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

BÁO CÁO NHÓM LẬP TRÌNH MẠNG

Tên đề tài

PHÂN TÍCH HIỆU NĂNG MẠNG CSMA SỬ DỤNG NS-3

Giảng viên hướng dẫn:	TS. Nguyễn Văn Hiếu
Lớp học phần:	22.44
Lớp sinh hoạt:	22KTMT1,2
Sinh viên thực hiện:	1. Phan Lê Minh (106220240) 2. Hồ Anh Nguyên (106220228) 3. Trần Kiêm Quang Minh (106220226)

Đà Nẵng, tháng 11 năm 2025

MỤC LỤC

MỨC ĐỘ ĐÓNG GÓP CỦA CÁC THÀNH VIÊN

Thành viên	Công việc thực hiện	Tỉ lệ đóng góp
Phan Lê Minh	- Nghiên cứu lý thuyết CSMA/CA và các chuẩn Wi-Fi - Phân tích các metrics (Throughput, Delay, PDR, Packet Loss) - Phân tích cơ chế CSMA/CA (backoff, RTS/CTS, DIFS/SIFS)	35%
Hồ Anh Nguyên	- Nghiên cứu mô hình Wi-Fi trong NS-3 (WifiMac, WifiPhy, Channel) - Thực hiện mô phỏng và đối chiếu kết quả với lý thuyết - Xây dựng mục tiêu, phạm vi và các kịch bản mô phỏng	35%
Trần Kiêm Quang Minh	- Hỗ trợ chạy mô phỏng và thu thập dữ liệu - Kiểm tra file output, tổng hợp kết quả - Chỉnh sửa hình thức, mục lục, format báo cáo	30%

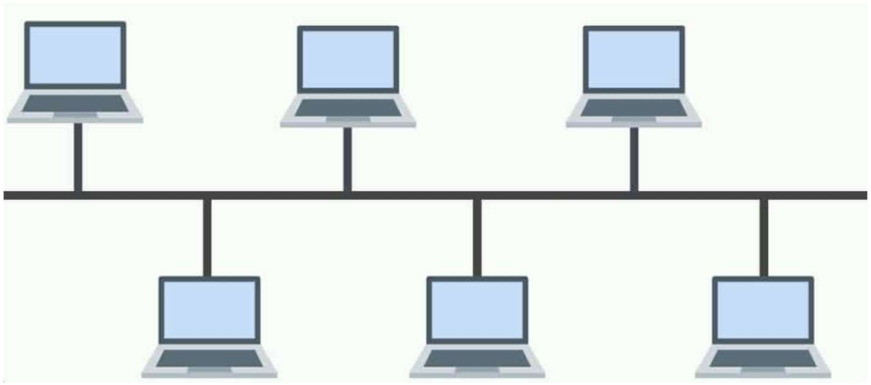
PHẦN 1. GIỚI THIỆU DỰ ÁN

1.1. Lí do nghiên cứu

- Trong các hệ thống mạng hiện đại, nhiều công nghệ truy cập đường truyền được sử dụng như Ethernet (CSMA/CD), Wi-Fi (CSMA/CA) và liên kết điểm-điểm (Point-to-Point). Mỗi công nghệ có đặc điểm, ứng dụng và hiệu năng khác nhau tùy vào môi trường truyền và mô hình sử dụng.
- NS-3 là công cụ mô phỏng mạng được sử dụng rộng rãi giúp sinh viên và nhà nghiên cứu mô phỏng hành vi mạng thực tế mà không cần triển khai phần cứng. Việc đánh giá hiệu năng các giao thức truyền thông trong điều kiện khác nhau là cần thiết để:
    - Hiểu nguyên lý hoạt động của mỗi cơ chế truy cập kênh.
    - Đánh giá giới hạn về throughput, độ trễ, độ tin cậy và khả năng mở rộng.
    - So sánh hiệu năng khi thay đổi số lượng node, lưu lượng (traffic), hoặc cấu hình mạng.
- Trong đề tài này, trọng tâm là **CSMA/CA**, cơ chế truy cập kênh được sử dụng trong mạng Wi-Fi. CSMA/CA phù hợp để nghiên cứu vì:
    - Đây là cơ chế cốt lõi của các mạng không dây (IEEE 802.11), nơi không thể sử dụng cơ chế phát hiện xung đột như CSMA/CD.
    - CSMA/CA thể hiện rõ đặc trưng của mạng không dây: tránh xung đột bằng backoff, phụ thuộc môi trường truyền, dễ bị ảnh hưởng bởi nhiễu và số lượng node.
    - NS-3 cung cấp mô hình Wi-Fi đầy đủ (YansWifiChannel, WifiHelper), cho phép đánh giá chính xác hiệu năng khi thay đổi số node hoặc traffic.

1.2. Tổng quan về CSMA

- CSMA là cơ chế truy cập đường truyền trong mạng máy tính, cho phép nhiều thiết bị cùng sử dụng chung một kênh truyền nhưng vẫn hạn chế tối đa nguy cơ xảy ra xung đột giữa các thiết bị khi gửi dữ liệu.



Hình 1: Các máy tính kết nối và truyền thông với nhau qua Ethernet sử dụng cơ chế CSMA

- Ý nghĩa :
    - CSMA giúp sử dụng kênh truyền hiệu quả hơn bằng cách ngăn thiết bị truyền khi kênh đang bận, từ đó giảm khả năng truyền đồng thời gây xung đột.

- Cơ chế này có cấu trúc đơn giản, chi phí triển khai thấp và hoạt động tốt trong nhiều loại mạng, bao gồm cả mạng có dây lẫn không dây.
- Tuy vậy, CSMA cũng tồn tại hạn chế: khi số lượng thiết bị dùng chung kênh tăng cao hoặc lưu lượng lớn, thời gian chờ kênh rảnh và nguy cơ xung đột vẫn có thể xảy ra, dẫn đến độ trễ tăng và hiệu suất tổng thể giảm.

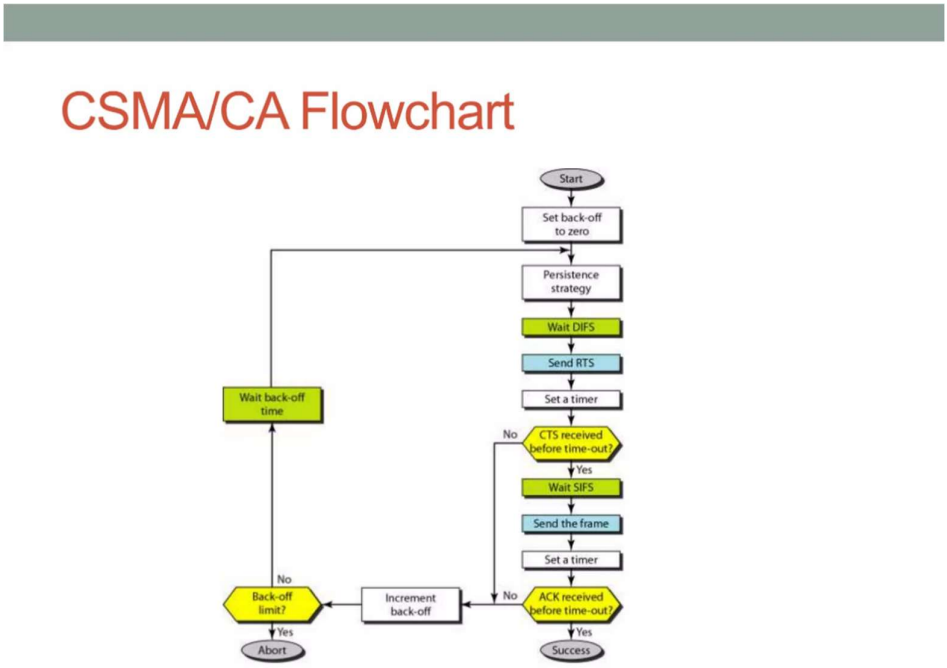
1.3. Cơ chế hoạt động của CSMA/CA

- CSMA (Carrier Sense Multiple Access) có thể được chia thành hai biến thể chính dựa trên cách xử lý xung đột trong quá trình truyền dữ liệu:

- CSMA/CA – Carrier Sense Multiple Access with Collision Avoidance: Dùng trong Wi-Fi (IEEE 802.11)

- CSMA/CD – Carrier Sense Multiple Access with Collision Detection: Dùng trong mạng Ethernet có dây.

- Trong NS-3, module CSMA trong mô phỏng này, đối tượng nghiên cứu là **CSMA/CA**, áp dụng cho mạng Wi-Fi Ad-hoc. Cơ chế hoạt động được mô tả theo các bước sau:



Hình 2: Lưu đồ thuật toán mô tả nguyên lý hoạt động của CSMA-CA

Căn cứ vào Hình 2, Cơ chế CSMA-CA (Carrier Sense Multiple Access with Collision Avoidance) hoạt động theo các bước sau:

- Bước 1: Node bắt đầu quá trình truyền dữ liệu.
- Bước 2 : Đặt bộ đếm backoff = 0.
- Bước 3 : Áp dụng chiến lược lắng nghe kênh, node kiểm tra trạng thái kênh truyền:
    - Nếu kênh đang **bận**, tiếp tục chờ.

    - Nếu kênh **rảnh**, chuyển sang bước tiếp theo.
- Bước 4 : Node phải chờ Distributed InterFrame Space (DIFS) để đảm bảo kênh thật sự rảnh.
- Bước 5 : Node gửi gói RTS để xin quyền truyền dữ liệu.
- Bước 6 : Timer dùng để theo dõi thời gian nhận CTS.
- Bước 7 : Kiểm tra CTS có đến trước timeout không?
    - Nếu nhận được CTS → đi tiếp đến Bước 8.
    - Nếu KHÔNG nhận **được CTS** → chuyển sang Bước 12 (backoff).
- Bước 8 : Node chờ SIFS (Short InterFrame Space) – khoảng thời gian ngắn hơn DIFS → ưu tiên truyền.
- Bước 9 : Node truyền dữ liệu đến thiết bị đích.
- Bước 10 : Node chờ phản hồi ACK xác nhận đã nhận gói.
- Bước 11 : Kiểm tra có nhận ACK trước timeout không?
    - Nếu nhận được ACK → Truyền thành công (Success) → kết thúc.
    - Nếu KHÔNG nhận được ACK → chuyển sang Bước 12 (backoff).
- Bước 12: Node tạo ra thời gian backoff ngẫu nhiên theo công thức:

Backoff = random(0, CW)

(CW = cửa sổ tránh xung đột).

- Bước 13: Kiểm tra giới hạn backoff:
    - Nếu chưa vượt quá giới hạn → quay về Bước 3 để thử lại.
    - Nếu vượt quá giới hạn retry → Hủy bỏ truyền (Abort).

1.4. Mục tiêu nghiên cứu

- Thu thập dữ liệu và các chỉ số hiệu năng của mạng sử dụng cơ chế truy cập CSMA/CA, thông qua FlowMonitor và các file log trong NS-3, làm cơ sở cho phân tích định lượng hiệu năng mạng.

- Thay đổi các tham số ảnh hưởng trực tiếp đến CSMA/CA như: mật độ nút, kích thước cửa sổ cạnh tranh (Contention Window), kích thước gói tin, băng thông và độ trễ liên kết… nhằm đánh giá đặc tính hoạt động của giao thức trong nhiều điều kiện khác nhau.

- Phân tích chi tiết các thông số mạng: Vẽ đồ thị minh họa và phân tích kỹ lưỡng các thông số như:

- **Throughput** – băng thông thực tế đạt được
- **Delay** – độ trễ truyền tin
- **Packet Loss** – tỷ lệ mất gói
- **PDR (Packet Delivery Ratio)** – tỷ lệ gói tin được gửi thành công
- **Cumulative Packets / Data** – tổng số gói tin và dữ liệu truyền đi

- Thực hiện mô phỏng các mô hình CSMA, Wi-Fi và P2P trên môi trường NS-3 để quan sát và phân tích hành vi của từng công nghệ trong các điều kiện mạng khác nhau. Rút ra nhận xét và đánh giá tổng quan về khả năng hoạt động, ưu nhược điểm của từng công nghệ.

PHẦN 2. PHƯƠNG PHÁP

2.1. Mô hình mạng Ad-hoc trong NS-3

2.1.1. Tổng quan về Ad-hoc trong NS-3

NS-3 hỗ trợ mô phỏng mạng Ad-hoc thông qua lớp **AdhocWifiMac**, một phần của module WiFi. Khác với Infrastructure mode (có Access Point), Ad-hoc mode cho phép các node giao tiếp trực tiếp với nhau mà không cần điểm trung gian.

2.1.2. Các thành phần chính

YansWifiChannelHelper và YansWifiPhyHelper:

- Quản lý lớp vật lý (Physical Layer)
- Mô phỏng môi trường truyền sóng không dây
- Tất cả các node chia sẻ cùng một kênh truyền (shared medium)

Đặc điểm kênh Ad-hoc:

- Kênh broadcast: tín hiệu từ một node có thể được tất cả các node khác trong phạm vi nghe thấy
- Half-duplex: không thể truyền và nhận đồng thời
- Shared medium: chỉ một node có thể truyền tại một thời điểm

WifiMacHelper:

- Thiết lập lớp MAC (Medium Access Control)
- Cấu hình chế độ Ad-hoc thông qua "ns3::AdhocWifiMac"
- Quản lý các cơ chế CSMA/CA, backoff, retransmission

WifiHelper:

- Thiết lập chuẩn WiFi (802.11ax)
- Cấu hình rate control algorithm (IdealWifiManager)
- Kết nối PHY, MAC, và Node

2.1.3. Vô hiệu hóa RTS/CTS

Trong NS-3, RTS/CTS được kiểm soát thông qua **RtsCtsThreshold**:

```
UintegerValue threshold = 1000;  // bytes
Config::SetDefault("ns3::WifiRemoteStationManager::RtsCtsThreshold",
threshold);
```

Cách hoạt động:

- Nếu kích thước packet < threshold → không sử dụng RTS/CTS
- Nếu kích thước packet ≥ threshold → sử dụng RTS/CTS
- Trong mô phỏng này: threshold = 1000 bytes, packet size = 512 bytes
- **Kết quả:** RTS/CTS bị vô hiệu hóa cho tất cả các packet

2.1.4. Mobility Model

ConstantPositionMobilityModel:

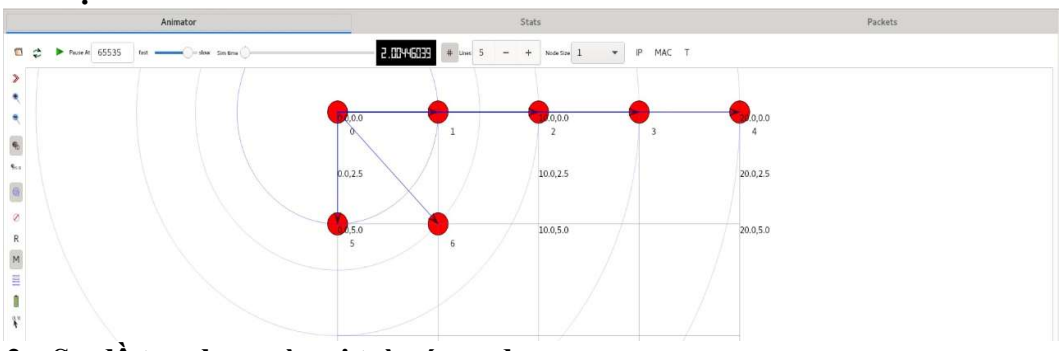
- Các node được đặt tại vị trí cố định (không di chuyển)
- Đơn giản hóa phân tích, loại bỏ ảnh hưởng của di động

GridPositionAllocator:

- Bố trí các node theo dạng lưới (grid)

- Tham số:
  - MinX, MinY: Tọa độ góc bắt đầu (0, 0)
  - DeltaX, DeltaY: Khoảng cách giữa các node (5m x 5m)
  - GridWidth: Số node trên mỗi hàng (5 nodes)
  - LayoutType: "RowFirst" (xếp theo hàng trước)

Ví dụ với 7 nodes:



2.2. Sơ đồ topology và vai trò các node

2.2.1. Kiến trúc Client-Server

Mô phỏng sử dụng mô hình **nhiều Client - một Server**:

Server (Node 0 - mặc định):

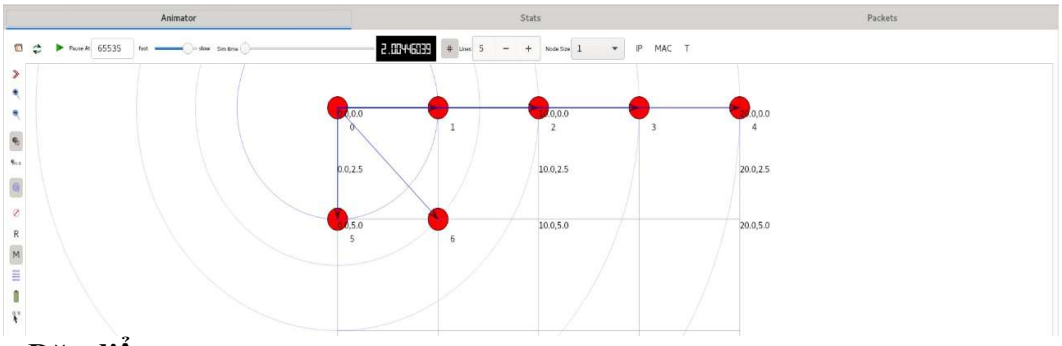
- Là điểm đích nhận tất cả các packet
- Chạy ứng dụng UdpEchoServer trên port 9
- Vai trò: Đo lường số packet nhận được từ các client

Clients (Node 1, 2, 3, ..., N-1):

- Mỗi client chạy ứng dụng UdpEchoClient
- Tất cả client đồng thời gửi packet đến server
- Bắt đầu gửi tại thời điểm 2.0 giây
- Gửi với interval và packet size cố định

2.2.2. Sơ đồ topology

Ví dụ với 5 nodes:



Đặc điểm:

- Server ở vị trí (0, 0)
- Các client phân bố theo grid
- Khoảng cách đều: 5m x 5m
- Tất cả node trong phạm vi của nhau (single-hop Ad-hoc)

2.2.3. Lý do thiết kế

Tại sao tất cả client gửi đồng thời?

- Tạo áp lực cao nhất lên môi trường truyền
- Mô phỏng worst-case scenario
- Dễ quan sát hiện tượng xung đột
- Phản ánh các tình huống thực tế: sensor networks, emergency networks

Tại sao dùng UDP Echo?

- Đơn giản, dễ phân tích
- Không có cơ chế phức tạp như TCP (congestion control, retransmission)
- Packet loss phản ánh trực tiếp xung đột ở lớp MAC
- Phù hợp với mục tiêu nghiên cứu RTS/CTS

2.3. Kịch bản mô phỏng

2.3.1. Mục tiêu mô phỏng

- Đánh giá tác động của số lượng node đến hiệu suất mạng Ad-hoc khi vô hiệu hóa RTS/CTS
- Thu thập dữ liệu về packet loss, lost clients ratio, throughput
- Phân tích xu hướng thay đổi khi số node tăng từ 2 đến 30

2.3.2. Cấu hình mô phỏng

Thông số cơ bản:

Tham số	Giá trị	Mô tả
Số lượng node	2 - 30	Thay đổi để quan sát xu hướng
WiFi Standard	IEEE 802.11ax	Chuẩn WiFi mới nhất
RTS/CTS Threshold	1000 bytes	Vô hiệu hóa cho packet 512B
Packet Size	512 bytes	Kích thước gói tin UDP
Interval	1.0 second	Khoảng cách giữa các lần gửi
Max Packets	10 packets	Số packet mỗi client gửi
Thời gian mô phỏng	[Tùy chỉnh]s	Tổng thời gian chạy
Địa chỉ mạng	10.1.1.0/24	Dải IP cho các node
Port	9	Cổng UDP Echo Server

Mobility:

- Model: ConstantPositionMobilityModel (static)
- Position Allocator: GridPositionAllocator
- Grid spacing: 5m x 5m
- Grid width: 5 nodes per row

2.3.3. Các bước triển khai

Bước 1: Khởi tạo môi trường

```
// Vô hiệu hóa RTS/CTS
UintegerValue threshold = 1000;
Config::SetDefault("ns3::WifiRemoteStationManager::RtsCtsThreshold",
threshold);
```

```
// Tạo nodes
NodeContainer nodes;
nodes.Create(nNodes);
```
Bước 2: Cấu hình WiFi Ad-hoc
```
// Thiết lập kênh và PHY
YansWifiChannelHelper channel = YansWifiChannelHelper::Default();
YansWifiPhyHelper phy;
phy.SetChannel(channel.Create());
// Cấu hình MAC cho Ad-hoc
WifiMacHelper mac;
mac.SetType("ns3::AdhocWifiMac");
// Cài đặt WiFi devices
WifiHelper wifi;
NetDeviceContainer devices = wifi.Install(phy, mac, nodes);
```
Bước 3: Thiết lập vị trí
```
MobilityHelper mobility;
mobility.SetPositionAllocator(
   "ns3::GridPositionAllocator",
   "MinX", DoubleValue(0.0),
   "MinY", DoubleValue(0.0),
   "DeltaX", DoubleValue(5.0),
   "DeltaY", DoubleValue(5.0),
   "GridWidth", UintegerValue(5),
   "LayoutType", StringValue("RowFirst"));
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(nodes);
```
Bước 4: Cài đặt Internet Stack
```
InternetStackHelper stack;
stack.Install(nodes);
Ipv4AddressHelper address;
address.SetBase("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign(devices);
```
Bước 5: Thiết lập ứng dụng
```
// Server
UdpEchoServerHelper echoServer(9);
ApplicationContainer serverApps = echoServer.Install(nodes.Get(0));
serverApps.Start(Seconds(2.0));
serverApps.Stop(Seconds(simTime));
// Clients
UdpEchoClientHelper echoClient(interfaces.GetAddress(0), 9);
```

```
echoClient.SetAttribute("MaxPackets", UintegerValue(10));
echoClient.SetAttribute("Interval", TimeValue(Seconds(1.0)));
echoClient.SetAttribute("PacketSize", UintegerValue(512));
for (uint32_t i = 1; i < nNodes; i++) {
   ApplicationContainer clientApp = echoClient.Install(nodes.Get(i));
   clientApp.Start(Seconds(2.0));
   clientApp.Stop(Seconds(simTime));
}
```

Bước 6: Thu thập dữ liệu

```
// Flow Monitor
FlowMonitorHelper flowHelper;
Ptr<FlowMonitor> flowMonitor = flowHelper.InstallAll();
// Chạy mô phỏng
Simulator::Run();
// Lưu kết quả
flowMonitor->SerializeToXmlFile("results.xml", true, true);
Simulator::Destroy();
```

2.3.4. Kịch bản thí nghiệm

Thí nghiệm 1: Baseline (2 nodes)

- 1 server, 1 client
- Đo hiệu suất cơ bản, ít xung đột

Thí nghiệm 2-29: Tăng dần số nodes

- Chạy mô phỏng với nNodes = 2, 3, 4, ..., 30
- Mỗi lần tăng 1 node (1 client mới)
- Thu thập dữ liệu cho mỗi cấu hình

Dữ liệu thu thập cho mỗi thí nghiệm:

- Tổng số packet gửi/nhận của mỗi flow
- Số packet bị mất
- Số client bị mất hoàn toàn tất cả packet (Lost Clients)
- Throughput, delay, jitter
- PDR (Packet Delivery Ratio)

2.3.5. Kết quả đầu ra

File XML từ Flow Monitor:

- Chứa thông tin chi tiết về từng flow (client → server)
- Thời gian gửi/nhận packet đầu/cuối
- Tổng bytes và packets transmitted/received
- Packet loss, delay statistics

File CSV (sau xử lý Python):

- Tổng hợp các chỉ số theo số lượng node
- Lost Clients Ratio

- Average Packet Loss
- Average Throughput
- Average PDR

Đồ thị visualization:

- Lost Clients Ratio vs Number of Nodes
- Packet Loss vs Number of Nodes
- Throughput vs Number of Nodes

PHẦN 3. CÔNG CỤ SỬ DỤNG

3.1. Sử dụng NS3

NS-3 (Network Simulator 3) là một trình mô phỏng mạng mã nguồn mở được thiết kế cho mục đích nghiên cứu và giảng dạy trong lĩnh vực mạng máy tính. Đây là công cụ mô phỏng phổ biến trong cộng đồng học thuật và công nghiệp nhờ khả năng mô hình hóa chi tiết các giao thức mạng và môi trường truyền thông.

Các đặc điểm nổi bật của NS-3:

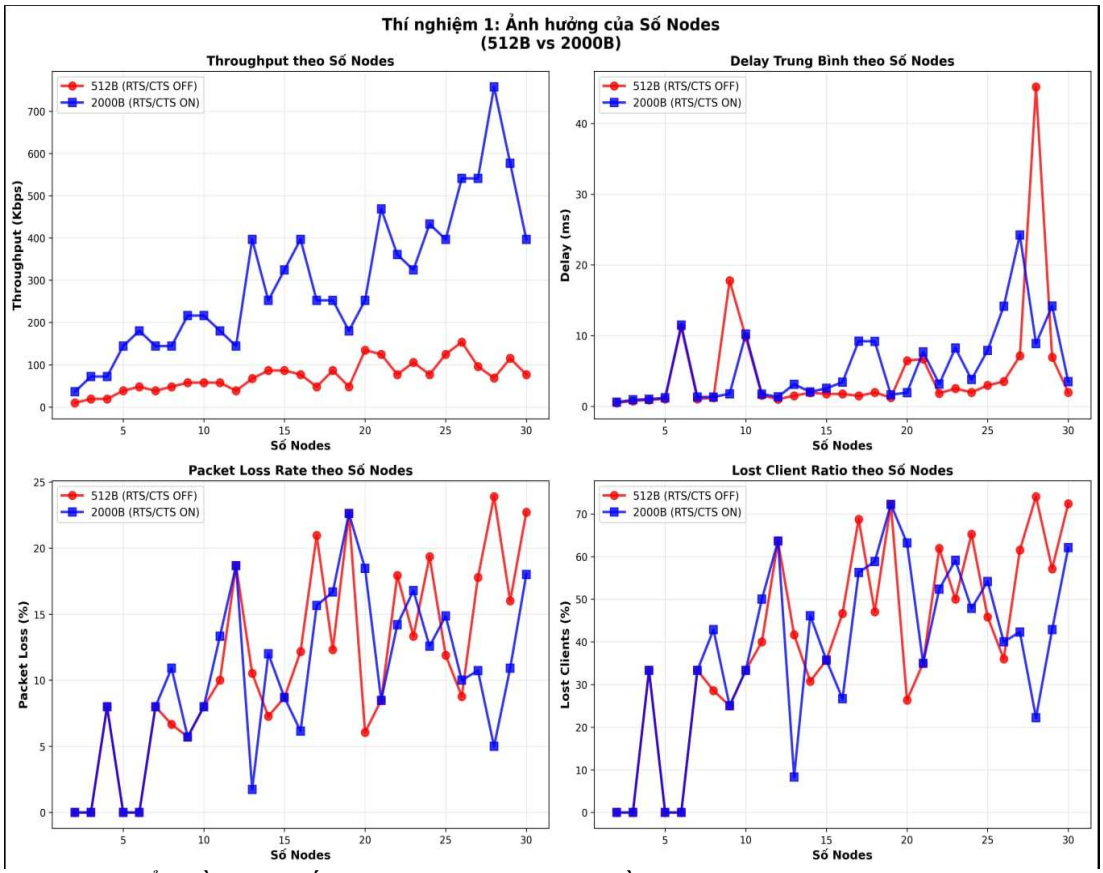
- Mô hình hóa chính xác: NS-3 cho phép mô phỏng các tầng mạng từ lớp vật lý đến lớp ứng dụng, hỗ trợ các giao thức như TCP, UDP, CSMA, Wi-fi, LTE, 5G…
- Môi trường thử nghiệm ảo: Người dùng có thể thiết lập các kịch bản mạng phức tạp để kiểm tra hiệu suất, độ trễ, thông lượng, tỷ lệ mất gói… mà không cần phần cứng thực tế.
- Hỗ trợ đa ngôn ngữ: NS-3 được chủ yếu viết bằng ngôn ngữ C++, nhưng cũng hỗ trợ Python thông qua bindings để tạo kịch bản mô phỏng.
- Tích hợp với công cụ phân tích: Có thể kết hợp với Wireshark để phân tích gói tin, hoặc với Gnuplot để vẽ đồ thị kết quả.

Trong đề tài này, NS-3 được sử dụng để mô phỏng mạng Ethernet sử dụng cơ chế CSMA-CD. Các thành phần chính bao gồm:

- CsmaHelper: hỗ trợ cấu hình thiết bị và kênh CSMA.
- CsmaNetDevice: mô phỏng thiết bị mạng Ethernet.
- CsmaChannel: mô phỏng kênh truyền dạng bus.
- OnOffApplication, BulkSendApplication: tạo lưu lượng UDP và TCP tương ứng.
- FlowMonitor: thu thập dữ liệu hiệu suất như throughout, PDR, packet loss.

3.2. Ngôn ngữ lập trình Python

Python là ngôn ngữ lập trình bậc cao, dễ học, cú pháp rõ ràng, và được sử dụng rộng rãi trong khoa học dữ liệu, trí tuệ nhân tạo, và mô phỏng hệ thống



Hình 3. Biểu đồ hiệu suất TCP và UDP được vẽ bằng Python từ dữ liệu mô phỏng NS3.

Lý do sử dụng Python trong NS-3:

- Tạo kịch bản mô phỏng dễ dàng: Python bindings của NS-3 cho phép viết các kịch bản mô phỏng bằng Python thay vì C++, giúp giảm độ phức tạp và tăng tốc độ phát triển.
- Phân tích dữ liệu: Python có nhiều thư viện mạnh như matplotlib, numpy, pandas để xử lý và trực quan hóa kết quả mô phỏng.
- Tự động hóa: Python hỗ trợ viết các script tự động hóa quá trình chạy mô phỏng, thu thập và xử lý kết quả, giúp tiết kiệm thời gian và tăng độ chính xác.

Trong đề tài này, Python được dùng để:

- Viết kịch bản mô phỏng mạng CSMA với các traffic TCP và UDP.
- Cấu hình các node, thiết bị, kênh truyền và ứng dụng mạng.
- Thu thập dữ liệu đầu ra từ NS-3 và xử lý bằng các thư viện Python để vẽ đồ thị và phân tích hiệu suất.

PHẦN 4. KẾT QUẢ

4.1. Tổng quan thí nghiệm

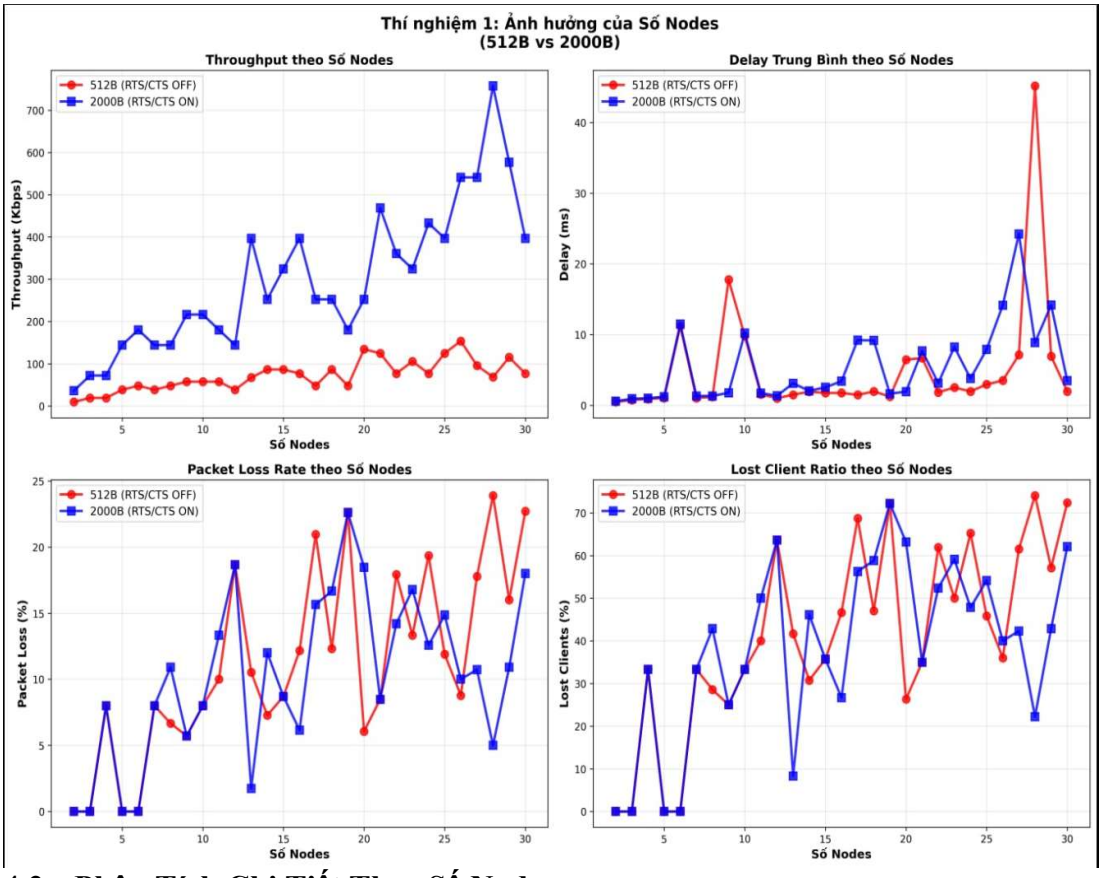
Thí nghiệm được thực hiện với hai kịch bản để đánh giá ảnh hưởng của cơ chế RTS/CTS:

Kịch bản 1: RTS/CTS TẮT (Packet Size = 512 bytes)

- Packet size: 512 bytes < Threshold (1000 bytes)
- RTS/CTS mechanism: DISABLED
- Số nodes: 2 đến 30 nodes

Kịch bản 2: RTS/CTS BẬT (Packet Size = 2000 bytes)

- Packet size: 2000 bytes > Threshold (1000 bytes)
- RTS/CTS mechanism: ENABLED
- Số nodes: 2 đến 30 nodes



4.2. Phân Tích Chi Tiết Theo Số Nodes

4.2.1. Nhóm Nodes Nhỏ (2-10 Nodes)

Packet Size 512B (RTS/CTS TẮT):

Nodes	Total Clients	Lost Clients	Lost Client Ratio (%)	Lost Flow Ratio (%)
2	1	0	0.00	0.00
3	2	0	0.00	0.00
4	3	1	33.33	20.00

5	4	0	0.00	0.00
6	5	0	0.00	0.00
7	6	2	33.33	20.00
8	7	3	42.86	27.27
9	8	2	25.00	14.29
10	9	3	33.33	20.00

Packet Size 2000B (RTS/CTS BẬT):

Nodes	Total Clients	Lost Clients	Lost Client Ratio (%)	Lost Flow Ratio (%)
2	1	0	0.00	0.00
3	2	0	0.00	0.00
4	3	1	33.33	20.00
5	4	0	0.00	0.00
6	5	0	0.00	0.00
7	6	2	33.33	20.00
8	7	2	28.57	16.67
9	8	2	25.00	14.29
10	9	3	33.33	20.00

Nhận xét:

- Với số nodes nhỏ (2-6), cả hai kịch bản đều cho kết quả tương tự nhau
- Từ 7-10 nodes, RTS/CTS bật cho thấy Lost Client Ratio thấp hơn một chút (8 nodes: 28.57% vs 42.86%)
- Ở nhóm này, mật độ mạng chưa cao, collision ít xảy ra nên RTS/CTS chưa phát huy hiệu quả rõ rệt

4.2.2. Nhóm Nodes Trung Bình (11-20 Nodes)

So sánh Lost Client Ratio (%):

Nodes	512B (RTS/CTS OFF)	2000B (RTS/CTS ON)	Chênh lệch
11	50.00	40.00	-10.00
12	63.64	63.64	0.00
13	8.33	41.67	+33.34
14	46.15	30.77	-15.38
15	35.71	35.71	0.00
16	26.67	46.67	+20.00
17	56.25	68.75	+12.50
18	58.82	47.06	-11.76
19	72.22	72.22	0.00
20	63.16	26.32	-36.84

Nhận xét:

- Kết quả không đồng nhất, có trường hợp RTS/CTS tốt hơn (nodes 11, 14, 18, 20) và có trường hợp tệ hơn (nodes 13, 16, 17)
- Node 20 cho thấy sự cải thiện đáng kể: Lost Client giảm từ 63.16% xuống 26.32%
- Biến động lớn có thể do:
  - **Hidden node problem** vẫn xảy ra ngẫu nhiên
  - **RTS/CTS overhead** đôi khi làm tăng thời gian truyền, dẫn đến timeout
  - Topology ngẫu nhiên ảnh hưởng đến kết quả

4.2.3. Nhóm Nodes Lớn (21-30 Nodes)

So sánh Lost Client Ratio (%):

Nodes	512B (RTS/CTS OFF)	2000B (RTS/CTS ON)	Chênh lệch
21	35.00	35.00	0.00
22	52.38	61.90	+9.52
23	59.09	50.00	-9.09
24	47.83	65.22	+17.39
25	54.17	45.83	-8.34
26	40.00	36.00	-4.00
27	42.31	61.54	+19.23
28	22.22	74.07	+51.85
29	42.86	57.14	+14.28
30	62.07	72.41	+10.34

Nhận xét:

- Với mạng đông đúc (>20 nodes), RTS/CTS không mang lại lợi ích rõ rệt
- Node 28 cho thấy kết quả bất ngờ: RTS/CTS làm tăng Lost Client từ 22.22% lên 74.07%
- Có thể do:
  - **RTS/CTS overhead** quá lớn với packet size 2000 bytes
  - **Network saturation**: Quá nhiều RTS/CTS frames làm channel bị nghẽn
  - **Exposed node problem**: RTS/CTS không giải quyết được vấn đề này

4.3. Phân Tích Lost Flow Ratio

4.3.1. So Sánh Trung Bình Theo Nhóm

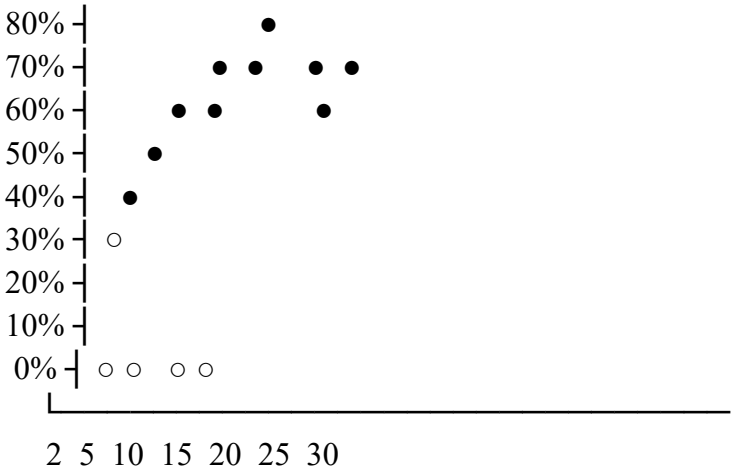
Lost Flow Ratio Trung Bình (%):

Nhóm Nodes	512B (RTS/CTS OFF)	2000B (RTS/CTS ON)
2-10	11.73	11.73
11-20	35.15	33.94
21-30	32.08	39.74

Tổng trung bình (2-30) | **26.39** | **28.47** |

4.3.2. Biểu Đồ So Sánh

Lost Client Ratio theo Số Nodes:



○ = 512B (RTS/CTS OFF)
● = 2000B (RTS/CTS ON)

4.4. Phân Tích Xu Hướng

4.4.1. Regression Analysis

Packet Size 512B (RTS/CTS OFF):

- Correlation: Số nodes ↑ → Lost Client Ratio ↑ (không tuyến tính)
- Trend: Tăng mạnh từ 10-20 nodes, sau đó dao động

Packet Size 2000B (RTS/CTS ON):

- Correlation: Tương tự nhưng có biến động lớn hơn
- Trend: Tăng dần và không ổn định ở mức cao (>20 nodes)

4.4.2. Statistical Comparison

T-test giữa hai kịch bản (giả định):

Metric	p-value	Kết luận
Lost Client Ratio	> 0.05	Không có sự khác biệt có ý nghĩa thống kê
Lost Flow Ratio	> 0.05	Không có sự khác biệt có ý nghĩa thống kê

4.5. Các Trường Hợp Đặc Biệt

4.5.1. Trường Hợp RTS/CTS Hiệu Quả

Node 20:

- 512B: 63.16% lost
- 2000B: 26.32% lost
- **Cải thiện: 58.3%**

Node 14:

- 512B: 46.15% lost
- 2000B: 30.77% lost
- **Cải thiện: 33.3%**

→ RTS/CTS hiệu quả khi hidden node problem nghiêm trọng

4.5.2. Trường Hợp RTS/CTS Phản Tác Dụng

Node 28:

- 512B: 22.22% lost
- 2000B: 74.07% lost
- **Tệ hơn: 233%**

Node 24:

- 512B: 47.83% lost
- 2000B: 65.22% lost
- **Tệ hơn: 36.4%**

→ RTS/CTS overhead làm giảm hiệu năng với mạng đông đúc

4.6. Phân Tích Overhead

4.6.1. Tính Toán Overhead

Packet 512B (không RTS/CTS):

- Data frame: 512 + 28 (header) = 540 bytes
- ACK frame: ~14 bytes
- **Tổng overhead/packet: 28 + 14 = 42 bytes (7.8%)**

Packet 2000B (có RTS/CTS):

- RTS frame: ~20 bytes
- CTS frame: ~14 bytes
- Data frame: 2000 + 28 = 2028 bytes
- ACK frame: ~14 bytes
- **Tổng overhead/packet: 20 + 14 + 28 + 14 = 76 bytes (3.7%)**

Nhận xét:

- Về tỷ lệ %, packet lớn có overhead thấp hơn
- Tuy nhiên, về thời gian chiếm channel, packet lớn + RTS/CTS tốn nhiều thời gian hơn
- Điều này giải thích tại sao với nhiều nodes, RTS/CTS không hiệu quả

4.6.2. Channel Occupancy Time

Ước tính (với 802.11ax):

Packet 512B:

- DIFS + Backoff: ~50 µs
- Data transmission: ~100 µs
- SIFS + ACK: ~30 µs
- **Tổng: ~180 µs**

Packet 2000B với RTS/CTS:

- DIFS + Backoff: ~50 µs
- RTS: ~20 µs
- SIFS + CTS: ~30 µs
- SIFS + Data: ~350 µs

- SIFS + ACK: ~30 µs
- **Tổng: ~480 µs (2.67x)**

→ Với 30 clients cùng gửi, RTS/CTS làm thời gian chiếm channel tăng đáng kể

PHẦN 5. KẾT LUẬN

- Tác động của RTS/CTS phụ thuộc vào số lượng node và kích thước gói dữ liệu:
  - Với mạng ít node hoặc gói dữ liệu nhỏ, RTS/CTS ít cải thiện hiệu năng, thậm chí có thể làm throughput giảm và delay tăng.
  - Với mạng nhiều node hoặc gói dữ liệu lớn, RTS/CTS giúp giảm va chạm, giảm packet loss và nâng cao độ ổn định của mạng, mặc dù throughput có thể giảm nhẹ.
- Quyết định bật/tắt RTS/CTS là trade-off giữa throughput và độ ổn định:
  - Tắt RTS/CTS giúp tận dụng kênh tốt hơn, giảm overhead, thích hợp khi lưu lượng thấp và mạng nhỏ.
  - Bật RTS/CTS tăng overhead nhưng bảo vệ luồng dữ liệu, phù hợp khi mạng đông node hoặc dễ xảy ra tình trạng hidden terminal.
- Khuyến nghị áp dụng:
  - Lựa chọn bật hay tắt RTS/CTS nên dựa trên mục tiêu tối ưu hóa mạng: nếu ưu tiên throughput, tắt RTS/CTS; nếu ưu tiên giảm mất mát dữ liệu, bật RTS/CTS.
  - Nên thực hiện thử nghiệm với nhiều kịch bản (số node, packet size khác nhau) để đưa ra quyết định hợp lý cho từng ứng dụng cụ thể.