Variable) với trung bình bằng 0 và phương sai bằng 1, để mô phỏng các nhiễu ngẫu nhiên xảy ra do nhiều nguồn khác nhau trong môi trường.

CODE:

```
// Khởi tạo Gaussian noise  
Ptr<NormalRandomVariable>  
    gaussianNoise = CreateObject<NormalRandomVariable>();  
    gaussianNoise->SetAttribute("Mean", DoubleValue(0.0));  
    gaussianNoise->SetAttribute("Variance", DoubleValue(1.0));
```

Giá trị công suất nhận bị nhiễu được tính bằng cách lấy công suất nhận không có nhiễu cộng thêm một giá trị ngẫu nhiên từ biến Gaussian. Điều này tạo ra dữ liệu công suất nhận với sự biến thiên do nhiễu nền.

CODE:

```
// Tính công suất nhận không có nhiễu  
double rxPowerDbm = friis->CalcRxPower(txPowerDbm, a, b);  
// Thêm Gaussian noise vào công suất nhận  
double gaussianRxPowerDbm = rxPowerDbm + gaussianNoise->GetValue();
```

Bước 3: Mô phỏng fading Rayleigh:

Để tạo fading Rayleigh, nhóm sử dụng hai biến ngẫu nhiên Gaussian độc lập, đại diện cho các thành phần phức của tín hiệu nhận do các sóng phản xạ đa đường.

CODE:

```
// Khởi tạo Rayleigh fading
Ptr<NormalRandomVariable>
rayleighFading1 = CreateObject<NormalRandomVariable>();
rayleighFading1->SetAttribute("Mean", DoubleValue(0.0));
rayleighFading1->SetAttribute("Variance", DoubleValue(1.0));
Ptr<NormalRandomVariable>
rayleighFading2 = CreateObject<NormalRandomVariable>();
rayleighFading2->SetAttribute("Mean", DoubleValue(0.0));
rayleighFading2->SetAttribute("Variance", DoubleValue(1.0));
```

Công suất nhận bị ảnh hưởng bởi fading Rayleigh được tính bằng cách lấy tổng bình phương của hai biến ngẫu nhiên Gaussian và chuyển đổi kết quả thành đơn vị dBm. Điều này giả lập sự thay đổi trong công suất nhận do phản xạ và nhiễu đa đường.

CODE:

```
// Tính Rayleigh fading bằng cách sử dụng hai biến Gaussian độc lập
double rayleighRxPowerDbm = rxPowerDbm
+ 10 * log10(sqrt(pow(rayleighFading1->GetValue(), 2)
+ pow(rayleighFading2->GetValue(), 2)));
```

Bước 4: Ghi dữ liệu vào file CSV:

Kết quả của mỗi lần mô phỏng được ghi vào file *interference-propagation-results.csv*, bao gồm các cột:

- **Distance (m):** Khoảng cách giữa hai thiết bị.
- **RxPower (dBm) without Noise:** Công suất nhận được khi không có nhiễu.
- **RxPower (dBm) with Gaussian Noise:** Công suất nhận với nhiễu Gaussian.
- **RxPower (dBm) with Rayleigh Fading:** Công suất nhận với fading Rayleigh.

Khoảng cách được tăng dần từ 10m đến 2500m với bước nhảy 100m, giúp tạo ra một loạt các dữ liệu công suất nhận tại các khoảng cách khác nhau để phân tích.

4. Kết quả

File *interference-propagation-results.csv* chứa kết quả mô phỏng với các giá trị công suất nhận tại từng khoảng cách trong ba trường hợp:

- **Không có nhiễu:** Chỉ có suy hao theo mô hình Friis.
- **Có nhiễu Gaussian:** Công suất nhận chịu ảnh hưởng của nhiễu Gaussian.

- **Có fading Rayleigh:** Công suất nhận bị ảnh hưởng bởi fading Rayleigh.

CODE:

```
// Ghi kết quả vào file
outFile << distance << ", " << rxPowerDbm << ", " << gaussianRxPowerDbm <<
", " << rayleighRxPowerDbm << "\n";
Simulator::Stop(Seconds(0.1));
Simulator::Run();
```

Các giá trị trong file CSV *Hình 3.1* cho phép đánh giá ảnh hưởng của khoảng cách và các yếu tố ngẫu nhiên lên tín hiệu nhận, từ đó hỗ trợ phân tích và tối ưu hóa các tham số truyền thông trong mạng không dây.

interference-propagation-results.csv /home/trunglogaric/ns-3-allinone/ns-3.29			
1	Distance (m), RxPower (dBm) without Noise, RxPower (dBm) with Gaussian Noise, RxPower (dBm) with Rayleigh Fading		
2	10, -46.6839, -45.4806, -45.5748		
3	110, -67.5118, -67.1056, -75.3112		
4	210, -73.1283, -73.2179, -74.4184		
5	310, -76.5112, -78.9093, -81.7182		
6	410, -78.9396, -79.4396, -79.551		
7	510, -80.8353, -79.6644, -76.9058		
8	610, -82.3905, -83.2726, -80.2443		
9	710, -83.7091, -83.5217, -82.2666		
10	810, -84.8536, -84.2757, -84.4834		
11	910, -85.8648, -85.6988, -86.3484		
12	1010, -86.7704, -85.4902, -85.5516		
13	1110, -87.5904, -87.3547, -90.855		
14	1210, -88.3396, -88.8364, -86.9222		
15	1310, -89.0294, -88.3718, -91.8908		
16	1410, -89.6683, -90.3922, -87.4876		
17	1510, -90.2635, -90.2396, -90.3788		
18	1610, -90.8204, -91.9776, -89.6387		
19	1710, -91.3439, -92.0049, -88.0791		
20	1810, -91.8375, -93.1909, -92.6265		
21	1910, -92.3046, -93.7832, -94.366		
22	2010, -92.7478, -92.1606, -93.5933		
23	2110, -93.1696, -93.185, -91.1843		
24	2210, -93.5718, -94.5829, -92.6258		
25	2310, -93.9562, -95.301, -98.9155		
26	2410, -94.3243, -93.8547, -96.2416		

Hình 3.1 Dữ liệu kết quả mô phỏng suy hao tín hiệu với nhiễu Gaussian, fading Rayleigh và không nhiễu

Cột 1: Distance (m)

- Đây là khoảng cách giữa hai thiết bị truyền thông không dây, được tính bằng mét (m).
- Giá trị của khoảng cách tăng dần từ 10m đến 2410m với bước nhảy là 100m. Điều này cho phép đánh giá mức độ suy hao tín hiệu theo khoảng cách.

Cột 2: RxPower (dBm) without Noise

- Đây là công suất tín hiệu nhận được (Received Power) tại thiết bị thu khi không có nhiễu.
- Công suất này được tính bằng đơn vị dBm (decibel-milliwatts) và được xác định dựa trên mô hình suy hao Friis. Mô hình này chỉ tính đến suy hao do khoảng cách mà không xem xét các yếu tố nhiễu hoặc fading khác.

Cột 3: RxPower (dBm) with Gaussian Noise

- Đây là công suất tín hiệu nhận được khi có thêm nhiễu Gaussian.
- Nhiễu Gaussian (Gaussian Noise) đại diện cho nhiễu nền trong kênh truyền thông. Nhieu này được mô phỏng bằng cách cộng một giá trị ngẫu nhiên từ phân phối Gaussian vào công suất nhận gốc.
- Mức công suất nhận được trong cột này sẽ dao động nhẹ quanh giá trị công suất không có nhiễu do sự thay đổi nhẹ của nhiễu Gaussian.

Cột 4: RxPower (dBm) with Rayleigh Fading

- Đây là công suất tín hiệu nhận được khi có ảnh hưởng của fading Rayleigh.
- Fading Rayleigh mô phỏng các tình huống tín hiệu bị phản xạ và truyền qua nhiều đường khác nhau trong môi trường (như trong thành phố với nhiều vật cản). Điều này gây ra sự dao động không đồng đều và lớn hơn so với nhiễu Gaussian.
- Công suất nhận được trong cột này dao động mạnh hơn so với cột có nhiễu Gaussian, vì fading Rayleigh có thể gây ra sự suy hao hoặc tăng đột biến ở công suất nhận dựa trên đường đi của tín hiệu qua các vật cản.

Để dễ dàng phân tích và so sánh công suất tín hiệu nhận được trong các trường hợp khác nhau, nhóm đã tạo thêm biểu đồ trực quan hóa dữ liệu từ file interference-propagation-results.csv. Biểu đồ này giúp chúng ta hiểu rõ hơn về suy hao công suất tín hiệu theo khoảng cách và tác động của các yếu tố nhiễu và fading đối với tín hiệu.

PYTHON CODE:

```

import pandas as pd
import matplotlib.pyplot as plt

# Đọc dữ liệu từ file CSV
data = pd.read_csv("interference-propagation-results.csv")

# Vẽ đồ thị cho từng cột
plt.figure(figsize=(10, 6))
plt.plot(data['Distance (m)'], data['RxPower (dBm) without Noise'], label='Without Noise')

```

```

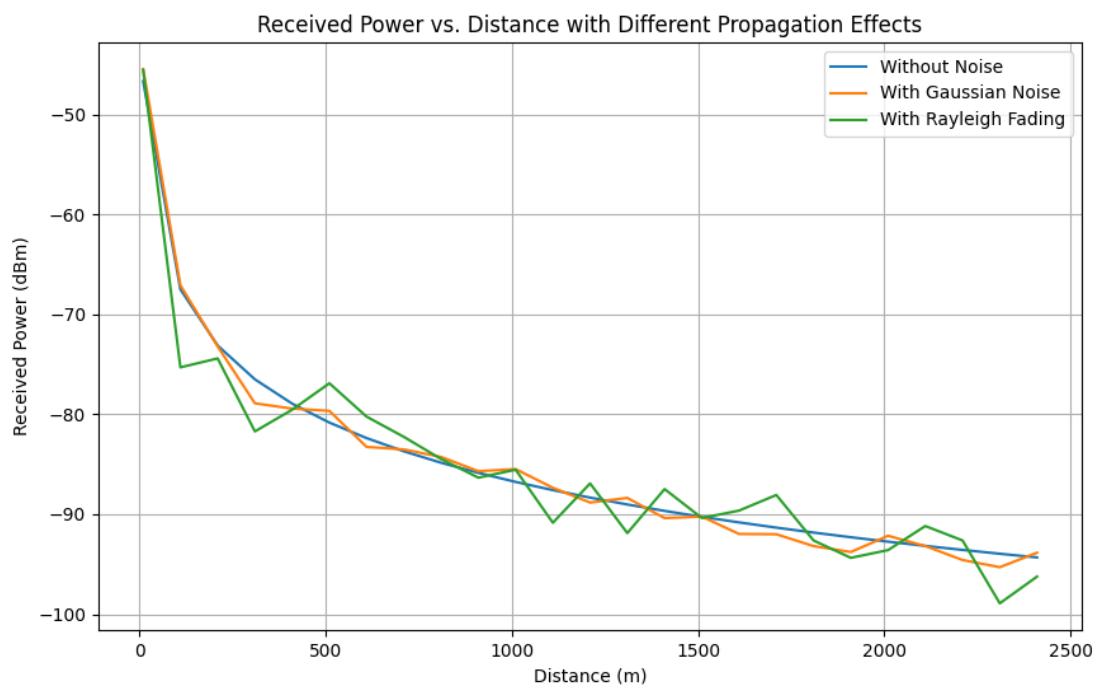
plt.plot(data['Distance (m)'], data['RxPower (dBm) with Gaussian Noise'],
label='With Gaussian Noise')
plt.plot(data['Distance (m)'], data['RxPower (dBm) with Rayleigh Fading'],
label='With Rayleigh Fading')

# Thêm nhãn và tiêu đề
plt.xlabel('Distance (m)')
plt.ylabel('Received Power (dBm)')
plt.title('Received Power vs. Distance with Different Propagation Effects')
plt.legend()
plt.grid(True)

# Lưu đồ thị thành file ảnh
plt.savefig("output.png")

```

KẾT QUẢ:



Hình 3.2 Đồ thị công suất tín hiệu theo khoảng cách với các điều kiện nhiễu khác nhau

Ý nghĩa của biểu đồ **Hình 3.2:**

- **Trục x (Distance):** Khoảng cách giữa hai thiết bị truyền thông, từ 10m đến 2410m.
- **Trục y (Received Power):** Công suất tín hiệu nhận được tại thiết bị thu, được đo bằng dBm.

Biểu đồ Hình x biểu diễn mối quan hệ giữa công suất tín hiệu nhận (Received Power) theo khoảng cách giữa hai thiết bị truyền thông không dây, với ba trường hợp:

1. **Without Noise (Không có nhiễu)** - Đường màu xanh nước.
2. **With Gaussian Noise (Có nhiễu Gaussian)** - Đường màu cam.
3. **With Rayleigh Fading (Có fading Rayleigh)** - Đường màu xanh lá cây.

Phân tích từng đường biểu diễn:

1. **Đường màu xanh nước (Without Noise)**
 - Đường này đại diện cho công suất tín hiệu nhận được khi không có nhiễu, chỉ phụ thuộc vào suy hao do khoảng cách theo mô hình Friis.
 - Ta thấy rằng công suất nhận giảm dần một cách đều đặn khi khoảng cách tăng lên. Đây là đặc điểm điển hình của mô hình suy hao theo khoảng cách, trong đó tín hiệu bị suy yếu khi khoảng cách truyền tăng.
2. **Đường màu cam (With Gaussian Noise)**
 - Đường này mô phỏng công suất nhận khi có thêm nhiễu Gaussian, mô phỏng nhiễu nền trong môi trường truyền thông.
 - Đường này vẫn giữ xu hướng tương tự như đường không có nhiễu, nhưng có sự dao động nhẹ quanh giá trị công suất không có nhiễu. Sự dao động này là do nhiễu Gaussian, tuy nhiên nhiễu Gaussian chỉ tác động nhỏ và không làm thay đổi đáng kể đường suy hao do khoảng cách.
3. **Đường màu xanh lá cây (With Rayleigh Fading)**
 - Đường này đại diện cho công suất nhận khi có fading Rayleigh, mô phỏng các môi trường có nhiều vật cản dẫn đến hiện tượng đa đường (multi-path).
 - Khác với nhiễu Gaussian, fading Rayleigh gây ra sự dao động mạnh hơn so với đường không có nhiễu. Đường này có những đỉnh và đáy đột ngột, thể hiện sự suy hao hoặc tăng đột biến trong công suất nhận do tín hiệu bị phản xạ và truyền qua nhiều đường khác nhau.
 - Fading Rayleigh phản ánh rõ ảnh hưởng của môi trường đô thị phức tạp hoặc các môi trường có nhiều vật cản, nơi tín hiệu có thể gấp phải hiện tượng tán xạ, khúc xạ, và phản xạ.

5. Đánh giá và nhận xét

5.1 Ưu điểm của tính năng mới

Độ chính xác và hiện thực hóa cao:

- Việc bổ sung nhiễu Gaussian và fading Rayleigh giúp mô phỏng không chỉ phản ánh mức suy hao tín hiệu đơn thuần, mà còn mô phỏng được những biến động do nhiễu gây ra trong công suất nhận. Điều này mang lại một bức tranh chân thực hơn về các điều kiện truyền thông không dây mà hệ thống phải đối mặt trong thực tế.

- Với việc có thể dự đoán các tình huống tín hiệu yếu hoặc bị ảnh hưởng bởi nhiễu nền và môi trường phản xạ phức tạp, mô phỏng này giúp nhà nghiên cứu và kỹ sư thiết kế mạng có cơ sở dữ liệu phong phú và chuẩn xác để đưa ra những quyết định quan trọng nhằm cải thiện độ tin cậy và hiệu năng của hệ thống.

Phù hợp với các kịch bản nghiên cứu đa dạng và phức tạp:

- Nhờ mô phỏng được các yếu tố phức tạp như nhiễu Gaussian và fading Rayleigh, tính năng này mở rộng phạm vi ứng dụng của mô hình trong các kịch bản thực tế, đặc biệt là trong các môi trường như đô thị với mật độ vật cản cao, hay các khu vực công nghiệp với nhiều nguồn nhiễu nền.
- Với khả năng mô phỏng sát thực tế hơn, công cụ này là lựa chọn hữu ích cho các nghiên cứu về tối ưu hóa mạng không dây, đặc biệt là trong các điều kiện địa hình hoặc môi trường khắc nghiệt mà mô hình lý thuyết đơn giản khó đáp ứng.

Khả năng lưu trữ và phân tích dữ liệu dễ dàng:

- Việc lưu trữ dữ liệu đầu ra vào file CSV giúp người dùng dễ dàng tiếp cận và xử lý dữ liệu bằng các công cụ phân tích phổ biến. Người dùng có thể so sánh chi tiết công suất tín hiệu theo khoảng cách, hoặc phân tích sâu hơn về tác động của từng loại nhiễu và fading lên hiệu năng truyền thông.
- Định dạng dữ liệu dễ xử lý này cũng cho phép thực hiện các phân tích nâng cao, từ đó cung cấp cơ sở để tối ưu hóa các thông số truyền thông và xây dựng chiến lược vận hành hiệu quả cho các mạng không dây phức tạp.

5.2 Hạn chế

Hạn chế trong phạm vi các loại nhiễu và fading:

- Hiện tại, mô phỏng chỉ bao gồm nhiễu Gaussian và fading Rayleigh, chưa bao quát được các hiện tượng fading khác như fading Rician - một loại fading thường thấy trong môi trường mà cả đường truyền trực tiếp và phản xạ cùng tồn tại.
- Ngoài ra, mô hình hiện tại chưa xét đến suy hao do các vật cản cụ thể như tòa nhà, cây cối hoặc các đặc điểm vật lý của môi trường truyền dẫn. Việc bổ sung thêm các yếu tố vật lý này có thể làm tăng độ chính xác, đặc biệt khi nghiên cứu các kịch bản môi trường cụ thể.

Phạm vi mô phỏng tuyển tính theo khoảng cách:

- Mô phỏng hiện tập trung vào suy hao theo khoảng cách tuyển tính mà chưa tính đến các yếu tố phi tuyển phức tạp hơn, chẳng hạn như sự phản xạ nhiều lần từ các bề mặt khác nhau, tán xạ từ vật liệu, hoặc ảnh hưởng của các điều kiện thời tiết.
- Trong bối cảnh nghiên cứu mạng không dây hiện đại, việc mở rộng mô hình để bao gồm các yếu tố phi tuyển sẽ giúp mô phỏng chính xác hơn mà còn

linh hoạt hơn, từ đó đáp ứng được nhu cầu nghiên cứu các môi trường truyền thông thực tế phức tạp hơn.

6. Kết luận

Việc phát triển tính năng mới này đã mang lại những bước tiến quan trọng trong việc nâng cao khả năng mô phỏng điều kiện truyền thông không dây thực tế. Bằng cách bổ sung các yếu tố nhiễu Gaussian và fading Rayleigh vào mô hình truyền sóng Friis, mô phỏng này không chỉ mô phỏng được mức suy hao tín hiệu đơn thuần mà còn phản ánh được những biến động ngẫu nhiên do môi trường tác động, giúp mô phỏng tiến gần hơn tới thực tế truyền thông không dây mà chúng ta gặp phải.

Với các kết quả thu được, tính năng này mở ra nhiều hướng nghiên cứu và ứng dụng:

- **Nghiên cứu nâng cao về tối ưu hóa mạng không dây:** Kết quả mô phỏng có thể hỗ trợ các nhà nghiên cứu đánh giá hiệu năng mạng không dây trong các kịch bản phức tạp, từ đó tối ưu hóa các tham số truyền thông và phát triển các giải pháp để đảm bảo độ tin cậy và hiệu năng cho hệ thống.
- **Tiềm năng mở rộng trong tương lai:** Mô hình hiện tại có thể được mở rộng thêm để bao gồm các dạng fading khác như fading Rician, hoặc các yếu tố vật lý đặc trưng của môi trường như ảnh hưởng của tòa nhà, cây cối, và các điều kiện thời tiết. Điều này sẽ giúp mô phỏng có khả năng ứng dụng rộng rãi hơn, đặc biệt trong việc đánh giá và triển khai các mạng không dây tại những địa điểm cụ thể và trong điều kiện khắc nghiệt.

Nhìn chung, tính năng mới này không chỉ mở rộng thêm quá trình truyền khi có nhiễu mà còn nâng cao tính linh hoạt của mô hình mô phỏng, tạo nền tảng vững chắc cho các nghiên cứu về truyền thông không dây trong tương lai, đồng thời hỗ trợ việc triển khai các mạng không dây chất lượng cao trong các môi trường phức tạp.