

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN, ĐHQG-HCM
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG



BÁO CÁO ĐỒ ÁN MÔN HỌC
MÔN HỌC: HỆ THỐNG NHÚNG VÀ MẠNG KHÔNG DÂY
ĐỀ TÀI: SỬ DỤNG NETSIMULYZER ĐỂ MÔ
PHỎNG LẠI UAV-BASED NETWORK

Môn học: NT131.P12 – Hệ thống nhúng và mạng không dây

Giảng viên hướng dẫn: ThS. Đặng Lê Bảo Chương

Thực hiện bởi nhóm 1, bao gồm:

1. Nguyễn Dương Hoàng Phúc	22521125	Trưởng nhóm
2. Phan Văn Tài	22521284	Thành viên
3. Phạm Ngọc Sơn	22521256	Thành viên

Thời gian thực hiện: 25/10/2024 – 15/12/2024

MỤC LỤC

MỤC LỤC	2
DANH SÁCH HÌNH ẢNH	6
DANH SÁCH BẢNG.....	10
TÓM TẮT ĐỒ ÁN	12
MỞ ĐẦU	13
CHƯƠNG I. MỞ ĐẦU ĐỀ TÀI	15
1.1 Tổng quan về mạng UAV	15
1.1.1 UAV (Unmanned Aerial Vehicle) là gì?	15
1.1.2 Vai trò và ứng dụng của UAV trong đời sống hiện đại.....	16
1.2 Thách thức khi triển khai mạng UAV	18
1.2.1 Hạn chế về năng lượng	18
1.2.2 Kết nối không ổn định và độ trễ cao.....	18
1.2.3 Quản lý không gian bay	19
1.2.4 An ninh mạng	19
1.2.5 Vấn đề pháp lý	19
1.2.6 Môi trường khắc nghiệt	20
1.3 Mục tiêu và phạm vi của đề tài	20
1.3.1 Mục tiêu tổng quát: Nghiên cứu và mô phỏng mạng UAV bằng NetSimulyzer.	20
1.3.2 Mục tiêu cụ thể:	21
1.3.3 Phạm vi nghiên cứu:	22
1.4 Phương pháp nghiên cứu và triển khai	22
1.4.1 Nghiên cứu lý thuyết.	22
1.4.2 Thiết kế mô hình mô phỏng trên ns-3.	23
1.4.3 Triển khai mô hình mô phỏng và thu thập dữ liệu bằng NetSimulyzer.	24
1.4.4 Phân tích và đánh giá kết quả.	25
CHƯƠNG II. CƠ SỞ LÝ THUYẾT	27
2.1 Mạng MANET.....	27

2.1.1 Giới thiệu chung về mạng MANET	27
2.1.2 Đặc điểm của mạng MANET	28
2.1.3 Kiểu kết nối và cơ chế hoạt động	29
2.1.4 Định tuyến trong MANET.....	35
2.2 NS-3	41
2.2.1 Giới thiệu về ns-3: Lịch sử, mục tiêu phát triển.....	41
2.2.2 Đặc điểm nổi bật của ns-3	42
2.2.3 Kiến trúc mô hình mạng cơ bản của ns-3	43
2.2.4 So sánh giữa ns-2 và ns-3	45
2.3 NetSimulyzer	46
2.3.1 Giới thiệu về NetSimulyzer: Mục tiêu, vai trò trong trực quan hóa mô phỏng mạng	46
2.3.2 Kiến trúc của NetSimulyzer	47
2.3.3 Quy trình luồng dữ liệu hoạt động của NetSimulyzer trong mô phỏng mạng	49
2.3.4 Cấu trúc tệp và thư mục của NetSimulyzer.....	49
2.4 Mạng UAV-based network	52
2.4.1 Tổng quan về mạng UAV-based	52
2.4.2 Mô hình mạng UAV trong NetSimulyzer:.....	52
CHƯƠNG III. THIẾT KẾ VÀ TRIỂN KHAI MÔ PHỎNG	56
3.1 Cài đặt môi trường mô phỏng.....	56
3.1.1 NS-3.....	56
3.1.2 NetSimulyzer	58
3.2 Phân tích Code.....	65
3.2.1 Khai báo biến:.....	65
3.2.2 Khai báo số lượng các node:	65
3.2.3 Cài đặt wifi và định tuyến cho các node:	66
3.2.4 Tạo Socket giao tiếp:	67
3.2.5 Cài đặt vị trí, mô hình di chuyển cho các node:	67
3.2.6 Mô hình cho NetSimulyzer:	68

3.2.7 Theo dõi năng lượng:.....	70
3.2.8 Theo dõi luồng dữ liệu:	71
3.2.9 NetAnim:	71
3.2.10 Wireshark:	72
3.2.11 Vẽ biểu đồ:.....	72
3.2.12 Các hàm con:	72
3.3 Triển khai kịch bản	75
3.3.1 Kịch bản 1: Sự khác nhau về số lượng UAV	76
3.3.2 Kịch bản 2: Sự khác nhau kích thước map.....	76
3.3.3 Kịch bản 3: Sự khác nhau về công suất phát.....	76
CHƯƠNG IV. KẾT QUẢ VÀ PHÂN TÍCH	78
4.1 Phân tích dữ liệu dựa trên các tool đã cài đặt.....	78
4.1.1 Wireshark để phân tích thông tin chi tiết các gói tin:	78
4.1.2 NetAnim để quan sát dưới góc nhìn 2D:	78
4.1.3 Kiểm tra thông tin mạng (IP,MAC) của các thiết bị trong NetAnim:	79
4.1.4 Phân tích luồng dữ liệu Flowmonitor:.....	79
4.1.5 Bảng định tuyến theo thời gian:	81
4.1.6 Vẽ biểu đồ để phân tích trong NetSimulyzer:	82
4.1.7 Các thông tin khác:	84
4.2 Kịch bản 1: Sự khác nhau về số lượng UAV	84
4.2.1 Số lượng 10 UAV:.....	84
4.2.2 Số lượng 25 UAV:.....	85
4.2.3 Số lượng 50 UAV:.....	86
4.2.4 So sánh.....	87
4.3 Kịch bản 2: Sự khác nhau về kích thước map	87
4.3.1 Kích thước 1250 x 1250 m	88
4.3.2 Kích thước 1500 x 1500 m	89
4.3.3 Kích thước 2000 x 2000 m	90
4.3.4 So sánh.....	90
4.4 Kịch bản 3: Sự khác nhau về công suất phát	91

4.4.1 Công suất phát 20 dBm	91
4.4.2 Công suất phát 45 dBm	92
4.4.3 Công suất phát 75 dBm	93
4.4.4 So sánh.....	94
CHƯƠNG V. KẾT LUẬN.....	96
5.1 Tóm tắt kết quả đạt được	96
5.2 Kết luận về vai trò của NetSimulyzer trong mô phỏng mạng UAV	97
5.3 Đóng góp của đồ án và giá trị thực tiễn.....	97
5.4 Hướng phát triển tiếp theo của đề tài đồ án.....	97
CHƯƠNG VI. TÀI LIỆU THAM KHẢO.....	100

DANH SÁCH HÌNH ẢNH

Hình 1: Minh họa mạng MANET	28
Hình 2: Biểu đồ mạng MANET	28
Hình 3: Mạng máy chủ di động	30
Hình 4: Hình minh họa mạng có các thiết bị di động không đồng nhất.....	30
Hình 5: Chế độ IEEE-ad hoc	31
Hình 6: Chế độ cơ sở hạ tầng	31
Hình 7: Singal-hop.....	32
Hình 8: Multi-hop	33
Hình 9: Mobile multi-hop.....	33
Hình 10 : Mô hình mạng phân cấp	34
Hình 11: Mô hình mạng Aggregate	35
Hình 12: Mô tả giao thức DSR	39
Hình 13: Network Simulator 3 - NS3	41
Hình 14: Kiến trúc mô hình mạng của ns-3	44
Hình 15: Giao diện mô phỏng NetSimulyzer	47
Hình 16: NetSimulyzer ns-3 Module	47
Hình 17: Các node UAV	53
Hình 18: Trạm mặt đất.....	54
Hình 19: Hệ điều hành của máy ảo sử dụng.....	56
Hình 20: Nâng cấp hệ thống	56
Hình 21: Cài đặt thư viện	57
Hình 22: Giao diện trang web nsnam	57
Hình 23: File ns-3 sau khi giải nén.....	57
Hình 24: Cài đặt ns-3 vào máy ảo	58
Hình 25: Chạy thử lệnh Hello Simulator.....	58
Hình 26: NetSimulyzer module sau khi giải nén.....	58
Hình 27: NetSimulyzer software sau khi giải nén.....	59
Hình 28: Chạy thử kịch bản NetSimulyzer	59

Hình 29: File .json thu được	60
Hình 30: Click 2 lần để mở NetSimulyzer	60
Hình 31: Thực hiện load file kịch bản.....	60
Hình 32: Chọn đúng file .json của kịch bản	61
Hình 33: Giao diện của NetSimulyzer khi load kịch bản.....	61
Hình 34: Nơi chứa các file đối tượng mặc định của NetSimulyzer	62
Hình 35: File hình ảnh của mô hình	62
Hình 36: Thư viện của NetSimulyzer	62
Hình 37: Cấu trúc dựa theo cấu trúc mặc định trong file	62
Hình 38: Chính sửa đường dẫn load ảnh	63
Hình 39: Chọn Preview Model	64
Hình 40: Chọn mô hình	64
Hình 41: Mô hình xe cứu hỏa	65
Hình 42: Các biến cho kịch bản	65
Hình 43: Mô hình cho NetSimulyzer	65
Hình 44: Số lượng node.....	65
Hình 45: Cài đặt wifi lên node	66
Hình 46: Gán IP cho các Node	66
Hình 47: Tạo Socket cho Ground Station và Drone.....	67
Hình 48: Các vị trí là ngẫu nhiên.....	67
Hình 49: Cấu hình di chuyển cho mô hình.....	68
Hình 50: Ghi vị trí các Node	68
Hình 51: Đối tượng chính xuất file .json.....	68
Hình 52: Giới hạn trong phạm vi hình chữ nhật.....	69
Hình 53: Ghi log NetSimulyzer.....	69
Hình 54: Để lại vết sáng khi di chuyển trong NetSimulyzer.....	69
Hình 55: Tạo nhiều ngọn núi phủ đầy phạm vi map	69
Hình 56: Mô hình, kích cỡ, vị trí của các Node	70
Hình 57: Năng lượng giảm 0.0174 mỗi lần truyền.....	70
Hình 58: Biến để theo dõi luồng dữ liệu	71

Hình 59: Truy xuất và phân tích luồng	71
Hình 60: Ghi dữ liệu thành file.....	71
Hình 61: Chỉnh các mô tả cho đối tượng trong 2D	71
Hình 62: Theo dõi và lưu các thông tin cần thiết	71
Hình 63: Bật theo dõi bằng Wireshark.....	72
Hình 64: Gọi callback để ghi vị trí liên tục	72
Hình 65: Cài đặt hiện biểu đồ gói tin đến.....	72
Hình 66: Hàm ghi log di chuyển	73
Hình 67: Hàm được gọi callback.....	73
Hình 68: Lên lịch gọi hàm.....	73
Hình 69: Cấu trúc lưu trữ thông tin	73
Hình 70: Tính khoảng cách giữa drone với con người.....	74
Hình 71: Gọi lại chính nó sau 3s	74
Hình 72: Hàm ReceivePacket.....	75
Hình 73: Hàm LogEnergy được gọi lại mỗi 1s	75
Hình 74: Số lượng Drone	76
Hình 75: Kích thước map	76
Hình 76: Công suất phát	76
Hình 77: Wireshak	78
Hình 78: NetAnim	79
Hình 79: IP, MAC	79
Hình 80: FlowMonitor.....	80
Hình 81: Flow Probes	80
Hình 82: Routing Table	81
Hình 83: Chart(biểu đồ ghi lại lịch sử đường bay của từng Drone).....	82
Hình 84: Chart(biểu đồ ghi lại luồng dữ liệu mà trạm mặt đất nhận được từ Drone) ..	83
Hình 85: Thông tin khác	84
Hình 86: Kết quả thu được 10 UAV	84
Hình 87: Flow Monitor 10 UAV.....	85
Hình 88: Kết quả thu được 25 UAV	85

Hình 89: Flow Monitor 25 UAV.....	86
Hình 90: Kết quả thu được 50 UAV	86
Hình 91: Flow Monitor 50 UAV.....	87
Hình 92: Kết quả thu được 1250 x 1250 m	88
Hình 93: Flow Monitor 1250 x 1250 m.....	88
Hình 94: Kết quả thu được 1500 x 1500 m	89
Hình 95: Flow Monitor 1500 x 1500 m.....	89
Hình 96: Kết quả thu được 2000 x 2000 m	90
Hình 97: Flow Monitor 2000 x 2000 m.....	90
Hình 98: Kết quả thu được 20 dBm.....	91
Hình 99: Flow Monitor 20 dBm	92
Hình 100: Kết quả thu được 45 dBm.....	92
Hình 101: Flow Monitor 45 dBm	93
Hình 102: Kết quả thu được 75 dBm.....	93
Hình 103: Flow Monitor 75 dBm	94
Hình 104: Thời gian đến của mức công suất 45 dBm	94
Hình 105: Thời gian đến của mức công suất 75 dBm	95

DANH SÁCH BẢNG

Bảng 1. Bảng danh mục từ viết tắt	11
Bảng 2: So sánh giữa ns-2 và ns-3	45
Bảng 3: Thông tin kịch bản 1	76
Bảng 4: Thông tin kịch bản 2	76
Bảng 5: Thông tin kịch bản 3	77
Bảng 6: Bảng thông tin kịch bản 1	84
Bảng 7: Bảng so sánh số lượng UAV	87
Bảng 8: Bảng thông tin kịch bản 2	88
Bảng 9: Bảng so sánh kích thước map	91
Bảng 10: Bảng thông tin kịch bản 3	91
Bảng 11: Bảng so sánh công suất phát	95
Bảng 12: Bảng phân công công việc	99

DANH MỤC TỪ VIẾT TẮT

STT	Ký hiệu chữ viết tắt	Chữ viết đầy đủ
1	MANET	Mobile Ad hoc Network
2	NS-3	Network Simulator 3
3	NS-2	Network Simulator 2
4	IEEE	Institute of Electrical and Electronics Engineers
5	AP	Access Point
6	TCP/IP	Transmission Control Protocol/Internet Protocol
7	UDP	User Datagram Protocol
8	LAN	Local Area Network
9	ID	Identifier
10	DV	Distance Vector
11	AODV	Ad-hoc On-demand Distance Vector
12	OLSR	Optimized Link State Routing
13	DSR	Dynamic Source Routing
14	MAC	Media Access Control
15	GPRS	General Packet Radio Service

Bảng 1. Bảng danh mục từ viết tắt

TÓM TẮT ĐỒ ÁN

Đồ án này tập trung vào việc nghiên cứu và ứng dụng công cụ NetSimulyzer để mô phỏng một mạng lưới dựa trên UAV (Unmanned Aerial Vehicle), nhằm khám phá và đánh giá các khía cạnh hiệu năng của mạng trong một môi trường động. Trong bối cảnh các ứng dụng của UAV ngày càng trở nên đa dạng và phức tạp, việc mô phỏng và quản lý hiệu quả các mạng lưới UAV là rất cần thiết. Đồ án sử dụng NetSimulyzer kết hợp với trình mô phỏng mạng ns-3 để xây dựng môi trường mô phỏng chi tiết và trực quan, cho phép các nhà nghiên cứu, quản trị mạng đánh giá khả năng kết nối, định tuyến và quản lý tài nguyên của mạng lưới UAV trong các điều kiện khác nhau.

Mục tiêu chính của đồ án là tạo ra một hệ thống mô phỏng linh hoạt và trực quan, có thể ứng dụng cho việc nghiên cứu các giao thức mạng và cơ chế định tuyến trong mạng MANET (Mobile Ad hoc Network) trên UAV-based network. Đồ án xây dựng các kịch bản mô phỏng khác nhau như thay đổi số lượng UAV, kích thước không gian mô phỏng, và công suất phát, đồng thời sử dụng giao thức định tuyến AODV. Kết quả từ mô phỏng được thu thập thông qua FlowMonitor của ns-3 và trực quan hóa bằng NetSimulyzer, cho phép đánh giá các thông số như độ trễ, thông lượng, jitter, và tỷ lệ mất gói tin. Bên cạnh đó, việc sử dụng NetAnim cũng được kết hợp để trực quan hóa vị trí của UAV và routing table.

Đồ án cũng đề cập chi tiết về cấu trúc và các tính năng của NetSimulyzer, các công cụ hỗ trợ của ns-3, trình bày các kết quả quan sát được như đồ thị vị trí của UAV, thống kê các thông số của mạng lưới như throughput, jitter, packet loss, và phân tích ảnh hưởng của số lượng UAV, kích thước không gian mô phỏng và công suất phát. Qua đó, đồ án không chỉ minh họa khả năng của NetSimulyzer trong việc trực quan hóa kết quả mô phỏng mà còn cung cấp các số liệu, giúp đánh giá và tối ưu hiệu năng mạng.

Các thách thức khi triển khai mạng UAV và hướng cải thiện được đề cập, bao gồm việc ứng dụng học tăng cường (Reinforcement Learning - RL) để tối ưu hóa các khía cạnh như định tuyến động, quản lý tài nguyên, và điều khiển UAV. Những hướng phát triển này có thể tiếp tục mở rộng phạm vi nghiên cứu, hướng đến việc xây dựng một hệ thống mạng UAV thông minh và linh hoạt, đáp ứng được các yêu cầu của các ứng dụng thực tiễn như giám sát môi trường, cứu hộ cứu nạn, và giao thông thông minh.

Cuối cùng, đồ án này khẳng định rằng, việc sử dụng kết hợp NetSimulyzer và ns-3 là một giải pháp hiệu quả, cung cấp khả năng mô phỏng chi tiết và trực quan cho mạng UAV, giúp các nhà nghiên cứu và quản trị mạng có cơ sở để đánh giá và tối ưu hóa hệ thống mạng trong thực tế. Đồng thời, kết quả từ đồ án cũng là nền tảng vững chắc cho việc tiếp tục nghiên cứu và phát triển các phương pháp quản lý mạng lưới UAV tiên tiến hơn.

MỞ ĐẦU

Trong kỷ nguyên công nghệ số ngày nay, hạ tầng mang không dây đã trở thành một phần không thể thiếu trong nhiều khía cạnh của đời sống xã hội và kinh tế. Đặc biệt, sự phát triển của các phương tiện bay không người lái (UAVs) đã mở ra những ứng dụng đầy tiềm năng trong nhiều lĩnh vực, từ giám sát môi trường, giao thông thông minh, đến cứu hộ cứu nạn và nhiều ứng dụng khác. Tuy nhiên, việc quản lý và tối ưu hóa các mạng lưới UAVs, với tính chất di động và tự cấu hình của chúng, đang đặt ra những thách thức mới cho các nhà nghiên cứu và quản trị mạng.

Mạng lưới dựa trên UAV (UAV-based networks) đòi hỏi một cách tiếp cận quản lý và mô phỏng khác biệt so với các mạng truyền thống. Các UAV di chuyển liên tục, tạo ra các thay đổi động trong cấu trúc liên kết và yêu cầu các giao thức định tuyến linh hoạt để đảm bảo kết nối ổn định và thông suốt. Việc giám sát và phân tích hiệu năng của các mạng lưới UAV phức tạp này là một nhiệm vụ không hề dễ dàng, đòi hỏi các công cụ mô phỏng và trực quan hóa chuyên dụng.

Trong bối cảnh đó, NetSimulyzer nổi lên như một công cụ đầy hứa hẹn, cung cấp khả năng trực quan hóa các kịch bản mô phỏng mạng 3D, đặc biệt hữu ích trong việc mô phỏng chuyển động của các UAV. Kết hợp với trình mô phỏng mạng ns-3, NetSimulyzer cho phép các nhà nghiên cứu xây dựng các mô hình mạng UAV chi tiết, đánh giá hiệu năng, cũng như thử nghiệm các phương pháp điều khiển và quản lý mạng khác nhau.

Đề tài "Sử dụng NetSimulyzer để mô phỏng lại UAV-based network" nhằm mục đích khai thác tiềm năng của hai công cụ này trong việc nghiên cứu các khía cạnh quan trọng của mạng UAV, đồng thời cung cấp một nền tảng thực nghiệm cho việc phát triển và tối ưu hóa các giao thức mạng tương lai. Đồ án này không chỉ giúp chúng em hiểu sâu hơn về giao thức mạng MANET áp dụng cho UAV mà còn đóng góp vào việc phát triển các giải pháp mô phỏng mạng thực tiễn, có giá trị ứng dụng cao trong các lĩnh vực đòi hỏi độ linh hoạt, tin cậy và tự động hóa cao.

Chúng em hy vọng rằng, những kết quả nghiên cứu và phân tích từ đề tài sẽ đóng góp một phần nhỏ vào việc nâng cao hiểu biết và kỹ năng quản lý, vận hành các mạng lưới UAV phức tạp, từ đó giúp chúng trở thành một phần quan trọng trong sự phát triển của công nghệ mạng không dây trong tương lai. Vì trình độ nghiên cứu thực tiễn và thời gian thực hiện còn hạn chế cùng với đó sự tiếp nhận kiến thức vẫn còn tồn tại một số hạn chế nhất định. Do đó, trong quá trình hoàn thành đồ án chắc chắn không tránh khỏi những thiếu sót. Bản thân nhóm em rất mong nhận được những góp ý đến từ thầy để báo cáo được hoàn thiện hơn.

Cuối cùng, chúng em xin gửi lời cảm ơn sâu sắc nhất đến Ths. Đặng Lê Bảo Chương đã giúp đỡ và hướng dẫn tận tình chúng em hoàn thiện đề tài này. Kính chúc thầy thật

nhiều sức khỏe, hạnh phúc và thành công trên con đường giảng dạy và nghiên cứu của mình.

Chúng em xin chân thành cảm ơn!



CHƯƠNG I. MỞ ĐẦU ĐỀ TÀI

1.1 Tổng quan về mạng UAV

1.1.1 UAV (Unmanned Aerial Vehicle) là gì?

⇒ **Định nghĩa mở rộng:**

⇒ UAV, thường được gọi là drone (máy bay không người lái) hoặc flycam (máy quay trên không), không chỉ đơn thuần là các phương tiện bay không có người lái. Chúng là các hệ thống phức tạp, tích hợp nhiều công nghệ tiên tiến như:

- **Hệ thống dẫn đường:** Các UAV hiện đại được trang bị GPS, IMU (Inertial Measurement Unit), và các cảm biến khác để định vị và duy trì vị trí chính xác.
- **Hệ thống điều khiển:** Các hệ thống điều khiển từ xa hoặc tự động cho phép UAV thực hiện các nhiệm vụ phức tạp, bay theo hành trình đã định hoặc tự động phản ứng với các điều kiện môi trường.
- **Hệ thống cảm biến:** Các UAV có thể mang theo nhiều loại cảm biến khác nhau (ví dụ: máy ảnh, camera hồng ngoại, LiDAR) để thu thập dữ liệu đa dạng.
- **Hệ thống truyền thông:** Các UAV sử dụng các kênh vô tuyến để giao tiếp với trạm điều khiển và truyền dữ liệu.
- **Tính linh hoạt:** UAV có thể hoạt động ở nhiều độ cao, tốc độ và môi trường khác nhau, từ các khu vực đô thị đến các vùng núi cao, hoặc các khu vực nguy hiểm mà con người khó tiếp cận.

⇒ **Phân loại chi tiết:**

⇒ **Theo hình dáng:**

- **Cánh cố định (Fixed-wing):** Bay bằng lực nâng của cánh, thích hợp cho các chuyến bay đường dài và tốc độ cao.
- **Cánh quạt (Multi-rotor):** Cất và hạ cánh thẳng đứng, cơ động linh hoạt, thích hợp cho các ứng dụng chụp ảnh và giám sát ở không gian hẹp.
- **Cánh nghiêng (VTOL - Vertical Take-off and Landing):** Kết hợp ưu điểm của cả hai loại trên, có thể cất cánh thẳng đứng và bay ở tốc độ cao.

○ **Theo kích thước:**

- **Micro UAV:** Kích thước nhỏ, thường được sử dụng cho các ứng dụng trong nhà hoặc các khu vực có không gian hạn chế.

- **Mini UAV:** Kích thước trung bình, thường dùng trong các ứng dụng giám sát và cứu hộ.
- **Medium UAV:** Kích thước lớn, có khả năng mang tải trọng lớn hơn, ứng dụng trong giao hàng và giám sát tầm xa.
- **Large UAV:** UAV có kích thước lớn, thường được sử dụng trong quân sự và các ứng dụng quy mô lớn.

* **Kết luận:** Sự đa dạng và linh hoạt của UAV đã tạo nên những công cụ mạnh mẽ, mang lại nhiều ứng dụng đột phá trong nhiều lĩnh vực của cuộc sống hiện đại.

1.1.2 Vai trò và ứng dụng của UAV trong đời sống hiện đại.

Luận điểm: Ứng dụng của UAV ngày càng mở rộng, không chỉ giới hạn trong các hoạt động quân sự mà còn mang lại lợi ích thiết thực trong nhiều khía cạnh của đời sống.

Phân tích chi tiết hơn các ứng dụng:

* Trong quân sự:

- + **Trinh sát và giám sát:** UAV cung cấp khả năng giám sát liên tục trên diện rộng, có thể thu thập dữ liệu hình ảnh, video và các thông tin tình báo khác một cách bí mật, hiệu quả hơn so với con người. Chúng giúp phát hiện sớm các hoạt động bất thường, giúp quân đội phản ứng kịp thời.
- + **Tấn công mục tiêu:** UAV có thể được trang bị tên lửa, bom hoặc các loại vũ khí khác, có khả năng tấn công các mục tiêu chính xác và giảm thiểu rủi ro cho quân đội. Điều này làm giảm đáng kể thương vong cho quân đội khi triển khai các nhiệm vụ quân sự.
- + **Hỗ trợ liên lạc:** UAV được sử dụng như các trạm tiếp sóng, giúp duy trì liên lạc ở các khu vực khó khăn hoặc trong điều kiện thiên tai, nơi mà các hệ thống liên lạc truyền thống có thể bị gián đoạn.

* Trong giao thông và logistics:

- + **Giao hàng và vận chuyển:** UAV có thể giao hàng trong các khu vực đô thị, giảm thiểu thời gian và chi phí vận chuyển. Ứng dụng trong giao hàng ở vùng sâu vùng xa, giúp đưa hàng hóa đến các khu vực khó tiếp cận một cách nhanh chóng. Vận chuyển thuốc men, thiết bị y tế khẩn cấp, và hàng hóa quan trọng khác.
- + **Quản lý giao thông:** UAV có khả năng giám sát và phân tích tình hình giao thông, giúp các cơ quan quản lý đưa ra quyết định về điều tiết giao thông một cách hiệu quả hơn. Phát hiện và báo cáo các tai nạn giao thông, giúp lực lượng chức năng đến hiện trường nhanh chóng hơn.

+ **Khảo sát và bảo trì cơ sở hạ tầng:** Kiểm tra đường dây điện trên cao, phát hiện các điểm hư hỏng hoặc các vị trí cần bảo trì, giúp giảm chi phí và thời gian bảo trì. * Kiểm tra cầu cống, đường ống dẫn dầu, và các công trình khác ở những nơi mà con người khó tiếp cận.

* **Trong nông nghiệp:**

+ **Giám sát mùa màng:** UAV giúp theo dõi sự phát triển của cây trồng trên diện rộng, xác định các khu vực bị sâu bệnh hoặc thiếu nước. Đưa ra các biện pháp chăm sóc cây trồng kịp thời và hiệu quả.

+ **Phân tích đất đai:** Các cảm biến trên UAV có thể phân tích thành phần dinh dưỡng, độ ẩm của đất. Giúp nông dân quyết định sử dụng phân bón và các chất điều hòa sinh trưởng hiệu quả hơn.

+ **Phun thuốc trừ sâu:** UAV có thể phun thuốc trừ sâu một cách chính xác, giảm thiểu lãng phí và tác động tiêu cực đến môi trường. Phun thuốc trên diện tích lớn một cách nhanh chóng và đồng đều hơn so với các phương pháp truyền thống.

* **Trong giải trí và sáng tạo nội dung:**

+ **Quay phim và chụp ảnh:** UAV mang đến những góc máy độc đáo và ấn tượng, không thể thực hiện được bằng các thiết bị thông thường.

+ **Các sự kiện và thể thao:** UAV có thể cung cấp góc quay toàn cảnh và chi tiết cho các sự kiện thể thao, các chương trình biểu diễn, làm tăng tính hấp dẫn và chân thực của chương trình.

+ **Ứng dụng nghệ thuật:** Tạo ra các tác phẩm nghệ thuật từ trên không, điều khiển ánh sáng với các UAV.

* **Trong ứng cứu khẩn cấp:**

+ **Tìm kiếm cứu nạn:** UAV giúp xác định vị trí nạn nhân trong các khu vực hiểm trở hoặc khó tiếp cận như các khu vực sạt lở, lũ lụt, giúp đội cứu hộ tiếp cận và giải cứu nhanh chóng hơn.

+ **Đánh giá thiệt hại:** UAV có thể bay vào các khu vực bị ảnh hưởng để thu thập thông tin, hình ảnh về mức độ thiệt hại, từ đó giúp cơ quan chức năng lên kế hoạch ứng phó và phục hồi hiệu quả.

+ **Hỗ trợ liên lạc:** Trong các thảm họa, các hệ thống liên lạc truyền thống có thể bị hỏng, UAV có thể được sử dụng như các trạm tiếp sóng di động để duy trì liên lạc và hỗ trợ các hoạt động cứu hộ.

=> **Kết luận:** Các ứng dụng của UAV là đa dạng, có tiềm năng cách mạng hóa nhiều ngành nghề và lĩnh vực của đời sống xã hội.

1.2 Thách thức khi triển khai mạng UAV

1.2.1 Hạn chế về năng lượng

Luận điểm mở rộng: Không chỉ giới hạn thời gian bay, mà vấn đề năng lượng còn tác động đến khả năng hoạt động liên tục, phạm vi tác chiến và khả năng thực hiện các nhiệm vụ phức tạp của UAV.

Giải thích: Nếu năng lượng không được quản lý hiệu quả, UAV có thể phải giảm bớt nhiệm vụ, hạn chế khu vực hoạt động hoặc phải trở về trạm sạc sớm.

Các yếu tố ảnh hưởng (mở rộng):

- **Công nghệ pin hiện tại:** Công nghệ pin hiện tại chưa đáp ứng được nhu cầu về thời gian bay và khối lượng nhẹ.
- **Tốc độ và độ cao:** Bay ở tốc độ cao hoặc ở độ cao lớn sẽ tiêu thụ nhiều năng lượng hơn.
- **Các thiết bị phụ tải:** Cảm biến, camera, bộ truyền dữ liệu, cũng làm tiêu thụ năng lượng lớn.
- **Quản lý năng lượng:** Việc quản lý năng lượng không hiệu quả cũng làm giảm thời gian hoạt động của UAV.

1.2.2 Kết nối không ổn định và độ trễ cao

Luận điểm mở rộng: Kết nối không ổn định và độ trễ cao không chỉ ảnh hưởng đến việc truyền dữ liệu mà còn có thể gây ra các vấn đề trong việc điều khiển UAV và phối hợp hoạt động trong mạng lưới.

Giải thích: Khi kết nối bị gián đoạn, các lệnh điều khiển có thể không được gửi đến UAV kịp thời, và thông tin phản hồi từ UAV về trạm điều khiển có thể bị chậm trễ. Điều này gây nguy hiểm và làm giảm hiệu quả của hệ thống UAV.

Các yếu tố ảnh hưởng (mở rộng):

- **Loại sóng vô tuyến:** Các dải tần khác nhau (2.4GHz, 5GHz) có đặc tính truyền sóng khác nhau, ảnh hưởng đến phạm vi và độ ổn định của kết nối.
- **Môi trường nhiều vật cản:** Các khu vực đô thị với nhiều tòa nhà cao tầng, cây cối, có thể làm suy yếu tín hiệu vô tuyến.
- **Số lượng kết nối:** Nếu có quá nhiều UAV hoạt động trong cùng một khu vực, nhiều sóng có thể tăng lên và làm giảm chất lượng kết nối.
- **Tự động thay đổi đường truyền:** Cần có những cơ chế tự động chuyển đổi các kênh sóng, hoặc tìm đường đi tốt nhất để giảm độ trễ trong trường hợp tín hiệu yếu.

1.2.3 Quản lý không gian bay

Luận điểm mở rộng: Quản lý không gian bay không chỉ đơn giản là tránh va chạm, mà còn phải đảm bảo UAV không can thiệp vào các hoạt động khác, cũng như tuân thủ các quy định về an toàn hàng không.

Giải thích: Việc điều phối hoạt động của nhiều UAV trong cùng một không gian, đặc biệt trong các khu vực đông dân cư hoặc gần các sân bay, là một thách thức không nhỏ.

Các yếu tố ảnh hưởng (mở rộng):

- **Yếu tố con người:** Cần xây dựng các hệ thống đào tạo chuyên nghiệp cho người điều khiển và các quy trình quản lý rủi ro.
- **Tính tự động hóa:** Các UAV cần có khả năng tự động tránh né chướng ngại vật và các UAV khác, đồng thời phải luôn có phương án dự phòng khi gặp sự cố.
- **Quy định và luật lệ:** Các quy định về không phận, cấp phép bay, và trách nhiệm pháp lý cần được rõ ràng, đồng bộ và thực thi nghiêm túc.

1.2.4 An ninh mạng

Luận điểm mở rộng: An ninh mạng không chỉ liên quan đến bảo vệ dữ liệu mà còn đảm bảo an toàn cho các UAV, tránh các cuộc tấn công có thể gây nguy hiểm đến tính mạng con người.

Giải thích: Việc mất quyền điều khiển UAV hoặc bị lộ thông tin có thể dẫn đến các hậu quả nghiêm trọng, đặc biệt trong các hoạt động quân sự hoặc cứu hộ.

Các nguy cơ (mở rộng):

- **Tấn công vào giao thức định tuyến:** Kẻ tấn công có thể làm thay đổi đường đi của dữ liệu hoặc làm gián đoạn giao tiếp giữa các UAV.
- **Tấn công vào hệ thống điều khiển:** Kẻ tấn công có thể giành quyền kiểm soát UAV và thay đổi hành vi của chúng.
- **Phần mềm độc hại:** UAV có thể bị nhiễm phần mềm độc hại qua các kết nối không dây hoặc các thiết bị lưu trữ.
- **Ăn cắp thông tin:** Thông tin cá nhân và các dữ liệu nhạy cảm có thể bị đánh cắp qua các cuộc tấn công vào hệ thống UAV.

1.2.5 Vấn đề pháp lý

Luận điểm mở rộng: Vấn đề pháp lý không chỉ liên quan đến việc bay mà còn liên quan đến các quyền sở hữu dữ liệu, trách nhiệm pháp lý khi xảy ra sự cố, và các quy định về an toàn.

Giải thích: Việc sử dụng UAV, đặc biệt trong các hoạt động thương mại, yêu cầu phải tuân thủ các quy định pháp lý nghiêm ngặt để bảo vệ quyền lợi của cả doanh nghiệp và người dân.

Các yếu tố ảnh hưởng (mở rộng):

- **Quy định về bảo hiểm:** Việc yêu cầu các UAV phải có bảo hiểm có thể gây khó khăn và tốn kém cho các nhà khai thác.
- **Quy định về bảo vệ dữ liệu:** Cần có quy định rõ ràng về việc thu thập, lưu trữ và xử lý dữ liệu từ các cảm biến của UAV.
- **Trách nhiệm pháp lý:** Việc xác định trách nhiệm khi UAV gây ra tai nạn là một vấn đề phức tạp và cần được giải quyết.
- **Quy định về xuất nhập khẩu:** Việc xuất nhập khẩu và sử dụng các UAV có thể bị hạn chế bởi các quy định của các quốc gia.

1.2.6 Môi trường khắc nghiệt

Luận điểm mở rộng: Điều kiện môi trường khắc nghiệt không chỉ gây khó khăn cho việc bay mà còn ảnh hưởng đến tuổi thọ và độ tin cậy của các thiết bị điện tử và cơ khí trên UAV.

Giải thích: Các yếu tố như nhiệt độ cao hoặc thấp, độ ẩm, gió lớn và bụi bặm có thể làm suy giảm hiệu suất, gây hỏng hóc cho các thành phần của UAV.

Các yếu tố ảnh hưởng (mở rộng):

- **Nhiệt độ cực đoan:** Nhiệt độ quá cao hoặc quá thấp ảnh hưởng đến hiệu suất của pin và các cảm biến.
- **Độ ẩm cao:** Độ ẩm cao có thể gây chập mạch hoặc hỏng các linh kiện điện tử.
- **Gió lớn và bão:** Có thể làm UAV mất ổn định hoặc bị cuốn trôi ra khỏi khu vực hoạt động.
- **Bụi và cát:** Bụi và cát có thể làm giảm hiệu suất của các cảm biến, hoặc làm nghẽn các động cơ.
- **Mưa lớn:** Mưa lớn có thể gây nhiễu sóng hoặc làm hỏng các linh kiện điện tử.

1.3 Mục tiêu và phạm vi của đề tài

1.3.1 Mục tiêu tổng quát: Nghiên cứu và mô phỏng mạng UAV bằng NetSimulyzer.

- Đề tài không chỉ tập trung vào việc xây dựng một mô hình mô phỏng mạng UAV, mà còn hướng đến việc **hiểu rõ bản chất của các giao thức mạng MANET, đánh giá hiệu năng mạng**, và từ đó đưa ra những giải pháp khả thi nhằm tối ưu hóa hiệu suất trong các tình huống thực tế.

- Khai thác sức mạnh của **NetSimulyzer**: Khả năng trực quan hóa mạnh mẽ giúp minh họa các hoạt động của mạng UAV trong môi trường không gian 3D, cung cấp góc nhìn chi tiết và trực quan hơn trong quá trình nghiên cứu.
- Tận dụng ns-3: Sử dụng khả năng mô phỏng linh hoạt và tùy chỉnh của ns-3 để tạo ra một môi trường mô phỏng chính xác, hỗ trợ phân tích các thông số mạng UAV trong các kịch bản khác nhau.

Mục tiêu dài hạn của đề tài là đóng góp vào việc xây dựng các hệ thống mạng UAV thông minh, hiệu quả hơn, giải quyết các bài toán thực tiễn như truyền thông không dây ổn định, tối ưu hóa định tuyến và giảm thiểu tiêu thụ năng lượng.

1.3.2 Mục tiêu cụ thể:

+ Thiết kế và triển khai mô phỏng giao thức mạng MANET cho UAV:

- **Nghiên cứu sâu các giao thức MANET**: Tìm hiểu các giao thức định tuyến như AODV, DSR, OLSR, đánh giá ưu nhược điểm và lựa chọn giao thức phù hợp nhất cho mạng UAV.
- **Tùy chỉnh các tham số của giao thức**: Điều chỉnh các thông số như thời gian timeout, kích thước bảng định tuyến, và các giá trị liên quan để tối ưu hóa hoạt động trong môi trường di động cao như mạng UAV.
- **Mô phỏng các kịch bản thực tế**: Thiết kế các kịch bản mô phỏng đa dạng, bao gồm thay đổi số lượng UAV, tốc độ di chuyển, và môi trường truyền dẫn.

+ Sử dụng ns-3 kết hợp với NetSimulyzer để tạo môi trường mô phỏng:

- Tích hợp khả năng lập trình của ns-3 với NetSimulyzer để xây dựng các mô hình mô phỏng chi tiết, minh họa các hoạt động của mạng UAV bằng hình ảnh 3D trực quan.
- **Phân tích dữ liệu mô phỏng**: Sử dụng NetSimulyzer để thu thập và phân tích các chỉ số hiệu năng quan trọng như độ trễ, thông lượng, jitter và năng lượng tiêu thụ.

+ Đánh giá các yếu tố ảnh hưởng đến hiệu năng mạng UAV:

- **Phân tích các chỉ số chính**: Đánh giá độ trễ (latency), thông lượng (throughput), jitter, tỷ lệ mất gói (packet loss), và mức tiêu thụ năng lượng.
- **Nghiên cứu các yếu tố ảnh hưởng**: Bao gồm số lượng UAV, tốc độ di chuyển, phạm vi truyền dẫn, và kiểu định tuyến.
- **Đưa ra giải pháp cải thiện hiệu năng**: Dựa trên kết quả phân tích, đề xuất các giải pháp tối ưu hóa hoạt động mạng UAV.

1.3.3 Phạm vi nghiên cứu:

+ **Đối tượng nghiên cứu:**

- Mạng MANET ứng dụng trong UAV-based network.
- Tập trung vào tính di động, khả năng tự tổ chức và các yêu cầu truyền thông đặc thù của UAV.

+ **Phạm vi kỹ thuật:**

- Sử dụng ns-3 làm công cụ chính để mô phỏng các giao thức mạng MANET.
- Sử dụng NetSimulyzer để trực quan hóa các kịch bản mô phỏng, giúp dễ dàng nhận diện các vấn đề trong mạng UAV.
- Tập trung vào các yếu tố cốt lõi:
 - **Khả năng kết nối:** Độ ổn định và tính sẵn sàng của kết nối mạng.
 - **Hiệu năng định tuyến:** Đánh giá hiệu quả của các giao thức định tuyến trong môi trường di động cao.
 - **Độ trễ và thông lượng:** Phân tích khả năng truyền tải dữ liệu trong các kịch bản khác nhau.

+ **Kịch bản mô phỏng:**

- Thiết lập nhiều kịch bản mô phỏng với các tham số thay đổi:
 - Số lượng UAV (từ nhỏ đến lớn).
 - Tốc độ di chuyển (chậm, trung bình, nhanh).
 - Môi trường truyền dẫn (khoảng cách ngắn, dài).
- **Bỏ qua các yếu tố phức tạp hơn** như ảnh hưởng thời tiết, nhiễu điện từ, hoặc môi trường 3D quá chi tiết để đơn giản hóa quá trình mô phỏng và tập trung vào mục tiêu chính của đề tài.

1.4 Phương pháp nghiên cứu và triển khai

1.4.1 Nghiên cứu lý thuyết.

Luận điểm:

- Nghiên cứu lý thuyết là bước đầu tiên và quan trọng nhất để tạo nền tảng kiến thức vững chắc. Quá trình này không chỉ dừng lại ở việc thu thập thông tin mà còn giúp hiểu sâu về vấn đề, xác định các hướng tiếp cận phù hợp và xây dựng các giả thuyết khoa học.

Nội dung nghiên cứu:

1.4.1.1. Mạng MANET:

- Nghiên cứu đặc điểm của mạng MANET, như khả năng tự tổ chức, tính di động cao, và cách thức truyền thông đa chặng (multi-hop communication).
- Phân tích các giao thức định tuyến phổ biến trong MANET như AODV, DSR, OLSR, và DSDV để hiểu rõ cách chúng xử lý quá trình định tuyến trong môi trường di động.
- So sánh hiệu năng của các giao thức trong các trường hợp cụ thể như thay đổi số lượng nút, tốc độ di chuyển, và kiểu mô hình di động.

1.4.1.2. ns-3:

- Tìm hiểu kiến trúc của ns-3, các module hỗ trợ giao thức mạng MANET, cách cài đặt và lập trình các kịch bản mô phỏng.
- Hiểu cách sử dụng các công cụ tích hợp như FlowMonitor để thu thập dữ liệu và đánh giá hiệu năng mạng.

1.4.1.3. NetSimulyzer:

- Phân tích cấu trúc và cách thức hoạt động của NetSimulyzer, đặc biệt là khả năng tích hợp với ns-3.
- Học cách cấu hình các tham số trực quan hóa, thu thập và hiển thị dữ liệu dưới dạng đồ họa 3D.

1.4.1.4. Mạng UAV:

- Tìm hiểu về các đặc điểm và yêu cầu đặc thù của mạng UAV, như tính di động cao, tiêu thụ năng lượng, và khả năng hoạt động trong môi trường không gian ba chiều.
- Xem xét các nghiên cứu liên quan đến ứng dụng của mạng MANET trong UAV-based networks để hiểu các thách thức và giải pháp hiện có.

1.4.2 Thiết kế mô hình mô phỏng trên ns-3.

Luận điểm:

- Việc thiết kế mô hình mô phỏng là bước quan trọng để đảm bảo rằng các kịch bản mô phỏng phản ánh đúng các tình huống thực tế và đáp ứng mục tiêu nghiên cứu.

Quy trình thiết kế:

1.4.2.1. Xác định các tham số mô phỏng:

- Số lượng UAV, tốc độ di chuyển, khoảng cách giữa các node, và phạm vi truyền dẫn.
- Thời gian mô phỏng, loại giao thức định tuyến, và loại môi trường truyền dẫn (phẳng hay không gian 3D).

1.4.2.2. Xây dựng mô hình di chuyển:

- Sử dụng các mô hình di chuyển phổ biến như Random Waypoint, Random Walk, hoặc Gauss-Markov để mô phỏng hành vi di chuyển thực tế của UAV.
- Cân nhắc thêm yếu tố khoảng cách tối thiểu giữa các UAV để tránh xung đột.

1.4.2.3. Lựa chọn giao thức định tuyến:

- Tích hợp giao thức AODV vào mô phỏng và điều chỉnh các tham số để phù hợp với môi trường di động cao của UAV.

1.4.2.4. Thiết lập kênh truyền dẫn:

- Chọn môi trường truyền dẫn phù hợp như WirelessChannel hoặc WifiChannel.
- Cấu hình các thông số liên quan đến công suất phát, độ nhạy thu, và tốc độ truyền dữ liệu.

1.4.2.5. Mô hình hóa kiến trúc mạng:

- Xây dựng cấu trúc mạng gồm các UAV, trạm điều khiển, và các nút hỗ trợ khác.
- Đảm bảo các UAV có khả năng kết nối và truyền thông qua giao thức định tuyến MANET.

1.4.3 Triển khai mô hình mô phỏng và thu thập dữ liệu bằng NetSimulyzer.

Luận điểm:

- Giai đoạn triển khai mô phỏng là bước thực tiễn hóa các thiết kế, cho phép kiểm tra tính chính xác của các giả thuyết và thu thập dữ liệu phục vụ phân tích.

Quy trình triển khai:

1.4.3.1. Viết code mô phỏng:

- Sử dụng ngôn ngữ C++ trong ns-3 để lập trình kịch bản mô phỏng, bao gồm khởi tạo các UAV, thiết lập kết nối mạng, và cài đặt giao thức định tuyến.

1.4.3.2. Cấu hình NetSimulyzer:

- Thiết lập các thông số hiển thị trong NetSimulyzer, như vị trí, tốc độ di chuyển của UAV và các thông số mạng khác.
- Tùy chỉnh các chế độ trực quan hóa để có góc nhìn rõ ràng về hoạt động mạng.

1.4.3.3. Chạy mô phỏng:

- Chạy kịch bản trên ns-3 để thu thập dữ liệu đầu ra dưới dạng file JSON hoặc các định dạng tương thích với NetSimulyzer.

1.4.3.4. Ghi nhận dữ liệu:

- Sử dụng FlowMonitor và NetSimulyzer để thu thập các chỉ số như độ trễ, thông lượng, jitter, và tỷ lệ mất gói.
- Đảm bảo dữ liệu được ghi nhận đầy đủ và không có lỗi trong quá trình chạy mô phỏng.

1.4.3.5. Trực quan hóa dữ liệu:

- Sử dụng NetSimulyzer để tạo các biểu đồ 3D minh họa hoạt động của mạng UAV, giúp phân tích dễ dàng hơn.

1.4.4 Phân tích và đánh giá kết quả.

Luận điểm:

- Việc phân tích và đánh giá kết quả giúp xác định hiệu quả của mạng UAV trong các kịch bản mô phỏng và cung cấp cơ sở cho các đề xuất cải tiến.

Quy trình phân tích:

1.4.4.1. Xử lý dữ liệu:

- Xử lý các dữ liệu thu thập được từ FlowMonitor và NetSimulyzer, loại bỏ các giá trị bất thường và chuẩn hóa số liệu để dễ dàng phân tích.

1.4.4.2. Biểu diễn dữ liệu:

- Sử dụng đồ thị, biểu đồ, và bảng để trình bày các kết quả, tập trung vào các chỉ số chính như độ trễ, thông lượng, jitter, và tỷ lệ mất gói.

1.4.4.3. So sánh các kịch bản:

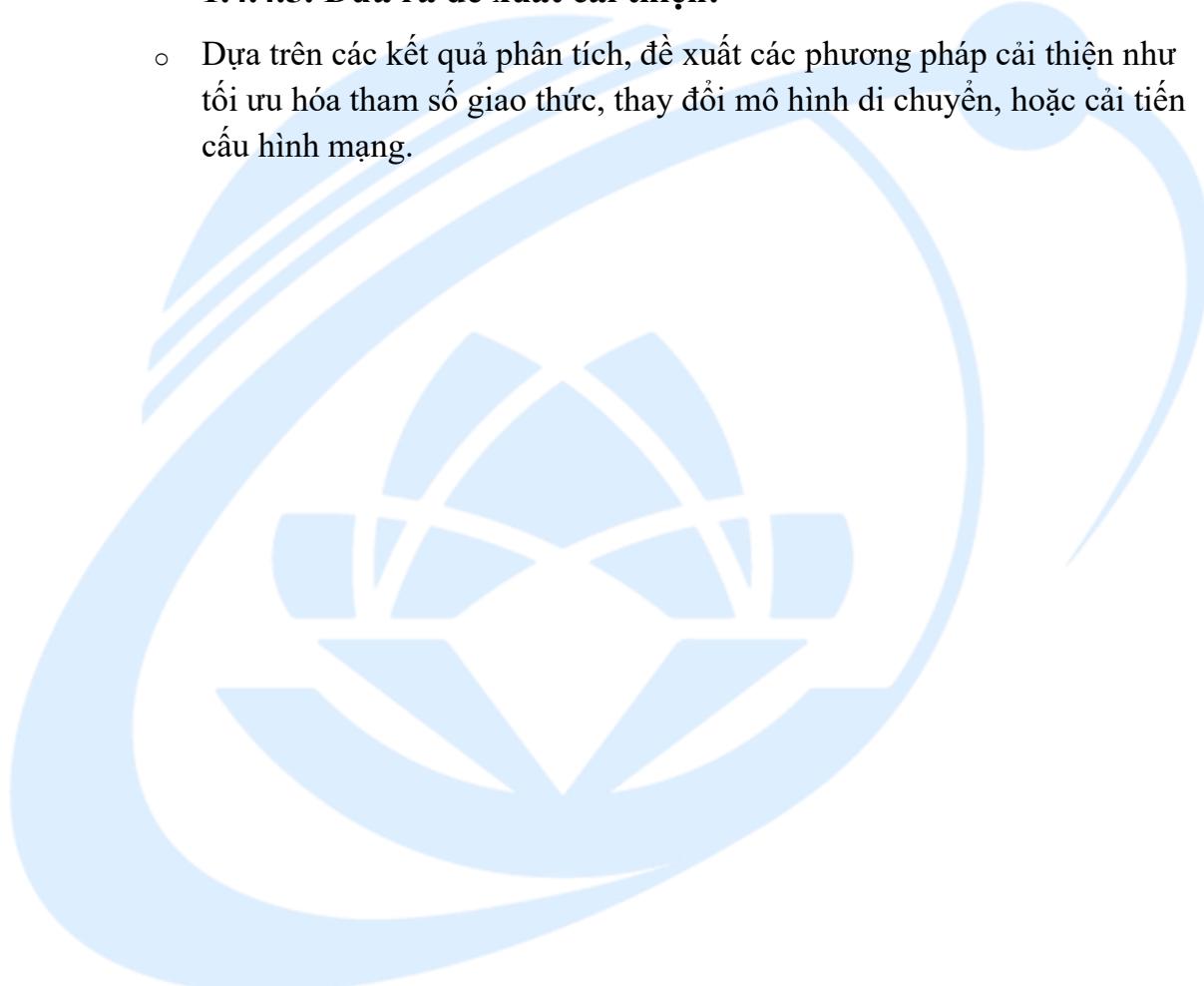
- So sánh kết quả trong các kịch bản khác nhau (số lượng UAV, tốc độ di chuyển, phạm vi truyền dẫn) để xác định các yếu tố ảnh hưởng lớn nhất đến hiệu năng mạng.

1.4.4.4. Đánh giá hiệu năng:

- Đánh giá hiệu năng của giao thức AODV trong môi trường mạng UAV, chỉ ra các điểm mạnh và hạn chế.

1.4.4.5. Đưa ra đề xuất cải thiện:

- Dựa trên các kết quả phân tích, đề xuất các phương pháp cải thiện như tối ưu hóa tham số giao thức, thay đổi mô hình di chuyển, hoặc cải tiến cấu hình mạng.



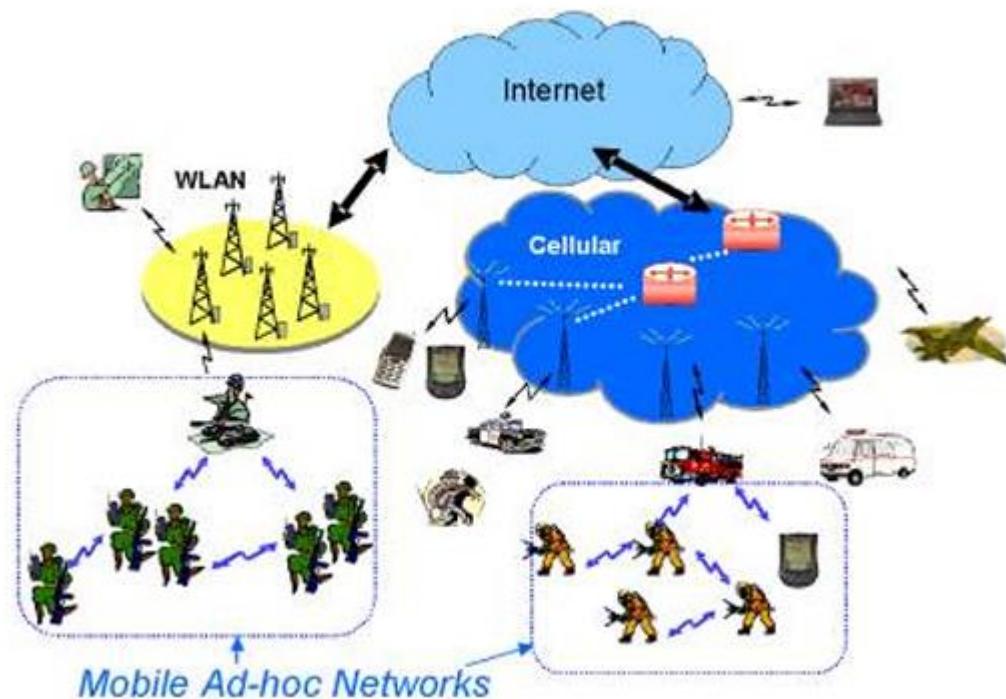
CHƯƠNG II. CƠ SỞ LÝ THUYẾT

2.1 Mạng MANET

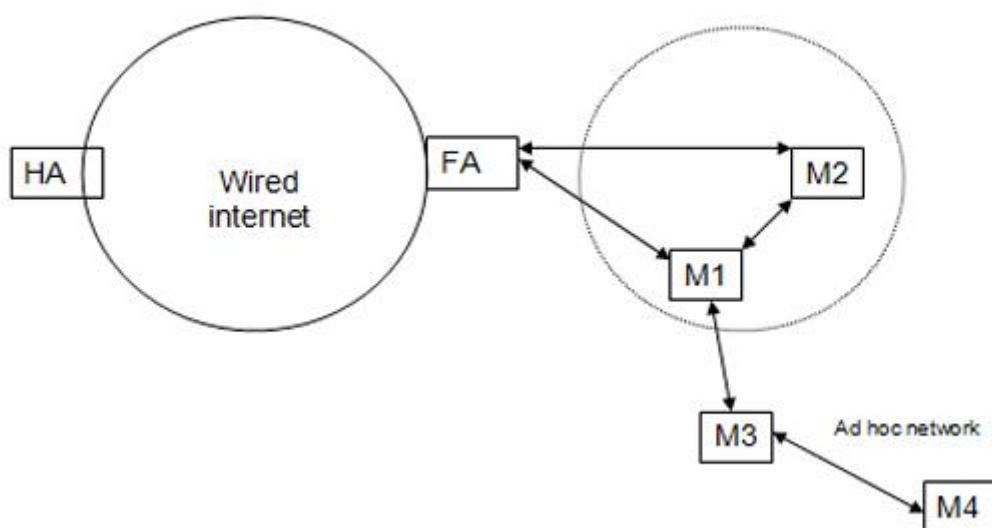
2.1.1 Giới thiệu chung về mạng MANET

Với hàng loạt các ưu điểm của công nghệ truyền thông không dây, các mạng di động không dây đã được phát triển rất mạnh trong thời gian gần đây. Mạng di động không dây có thể chia thành hai kiểu mạng: mạng hạ tầng và mạng không hạ tầng. Trong mạng hạ tầng, truyền thông giữa các phần tử mạng phụ thuộc vào sự hỗ trợ của hạ tầng mạng, các thiết bị đầu cuối di động truyền thông đơn bước không dây qua các điểm truy nhập (các trạm cơ sở) để tới hạ tầng mạng cố định. Kiểu mạng không phụ thuộc hạ tầng còn được gọi chung là các mạng tùy biến di động MANET (mobile ad hoc network) là một tập hợp của những node mạng không dây, những node này có thể được thiết lập tại bất kỳ thời điểm và tại bất cứ nơi nào. Mạng MANET không dùng bất kỳ cơ sở hạ tầng nào. Nó là một hệ thống tự trị mà máy chủ di động được kết nối bằng đường vô tuyến và có thể di chuyển tự do, thường hoạt động như một router.

Mạng Adhoc di động (MANET) bao gồm các miền router kết nối lỏng với nhau. Một mạng MANET được đặc trưng bởi một hoặc nhiều giao diện mạng MANET, các giao diện được phân biệt bởi “khả năng tiếp cận không đối xứng” thay đổi theo thời gian của nó đối với các router lân cận. Các router này nhận dạng và duy trì một cấu trúc định tuyến giữa chúng. Các router có thể giao tiếp thông qua các kênh vô tuyến động với khả năng tiếp cận không đối xứng, có thể di động và có thể tham gia hoặc rời khỏi mạng bất kì thời điểm nào. Để giao tiếp với nhau, các nút mạng adhoc cần cấu hình giao diện mạng của nó với địa chỉ địa phương có giá trị trong khu vực của mạng adhoc đó. Các nút mạng adhoc có thể phải cấu hình các địa chỉ toàn cầu có thể được định tuyến, để giao tiếp với các thiết bị khác trên mạng Internet. Nhìn từ góc độ lớp IP, mạng MANET có vai trò như một mạng multi-hop lớp 3 được tạo thành bởi các liên kết. Do vậy mỗi nút mạng adhoc trong mạng MANET sẽ hoạt động như một router lớp 3 để cung cấp kết nối với các nút khác trong mạng. Mỗi nút adhoc duy trì các tuyến tới các nút khác trong mạng MANET và các tuyến mạng tới các nút đích ở ngoài mạng MANET đó. Nếu đã được kết nối với mạng Internet, các mạng MANET sẽ trở thành mạng rìa (edge network), nghĩa là biên giới của chúng được xác định bởi các router rìa (edge-router). Do bản chất của các liên kết tạo nên mạng MANET, các nút adhoc trong mạng không chia sẻ truy nhập cho liên Downloaded by Phúc Nguy?n (lylishandy@gmail.com) IOMoARcPSD|48180357 kết đơn báo hiệu đa điểm (multicast). Như vậy, trong mạng MANET không dự trữ hay dành riêng liên kết đa điểm multicast và liên kết quảng bá broadcast.



Hình 1: Minh họa mạng MANET



Hình 2: Biểu đồ mạng MANET

2.1.2 Đặc điểm của mạng MANET

- Thiết bị tự trị đầu cuối (Autonomous terminal): Trong Manet, mỗi thiết bị di động đầu cuối là một node tự trị. Nó có thể mang chức năng của host và router. Bên cạnh khả năng xử lý cơ bản của một host, các node di động này có thể chuyển đổi

chức năng như một router. Vì vậy, thiết bị đầu cuối và chuyển mạch là không thể phân biệt được trong mạng Manet

- Phân chia hoạt động (Distributed operation): Vì không có hệ thống mạng nền tảng cho trung tâm kiểm soát hoạt động của mạng nên việc kiểm soát và quản lý hoạt động của mạng được chia cho các thiết bị đầu cuối. Các node trong MANET đòi hỏi phải có sự phối hợp với nhau. Khi cần thiết các node hoạt động như một relay để thực hiện chức năng của mình như bảo mật và định tuyến.- Định tuyến đa đường: Thuật toán định tuyến không dây cơ bản có thể định tuyến một chặng và nhiều chặng dựa vào các thuộc tính liên kết khác nhau và giao thức định tuyến. Singalhop Manet đơn giản hơn multihop ở vấn đề cấu trúc và thực hiện với chi phí thấp và ít ứng dụng. Khi truyền các gói dữ liệu từ một nguồn của nó đến điểm trong phạm vi truyền tải trực tiếp không dây, các gói dữ liệu sẽ được chuyển tiếp qua một hoặc nhiều trung gian các nút.

- Cấu hình động (dynamic network topology): Vì các node là di động, nên cấu trúc mạng có thể thay đổi nhanh và không thể biết trước, các kết nối giữa các thiết bị đầu cuối có thể thay đổi theo thời gian. MANET sẽ thích ứng tự động và điều kiện lan truyền giống như mẫu di động và các node mạng di động. Các node di động trong mạng thiết lập định tuyến động với nhau khi chúng di chuyển, hình thành mạng riêng của chúng trong không trung. Hơn nữa, một User trong Manet có thể không chỉ hoạt động trong mạng lưới di động đặc biệt, mà còn có thể yêu cầu truy cập vào một mạng cố định công cộng như Internet.

- Dao động về dung lượng liên kết (Fluctuating link capacity): Bản chất tỉ lệ bit lỗi cao của kết nối không dây cần quan tâm trong mạng MANET. Từ đầu cuối này đến đầu cuối kia có thể được chia sẻ qua một vài chặng. Kênh giao tiếp ở đầu cuối chịu ảnh hưởng của nhiều, hiệu ứng đa đường, sự giao thoa và băng thông của nó ít hơn so với mạng có dây. Trong một vài tình huống, truy cập của hai người dùng có thể qua nhiều liên kết không dây và các liên kết này có thể không đồng nhất.

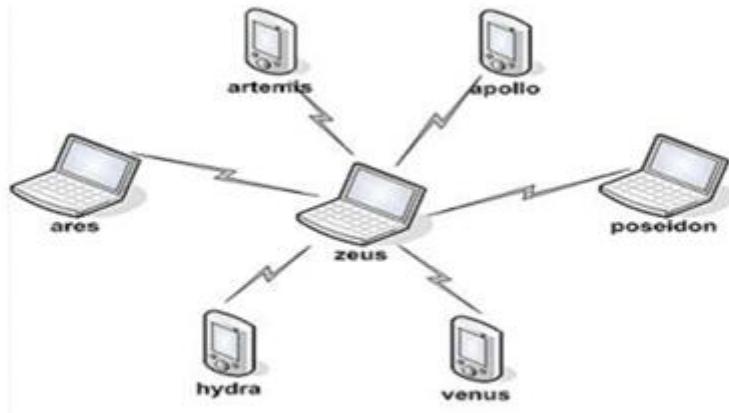
- Tối ưu hóa cho thiết bị đầu cuối (light-weight terminals): Trong hầu hết các trường hợp các node trong mạng MANET là thiết bị với tốc độ xử lý của CPU thấp, bộ nhớ ít và lưu trữ điện năng ít. Vì vậy cần phải tối ưu hóa các thuật toán và cơ chế.

2.1.3 Kiểu kết nối và cơ chế hoạt động

2.1.3.1. Các kiểu kết nối topo mạng

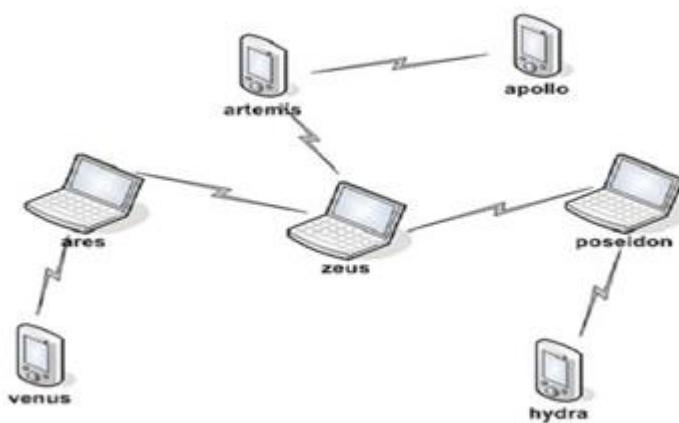
a) Mạng máy chủ di động

- Ở topo này các thiết bị chỉ liên kết với một máy chủ duy nhất. Các thiết bị khác liên kết qua máy chủ đó như hình vẽ:



Hình 3: Mạng máy chủ di động

- b) Mạng có các thiết bị di động không đồng nhất
 - Ở topo này các máy có thể liên kết trực tiếp với nhau trong phạm vi phủ sóng của mình:

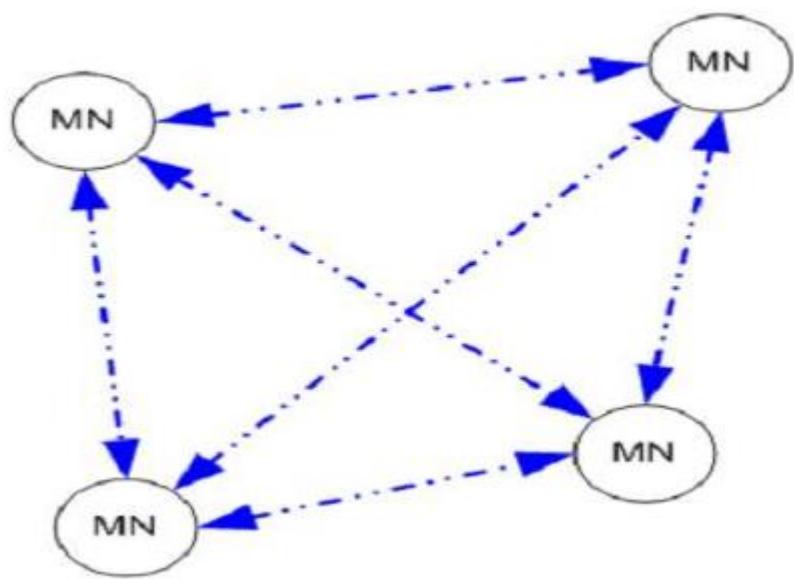


Hình 4: Hình minh họa mạng có các thiết bị di động không đồng nhất

2.1.3.2. Chế độ hoạt động

- a) Chế độ IEEE-ad hoc

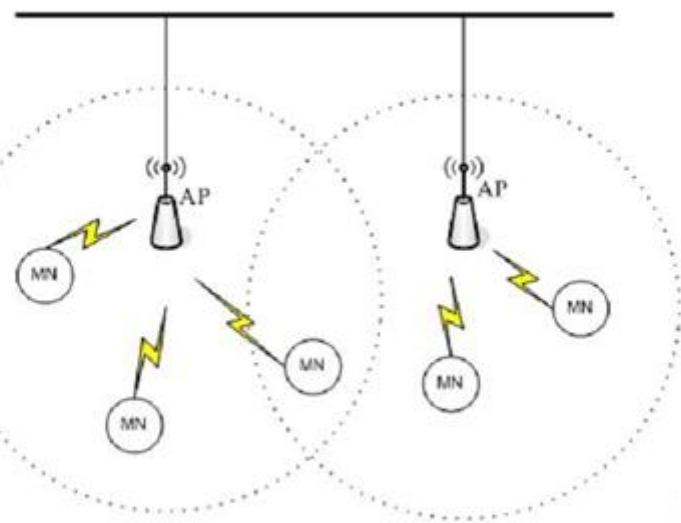
- Chế độ này thì các node di động truyền thông trực tiếp với nhau mà không cần tới một cơ sở hạ tầng nào cả. Trong chế độ này thì các liên kết không thể thực hiện qua nhiều chặng



Hình 5: Chế độ IEEE-ad hoc

b) Chế độ cơ sở hạ tầng

- Chế độ này thì mạng bao gồm các điểm truy cập AP cố định và các node di động tham gia vào mạng, thực hiện truyền thông qua các điểm truy cập. Trong chế độ này thì các liên kết có thể thực hiện qua nhiều chặng



Hình 6: Chế độ cơ sở hạ tầng

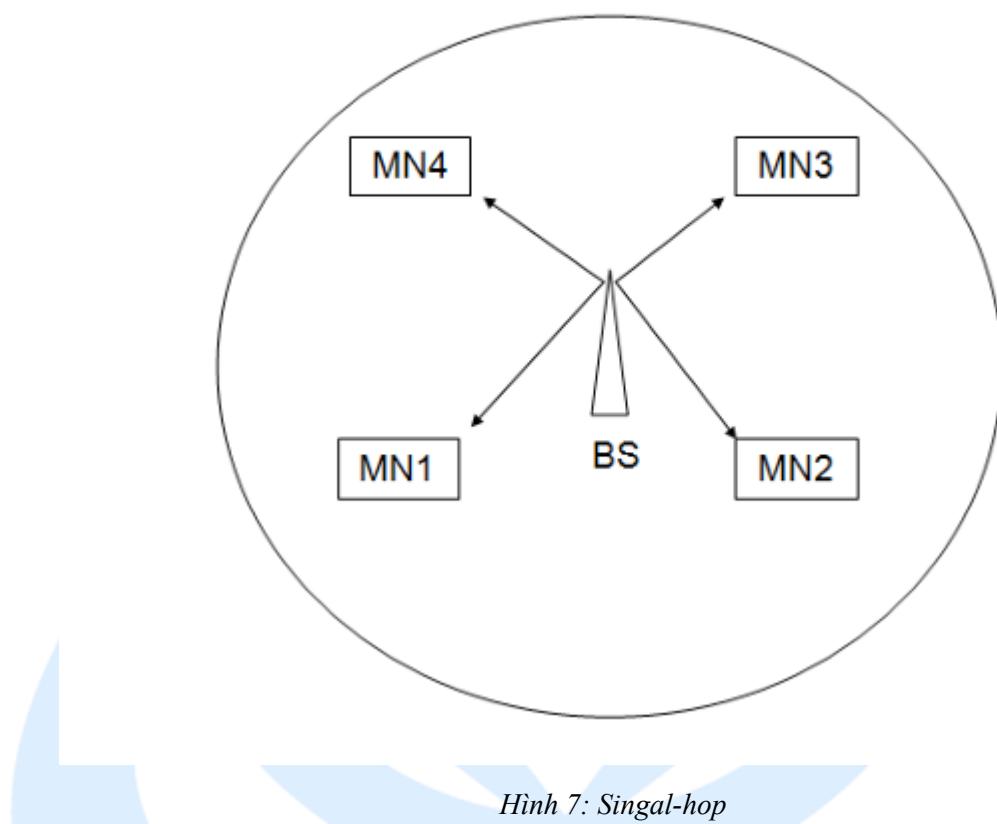
2.1.3.3. Phân loại MANET

*Theo giao thức:

- Singal-hop:

+ Mạng Manet định tuyến singal-hop là loại mô hình mạng ad-hoc đơn giản nhất. Trong đó, tất cả các node đều nằm trong cùng một vùng phủ sóng, nghĩa là các node có thể kết nối trực tiếp với nhau mà không cần các node trung gian.

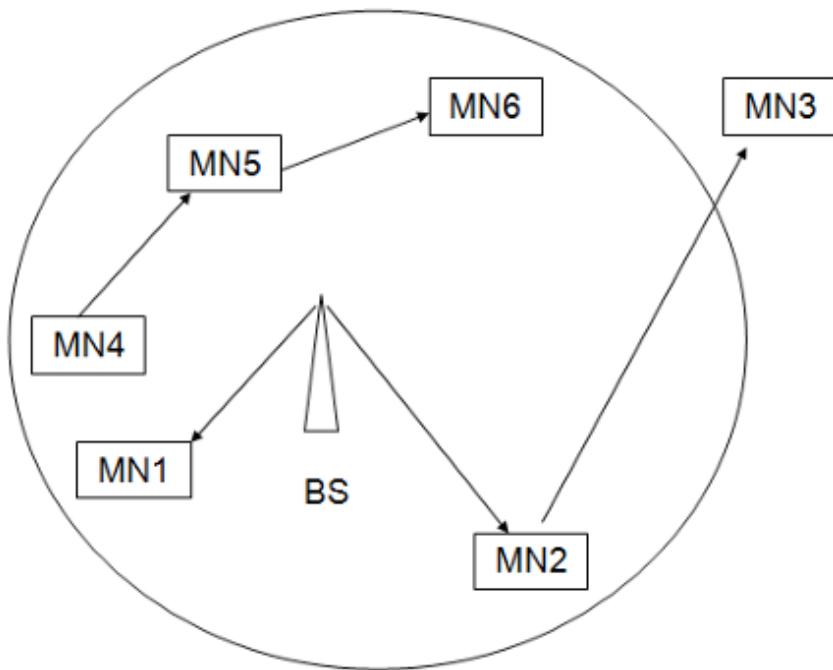
+ Mô hình này các node có thể di chuyển tự do nhưng chỉ trong một phạm vi nhất định đủ để các node liên kết trực tiếp với các node khác trong mạng.



Hình 7: Singal-hop

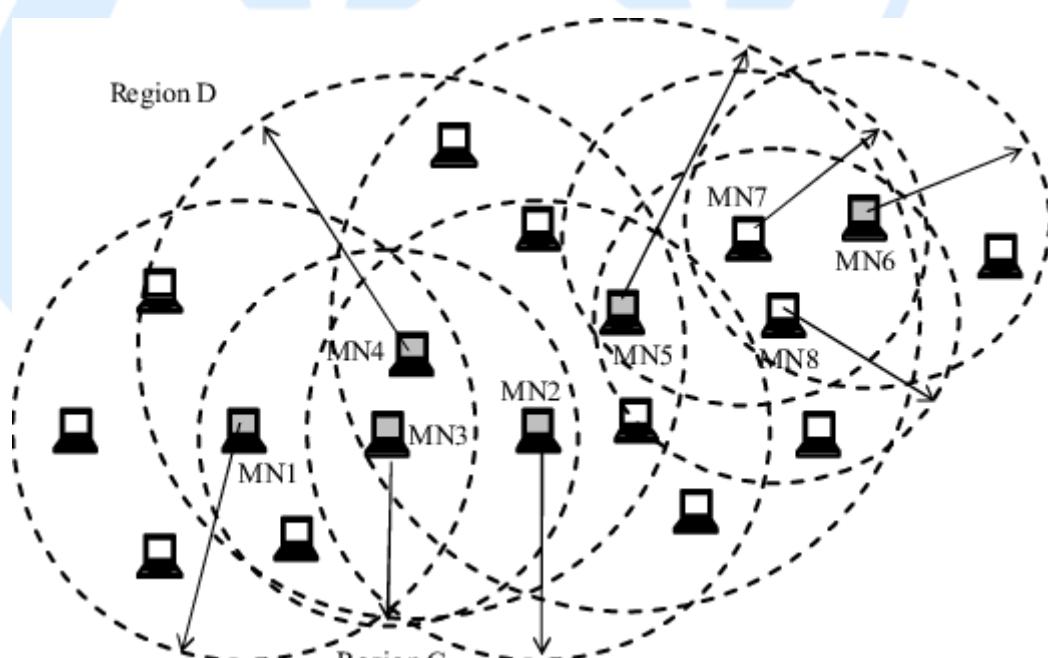
- Multi-hop:

+ Đây là mô hình phổ biến nhất trong mạng MANET, nó khác với mô hình trước là các node có thể kết nối với các node khác trong mạng mà có thể không cần kết nối trực tiếp với nhau. Các node có thể định tuyến với các node khác thông qua các node trung gian trong mạng. Để mô hình này hoạt động một cách hoàn hảo thì cần phải có giao thức định tuyến phù hợp với mô hình mạng MANET.



Hình 8: Multi-hop

- Mobile multi-hop:
 - + Mô hình này cũng tương tự với mô hình thứ hai nhưng sự khác biệt ở đây là mô hình này tập trung vào các ứng dụng có tính chất thời gian thực như audio, video...



Hình 9: Mobile multi-hop

*Theo chức năng:

- Mạng MANET đẳng cấp (Flat)

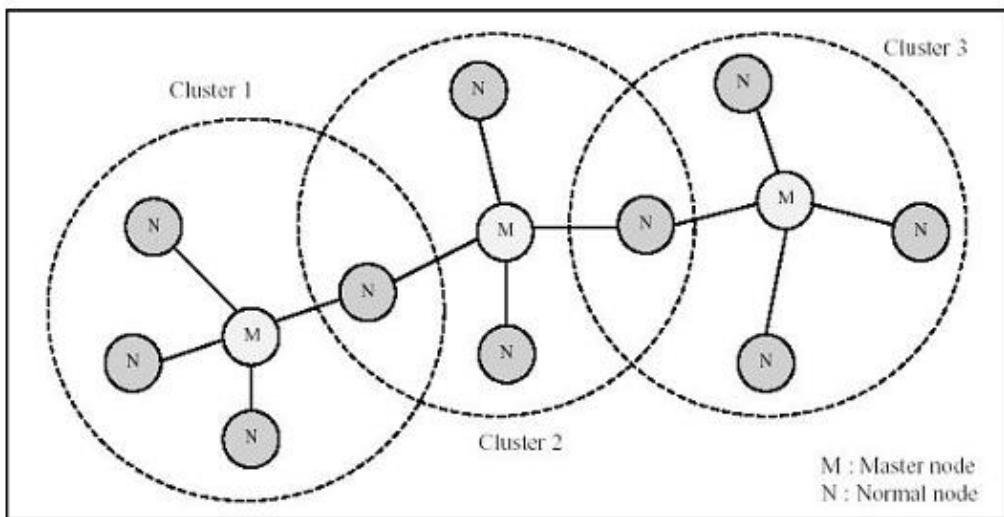
+ Trong kiến trúc này tất cả các node có vai trò ngang hàng với nhau (peer-to-peer) và các node đóng vai trò như các router định tuyến dữ liệu gói trên mạng. Trong những mạng lớn thì cấu trúc Flat không tối ưu hóa việc sử dụng tài nguyên băng thông của mạng vì những thông tin điều khiển phải truyền trên toàn bộ mạng. Tuy nhiên nó thích hợp trong những topo có các node di chuyển nhiều.

- Mạng Manet phân cấp (Hierarchical)

+ Đây là mô hình sử dụng phổ biến nhất. Trong mô hình này thì mạng chia thành các domain, trong mỗi domain bao gồm một hoặc nhiều cluster, mỗi cluster chia thành nhiều node. Có hai loại node là master node và normal node.

- Master node: là node quản trị một router có nhiệm vụ chuyển dữ liệu của các node trong cluster đến các node trong cluster khác và ngược lại. Nói cách khác nó có nhiệm vụ như một gateway.

- Normal node: là các node nằm trong cùng một cluster. Nó có thể kết nối với các node trong cluster hoặc kết nối với các cluster khác thông qua master node.



Hình 10 : Mô hình mạng phân cấp

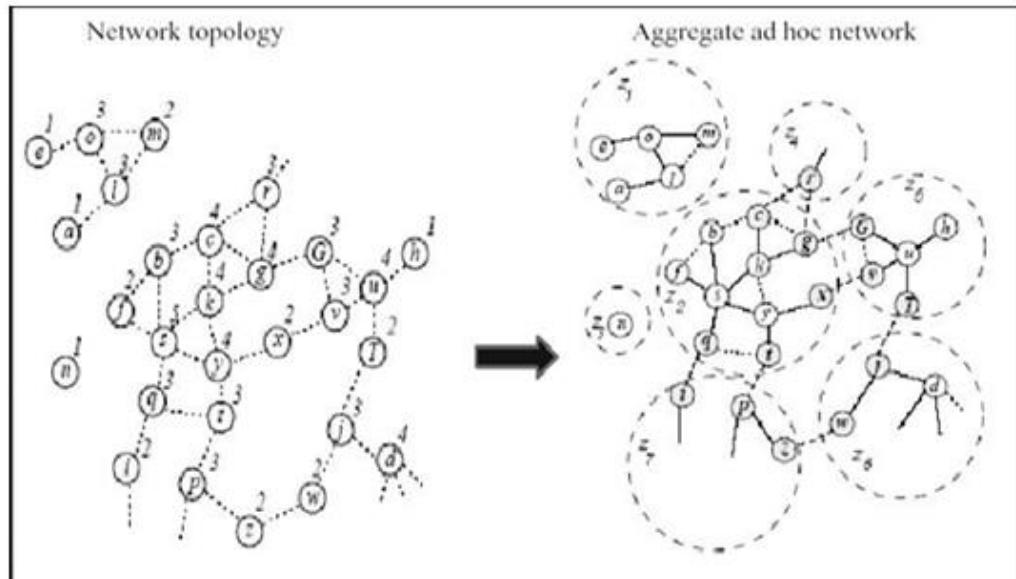
+ Với các cơ chế trên mạng sử dụng tài nguyên băng thông hiệu quả hơn vì các tin nhắn chỉ phải truyền trong 1 cluster. Tuy nhiên việc quản lý tính chuyển động của các node trở nên phức tạp hơn. Kiến trúc mạng phân cấp thích hợp cho các mạng có tính chuyển động thấp.

- Mạng MANET kết hợp (Aggregate)

+ Mạng = Zones, Zone = nodes

+ Mỗi node bao gồm hai mức topo : Topo mức thấp (node level), và topo mức cao (zone level)

+ Mỗi node đặc trưng bởi: node ID và zone ID. Trong một Zone có thể áp dụng kiến trúc đẳng cấp hoặc kiến trúc phân cấp



Hình 11: Mô hình mạng Aggregate

2.1.4 Định tuyến trong MANET

2.1.4.1. Những loại định tuyến

Trong mạng thông tin vô tuyến nói chung và mạng Ad hoc nói riêng do mỗi nút mạng đều có khả năng di chuyển nên topo mạng cũng thay đổi theo thời gian. Đặc điểm này gây ra khó khăn trong việc truyền tải gói tin. Riêng mạng Ad hoc gói tin muốn đến được đích thì phải truyền qua nhiều trạm và nút mạng do đó để gói tin đến được đích thì nút mạng phải sử dụng phương pháp định tuyến . Giao thức định tuyến có hai chức năng: Tìm, chọn đường đi tốt nhất và chuyển gói tin đến đúng đích. Ta sẽ đề cập sâu hơn về việc tìm, chọn đường của các nút.

a) Định tuyến Bellman-Ford

Trong thuật toán Bellman-Ford, mỗi nút duy trì một bảng định tuyến hay ma trận chứa thông tin khoảng cách và thông tin về nút kế tiếp của mình trên đường đi ngắn nhất tới đích bất kỳ, trong đó khoảng cách chính là chiều dài ngắn nhất từ nút tới đích. - Để cập nhật thông tin về đường đi ngắn nhất mỗi nút sẽ thường xuyên trao đổi bảng định tuyến với các nút bên cạnh nó. Dựa trên bảng định tuyến từ các nút lân cận đó, nút nào đó biết được khoảng cách ngắn nhất từ các lân cận của nó tới nút đích bất kỳ. Do đó, với mỗi nút đích, nút xuất phát sẽ chọn một nút trung gian cho chặng kế tiếp sao cho khoảng cách từ nó qua nút trung gian tới nút đích là nhỏ nhất.

Các thông tin tính toán mới này sẽ được lưu trữ vào bảng định tuyến của nút này và được trao đổi ở vòng cập nhật định tuyến tiếp theo. - Định tuyến này có ưu điểm là đơn giản và tính toán hiệu quả do đặc điểm phân bố. Tuy nhiên nhược điểm của nó là hội tụ chậm khi topo mạng thay đổi và có xu hướng tạo các vòng lặp định tuyến đặc biệt là khi các điều kiện liên kết không ổn định.

b) Định tuyến tìm đường

Các giao thức mới như DSDV (Destination Sequenced Distance Vector) và WRP (Wireless Routing Protocol) dựa trên DBF để cung cấp định tuyến lặp tự do. Cho dù là vấn đề đã được giải quyết thì vẫn còn tồn tại vấn đề về độ thiếu chính xác trong định tuyến DBF, vấn đề này có thể gây ra suy giảm hiệu suất mạng. Nguyên nhân dẫn đến sự thiếu chính xác là do nút mạng không có được các thông tin trạng thái toàn mạng dẫn đến các quyết định đưa ra chỉ tối ưu trong phạm vi cục bộ, nó không đảm bảo một giải pháp tối ưu trong môi trường di động.Thêm vào đó khi DBF chỉ duy trì một đường đi duy nhất tới đích, nó thiếu khả năng thích nghi với các lỗi liên kết và yêu cầu nghiên cứu mở rộng cho các hỗ trợ multicasting.

c) Định tuyến on-demand

Định tuyến On-demand được biết đến như DC (Diffusion Computation) cũng được sử dụng trong mạng không dây. Trong lược đồ định tuyến On-demand, một nút xây dựng đường đi bằng cách chất vấn tất cả các nút trong mạng. Gói chất vấn tìm được ID của các nút trung gian và lưu giữ ở phần Path. Khi dò tìm các chất vấn, nút đích hay các nút đã biết đường đi tới đích trả lại chất vấn bằng cách phúc đáp “source routed” cho nơi gửi. Do nhiều phúc đáp nên có nhiều đường đi được tính toán và duy trì. Sau khi tính toán đường đi nút liên kết bắt kỳ bắt đầu các chất vấn , phúc đáp khác nên luôn cập nhật định tuyến. Mặc dù các tiếp cận dựa trên cơ sở DC có độ chính xác cao hơn và phản ứng nhanh hơn với sự thay đổi mạng nhưng phụ trợ điều khiển quá mức do thường xuyên yêu cầu flooding đặc biệt khi tính di động cao hơn và lưu lượng dày đặc phân bố đều nhau. Kết quả là các giao thức định tuyến On-demand chỉ phù hợp với mạng không dây băng thông rộng trễ truyền gói nhỏ và lưu lượng rất nhỏ.

d) Định tuyến vùng

Định tuyến vùng là một giao thức định tuyến khác thiết kế trong môi trường Ad hoc. Đây là giao thức lai giữa định tuyến On-demand với một giao thức bất kỳ đã tồn tại. Trong định tuyến vùng mỗi nút xác định vùng riêng khi nút ở khoảng cách nhất định. Định tuyến vùng trung gian sẽ dùng định tuyến On-demand để tìm đường đi. Ưu điểm của định tuyến vùng là khả năng mở

rộng cấp độ khi nhu cầu lưu trữ cho bảng định tuyến giảm xuống. Tuy nhiên do gần giống với định tuyến On-demand nên định tuyến vùng cũng gặp phải vấn đề về trễ kết nối và điểm kết thúc của các gói yêu cầu.

2.1.4.2. Các giao thức định tuyến

***Phân loại giao thức định tuyến:**

- Định tuyến theo bảng (proactive)

Trong các giao thức định tuyến theo bảng, tất cả các node cần duy trì thông tin về cấu hình mạng. Khi cấu hình mạng thay đổi, các cập nhật được truyền lan trong mạng nhằm thông báo sự thay đổi. Hầu hết các giao thức định tuyến theo bảng đều kể thừa và sửa đổi đặc tính tương thích từ các thuật toán chọn đường dẫn ngắn nhất trong các mạng hữu tuyến truyền thống. Các thuật toán định tuyến theo bảng được sử dụng cho các node cập nhật trạng thái mạng và duy trì tuyến bất kỳ có lưu lượng hay không. Vì vậy, tiêu đề thông tin để duy trì cấu hình mạng đối với các giao thức này thường là lớn. Một số giao thức định tuyến điển hình theo bảng trong MANET gồm:

- + Giao thức định tuyến không dây WRP (Wireless Routing Protocol)
- + Định tuyến vector khoảng cách tuần tự đích DSDV (Destination Sequence Distance Vector)
- + Định tuyến trạng thái tối ưu liên kết OLSR (Optimized Link State Routing)
- Định tuyến theo yêu cầu (reactive)

Trong mạng MANET, các tuyến hoạt động có thể ngừng do tính di động của node. Vì vậy, thông tin duy trì tuyến là tối quan trọng đối với các giao thức định tuyến theo yêu cầu. So với các giao thức định tuyến theo bảng, các giao thức định tuyến theo yêu cầu thường có tiêu đề trao đổi thông tin định tuyến nhỏ hơn. Vì vậy, về mặt nguyên tắc, các giao thức này có khả năng mở rộng tốt hơn so với các giao thức định tuyến theo bảng. Tuy nhiên, vấn đề lớn nhất của các giao thức định tuyến theo yêu cầu là trễ do tìm kiếm tuyến trước khi chuyển tiếp thông tin dữ liệu. Ví dụ về một số giao thức định tuyến theo yêu cầu gồm:

- + Giao thức định tuyến nguồn động DSR (Dynamic Source Routing)
- + Giao thức định tuyến vector khoảng cách theo yêu cầu AODV (Ad hoc On-demand Distance Vector routing)
- + Giao thức định tuyến theo thứ tự tạm thời TORA (Temporally Ordered Routing Algorithm).

- Giao thức định tuyến lai ghép

- + Các giao thức định tuyến lai ghép được đề xuất để kết hợp các đặc tính ưu điểm của các giao thức định tuyến theo bảng và theo yêu cầu. Thông thường, các giao

thức định tuyến lai ghép Manet được sử dụng trong kiến trúc phân cấp. Các giao thức định tuyến theo bảng và theo yêu cầu được triển khai trong các cấp thích hợp

+ Một số ví dụ về giao thức định tuyến lai ghép:

- Giao thức định tuyến vùng ZRP (Zone Routing Protocol)
- Giao thức định tuyến trạng thái liên kết dựa trên vùng ZHLS (Zone-based Hierarchical Link State routing)
- Giao thức định tuyến mạng tuỳ biến lai HARP (Hybrid Ad hoc Routing Protocol)

Ngoài ra, chúng cũng được phân loại theo cách khác:

- *Link state protocol* : Trong các giao thức loại này, các router sẽ trao đổi LSA (Link state advertisement) với các router khác để xây dựng và duy trì cơ sở dữ liệu về trạng thái của toàn mạng (Network topology database). Các thông tin này được trao đổi dưới dạng multicast (Một router đến nhiều router khác). Như vậy mỗi router sẽ có một cái nhìn đầy đủ và độc lập về toàn mạng (Routing table chung) và từ đó sẽ tìm cách xây dựng đường đi ngắn nhất đến đích

- *Distance vector protocol* : Trong giao thức loại này, các router sẽ chỉ trao đổi bảng định tuyến (Routing table) riêng của mình đến các router lân cận được kết nối trực tiếp với mình. Như vậy, các router này không tự biết được đường đi đến đích, không biết các router trung gian mà phải dựa vào bảng định tuyến của router lân cận (Bị chi phối bởi các router lân cận)

*Các giao thức định tuyến cơ bản:

a) Giao thức DSDV(Destination Sequence Distance Vector)

- Mô tả:

+ DSDV là giao thức định tuyến vector khoảng cách theo kiểu từng bước: Trong mỗi nút mạng duy trì bảng định tuyến lưu trữ đích có thể đến ở bước tiếp theo của định tuyến và số bước để đến đích. DSDV yêu cầu nút mạng phải gửi đều đặn thông tin định tuyến quảng bá trên mạng

+ Ưu điểm của DSDV là đảm bảo không có đường định tuyến kín bằng cách sử dụng số thứ tự để đánh dấu mỗi đường. Số thứ tự cho biết mức độ “mới” của đường định tuyến, số càng lớn thì mức độ đảm bảo càng cao (đường R được coi là tốt hơn R' nếu số thứ tự của R lớn hơn, trong trường hợp có cùng số thứ tự thì R phải có số bước nhỏ hơn). Số thứ tự sẽ tăng khi nút A phát hiện ra đường đến đích D bị phá vỡ, sau đó nút A quảng bá đường định tuyến của nó tới nút D với số bước không giới hạn và số thứ tự sẽ tăng lên

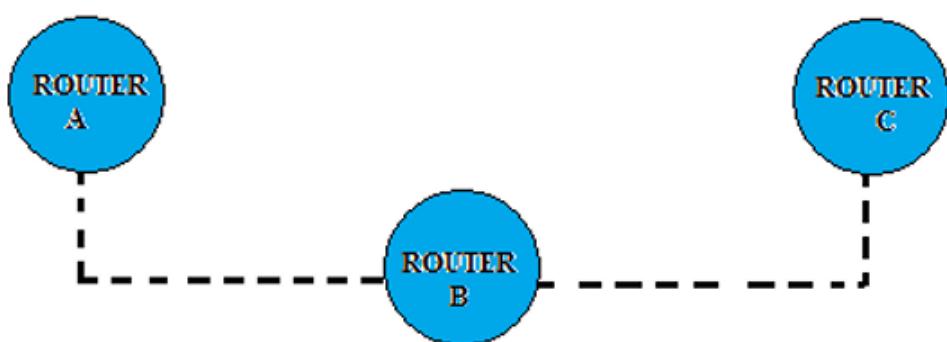
- Đặc điểm:

+ DSDV phụ thuộc vào thông tin quảng bá định kỳ nên nó sẽ tiêu tốn thời gian để tổng hợp thông tin trước khi đường định tuyến được đưa vào sử dụng. Thời gian này là không đáng kể đối với mạng có cấu trúc cố định nói chung (bao gồm cả mạng có dây), nhưng với mạng Ad hoc thời gian này là đáng kể, có thể gây ra mất gói tin trước khi tìm ra được định tuyến hợp lý. Ngoài ra, bản tin quảng cáo định kỳ cũng là nguyên nhân gây ra lãng phí tài nguyên mạng

b) Giao thức DSR (Dynamic source routing)

- Mô tả:

Đây là một giao thức thuộc dạng Distance Vector được dùng trong Manet. Khi một node mạng cần chuyển dữ liệu nhưng chưa biết được đường dẫn đến một địa chỉ nào đó, node mạng này bắt đầu quá trình tìm kiếm đường dẫn (Route discovery). Vì vậy, DSR là giao thức bị động (Chỉ cập nhật trạng thái mạng và tìm đường dẫn khi có yêu cầu). Một ưu điểm của DSR là không có gói tìm đường nào được phát đi định kỳ (vì không cần phải cập nhật trạng thái mạng thường xuyên – trái ngược với giao thức Link state). DSR còn có khả năng điều hành đường dẫn một chiều. Vì DSR tìm đường theo yêu cầu nên nó không thích hợp cho các mạng dung lượng lớn và có tính di động cao. Giao thức DSR cũng có hai hoạt động chính: Tìm đường và bảo trì đường dẫn (Router maintenance). Hình dưới đây cho ta thấy một ví dụ đơn giản của DSR. Router A, B và C lập thành một mạng Manet. Router A và C không kết nối với nhau trong khi cả hai cùng kết nối với router B.



Hình 12: Mô tả giao thức DSR

- Thủ tục tìm kiếm đường:

Giả định rằng ban đầu bộ nhớ đệm trong tất cả các router đều trống (những router này chưa biết gì về sự có mặt của nhau và những đường dẫn có thể có giữa chúng). Khi router A muốn gửi dữ liệu đến router C, nó phát ra tín hiệu yêu cầu tìm đường dẫn, và quá trình tìm đường dẫn lúc này mới được kích hoạt. Router B nhận được yêu cầu của A vì nó nằm trong vùng phủ sóng của A. Router C là địa chỉ của yêu cầu đó và B chưa có thông tin nào về địa chỉ của C lúc này, vì vậy router B gán

ID của nó vào trong danh sách các router trung gian được đính kèm trong yêu cầu của A và chuyển tiếp yêu cầu đó đến những router khác. Khi C nhận được yêu cầu được gửi đến từ B, nó nhận biết rằng địa chỉ của nó trùng với địa chỉ đích đến. Vì vậy một đường dẫn từ A đến C được tìm thấy. Để giúp cho router nguồn (A) và những router trung gian (B) thiết lập đúng đường dẫn, router C gửi một thông điệp trả lời về A trong trường hợp đây là đường dẫn hai chiều. Quá trình này được thực hiện dễ dàng vì ID của những router trung gian đều nằm trong gói yêu cầu được gửi đến C. Những router trung gian này sẽ xây dựng cho mình bảng định tuyến ngay khi chúng nhận được trả lời từ router C. Vì vậy, một đường dẫn từ A đến C được thiết lập

- Đặc điểm:

- + Trong quá trình tìm đường, các router duy trì danh sách ID của những router trung gian trong các yêu cầu tìm kiếm gần thời điểm đó để tránh phải xử lý cùng một yêu cầu tìm kiếm (lặp). Yêu cầu tìm kiếm bị bỏ qua trong trường hợp chúng đã được xử lý gần thời điểm đó và được xác định là một yêu cầu lặp. Khi một router nhận được yêu cầu và nhận ra rằng ID của nó đã nằm sẵn trong danh sách router trung gian của yêu cầu đó thì yêu cầu này sẽ bị bỏ qua

- + Quá trình bảo trì đường dẫn diễn ra khi đường dẫn trở nên không thể sử dụng được vì sự di chuyển không đoán trước của các router (đặc trưng của MANET). Mỗi router quản lý tất cả đường dẫn để chuyển tiếp các gói, khi một đường dẫn hỏng, một gói báo cáo lỗi đường dẫn (Route error) lập tức được gửi về router nguồn và đường dẫn tương ứng. Vì vậy, đường dẫn bị hỏng sẽ bị bỏ qua.

- + Để quản lý việc truyền gói dữ liệu điều khiển vốn không đảm bảo (topo mạng luôn thay đổi), DSR phải dựa vào giao thức ngầm định MAC (XX) để đảm bảo nơi nhận luôn nhận được dữ liệu hoặc nó sẽ gửi gói dữ liệu điều khiển một số lần nhất định. Vì DSR là một giao thức bị động, nó không thể biết được router đích bị ngắt kết nối hay yêu cầu tìm đường bị mất. Vì vậy, chi phí vận hành sẽ lớn trong trường hợp giao thức MAC không đảm bảo dữ liệu luôn tới được đích. Đây là một vấn đề phổ biến của các giao thức bị động, bởi vì khi không nhận được trả lời từ router đích, router có giao thức bị động sẽ không thể phân biệt được hai trường hợp lỗi xảy ra trong quá trình truyền dẫn hoặc một hoặc nhiều node mạng trở nên không thể sử dụng được. Giao thức bị động thường sử dụng nhiều gói xác nhận (Acknowledgement) hoặc gửi dữ liệu đi nhiều lần để khắc phục vấn đề này, tuy nhiên phương pháp này lại làm tăng chi phí hoạt động. Giao thức chủ động phát đi các gói điều khiển định kỳ và bỏ qua các node mạng khi chúng không trả lời sau một số lần phát nhất định, vì vậy giao thức này không mắc phải vấn đề trên, tuy nhiên việc phát các gói điều khiển một cách định kỳ như vậy cũng làm tăng chi phí.

2.2 NS-3

2.2.1 Giới thiệu về ns-3: Lịch sử, mục tiêu phát triển

- *Lịch sử hình thành:*

- + **Từ ns-2 đến ns-3:** Sự ra đời của ns-3 không chỉ là một sự thay thế đơn thuần mà còn là một bước tiến lớn trong công nghệ mô phỏng mạng. ns-2 dù vẫn được sử dụng trong một số môi trường nghiên cứu nhưng đã không còn đáp ứng được các yêu cầu về tính linh hoạt, hiệu năng, khả năng mở rộng trong bối cảnh mạng ngày càng phát triển. ns-3 được thiết kế lại hoàn toàn, sử dụng C++ cho cả lõi và API, giúp tận dụng tối đa sức mạnh của ngôn ngữ này.
- + **Sự phát triển của cộng đồng:** Dự án ns-3 được xây dựng trên tinh thần hợp tác và đóng góp từ cộng đồng, do đó nó luôn được cập nhật và cải tiến liên tục. Nhiều nhà nghiên cứu và lập trình viên trên thế giới đã tham gia vào quá trình phát triển và kiểm thử, giúp ns-3 ngày càng hoàn thiện.
- + **Ảnh hưởng của các công nghệ mới:** ns-3 cũng không ngừng được cập nhật để phù hợp với các công nghệ mạng mới, như 5G, IoT, SDN. Điều này cho thấy ns-3 không chỉ là một trình mô phỏng mạng mà còn là một nền tảng nghiên cứu và phát triển cho các công nghệ mạng tương lai.



Hình 13: Network Simulator 3 - NS3

- *Mục tiêu phát triển:*

- + **Mô phỏng chính xác và hiệu quả:** Ngoài việc khắc phục các hạn chế của ns-2, ns-3 còn đặt mục tiêu cung cấp các mô phỏng mạng chính xác và hiệu quả, giúp người dùng tin tưởng vào kết quả.
- + **Tính linh hoạt và mở rộng:** ns-3 được thiết kế để linh hoạt và dễ mở rộng, cho phép người dùng tùy biến và thêm các module mới một cách dễ dàng, đáp ứng nhu cầu nghiên cứu đa dạng.

+ **Tạo môi trường nghiên cứu chuyên nghiệp:** Cung cấp các công cụ và mô-đun hỗ trợ các nhà nghiên cứu trong việc phát triển và đánh giá các giao thức mạng mới, các kiến trúc mạng tiên tiến.

+ **Kết nối lý thuyết và thực hành:** ns-3 không chỉ cung cấp một môi trường mô phỏng mà còn là công cụ kết nối lý thuyết và thực hành, giúp sinh viên và người học hiểu rõ hơn về các khái niệm và cơ chế trong mạng.

2.2.2 Đặc điểm nổi bật của ns-3

- *Mã nguồn mở và miễn phí :*

+ **Tính minh bạch:** Mã nguồn mở giúp người dùng có thể kiểm tra và hiểu rõ cách hoạt động của trình mô phỏng, đảm bảo tính minh bạch và tin cậy của kết quả.

+ **Cộng đồng rộng lớn:** Nhiều nhà nghiên cứu có thể đóng góp vào sự phát triển, và có thể tiếp cận dễ dàng các phiên bản mới.

+ **Tùy biến dễ dàng:** Người dùng có thể chỉnh sửa và tùy biến các module, thêm các tính năng mới.

- *Hỗ trợ mô phỏng thực tế:*

+ **Mô hình kênh truyền phức tạp:** Cho phép mô phỏng các môi trường truyền dẫn thực tế (như hiệu ứng fading, nhiễu sóng, kênh đa đường), giúp mô phỏng mạng không dây chính xác hơn.

+ **Mô hình năng lượng:** Cung cấp các mô hình pin và tiêu thụ năng lượng, giúp nghiên cứu các giải pháp tiết kiệm năng lượng trong mạng không dây.

+ **Hỗ trợ đa giao thức:** ns-3 hỗ trợ một loạt các giao thức từ tầng thấp đến tầng cao, bao gồm các giao thức định tuyến, giao thức vận tải, các ứng dụng mạng.

+ **Mô hình node chi tiết:** Có thể mô phỏng hành vi chi tiết của các node, như hàng đợi, chuyển mạch, CPU và bộ nhớ.

- *Tính Mô-đun:*

+ **Module độc lập:** Mỗi module có thể được phát triển, kiểm thử và cập nhật một cách độc lập mà không ảnh hưởng đến các module khác.

+ **Dễ dàng tái sử dụng:** Người dùng có thể tái sử dụng các module đã có trong các mô phỏng khác nhau, giúp tiết kiệm thời gian và công sức.

+ **Khả năng mở rộng:** Người dùng có thể dễ dàng thêm các module mới để mở rộng khả năng của trình mô phỏng, phù hợp với yêu cầu của các công nghệ mạng mới.

- *Khả năng tích hợp:*

+ **NetAnim:** Tích hợp với NetAnim để trực quan hóa sự di chuyển của các node, lưu lượng mạng theo thời gian.

- + **PyViz:** Kết hợp với PyViz để visualize đồ thị và các số liệu thu được từ các simulation.
- + **Wireshark:** Cho phép capture các packet trên các node và phân tích chi tiết bằng Wireshark.
- + **Các thư viện:** Để dàng kết nối với các thư viện chuyên dụng, giúp thêm các tính năng cụ thể.
- **Cộng đồng phát triển mạnh mẽ:**
 - + **Hỗ trợ nhiệt tình:** Các thành viên trong cộng đồng luôn sẵn sàng trả lời các câu hỏi, giải quyết các vấn đề và cung cấp các hướng dẫn chi tiết.
 - + **Nguồn tài liệu phong phú:** Có nhiều tài liệu hướng dẫn, ví dụ, bài báo khoa học và các diễn đàn thảo luận về ns-3, giúp người dùng nhanh chóng tiếp cận và sử dụng hiệu quả.
 - + **Phát triển module:** Cộng đồng cũng phát triển nhiều module, có thể dễ dàng dùng lại và chỉnh sửa.

2.2.3 Kiến trúc mô hình mạng cơ bản của ns-3

- **Nodes (Nút mạng):**

- + Các phương thức kết nối: Các Node có thể kết nối với nhau bằng các giao thức khác nhau, ví dụ như point-to-point link, wireless link.
- + Tùy chỉnh các thuộc tính: Có thể chỉnh sửa rất nhiều thuộc tính của node, từ vị trí, vận tốc, hướng di chuyển đến các lớp giao thức và ứng dụng của nó.
- + NodeContainer: Được dùng để quản lý số lượng lớn các node một cách hiệu quả.

- **Net Device (Thiết bị mạng):**

* Net Device trong ns-3:

- + Không chỉ đại diện cho card mạng mà còn bao gồm các trình điều khiển giao diện và các giao thức liên kết.
- + Cấu hình giao thức: Người dùng có thể cấu hình các giao thức liên kết, như Ethernet, Wi-Fi, LTE để điều chỉnh cách thiết bị truyền và nhận dữ liệu.

* NetDeviceContainer: Quản lý các NetDevice, và giúp tạo một list các netdevice có chung thuộc tính.

- **Channel (Kênh truyền):**

- + Các loại kênh: Có nhiều loại channel như PointToPointChannel (dùng cho các kết nối hữu tuyến), WifiChannel và YansWifiChannel (dùng cho kết nối không dây), CsmaChannel,...

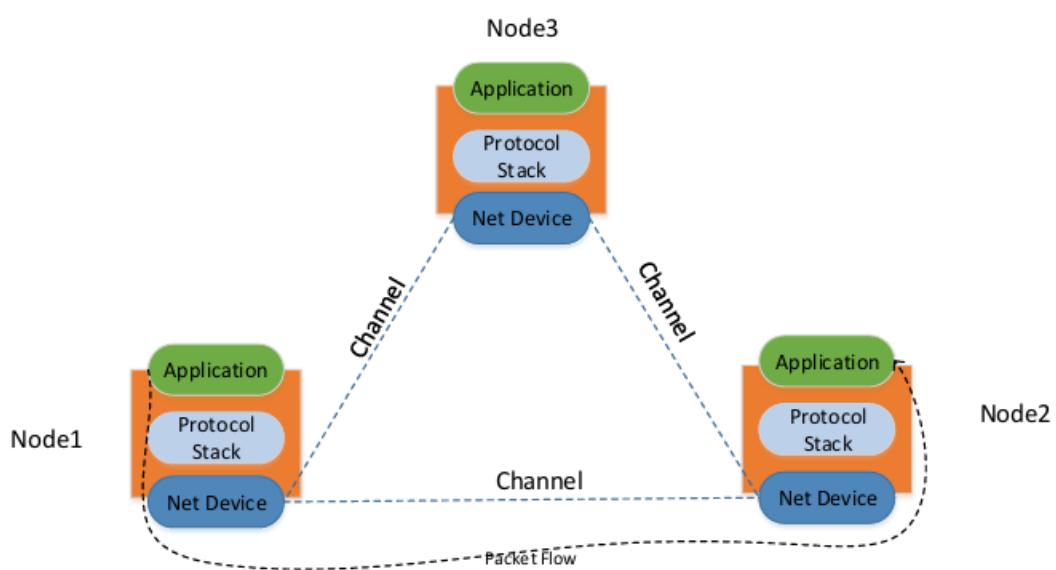
- + Mô phỏng suy hao: Channel hỗ trợ mô phỏng các hiện tượng suy hao tín hiệu (pathloss, fading).
- + Mô phỏng nhiễu: Mô phỏng các yếu tố gây nhiễu sóng vô tuyến và ảnh hưởng đến quá trình truyền dữ liệu.

- Protocol Stack (Ngăn xếp giao thức):

- + Các giao thức thường dùng: TCP, UDP, IP, AODV, OLSR,...
- + Mô phỏng các giao thức: Mỗi giao thức có thể được mô phỏng chi tiết, bao gồm các tham số và các cơ chế hoạt động.
- + Linh hoạt: Có thể tùy chỉnh và thay thế từng module của các giao thức để đánh giá ảnh hưởng của chúng.
- + Ứng dụng nhiều layer: Protocol Stack hỗ trợ các giao thức trên tất cả các layer, từ vật lý đến ứng dụng.

- Application (Ứng dụng):

- + Các ứng dụng cơ bản: Cung cấp các class ứng dụng cơ bản như UdpClient, UdpServer, OnOffApplication và nhiều ứng dụng khác.
- + Tùy chỉnh ứng dụng: Có thể tùy chỉnh hoặc xây dựng các ứng dụng mới phù hợp với mục tiêu mô phỏng.



Hình 14: Kiến trúc mô hình mạng của ns-3

2.2.4 So sánh giữa ns-2 và ns-3

<i>Tiêu chí</i>	<i>ns-2</i>	<i>ns-3</i>
Ngôn ngữ lập trình	C++ kết hợp với TCL, TCL dùng để viết kịch bản nên có thể khá khó khăn cho người mới bắt đầu.	C++ hoàn toàn, giúp tận dụng triệt để sức mạnh của ngôn ngữ này, dễ dàng mở rộng và tích hợp các module mới. Việc dùng Python cho các kịch bản giúp cho việc viết code đơn giản và dễ debug hơn.
Kiến trúc	Kiến trúc không được module hóa, khiến việc bảo trì, mở rộng và tùy biến khó khăn.	Module hóa một cách triệt để, code có tính tái sử dụng cao, dễ dàng mở rộng thêm các module.
Giao diện người dùng	Sử dụng TCL để viết kịch bản gây khó khăn cho người mới tiếp cận.	Sử dụng Python làm cho code dễ đọc, dễ viết và dễ debug hơn.
Hiệu năng mô phỏng	Xử lý tương đối chậm, không tối ưu cho các mô phỏng lớn.	Được thiết kế để tối ưu hóa hiệu năng, xử lý các kịch bản mô phỏng phức tạp hoặc quy mô lớn tốt hơn.
Khả năng tích hợp	Khó khăn trong việc tích hợp với các phần mềm hoặc thư viện khác.	Hỗ trợ tích hợp với nhiều công cụ như NetAnim, PyViz, và các mô-đun phân tích khác.
Lĩnh vực sử dụng	Nghiên cứu mạng cơ bản, mô phỏng giao thức truyền thông.	Mô phỏng các giao thức hiện đại, nghiên cứu mạng 5G, IoT, SDN, MANET.
Khả năng ứng dụng	Nghiên cứu cơ bản thì thích hợp, nghiên cứu hiện đại thì không phù hợp và trong ứng dụng thực tế thì hạn chế	Nghiên cứu cơ bản thì thích hợp hơn, nghiên cứu hiện đại thì đáp ứng rất tốt và trong ứng dụng thực tế thì rất khả thi, phù hợp

Bảng 2: So sánh giữa ns-2 và ns-3

2.3 NetSimulyzer

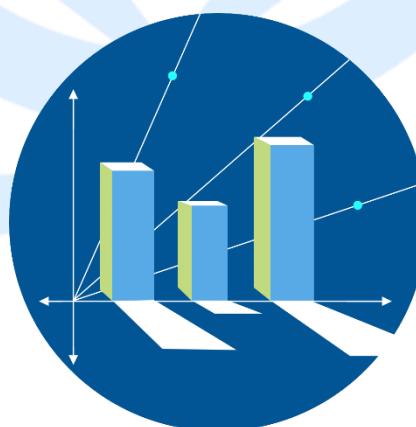
2.3.1 Giới thiệu về NetSimulyzer: Mục tiêu, vai trò trong trực quan hóa mô phỏng mạng

- Mục tiêu:

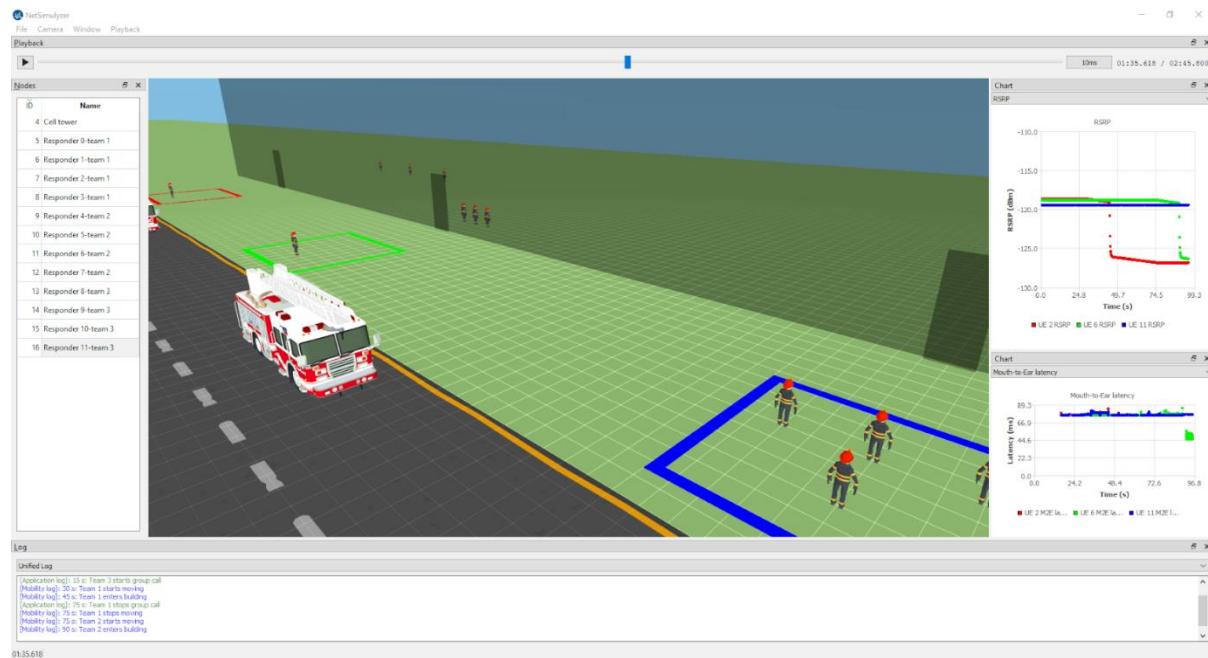
- + **Tối ưu hóa việc phân tích dữ liệu:** NetSimulyzer không chỉ hiển thị dữ liệu mà còn giúp người dùng phân tích dữ liệu một cách trực quan, từ đó có cái nhìn sâu sắc hơn về các hiện tượng mạng.
- + **Đơn giản hóa quá trình:** Các công cụ trực quan hóa mặc định có thể không đủ, nên NetSimulyzer có mục tiêu làm cho việc phân tích và hiểu dữ liệu trở nên đơn giản và dễ dàng hơn đối với mọi đối tượng.
- + **Hỗ trợ thiết kế mạng:** NetSimulyzer còn hỗ trợ việc thiết kế và tối ưu hóa mạng, giúp người dùng đưa ra các quyết định thông minh hơn.

- Vai trò quan trọng:

- + **Tạo ra trải nghiệm tương tác:** NetSimulyzer cung cấp giao diện người dùng thân thiện, cho phép người dùng tương tác với mô phỏng, xem dữ liệu từ nhiều góc độ khác nhau, thay đổi góc nhìn và zoom in/out để xem chi tiết.
- + **Hỗ trợ nghiên cứu đa dạng:** Không chỉ hỗ trợ nghiên cứu mạng UAV, NetSimulyzer còn có thể được sử dụng trong nhiều lĩnh vực khác như mô phỏng giao thông, mô phỏng nhà máy, đô thị,...
- + **Kết luận:** NetSimulyzer là một công cụ mạnh mẽ, không chỉ đơn giản là hiển thị hình ảnh mà còn cung cấp một cái nhìn trực quan, tương tác và toàn diện về hoạt động của mạng.

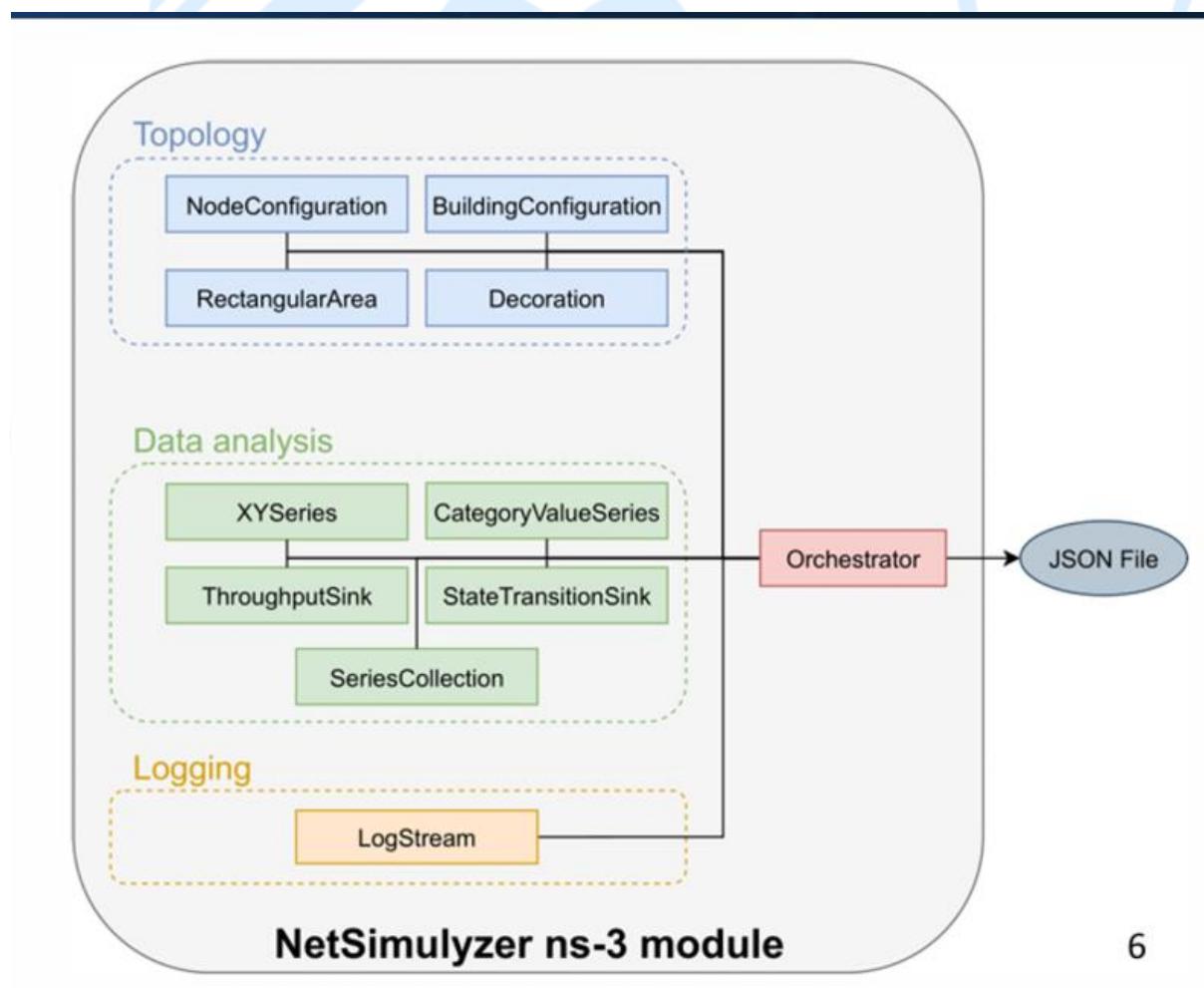


NetSimulyzer



Hình 15: Giao diện mô phỏng NetSimulyzer

2.3.2 Kiến trúc của NetSimulyzer



Hình 16: NetSimulyzer ns-3 Module

- Topology (Cấu hình mạng lưới):

- + NodeConfiguration: Không chỉ định nghĩa các thông số node cơ bản mà còn cung cấp khả năng tùy chỉnh hình dạng, kích thước, màu sắc của node, thậm chí có thể load các model 3D, tên, vị trí, các interface kết nối mạng.
- + BuildingConfiguration: Cho phép người dùng định nghĩa các công trình trong không gian 3D, ví dụ như các tòa nhà, chướng ngại vật, để tạo ra môi trường mô phỏng phức tạp và thực tế hơn.
- + RectangularArea & Decoration: Cho phép tùy chỉnh kích thước, màu sắc và chất liệu của các bề mặt, tạo ra các cảnh quan khác nhau trong mô phỏng, giúp tăng tính trực quan và dễ dàng phân biệt các khu vực.

- Data Analysis (Phân tích dữ liệu):

- + XYSeries & CategoryValueSeries: XYSeries dùng để hiển thị các dữ liệu theo trục x, y. CategoryValueSeries dùng để hiển thị các dữ liệu theo các category khác nhau. Hỗ trợ người dùng tạo ra đồ thị dễ dàng.
- + ThroughputSink: Chuyên dùng để thu thập và tính toán thông lượng trên các luồng dữ liệu, giúp theo dõi và đánh giá hiệu suất truyền tải.
- + StateTransitionSink: Cho phép thu thập và phân tích các sự thay đổi trạng thái của thiết bị.
- + SeriesCollection: Kết hợp các loại chuỗi dữ liệu (XYSeries, CategoryValueSeries), giúp chúng dễ dàng được nhóm lại và sử dụng trong các đồ thị.

- Logging (Ghi nhật ký):

- + LogStream: Cho phép ghi lại các sự kiện trong quá trình mô phỏng, bao gồm các thông báo của các module trong NetSimulyzer và thông tin từ các thiết bị mô phỏng. Các log stream này có thể được đặt tên, màu sắc và hiển thị theo nhiều cách khác nhau, giúp người dùng theo dõi và debug mô phỏng dễ hơn.
- * Unified Log: NetSimulyzer cung cấp một cửa sổ chung để hiển thị tất cả các log, giúp người dùng xem được một cách tổng quan các sự kiện xảy ra.

- Orchestrator (Điều phối):

- + Chức năng chính: Thu thập dữ liệu từ các lớp topology, data analysis, và logging, chuẩn hóa dữ liệu và xuất ra file JSON để hiển thị.
- + Đảm bảo tính nhất quán và toàn vẹn của dữ liệu khi được truyền đi để visualization.
- + Có thể tùy chỉnh các parameter để tạo ra các file JSON phù hợp.

2.3.3 Quy trình luồng dữ liệu hoạt động của NetSimulyzer trong mô phỏng mạng

- Cấu hình mạng:

- + Người dùng có thể tạo các file JSON cấu hình, thiết lập vị trí, tốc độ, hướng di chuyển của UAV, và các thông số liên quan đến mạng.
- + Cho phép người dùng thay đổi các tham số của UAV, môi trường, và các giao thức.
- + Các file cấu hình này có thể được tạo một cách tự động hoặc thủ công, tùy theo yêu cầu của kịch bản mô phỏng.

- Thu thập dữ liệu:

- + Các module Data Analysis sẽ thu thập dữ liệu về throughput, độ trễ, và các thông số khác, sử dụng các hook hoặc callback được cung cấp bởi ns-3.
- + Dữ liệu được thu thập một cách liên tục hoặc theo các sự kiện cụ thể.

- Ghi nhật ký (chi tiết):

- + LogStream ghi lại thông tin quan trọng về các node, các sự kiện, và các trạng thái của mạng.
- + Các thông tin debug cũng có thể được ghi lại, hỗ trợ việc kiểm tra và sửa lỗi.

- Xử lý và xuất file JSON:

- + Orchestrator chuẩn hóa các dữ liệu theo một định dạng thống nhất.
- + Xuất dữ liệu ra định dạng JSON giúp NetSimulyzer dễ dàng đọc và sử dụng để visualization.

2.3.4 Cấu trúc tệp và thư mục của NetSimulyzer

LINK SƠ ĐỒ CẤU TRÚC TỆP:

https://drive.google.com/drive/folders/1_Y3eg3PeTFBN7-dqN3kZNlmgIFF5z59W

Dưới đây là phân tích chi tiết vai trò của từng tệp và thư mục trong cấu trúc NetSimulyzer:

Cấu trúc gốc

- **CMakeLists.txt:** Tệp cấu hình CMake để biên dịch dự án. Nó xác định các tệp nguồn, thư viện cần thiết và các tham số biên dịch.
- **LICENSE.md:** Tệp chứa thông tin về giấy phép sử dụng phần mềm.
- **README.md:** Hướng dẫn cơ bản và giới thiệu về dự án.

- **THIRD_PARTY_LICENSES.md:** Danh sách và giấy phép của các thư viện bên thứ ba được sử dụng trong dự án.
 - **wscript:** Kịch bản Waf, một công cụ xây dựng thay thế cho CMake.
-

Thư mục doc

Chứa tài liệu hướng dẫn sử dụng và cấu trúc của NetSimulyzer:

- **make.bat & Makefile:** Tệp dùng để biên dịch tài liệu (thường với Sphinx).
 - **source:** Tài liệu nguồn:
 - **.rst:** Các tệp tài liệu định dạng reStructuredText (dùng với Sphinx).
 - **architecture.rst:** Mô tả kiến trúc hệ thống.
 - **areas.rst, buildings.rst:** Giải thích về các mô hình khu vực và tòa nhà.
 - **colors.rst, decorations.rst:** Hướng dẫn về sử dụng màu sắc và các yếu tố trang trí.
 - **linking-module.rst:** Hướng dẫn liên kết module với NetSimulyzer.
 - **quickstart.rst:** Tài liệu bắt đầu nhanh.
 - **_static:** Thư mục chứa các hình ảnh và tài nguyên bổ sung cho tài liệu.
 - **_templates:** Mẫu cho các tài liệu.
-

Thư mục examples

Chứa các ví dụ minh họa:

- **.cc:** Các ví dụ bằng C++ để người dùng hiểu cách sử dụng NetSimulyzer:
 - **wifi-bianchi-netsimulyzer.cc:** Mô phỏng mô hình WiFi.
 - **ecdf-sink-example-netsimulyzer.cc:** Ví dụ về phân tích ECDF (empirical cumulative distribution function).
 - **mobility-buildings-example.cc:** Mô phỏng di chuyển trong môi trường có tòa nhà.
 - **CMakeLists.txt:** Để biên dịch các ví dụ.
-

Thư mục helper

Cung cấp các tiện ích và container cho cấu hình:

- **building-configuration-***: Quản lý cấu hình các tòa nhà trong mô phỏng.
 - **node-configuration-***: Hỗ trợ cấu hình các node (thiết bị) trong mô phỏng.
-

Thư mục library

- **json.hpp**: Thư viện JSON header-only dùng để xử lý dữ liệu JSON, hỗ trợ việc trao đổi dữ liệu cấu hình và kết quả mô phỏng.
-

Thư mục model

Chứa các tệp chính để mô phỏng:

- **building-configuration.***: Mô phỏng cấu trúc tòa nhà.
 - **category-***: Mô hình trực và giá trị theo danh mục (category axis).
 - **color.***: Mô phỏng và quản lý màu sắc.
 - **orchestrator.***: Điều phối các thành phần của mô phỏng.
 - **state-transition-sink.***: Ghi lại các trạng thái chuyển tiếp.
 - **value-axis.***, **xy-series.***: Mô phỏng dữ liệu chuỗi số và các giá trị trực.
-

Thư mục test

- **examples-to-run.py**: Kịch bản để kiểm tra tính đúng đắn của các ví dụ.
-

Vai trò tổng thể

- **CMakeLists.txt và wscript**: Hỗ trợ xây dựng và biên dịch dự án.
- **doc**: Tài liệu chi tiết để hướng dẫn sử dụng.
- **examples**: Cung cấp các ví dụ minh họa về cách sử dụng và triển khai NetSimulyzer.
- **helper và model**: Các tệp nguồn chính xử lý logic và cấu hình của mô phỏng.
- **library**: Xử lý dữ liệu JSON để tích hợp và cấu hình mô phỏng.

2.4 Mạng UAV-based network

2.4.1 Tổng quan về mạng UAV-based

- Tính chất mạng động:

- + Không chỉ di chuyển liên tục, mà tốc độ, hướng đi, độ cao của UAV cũng thay đổi liên tục theo thời gian.

- + Tính chất động này tạo nên một môi trường mạng vô cùng phức tạp, đòi hỏi các giao thức và cơ chế quản lý phải thật sự linh hoạt.

* Giao tiếp đa dạng:

- + Các UAV có thể giao tiếp trực tiếp với nhau, thông qua các UAV trung gian, hoặc với trạm điều khiển.

- + Mỗi kiểu giao tiếp có một đặc điểm riêng về phạm vi, tốc độ, và độ tin cậy.

- Tính không đồng nhất:

- + Các UAV có thể có nhiều loại khác nhau (fixed-wing, multirotor) và mục đích khác nhau.

- + Điều này đòi hỏi có các cơ chế quản lý tài nguyên linh hoạt để phù hợp với từng loại UAV.

- Khả năng chịu lỗi:

- + Mạng lưới UAV phải có khả năng chịu lỗi khi một số UAV bị hỏng hoặc mất liên lạc.

- * Các giao thức định tuyến và quản lý mạng cần hỗ trợ tính năng phục hồi và định tuyến lại khi có sự cố.

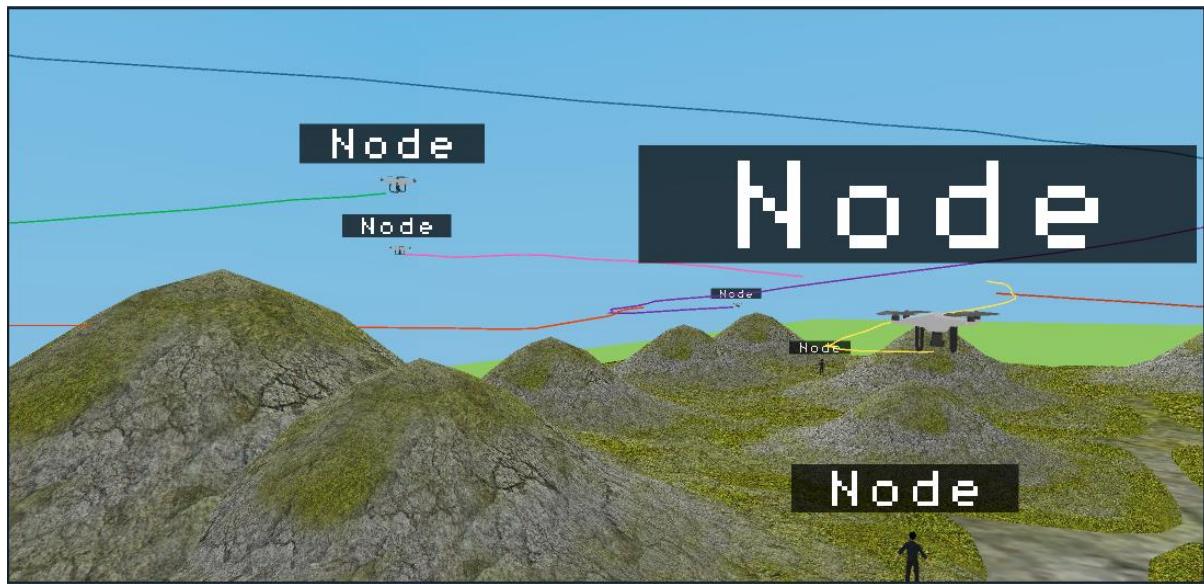
2.4.2 Mô hình mạng UAV trong NetSimulyzer:

- Node UAV:

- + Mô hình hóa hành vi: Cho phép mô hình hóa hành vi của UAV (di chuyển, tốc độ, độ cao) thông qua các bộ điều khiển và các thuật toán di chuyển.

- + Tích hợp cảm biến: Có thể tùy chỉnh để các Node có thể thu thập thông tin qua các cảm biến, từ đó gửi dữ liệu về cho trạm điều khiển.

- + Tùy chỉnh thuộc tính: Có thể tùy chỉnh nhiều thuộc tính như mô hình 3D, thông tin kết nối, tên, label,...



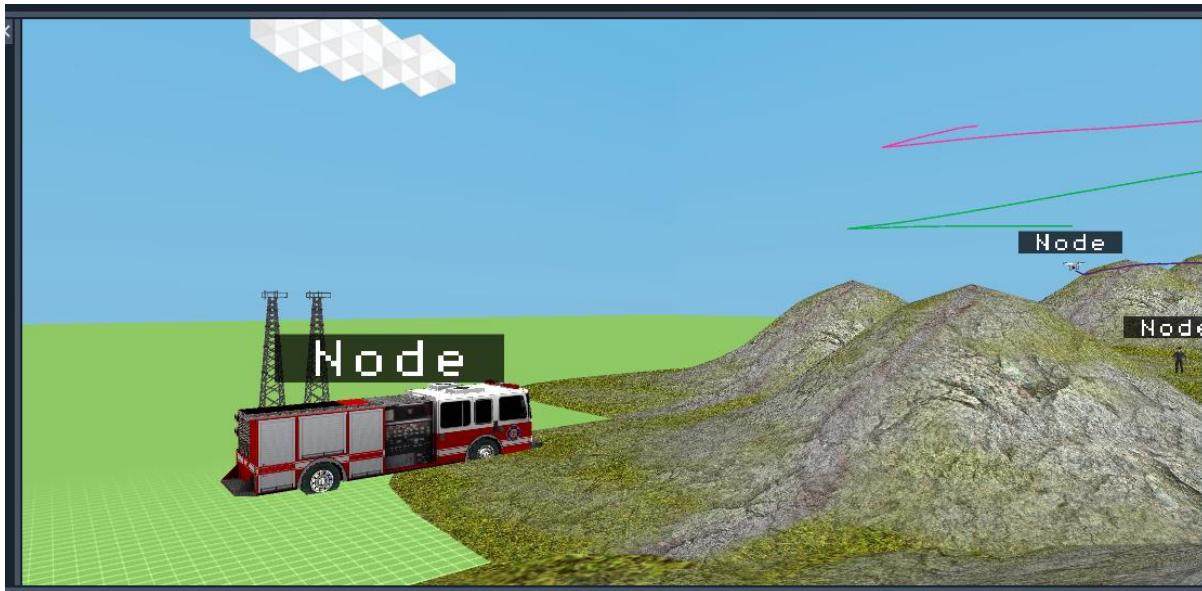
Hình 17: Các node UAV

- Channel:

- + Mô hình truyền sóng: Hỗ trợ các mô hình truyền sóng khác nhau, từ mô hình lý tưởng đến các mô hình thực tế có các yếu tố như suy hao, nhiễu, và các tác động của môi trường.
- + Đa dạng giao thức: Hỗ trợ các giao thức không dây phổ biến (ví dụ Wi-Fi, LTE, 5G) hoặc các giao thức riêng biệt để phù hợp với các trường hợp sử dụng khác nhau.
- + Tính linh hoạt: Cho phép cấu hình nhiều tham số kênh truyền để mô phỏng các điều kiện thực tế, từ đó đánh giá một cách chính xác nhất.

- Ground Station (Trạm mặt đất):

- + Đa chức năng: Trạm mặt đất có thể đóng vai trò điều khiển, giám sát, hoặc là nơi thu thập dữ liệu từ UAV.
- + Mô hình kết nối: Có thể thiết lập các kết nối có dây hoặc không dây với mạng internet hoặc các hệ thống khác.
- + Quản lý các UAV: Các trạm điều khiển có thể được xây dựng để quản lý một nhóm các UAV cùng một lúc.



Hình 18: Trạm mặt đất

- Traffic Flow:

- + Giao thức truyền tải: Có thể tùy chỉnh các giao thức truyền tải dữ liệu (TCP, UDP) theo mục đích sử dụng, thiết lập thời gian truyền, kích thước gói tin.
- + Loại dữ liệu: Mô phỏng các loại dữ liệu khác nhau (ví dụ, video, hình ảnh, dữ liệu cảm biến) với các đặc tính riêng biệt, để có thể tùy chỉnh trong quá trình mô phỏng.
- + Giao tiếp giữa các UAV: Không chỉ là truyền từ UAV đến trạm điều khiển mà còn cần các luồng dữ liệu khác, ví dụ như trao đổi thông tin giữa các UAV, kết nối và giao tiếp giữa các node.

- Cơ chế hoạt động:

*Tạo mạng:

- + Hỗ trợ các giao diện GUI để người dùng thiết kế và cấu hình một mạng lưới UAV, các thiết bị, thông số.
- + Cho phép lưu lại các cấu hình để có thể sử dụng cho nhiều lần sau, tiết kiệm thời gian khi muốn tạo một hệ thống tương tự.

*Định tuyến dữ liệu:

- + Cho phép lựa chọn, cấu hình và đánh giá các giao thức định tuyến khác nhau (AODV, DSR, OLSR,...) và xem chúng hoạt động trong môi trường động của UAV.
- + Có thể visualize các đường đi của dữ liệu để hiểu rõ hơn về cách thức hoạt động của mạng.

*Theo dõi chuyên động:

+ Người dùng có thể thiết kế các đường đi cụ thể để cho UAV di chuyển theo một lộ trình đã được lập sẵn.

+ Hỗ trợ các API để xây dựng các mô hình di chuyển phức tạp (ví dụ như tránh chướng ngại vật).

*Thu thập và giám sát dữ liệu:

+ Có thể dùng NetSimulyzer để hiển thị thông tin một cách trực quan, đồng thời có thể kết hợp với các công cụ khác để thu thập các thông tin chi tiết về băng thông, độ trễ,...



CHƯƠNG III. THIẾT KẾ VÀ TRIỂN KHAI MÔ PHỎNG

3.1 Cài đặt môi trường mô phỏng

3.1.1 NS-3

3.1.1.1. Chuẩn bị

- Thực hiện trên máy ảo Ubuntu 22.04 OS:

OS Name	Ubuntu 22.04.5 LTS
OS Type	64-bit
GNOOME Version	42.9
Windowing System	Wayland
Virtualization	VMware
Software Updates	>

Hình 19: Hệ điều hành của máy ảo sử dụng

- Tiến hành cài đặt các lệnh cần thiết:

\$ sudo apt update:

```
vantai@vantai-virtual-machine:~/ns-allinone-3.38/ns-3.38$ sudo apt update
```

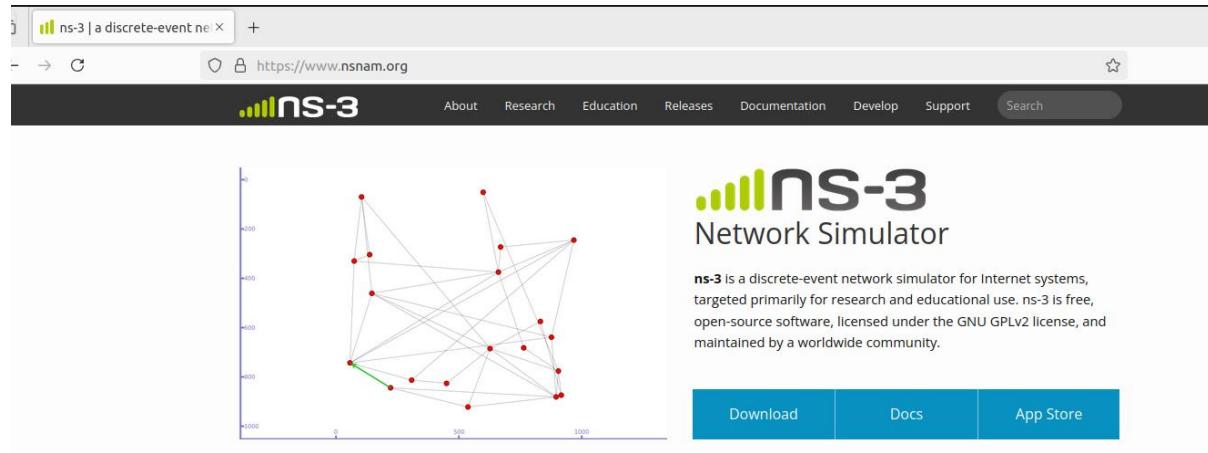
Hình 20: Nâng cấp hệ thống

```
$ sudo apt install g++ python3 cmake ninja-build git gir1.2-goocanvas-2.0 python3-gi
python3-gi-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython3 tcpdump wireshark
sqlite sqlite3 libsqlite3-dev qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools
openmpi-bin openmpi-common openmpi-doc libopenmpi-dev doxygen graphviz
imagemagick python3-sphinx dia imagemagick texlive dvipng latexmk texlive-extra-
utils texlive-latex-extra texlive-font-utils libeigen3-dev gsl-bin libgsl-dev libgslcblas0
libxml2 libxml2-dev libgtk-3-dev lxc-utils lxc-templates vtun uml-utilities ebttables
bridge-utils libxml2 libxml2-dev libboost-all-dev
```

```
vantai@vantai-virtual-machine:~/ns-allinone-3.38/ns-3.38$ sudo apt install g++ python3
cmake ninja-build git gir1.2-goocanvas-2.0 python3-gi python3-gi-cairo python3-pygraphviz
gir1.2-gtk-3.0 ipython3 tcpdump wireshark sqlite3 libsqlite3-dev qtbase5-dev
qtchooser qt5-qmake qtbase5-dev-tools openmpi-bin openmpi-common openmpi-doc libopenmpi
-dev doxygen graphviz imagemagick python3-sphinx dia imagemagick texlive dvipng latexmk
texlive-extra-utils texlive-latex-extra texlive-font-utils libeigen3-dev gsl-bin libgs
l-dev libgslcblas0 libxml2 libxml2-dev libgtk-3-dev lxc-utils lxc-templates vtun uml-ut
ilities ebtables bridge-utils libxml2 libxml2-dev libboost-all-dev
```

Hình 21: Cài đặt thư viện

- Tải và cài đặt ns-3.38 trên nsnam.org:

*Hình 22: Giao diện trang web nsnam*

- File sau khi đã giải nén:

*Hình 23: File ns-3 sau khi giải nén*

3.1.1.2. Cài đặt và chạy thử

- Tiến hành cài đặt ns-3 lên máy ảo

```
vantai@vantai-virtual-machine:~/ns-allinone-3.38$ ./build.py --enable-examples --enable-tests
# Build NetAnim
Entering directory `netanim-3.109'
=> qmake -v
QMake version 3.1
Using Qt version 5.15.3 in /usr/lib/x86_64-linux-gnu
qmake found
=> qmake NetAnim.pro
=> make
make: Nothing to be done for 'first'.
Leaving directory `netanim-3.109'
# Building examples (by user request)
```

Hình 24: Cài đặt ns-3 vào máy ảo

- Sau khi cài đặt xong tiến hành chạy thử file code:

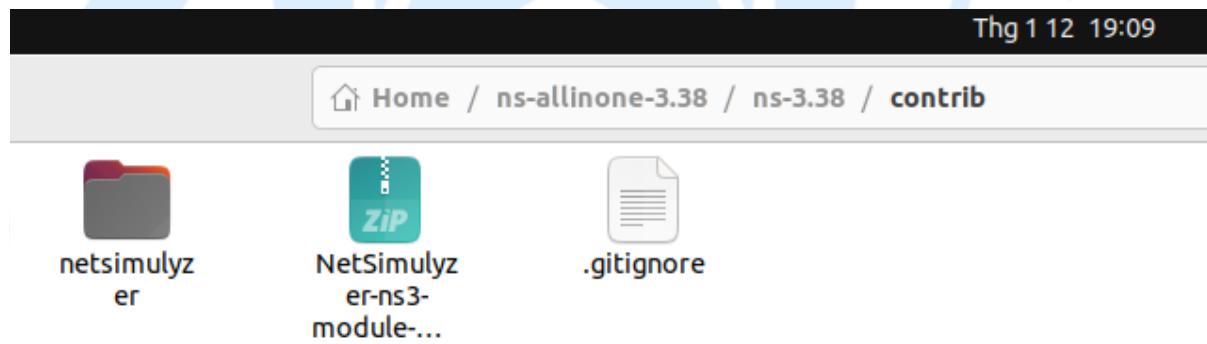
```
vantai@vantai-virtual-machine:~/ns-allinone-3.38/ns-3.38$ ./ns3 run hello-simulator
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ../build/examples/tutorial/ns3.38-hello-simulator-default
Hello Simulator
vantai@vantai-virtual-machine:~/ns-allinone-3.38/ns-3.38$
```

Hình 25: Chạy thử lệnh Hello Simulator

3.1.2 NetSimulyzer

3.1.2.1. NetSimulyzer module 1.0.7

- Git file từ github về bằng lệnh:
- ```
$ wget https://github.com/usnistgov/NetSimul... -O NetSimulyzer-ns3-module-master.zip
```
- Giải nén file vào thư mục contrib như hình dưới:

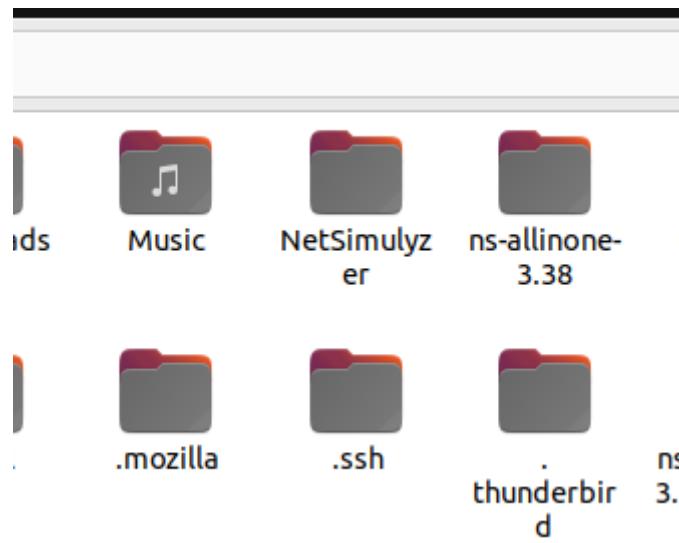
*Hình 26: NetSimulyzer module sau khi giải nén*

- Tiến hành cài đặt NetSimulyzer lên ns-3 bằng lệnh:

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

#### 3.1.2.2. NetSimulyzer software

- Tải về từ Github:
- ```
$ git clone --recursive https://github.com/usnistgov/NetSimul...
```
- Giải nén ta được thư mục NetSimulyzer:



Hình 27: NetSimulyzer software sau khi giải nén

- Thực hiện lần lượt các lệnh dưới đây để cài đặt lên máy ảo:

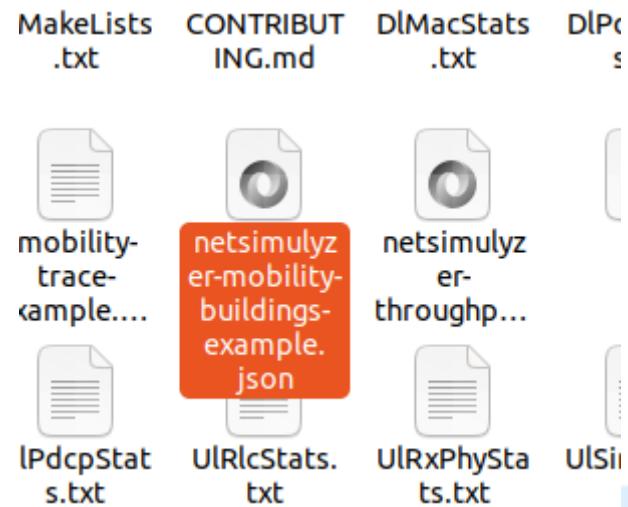
```
$ cd NetSimulyzer/
$ mkdir build
$ cd build
$ cmake -DCMAKE_BUILD_TYPE=Release
$ cmake --build
```

- Sau khi chạy xong để chạy thử ta copy các file .cc từ ***ns-allinone-3.38/ns-3.38/contrib/NetSimulyzer/examples*** đến ***ns-allinone-3.38/ns-3.38/scratch***
- Tiến hành chạy thử một kịch bản:

```
vantai@vantai-virtual-machine:~/ns-allinone-3.38/ns-3.38$ ./ns3 run scratch/mobility-buildings-example.cc
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ..../build/scratch/ns3.38-mobility-buildings-example-default
vantai@vantai-virtual-machine:~/ns-allinone-3.38/ns-3.38$
```

Hình 28: Chạy thử kịch bản NetSimulyzer

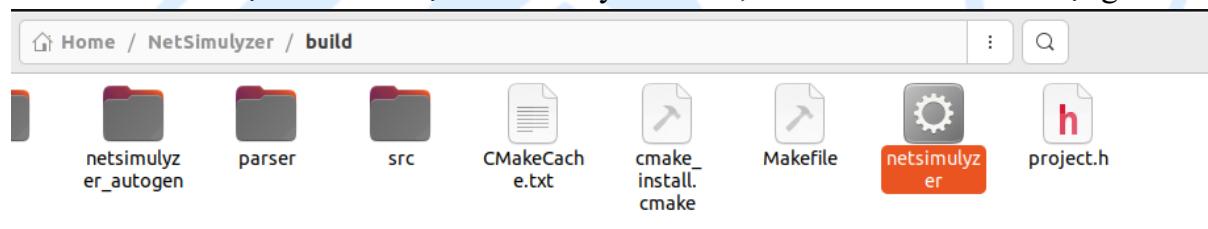
- Sau đó thu được file .json như sau:



Hình 29: File .json thu được

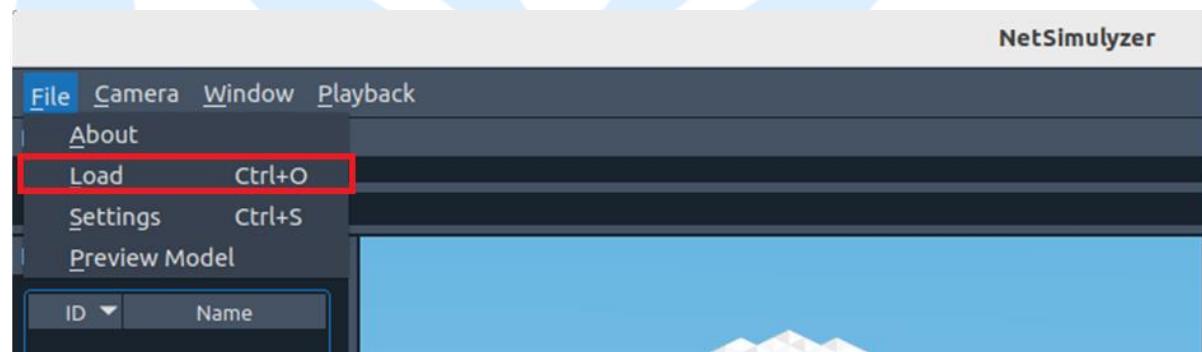
- Tiến hành mở NetSimulyzer để chạy thử kịch bản mẫu:
 - + Có thể chạy bằng lệnh:


```
$ cd NetSimulyzer/build
$ ./netsimulyzer
```
 - + Hoặc vào thư mục NetSimulyzer/build, click 2 lần vào biểu tượng:

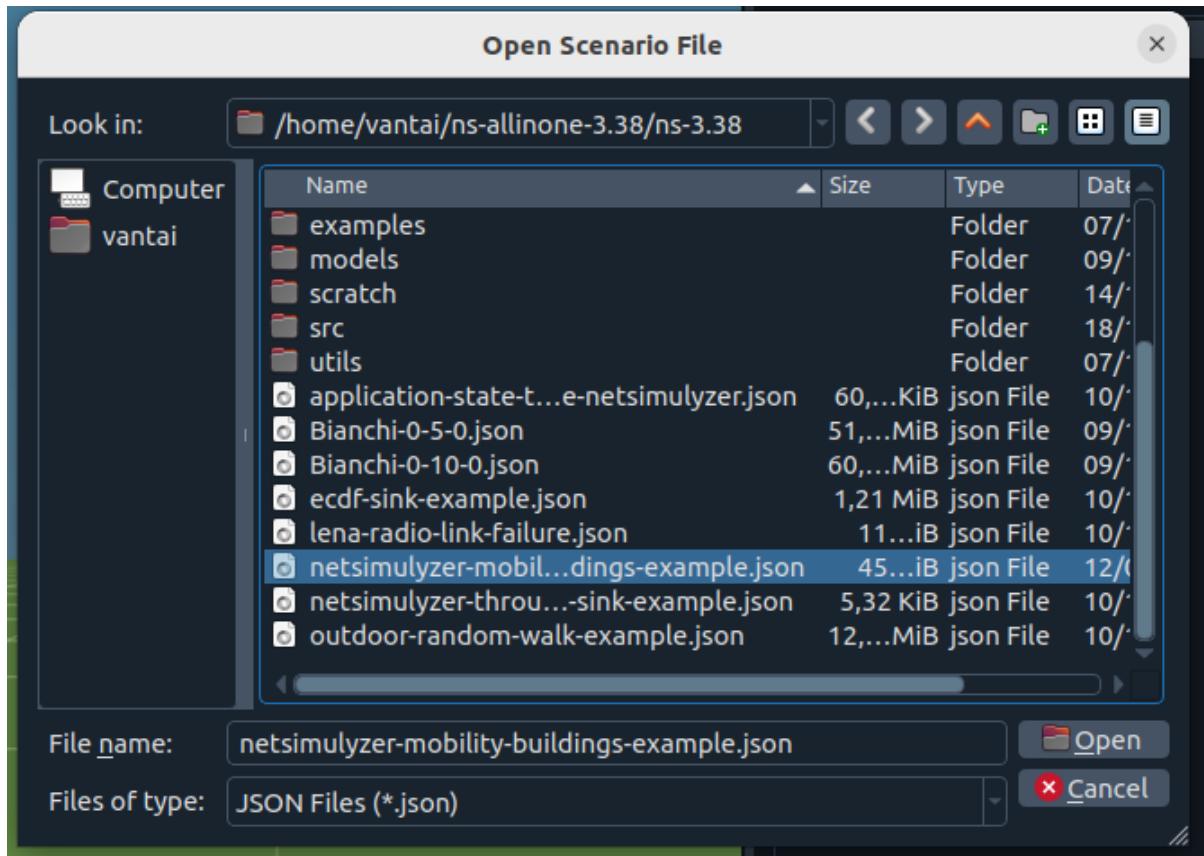


Hình 30: Click 2 lần để mở NetSimulyzer

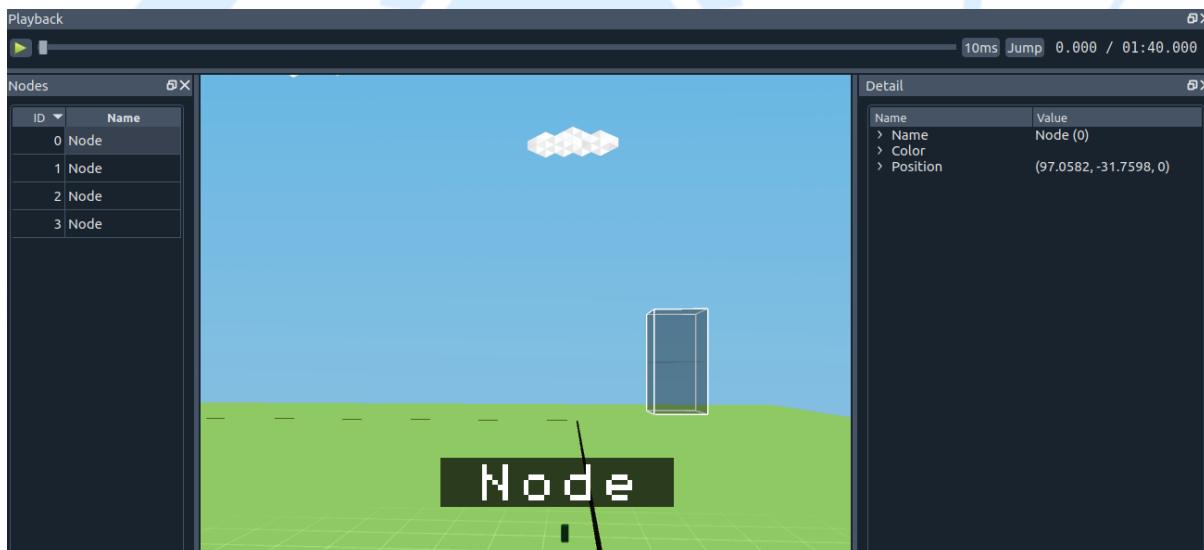
- Tiến hành load file .json và mô phỏng kịch bản mẫu:



Hình 31: Thực hiện load file kịch bản



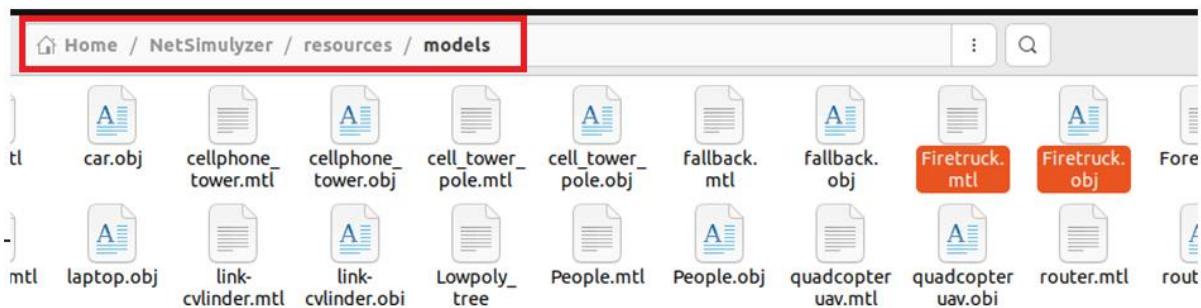
Hình 32: Chọn đúng file .json của kịch bản



Hình 33: Giao diện của NetSimulyzer khi load kịch bản

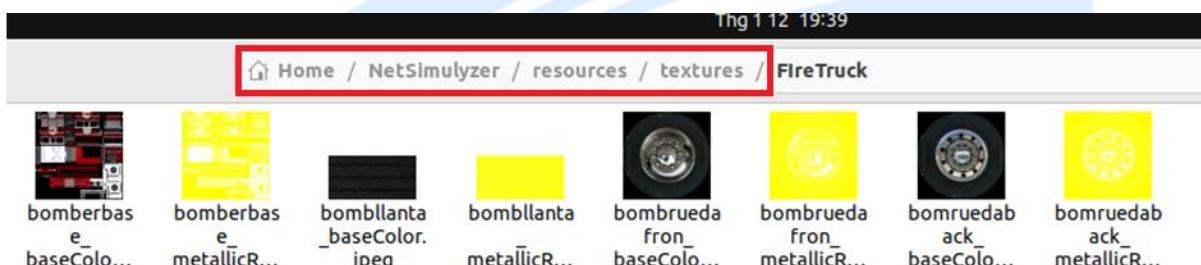
3.1.2.3. Thêm mô hình bên ngoài vào NetSimulyzer

- Tải mô hình gồm 3 thành phần chính:
 - File .obj
 - File .mtl
 - Các file hình ảnh cho mô hình
- Để file .obj và file .mtl tại đường dẫn:



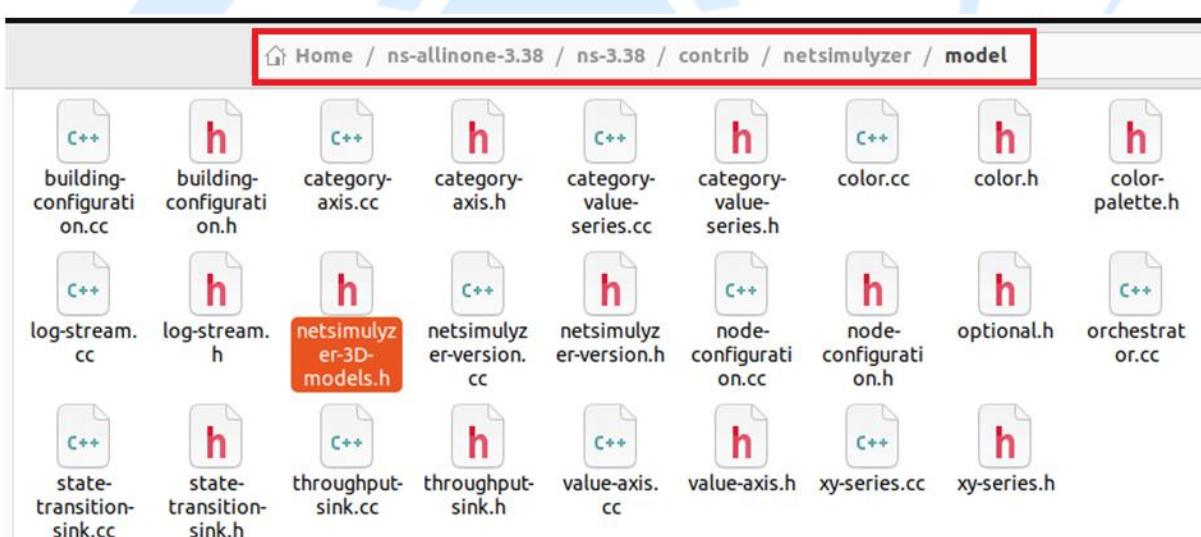
Hình 34: Nơi chứa các file đối tượng mặc định của NetSimulyzer

- Để các file hình ảnh tại đường dẫn (nên tạo 1 thư mục riêng chứa ảnh để dễ phân loại):



Hình 35: File hình ảnh của mô hình

- Mở file netsimulyzer-3D-models.h tại đường dẫn:



Hình 36: Thư viện của NetSimulyzer

- Thêm 2 dòng định nghĩa vào file (tên file .obj giống với file .obj đã thêm vào ở bước trên):

```

77
78 const std::string FIRETRUCK{"models/Firetruck.obj"};
79 const StringValue FIRETRUCK_VALUE{FIRETRUCK};

```

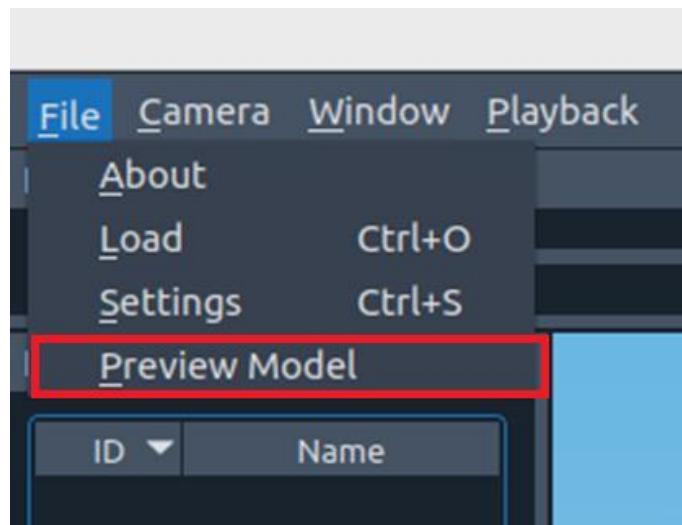
Hình 37: Cấu trúc dựa theo cấu trúc mặc định trong file

- Mở file .mtl để chỉnh đường dẫn hình ảnh bằng cách thêm vào mỗi tên ảnh 2 ký tự “.\”:

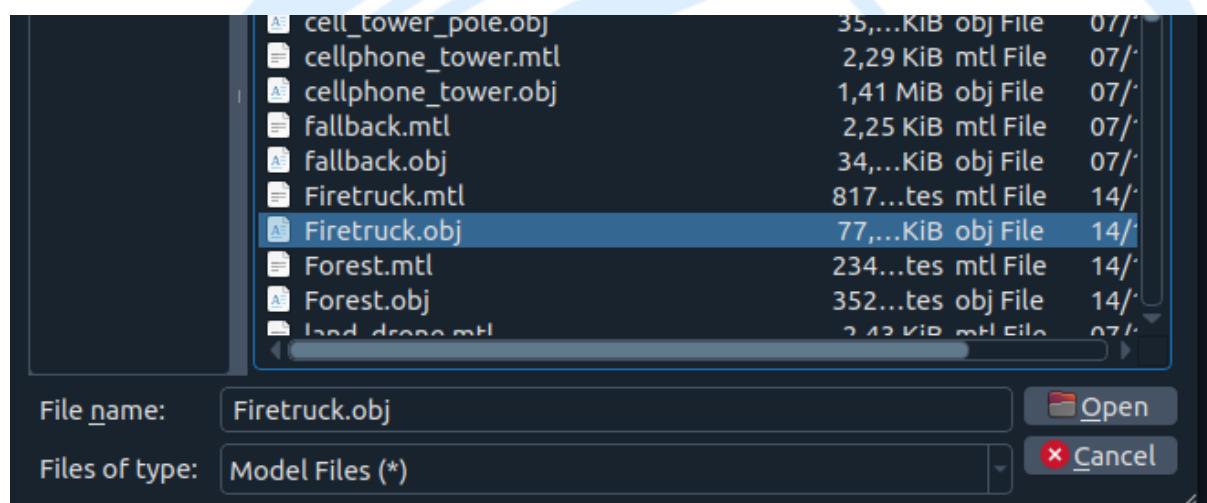
```
1 # Blender 4.3.0 MTL File: 'None'
2 # www.blender.org
3
4 newmtl bomberbase
5 Ns 600.000000
6 Ka 1.000000 1.000000 1.000000
7 Ks 0.752941 0.752941 0.752941
8 Ke 0.000000 0.000000 0.000000
9 Ni 1.500000
0 d 1.000000
1 illum 2
2 map_Kd .\bomberbase_baseColor.jpeg
3
4 newmtl bomblanta
5 Ns 600.000000
6 Ka 1.000000 1.000000 1.000000
7 Ks 0.752941 0.752941 0.752941
8 Ke 0.000000 0.000000 0.000000
9 Ni 1.500000
0 d 1.000000
1 illum 2
2 map_Kd .\bomblanta_baseColor.jpeg
3
4 newmtl bombruedafron
5 Ns 600.000000
6 Ka 1.000000 1.000000 1.000000
7 Ks 0.752941 0.752941 0.752941
8 Ke 0.000000 0.000000 0.000000
9 Ni 1.500000
0 d 1.000000
1 illum 2
2 map_Kd .\bombruedafron_baseColor.jpeg
3
4 newmtl bombruedaback
```

Hình 38: Chỉnh sửa đường dẫn load ảnh

- Thử mô hình mô phỏng :



Hình 39: Chọn Preview Model



Hình 40: Chọn mô hình



Hình 41: Mô hình xe cứu hỏa

3.2 Phân tích Code

3.2.1 Khai báo biến:

```
double detectionRange = 75 ;//Bán kính phạm vi mà các Drone có thể phát hiện con người (targets)
std::vector<Ptr<netsimulyzer::XYSeries>> posSeriesVec;

std::vector<Ptr<Socket>> droneSockets;
// Phạm vi tối thiểu và tối đa mà các Drone di chuyển trong không gian 2D và 3D ( ở đây setup mô phỏng map kích thước 500x500(m) )
double minNodePosition = 0;
double maxNodePosition = 500;
// Những giá trị này phải luôn dương (vì RandomDirection2dMobilityModel chỉ chấp nhận các giá trị dương)
double minSpeed = .1;
double maxSpeed = 5;
double duration = 400;// Thời gian chạy mô phỏng (tính bằng giây)
/// Thiết bị được sử dụng trong kịch bản
NetDeviceContainer devices;
NetDeviceContainer dronesnet;

/// Giao diện được sử dụng trong kịch bản
Ipv4InterfaceContainer interfaces;
Ipv4InterfaceContainer interfaces1;
```

Hình 42: Các biến cho kịch bản

```
std::string outputFileName = "UAV-1.json";
// Tập đầu ra dùng để lưu thông tin mô phỏng trong định dạng JSON, sử dụng bởi NetSimulyzer để trực quan hóa.
//Các biến này lưu trữ đường dẫn đến các mô hình 3D đại diện cho các đối tượng trong mô phỏng
std::string nui = netsimulyzer::models::NUI1;// mô hình núi
std::string cothusong = netsimulyzer::models::CELL_TOWER;// mô hình cột thu sóng
std::string thamrung = netsimulyzer::models::THAMNUI;// mô hình thảm cỏ rừng
std::string rung = netsimulyzer::models::RUNG;
std::string droneModelPath = netsimulyzer::models::QUADCOPTER_UAV;// mô hình UAV
std::string phoneModelPath = netsimulyzer::models::XECUUTHUONG;// mô hình xe ô tô (tượng trưng cho ground staion)
std::string targets1 = netsimulyzer::models::NGUOI3D1;// mô hình lego (đại diện con người cần tìm kiếm cứu hộ)
```

Hình 43: Mô hình cho NetSimulyzer

3.2.2 Khai báo số lượng các node:

- 1 phones ứng với 1 ground station, 3 targets ứng với 3 con người, số lượng drones là 25:

```
// ---- Nodes ----

NodeContainer phones;
phones.Create(1);

NodeContainer targets;
targets.Create(3);

NodeContainer drones;
drones.Create(25);
```

Hình 44: Số lượng node

3.2.3 Cài đặt wifi và định tuyến cho các node:

```
// Cấu hình Wifi và công suất phát tín hiệu
WifiMacHelper wifiMac;
wifiMac.SetType("ns3::AdhocWifiMac");
YansWifiPhyHelper wifiPhy;
YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default();
wifiPhy.SetChannel(wifiChannel.Create());
WifiHelper wifi;
wifiPhy.Set("TxPowerStart", DoubleValue(30.0)); // Công suất phát tối thiểu (dBm)
wifiPhy.Set("TxPowerEnd", DoubleValue(30.0)); // Công suất phát tối đa (dBm)
// Cấu hình quản lý tốc độ truyền cố định
// Sử dụng tốc độ truyền cố định là 6 Mbps
// Kích hoạt RTS/CTS để giảm xung đột truyền gói tin
wifi.SetRemoteStationManager("ns3::ConstantRateWifiManager",
    "DataMode",
    StringValue("OfdmRate6Mbps"),
    "RtsCtsThreshold",
    UIntegerValue(0));
// Gắn Wifi lên Drone và Ground Station
dronesnet = wifi.Install(wifiPhy, wifiMac, drones);
devices = wifi.Install(wifiPhy, wifiMac, phones);
```

Hình 45: Cài đặt wifi lên node

```
AodvHelper aodv;
// Cài đặt giao thức định tuyến AODV
InternetStackHelper stack;
InternetStackHelper stack1;
stack.SetRoutingHelper(aodv);
stack1.SetRoutingHelper(aodv);
stack.Install(drones); // Cài đặt AODV lên Drone
stack1.Install(phones); // Cài đặt AODV lên Ground Station
// Cấu hình địa chỉ IPv4
Ipv4AddressHelper address;
address.SetBase("10.0.0.0", "255.255.224.0");
// Phân bổ địa chỉ IP cho Drone và Ground Station
interfaces = address.Assign(devices);
interfaces1 = address.Assign(dronesnet);
```

Hình 46: Gán IP cho các Node

3.2.4 Tạo Socket giao tiếp:

```
// Tạo Socket UDP cho Ground Station
Ptr<Socket> serverSocket = Socket::CreateSocket(phones.Get(0), UdpSocketFactory::GetTypeId());
InetSocketAddress localAddr(Ipv4Address::GetAny(), 8080); // Lắng nghe trên tất cả các IP
serverSocket->Bind(localAddr);
serverSocket->SetRecvCallback(MakeCallback(&ReceivePacket));

for (uint32_t i = 0; i < drones.GetN(); ++i) {
    // Tạo Socket UDP cho Drone
    Ptr<Socket> droneSocket = Socket::CreateSocket(drones.Get(i), UdpSocketFactory::GetTypeId());
    InetSocketAddress serverAddr(Ipv4Address("10.0.0.1"), 8080); // Địa chỉ Ground Station và cổng
    droneSocket->Bind(); // Bắt buộc để socket lắng nghe
    droneSocket->Connect(serverAddr); // Kết nối đến Ground Station

    // Lưu socket vào vector
    droneSockets.push_back(droneSocket);
}
```

Hình 47: Tạo Socket cho Ground Station và Drone

3.2.5 Cài đặt vị trí, mô hình di chuyển cho các node:

```
//Cài đặt min và max cho dài giá trị vị trí
auto positionAllocator = CreateObject<RandomBoxPositionAllocator>();
auto positionStream = CreateObject<UniformRandomVariable>();
positionStream->SetAttribute("Min", DoubleValue(minNodePosition));
positionStream->SetAttribute("Max", DoubleValue(maxNodePosition));

//Cài đặt vị trí ban đầu vào biến positionAllocator
positionAllocator->SetX(positionStream);
positionAllocator->SetY(positionStream);
positionAllocator->SetAttribute("Z", StringValue("ns3::ConstantRandomVariable[Constant=0.0]"));
```

Hình 48: Các vị trí là ngẫu nhiên

```
// Cấu hình mô hình Drone di chuyển
MobilityHelper mobility;
mobility.SetMobilityModel("ns3::GaussMarkovMobilityModel",
    "Bounds", BoxValue(Box(0, maxNodePosition, 0, maxNodePosition, 20, 40)),
    "TimeStep", TimeValue(Seconds(0.5)),
    "Alpha", DoubleValue(0.85),
    "MeanVelocity", StringValue("ns3::UniformRandomVariable[Min=10|Max=20]"),
    "MeanDirection", StringValue("ns3::UniformRandomVariable[Min=0|Max=6.283185307]"),
    "MeanPitch", StringValue("ns3::UniformRandomVariable[Min=0.05|Max=0.05]"),
    "NormalVelocity", StringValue("ns3::NormalRandomVariable[Mean=0|Variance=0.1|Bound=0.4]"),
    "NormalDirection", StringValue("ns3::NormalRandomVariable[Mean=0.0|Variance=0.2|Bound=0.4]"),
    "NormalPitch", StringValue("ns3::NormalRandomVariable[Mean=0.0|Variance=0.02|Bound=0.04]")
);
//Cài đặt vị trí ban đầu và di chuyển vào drones
mobility.SetPositionAllocator(positionAllocator);
mobility.Install(drones);
// Mô hình Ground Station và con người sẽ đứng cố định
MobilityHelper mobility1;
MobilityHelper mobility2;
//Ground Station sẽ được đặt cố định và có vị trí mặc định là (0,0,0)
mobility1.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility1.Install(phones);
//Cài đặt vị trí ban đầu là ngẫu nhiên cho người và không di chuyển
mobility2.SetPositionAllocator(positionAllocator);
mobility2.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility2.Install(targets);
```

Hình 49: Cấu hình di chuyển cho mô hình

```
//Thực hiện ghi lại sự dịch chuyển của các Node để thông báo ra NetSimulyzer
for (auto iter = NodeList::Begin(); iter != NodeList::End(); iter++)
{
    auto m = (*iter)->GetObject<MobilityModel>();
    if (!m)
        continue;
    //Sẽ được gọi lại mỗi khi di chuyển
    m->TraceConnectWithoutContext("CourseChange", MakeCallback(&CourseChanged));
}
```

Hình 50: Ghi vị trí các Node

3.2.6 Mô hình cho NetSimulyzer:

```
// ---- NetSimulyzer ----
// Tạo một đối tượng Orchestrator để quản lý các mô hình đồ họa trong NetSimulyzer
auto orchestrator = CreateObject<netsimulyzer::Orchestrator>(outputFileName);
```

Hình 51: Đối tượng chính xuất file .json

- Đánh dấu vùng giới hạn của kịch bản:

```
// Đánh dấu khu vực khả thi cho các Node
auto possibleNodeLocations = CreateObject<netsimulyzer::RectangularArea>(
    orchestrator,
    Rectangle(minNodePosition, maxNodePosition, minNodePosition, maxNodePosition));

// Đặt tên khu vực
possibleNodeLocations->SetAttribute("Name", StringValue("Possible Node Locations"));

// Đặt màu sắc cho khu vực
possibleNodeLocations->SetAttribute("FillColor", netsimulyzer::Color3Value{204u, 255u, 204u});
```

Hình 52: Giới hạn trong phạm vi hình chữ nhật

- Ghi log các thông tin của Drones:

```
//Ghi log các thông tin cấu hình cơ bản của kịch bản (như vị trí node, tốc độ node).
auto infoLog = CreateObject<netsimulyzer::LogStream>(orchestrator);
//Ghi log các sự kiện trong quá trình mô phỏng.
eventLog = CreateObject<netsimulyzer::LogStream>(orchestrator);

// Ghi thông tin cấu hình vào log
*infoLog << "----- Scenario Settings ----- \n";
*infoLog << "Node Position Range: [" << minNodePosition << ',' << maxNodePosition << "] \n";
*infoLog << "Node Speed Range: [" << minSpeed << ',' << maxSpeed << "] \n";
*infoLog << "Models: Phone [" << rung << "], Drone [" << droneModelPath << "] \n";
*infoLog << "Scenario Duration (Seconds): " << duration << '\n';
```

Hình 53: Ghi log NetSimulyzer

```
//Bật tính năng hiển thị vết di chuyển của các node trong giao diện Netsimulyzer.
netsimulyzer::NodeConfigurationHelper nodeConfigHelper(orchestrator);
nodeConfigHelper.Set("EnableMotionTrail", BooleanValue(true));
```

Hình 54: Để lại vết sáng khi di chuyển trong NetSimulyzer

- Chính các thông số cho background mô phỏng gồm nhiều ngọn núi:

```
// Cấu hình và cài đặt mô hình cho map đồi núi (49 đồi núi nằm liền kề nhau với các độ cao khác nhau )
auto decoration1 = CreateObject<netsimulyzer::Decoration>(orchestrator);
decoration1->SetAttribute("Model", StringValue(nui));
decoration1->SetAttribute("Scale", StringValue("1.97"));
decoration1->SetAttribute("Position", Vector3DValue(Vector(0,75,0)));
auto decoration2 = CreateObject<netsimulyzer::Decoration>(orchestrator);
decoration2->SetAttribute("Model", StringValue(nui));
decoration2->SetAttribute("Scale", StringValue("1.39"));
decoration2->SetAttribute("Position", Vector3DValue(Vector(0,150,0)));
```

Hình 55: Tạo nhiều ngọn núi phủ đầy phạm vi map

- Cấu hình trang trí cho các vật thể (Scale đại diện cho độ lớn nhỏ):

```

// Cấu hình và cài đặt mô hình cho cột thu sóng (2 cột thu sóng)
auto decoration289=CreateObject<netsimulyzer::Decoration>(orchestrator);
decoration289->SetAttribute("Model", StringValue(cotthusong));
decoration289->SetAttribute("Scale", StringValue("0.25"));
decoration289->SetAttribute("Position", Vector3DValue(Vector(-10,-8,0)));
auto decoration290=CreateObject<netsimulyzer::Decoration>(orchestrator);
decoration290->SetAttribute("Model", StringValue(cotthusong));
decoration290->SetAttribute("Scale", StringValue("0.25"));
decoration290->SetAttribute("Position", Vector3DValue(Vector(-10,-4,0)));
// Cấu hình và cài đặt mô hình cho thảm cỏ
auto decoration291=CreateObject<netsimulyzer::Decoration>(orchestrator);
decoration291->SetAttribute("Model", StringValue(thamrung));
decoration291->SetAttribute("Position", Vector3DValue(Vector(250,250,0)));

// Cấu hình và cài đặt mô hình cho Drone
nodeConfigHelper.Set("Model", StringValue(droneModelPath));
nodeConfigHelper.Set("Scale", StringValue("2"));
nodeConfigHelper.Install(drones);
// Cấu hình và cài đặt mô hình cho Ground Station
nodeConfigHelper.Set("Model", StringValue(phoneModelPath));
nodeConfigHelper.Set("Scale", StringValue("0.35"));
nodeConfigHelper.Install(phones);
// Cấu hình và cài đặt mô hình con người (đại diện con người cần tìm kiếm)
nodeConfigHelper.Set("Model", StringValue(targets1));
nodeConfigHelper.Set("Scale", StringValue("3"));
nodeConfigHelper.Install(targets);

```

Hình 56: Mô hình, kích cỡ, vị trí của các Node

3.2.7 Theo dõi năng lượng:

```

// Mô hình năng lượng
BasicEnergySourceHelper energySourceHelper;
energySourceHelper.Set("BasicEnergySourceInitialEnergyJ", DoubleValue(450));
energySourceHelper.Set("PeriodicEnergyUpdateInterval", TimeValue(Seconds(1)));
EnergySourceContainer sources = energySourceHelper.Install(drones);
EnergySourceContainer sources1 = energySourceHelper.Install(phones);
WifiRadioEnergyModelHelper radioEnergyHelper;
// Cài đặt giá trị năng lượng bị giảm khi truyền gói tin
radioEnergyHelper.Set ("TxCurrentA", DoubleValue (0.0174));
// Cài đặt vào các thiết bị
DeviceEnergyModelContainer deviceModels = radioEnergyHelper.Install (devices, sources1);
DeviceEnergyModelContainer deviceModels1 = radioEnergyHelper.Install (dronesnet, sources);

```

Hình 57: Năng lượng giảm 0.0174 mỗi lần truyền

3.2.8 Theo dõi luồng dữ liệu:

```
// Cài đặt FlowMonitor
FlowMonitorHelper flowmon;
Ptr<FlowMonitor> monitor = flowmon.InstallAll();
// Kiểm tra các gói tin bị mất
monitor->CheckForLostPackets();
```

Hình 58: Biến để theo dõi luồng dữ liệu

```
// Truy xuất và phân tích lưu lượng
Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier>(flowmon.GetClassifier());
std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats();
```

Hình 59: Truy xuất và phân tích luồng

```
// Ghi lại dữ liệu hiệu năng của các luồng
monitor->SerializeToXmlFile ("firstanim_flowmetrics-1.xml", true, true);
// ...
```

Hình 60: Ghi dữ liệu thành file

3.2.9 NetAnim:

```
// Ghi dữ liệu với NetAnim
AnimationInterface anim("Simulation-1.xml");
anim.EnablePacketMetadata(true); // Ghi dữ liệu gói tin
// Mô tả ghi chú cho Ground Station, Drone và con người cần tìm kiếm
anim.UpdateNodeDescription(0, "Ground Station");
anim.UpdateNodeColor(0, 0, 255, 0);
anim.UpdateNodeDescription(1, "Human 1");
anim.UpdateNodeColor(1, 0, 0, 255);
anim.UpdateNodeDescription(2, "Human 2");
anim.UpdateNodeColor(2, 0, 0, 255);
anim.UpdateNodeDescription(3, "Human 3");
anim.UpdateNodeColor(3, 0, 0, 255);
for (uint32_t i = 0; i < drones.GetN(); ++i) {
    anim.UpdateNodeDescription(i+4, "Drone "+std::to_string(i+1));
}
```

Hình 61: Chính các mô tả cho đối tượng trong 2D

```
// Theo dõi và lưu trữ dữ liệu bảng định tuyến
anim.EnableIpv4RouteTracking ("firstanim_routetable-1.xml", Seconds (0), Seconds (100), Seconds (0.25));
anim.EnableIpv4L3ProtocolCounters (Seconds (0), Seconds (10));
anim.EnableQueueCounters (Seconds (0), Seconds (10));
anim.EnableWifiMacCounters(Seconds(0), Seconds(10)); // Optional
anim.EnableWifiPhyCounters(Seconds(0), Seconds(10)); // Optional
```

Hình 62: Theo dõi và lưu các thông tin cản thiết

3.2.10 WireShark:

```
// Lưu trữ dữ liệu mô phỏng với PCAP để phân tích bằng Wireshark
wifiPhy.EnablePcap("wifiPackets-1", devices.Get(0));
```

Hình 63: Bật theo dõi bằng Wireshark

3.2.11 Vẽ biểu đồ:

- Đường đi của các Drones:

```
// Theo dõi vị trí di động của các Drone để hiện thị biểu đồ lịch sử đường di chuyển của Drone
for (uint32_t i = 0; i < drones.GetN(); ++i) {
    Ptr<netsimulyzer::XYSeries> posSeries = CreateObject<netsimulyzer::XYSeries>(orchestrator);
    posSeries->SetAttribute("Name", StringValue("Drone " + std::to_string(i+1) + " position"));
    posSeries->SetAttribute("LabelMode", StringValue("Hidden"));
    posSeries->SetAttribute("Color", netsimulyzer::BLUE_VALUE);
    posSeries->GetXAxis()->SetAttribute("Name", StringValue("X position (m)"));
    posSeries->GetYAxis()->SetAttribute("Name", StringValue("Y position (m)"));
    posSeriesVec.push_back(posSeries);
}

for (size_t i = 0; i < posSeriesVec.size(); ++i) {
    Ptr<netsimulyzer::XYSeries> posSeries = posSeriesVec[i];

    // Gắn callback cho node hiện tại
    Config::Connect((" NodeList/" + std::to_string(i+4) + "/$ns3::MobilityModel/CourseChange").c_str(),
                    MakeBoundCallback(&CourseChanged1, posSeries));
}
```

Hình 64: Gọi callback để ghi vị trí liên tục

- Biểu đồ phát hiện gói tin đến Ground Station:

```
// Đặt callback nhận gói tin

serverRxTraceSeries = CreateObject<netsimulyzer::ThroughputSink>(orchestrator, "Rx Server");
serverRxTraceSeries->SetAttribute("Unit", StringValue("Mb/s"));
serverRxTraceSeries->SetAttribute("Interval", TimeValue(Seconds(0.2)));
PointerValue rxXySeries1;
serverRxTraceSeries->GetAttribute("XYSeries", rxXySeries1);
rxXySeries1.Get<netsimulyzer::XYSeries>()->SetAttribute("LabelMode", StringValue("Hidden"));
rxXySeries1.Get<netsimulyzer::XYSeries>()->SetAttribute("Color", netsimulyzer::RED_VALUE);
```

Hình 65: Cài đặt hiển thị đồ gói tin đến

3.2.12 Các hàm con:

- Hàm ghi log di chuyển của Drones:

```
// Hàm ghi lại sự thay đổi hướng di chuyển của một Drone
void CourseChanged(Ptr<const MobilityModel> model)
{
    const auto nodeId = model->GetObject<Node>()->GetId(); // Lấy thông tin id của Drone
    const auto position = model->GetPosition(); // Lấy thông tin vị trí của Drone
    const auto velocity = model->GetVelocity(); // Lấy thông tin vận tốc của Drone

    // Ghi log các thông tin về thời điểm, vị trí và vận tốc của node khi có thay đổi trong NetSimulyzer.
    *eventLog << Simulator::Now().GetMilliSeconds() << ": Node [" << nodeId
    << "] Thay đổi hướng di chuyển - Vị trí: [" << position.x << ", " << position.y << ", "
    << position.z << "] "
    << "với vận tốc: [" << abs(velocity.x) << ", " << abs(velocity.y) << ", " << abs(velocity.z) << "]\n";
}
```

Hình 66: Hàm ghi log di chuyển

- Hàm ghi vị trí của Drones sang biểu đồ:

```
// Hàm cập nhật dữ liệu vị trí của node trên biểu đồ tọa độ
void CourseChanged1(Ptr<netsimulyzer::XYSeries> posSeries, std::string context, Ptr<const MobilityModel> model)
{
    const auto position = model->GetPosition(); // Lấy vị trí hiện tại của Drone từ mô hình di chuyển
    // Thêm điểm mới vào chuỗi tọa độ posSeries để hiển thị sự di chuyển của Drone trên biểu đồ.
    posSeries->Append(position.x, position.y);
}
```

Hình 67: Hàm được gọi callback

- Hàm phát hiện người và gửi gói tin:
 - + Hàm sẽ được gọi sau 0.1s mô phỏng:

```
// Mô phỏng phát hiện tại thời điểm khác nhau
Simulator::Schedule(Seconds(0.1), &DetectAndReport, droneSockets, drones, targets, detectionRange);
```

Hình 68: Lên lịch gọi hàm

- + Cấu trúc, biến lưu trữ:

```
// Cấu trúc của các thông tin cần lưu trữ và gửi đi
struct DetectionInfo {
    double x;
    double y;
    double z;
    int64_t timeMs;
};

// Tập hợp các vị trí đã được phát hiện
std::set<std::string> detectedPositions;
int biendem=1;
```

Hình 69: Cấu trúc lưu trữ thông tin

- + Hàm chính:
 - Chạy 2 vòng for lồng nhau để duyệt qua drone và target => Tính khoảng cách giữa drone và target => Nếu bé hơn tầm phát hiện của drone => Tiến hành tạo gói tin và gửi cho Ground Station

```
// Hàm phát hiện con người (targets) trong phạm vi của Drone và gửi thông tin phát hiện qua socket.
void DetectAndReport(std::vector<Ptr<Socket>> droneSockets, NodeContainer drones, NodeContainer targets, double detectionRange) {
    //Vòng lặp qua các drone
    for (NodeContainer::Iterator droneIt = drones.Begin(); droneIt != drones.End(); ++droneIt) {
        Ptr<Node> drone = *droneIt;
        Vector dronePosition = drone->GetObject<MobilityModel>()->GetPosition(); //Lấy thông tin vị trí hiện tại của Drone
        uint32_t index = std::distance(drones.Begin(), droneIt);
        Ptr<Socket> droneSocket = droneSockets[index];
        Ptr<Ipv4> ipv4 = drone->GetObject<Ipv4>();
        Ipv4Address droneIp = ipv4->GetAddress(1, 0).GetLocal(); // Lấy thông tin địa chỉ IP của Drone
        std::ostringstream oss;
        oss << droneIp;
        std::string ipString = oss.str();

        //Vòng lặp qua các con người cần tìm kiếm (targets)
        for (uint32_t i = 0; i < targets.GetN(); ++i) {
            Ptr<Node> target = targets.Get(i);
            Vector targetPosition = target->GetObject<MobilityModel>()->GetPosition(); // Lấy thông tin vị trí của con người
            double distance = CalculateDistance(dronePosition, targetPosition); // Tính khoảng cách giữa Drone và con người
            if (distance <= detectionRange) {
                // Ghi nhận phát hiện
                // Tạo chuỗi biểu diễn vị trí để kiểm tra
                std::ostringstream positionKey;
                positionKey << targetPosition.x << "," << targetPosition.y << "," << targetPosition.z;
                std::string positionStr = positionKey.str();
                // Tạo gói dữ liệu
                if (detectedPositions.find(positionStr) == detectedPositions.end()) {
                    // Ghi nhận phát hiện
                    uint32_t droneId = drone->GetId();
                    Time detectionTime = Simulator::Now();
                    DetectionInfo info = {targetPosition.x, targetPosition.y, targetPosition.z, detectionTime.GetMilliSeconds()};
                    detectedPositions.insert(positionStr);
                    biendum++;
                }
            }
        }
    }
}
```

Hình 70: Tính khoảng cách giữa drone với con người

- Tạo gói dữ liệu để gửi cho Ground Station. Hàm gọi lại chính nó sau 3s:

```
// Tạo gói dữ liệu
if (detectedPositions.find(positionStr) == detectedPositions.end()) {
    // Ghi nhận phát hiện
    uint32_t droneId = drone->GetId();
    Time detectionTime = Simulator::Now();
    DetectionInfo info = {targetPosition.x, targetPosition.y, targetPosition.z, detectionTime.GetMilliSeconds()};

    std::string message = "Drone " + std::to_string(droneId-3) + " (IP: " + ipString +
        ") phát hiện người thứ " + std::to_string(biendum) + " tại (" +
        std::to_string(info.x) + ", " +
        std::to_string(info.y) + ", " +
        std::to_string(info.z) + ") lúc " +
        std::to_string(info.timeMs) + "ms";
    NS_LOG_UNCOND(message);
    // Tạo gói tin từ nội dung thông điệp và gửi qua socket của Drone
    Ptr<Packet> packet = Create<ns3::Packet>((uint8_t*)message.c_str(), message.size());
    droneSocket->Send(packet);
    //NS_LOG_UNCOND(message);
    detectedPositions.insert(positionStr); // Kiểm tra vị trí con người đó đã được phát hiện chưa, nếu chưa thì ghi nhận
    biendum++;
}
}

// Hàm được tái thực thi sau mỗi 3 giây để phát hiện liên tục
Simulator::Schedule(Seconds(3), &DetectAndReport, droneSockets, drones, targets, detectionRange);
}
```

Hình 71: Gọi lại chính nó sau 3s

- Hàm xử lý gói tin đến của Ground Station:

```
//Hàm nhận và xử lý các gói tin được gửi đến ground station
void ReceivePacket(Ptr<Socket> socket)
{
    // Kiểm tra nếu gói tin không rỗng
    while (Ptr<Packet> packet = socket->Recv())
    {
        uint32_t packetSize = packet->GetSize(); // Lấy kích thước gói tin
        serverRxTraceSeries->AddPacketSize(packetSize);
        if (packetSize > 0) {
            // Tạo bộ đệm để sao chép dữ liệu
            char buffer[packetSize + 1]; // Thêm 1 ký tự để kết thúc chuỗi
            packet->CopyData((uint8_t*)buffer, packetSize); // Sao chép dữ liệu vào bộ đệm
            buffer[packetSize] = '\0'; // Đảm bảo kết thúc chuỗi

            // In thông báo nhận được
            NS_LOG_UNCOND("Server đã nhận được thông tin: " << buffer);
        }
    }
}
```

Hình 72: Hàm ReceivePacket

- Hàm ghi log năng lượng vào file energy_log-1.txt:

```
std::ofstream energyLogFile("energy_log-1.txt");
// Hàm ghi log năng lượng còn lại của các drone định kỳ
void LogEnergy(NodeContainer nodes)
{
    energyLogFile << '\n' << Simulator::Now().GetSeconds() << "s :\n";
    for (NodeContainer::Iterator it = nodes.Begin(); it != nodes.End(); ++it)
    {
        Ptr<Node> node = *it;
        // Lấy nguồn năng lượng của Drone
        Ptr<BasicEnergySource> energySource = DynamicCast<BasicEnergySource>(
            node->GetObject<EnergySourceContainer>()->Get(0));

        if (energySource)// Nếu nguồn năng lượng hợp lệ(khác nullptr)
        {
            double remainingEnergy = energySource->GetRemainingEnergy();
            energyLogFile << "Drone " << node->GetId()-3 << " Năng lượng còn lại = "
            << remainingEnergy << "J" << std::endl;
        }
    }
    // Lên lịch lặp ghi tiếp theo
    Simulator::Schedule(Seconds(1), &LogEnergy, nodes);
}
```

Hình 73: Hàm LogEnergy được gọi lại mỗi 1s

3.3 Triển khai kịch bản

- Lần lượt tiến hành kiểm sự hiệu quả của quá trình cứu hộ khi thay đổi các thông số: **số lượng UAV, kích thước map** và **công suất phát**
- Thay đổi số lượng UAV:

```
NodeContainer drones;
drones.Create(25);
```

Hình 74: Số lượng Drone

- Thay đổi kích thước map:

```
double minNodePosition = 0;
double maxNodePosition = 500;
```

Hình 75: Kích thước map

- Thay đổi công suất phát:

```
wifiPhy.Set("TxPowerStart", DoubleValue(30.0)); // Công suất phát tối thiểu (dBm)
wifiPhy.Set("TxPowerEnd", DoubleValue(30.0)); // Công suất phát tối đa (dBm)
```

Hình 76: Công suất phát

- VIDEO DEMO:

https://drive.google.com/drive/folders/1_Y3eg3PeTFBN7-dqN3kZNlmgIFF5z59W

3.3.1 Kịch bản 1: Sự khác nhau về số lượng UAV

Số lượng UAV	10	25	50
Tốc độ UAV	10m/s-20m/s	10m/s-20m/s	10m/s-20m/s
Kích thước map	500x500(m)	500x500(m)	500x500(m)
Công suất phát	30dBm	30dBm	30dBm

Bảng 3: Thông tin kịch bản 1

3.3.2 Kịch bản 2: Sự khác nhau kích thước map

Số lượng UAV	100	100	100
Tốc độ UAV	10m/s-20m/s	10m/s-20m/s	10m/s-20m/s
Kích thước map	1250x1250(m)	1500x1500(m)	2000x2000(m)
Công suất phát	40dBm	40dBm	40dBm

Bảng 4: Thông tin kịch bản 2

3.3.3 Kịch bản 3: Sự khác nhau về công suất phát

Số lượng UAV	25	25	25
Tốc độ UAV	10m/s-20m/s	10m/s-20m/s	10m/s-20m/s

Kích thước map	750x750(m)	750x750(m)	750x750(m)
Công suất phát	20dBm	45dBm	75dBm

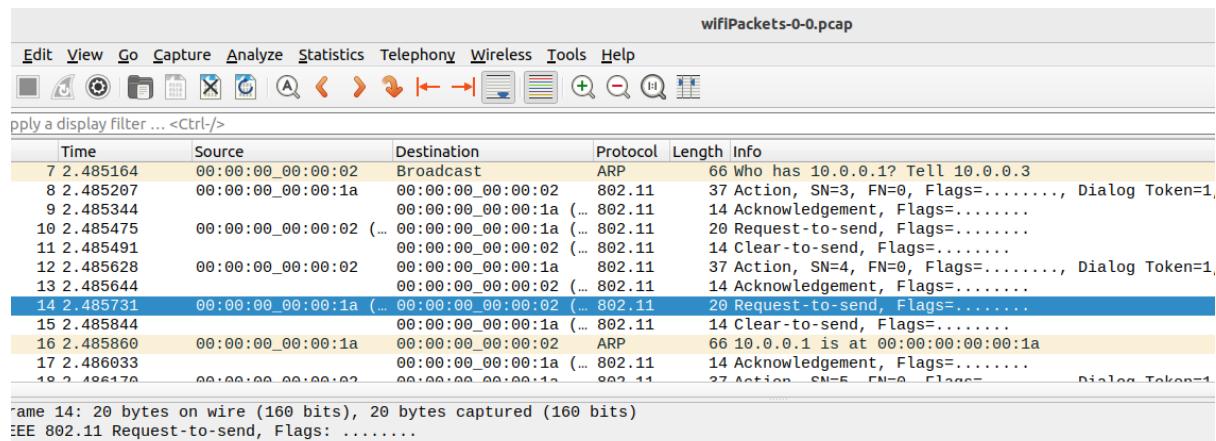
Bảng 5: Thông tin kích bản 3



CHƯƠNG IV. KẾT QUẢ VÀ PHÂN TÍCH

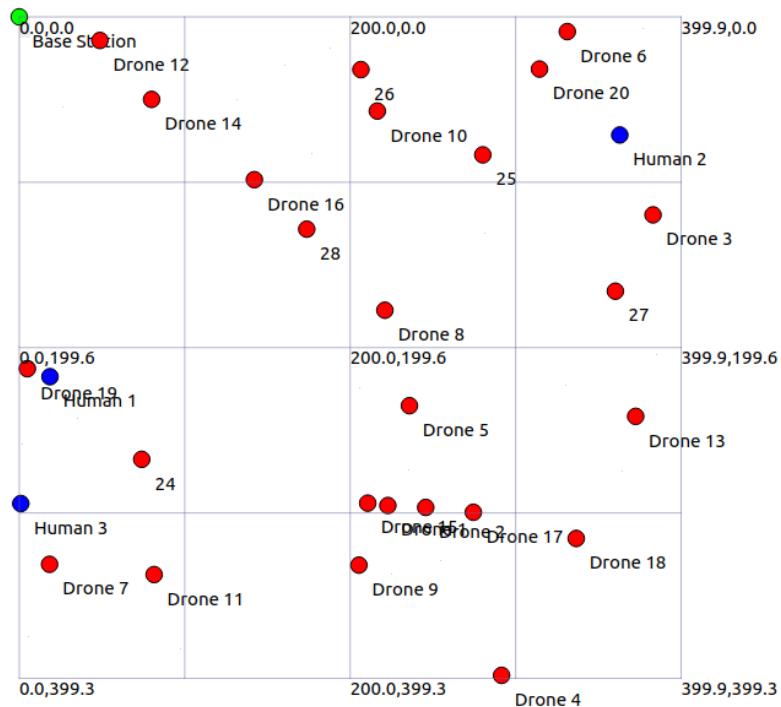
4.1 Phân tích dữ liệu dựa trên các tool đã cài đặt

4.1.1 Wireshark để phân tích thông tin chi tiết các gói tin:



Hình 77: Wireshak

4.1.2 NetAnim để quan sát dưới góc nhìn 2D:



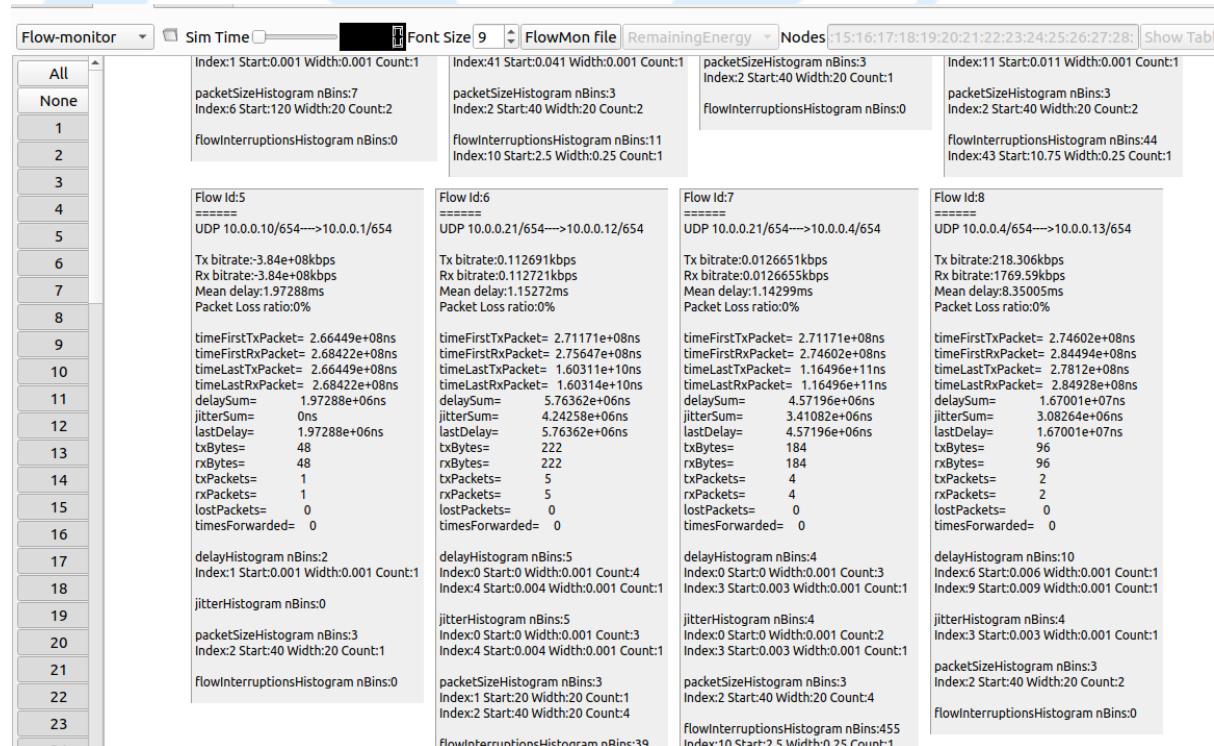
Hình 78: NetAnim

4.1.3 Kiểm tra thông tin mạng (IP,MAC) của các thiết bị trong NetAnim:

Node:9 IP: 10.0.0.2 127.0.0.1 IPv6: ::1 MAC: 00:00:00:00:00:01	Node:10 IP: 10.0.0.3 127.0.0.1 IPv6: ::1 MAC: 00:00:00:00:00:02	Node:11 IP: 10.0.0.4 127.0.0.1 IPv6: ::1 MAC: 00:00:00:00:00:03	Node:12 IP: 127.0.0.1 10.0.0.5 IPv6: ::1 MAC: 00:00:00:00:00:04	Node:13 IP: 10.0.0.6 127.0.0.1 IPv6: ::1 MAC: 00:00:00:00:00:05	Node:14 IP: 10.0.0.7 127.0.0.1 IPv6: ::1 MAC: 00:00:00:00:00:06
Node:15 IP: 127.0.0.1 10.0.0.8 IPv6: ::1 MAC: 00:00:00:00:00:07	Node:16 IP: 127.0.0.1 10.0.0.9 IPv6: ::1 MAC: 00:00:00:00:00:08	Node:17 IP: 127.0.0.1 10.0.0.10 IPv6: ::1 MAC: 00:00:00:00:00:09	Node:18 IP: 127.0.0.1 10.0.0.11 IPv6: ::1 MAC: 00:00:00:00:00:0a	Node:19 IP: 127.0.0.1 10.0.0.12 IPv6: ::1 MAC: 00:00:00:00:00:0b	Node:20 IP: 10.0.0.13 127.0.0.1 IPv6: ::1 MAC: 00:00:00:00:00:0c
Node:21 IP: 127.0.0.1 10.0.0.14 IPv6: ::1 MAC: 00:00:00:00:00:0d	Node:22 IP: 127.0.0.1 10.0.0.15 IPv6: ::1 MAC: 00:00:00:00:00:0e	Node:23 IP: 10.0.0.16 127.0.0.1 IPv6: ::1 MAC: 00:00:00:00:00:0f	Node:24 IP: 10.0.0.17 127.0.0.1 IPv6: ::1 MAC: 00:00:00:00:00:10	Node:25 IP: 10.0.0.18 127.0.0.1 IPv6: ::1 MAC: 00:00:00:00:00:11	Node:26 IP: 127.0.0.1 10.0.0.19 IPv6: ::1 MAC: 00:00:00:00:00:12
Node:27 IP: 10.0.0.20 127.0.0.1 IPv6: ::1 MAC: 00:00:00:00:00:13	Node:28 IP: 127.0.0.1 10.0.0.21 IPv6: ::1 MAC: 00:00:00:00:00:14				

Hình 79: IP, MAC

4.1.4 Phân tích luồng dữ liệu Flowmonitor:



Hình 80: FlowMonitor

⇒ Các port /654 là do aodv tự tạo luồng giao tiếp để giao tiếp giữa các thiết bị với nhau

Flow Probes:

```

Index:0
FlowId:1 Packets:2 Bytes:240 DelayFromFirstProbeSum:6.08334e+08ns
FlowId:3 Packets:1 Bytes:48 DelayFromFirstProbeSum:6.96655e+06ns
FlowId:5 Packets:1 Bytes:48 DelayFromFirstProbeSum:1.97288e+06ns
FlowId:19 Packets:1 Bytes:123 DelayFromFirstProbeSum:2.89993e+08ns
FlowId:21 Packets:6 Bytes:272 DelayFromFirstProbeSum:1.61459e+07ns
FlowId:83 Packets:1 Bytes:30 DelayFromFirstProbeSum:0ns

Index:3

Index:4
FlowId:10 Packets:1 Bytes:122 DelayFromFirstProbeSum:6.91394e+08ns
FlowId:16 Packets:11 Bytes:472 DelayFromFirstProbeSum:1.82e+07ns
FlowId:17 Packets:7 Bytes:294 DelayFromFirstProbeSum:0ns
FlowId:34 Packets:1 Bytes:48 DelayFromFirstProbeSum:1.2892e+06ns
FlowId:35 Packets:1 Bytes:48 DelayFromFirstProbeSum:4.90114e+06ns
FlowId:36 Packets:2 Bytes:96 DelayFromFirstProbeSum:0ns

Index:5

Index:6
FlowId:1 Packets:2 Bytes:240 DelayFromFirstProbeSum:5.87385e+08ns
FlowId:7 Packets:4 Bytes:184 DelayFromFirstProbeSum:4.57196e+06ns
FlowId:8 Packets:2 Bytes:96 DelayFromFirstProbeSum:0ns
FlowId:10 Packets:1 Bytes:122 DelayFromFirstProbeSum:7.07116e+08ns
FlowId:12 Packets:1 Bytes:48 DelayFromFirstProbeSum:1.84828e+06ns
FlowId:13 Packets:1 Bytes:48 DelayFromFirstProbeSum:0ns
FlowId:14 Packets:3 Bytes:136 DelayFromFirstProbeSum:0ns
FlowId:18 Packets:2 Bytes:96 DelayFromFirstProbeSum:1.86246e+06ns
FlowId:49 Packets:0 Bytes:0 DelayFromFirstProbeSum:0ns
FlowId:51 Packets:2 Bytes:88 DelayFromFirstProbeSum:2.01084e+09ns
FlowId:53 Packets:3 Bytes:136 DelayFromFirstProbeSum:9.49219e+06ns
FlowId:54 Packets:1 Bytes:48 DelayFromFirstProbeSum:1.00286e+09ns
FlowId:56 Packets:2 Bytes:70 DelayFromFirstProbeSum:0ns
FlowId:57 Packets:4 Bytes:184 DelayFromFirstProbeSum:0ns
FlowId:58 Packets:1 Bytes:30 DelayFromFirstProbeSum:0ns
FlowId:35 Packets:1 Bytes:48 DelayFromFirstProbeSum:4.90114e+06ns
FlowId:36 Packets:2 Bytes:96 DelayFromFirstProbeSum:0ns

Index:5

Index:6
FlowId:1 Packets:2 Bytes:240 DelayFromFirstProbeSum:5.87385e+08ns
FlowId:7 Packets:4 Bytes:184 DelayFromFirstProbeSum:4.57196e+06ns
FlowId:8 Packets:2 Bytes:96 DelayFromFirstProbeSum:0ns
FlowId:10 Packets:1 Bytes:122 DelayFromFirstProbeSum:7.07116e+08ns

```

Hình 81: Flow Probes

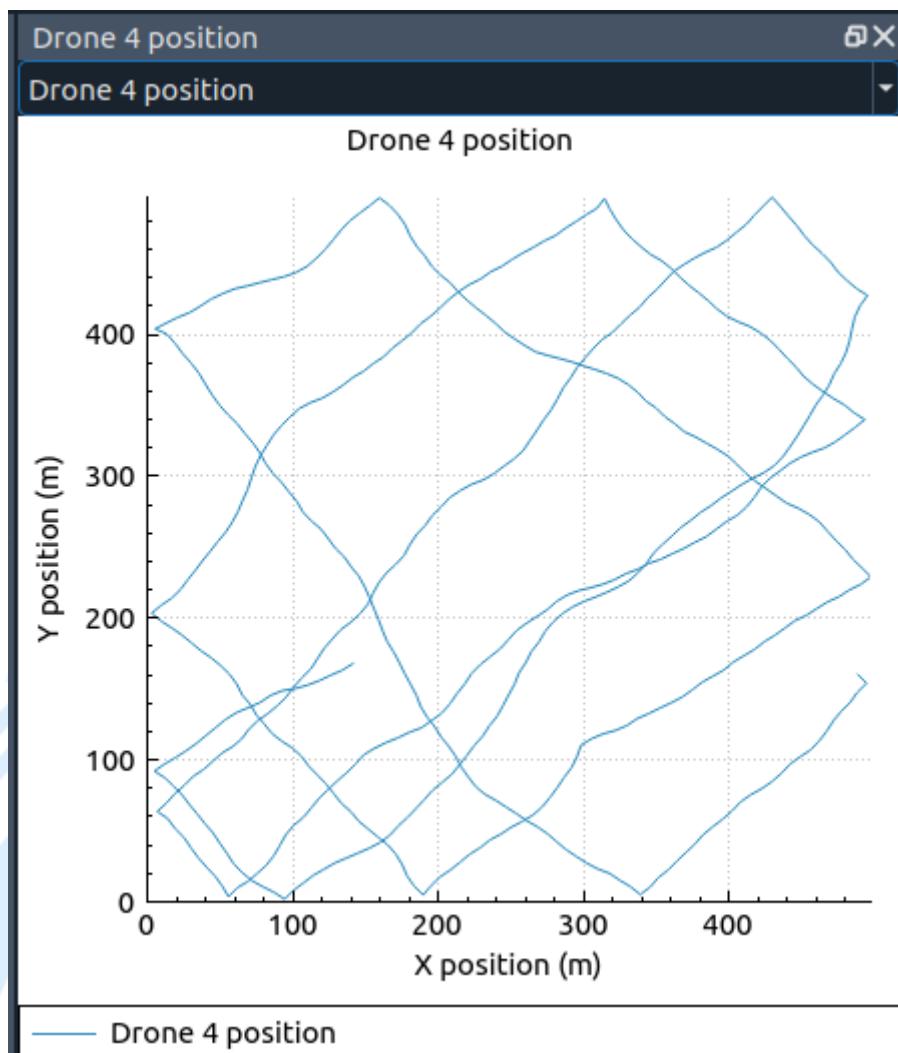
4.1.5 Bảng định tuyến theo thời gian:

Routing ▾ Sim Time Font Size 8 FlowMon file RemainingEnergy Nodes 15:16:17:18:19:20:21:22:23:24:25:26:27:28

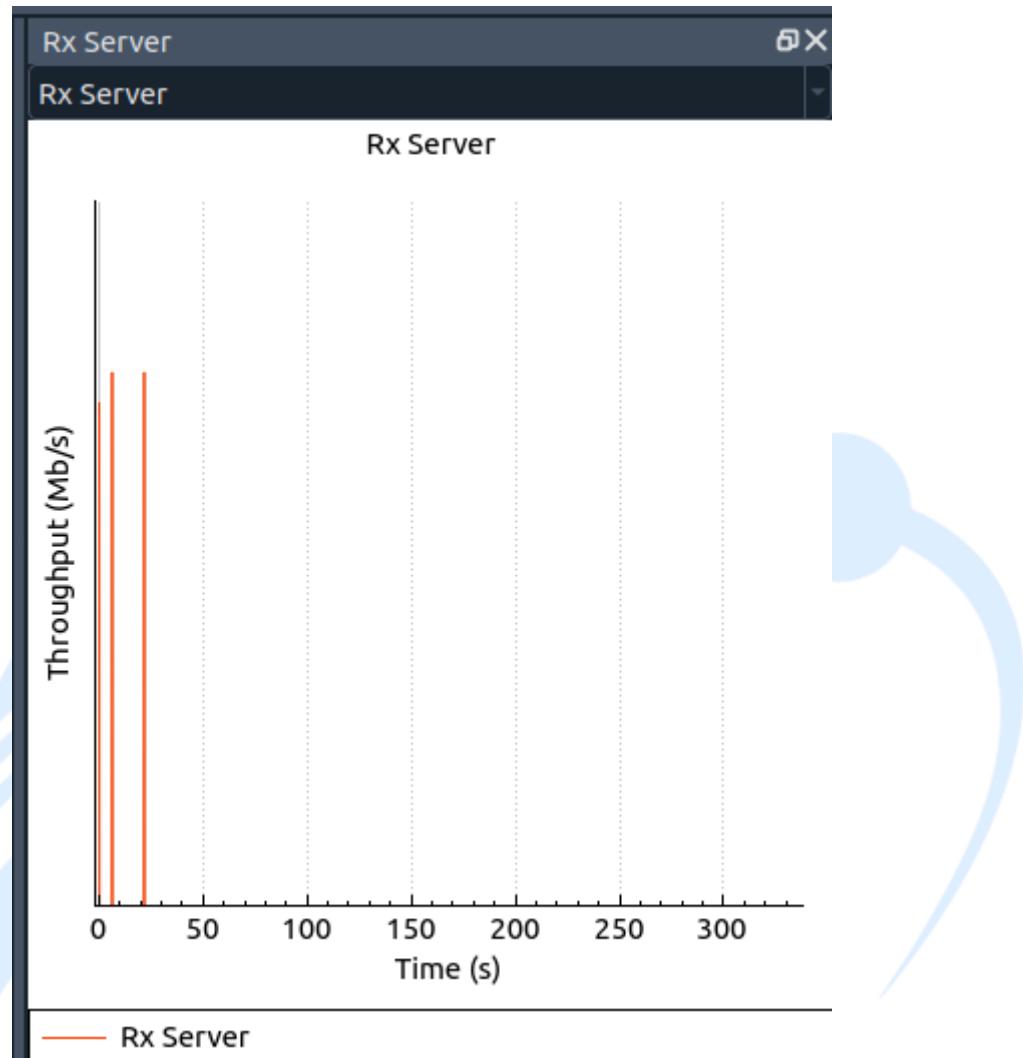
Node	Time	Local time	AODV Routing table
0	+100s	+100s	Destination Gateway Interface Flag Expire Hops 10.0.0.8 10.0.0.8 10.0.0.1 DOWN +9.3s 1 10.0.0.14 10.0.0.14 10.0.0.1 DOWN +9.3s 1 10.0.0.15 10.0.0.15 10.0.0.1 UP +2.3s 1 10.0.31.255 10.0.31.255 10.0.0.1 UP +9.2e+09s 1 127.0.0.1 127.0.0.1 127.0.0.1 UP +9.2e+09s 1
4	+100s	+100s	Destination Gateway Interface Flag Expire Hops 10.0.0.2 10.0.0.2 10.0.0.3 UP +2.3s 1 10.0.0.4 10.0.0.4 10.0.0.3 UP +4.3s 1 10.0.0.6 10.0.0.6 10.0.0.3 UP +5.8s 1 10.0.0.7 10.0.0.7 10.0.0.3 DOWN +11s 1 10.0.0.10 10.0.0.10 10.0.0.3 DOWN +4.8s 1 10.0.0.12 10.0.0.12 10.0.0.3 UP +10s 1
8	+100s	+100s	Destination Gateway Interface Flag Expire Hops 10.0.0.4 10.0.0.4 10.0.0.5 DOWN +0.29s 1 10.0.0.6 10.0.0.6 10.0.0.5 UP +2.3s 1 10.0.0.7 10.0.0.7 10.0.0.5 UP +2.3s 1 10.0.0.10 10.0.0.10 10.0.0.5 UP +2.3s 1 10.0.0.17 10.0.0.17 10.0.0.5 UP +2.3s 1 10.0.0.19 10.0.0.19 10.0.0.5 UP +2.3s 1
12	+100s	+100s	Destination Gateway Interface Flag Expire Hops 10.0.0.2 10.0.0.2 10.0.0.7 DOWN +12s 1 10.0.0.3 10.0.0.3 10.0.0.7 DOWN +12s 1 10.0.0.4 10.0.0.4 10.0.0.7 UP +2.3s 1 10.0.0.5 10.0.0.5 10.0.0.7 UP +2.3s 1 10.0.0.6 10.0.0.6 10.0.0.7 UP +2.3s 1 10.0.0.10 10.0.0.10 10.0.0.7 UP +9.3s 1
16	+100s	+100s	Destination Gateway Interface Flag Expire Hops 10.0.0.1 10.0.0.1 10.0.0.8 DOWN +9.3s 1 10.0.0.2 10.0.0.2 10.0.0.8 UP +2.3s 1 10.0.0.11 10.0.0.11 10.0.0.8 UP +2.3s 1 10.0.0.12 10.0.0.12 10.0.0.8 UP +2.3s 1 10.0.0.13 10.0.0.13 10.0.0.8 UP +2.3s 1 10.0.0.14 10.0.0.14 10.0.0.8 DOWN +6.3s 1
20	+100s	+100s	Destination Gateway Interface Flag Expire Hops 10.0.0.3 10.0.0.3 10.0.0.10 UP +2.3s 1 10.0.0.4 10.0.0.4 10.0.0.10 UP +2.3s 1 10.0.0.5 10.0.0.5 10.0.0.10 UP +2.3s 1 10.0.0.6 10.0.0.6 10.0.0.10 UP +2.3s 1 10.0.0.7 10.0.0.7 10.0.0.10 UP +2.3s 1 10.0.0.8 10.0.0.8 10.0.0.10 UP +2.3s 1 10.0.0.9 10.0.0.9 10.0.0.10 UP +2.3s 1 10.0.0.10 10.0.0.10 10.0.0.10 UP +2.3s 1

Hình 82: Routing Table

4.1.6 Vẽ biểu đồ để phân tích trong NetSimulyzer:

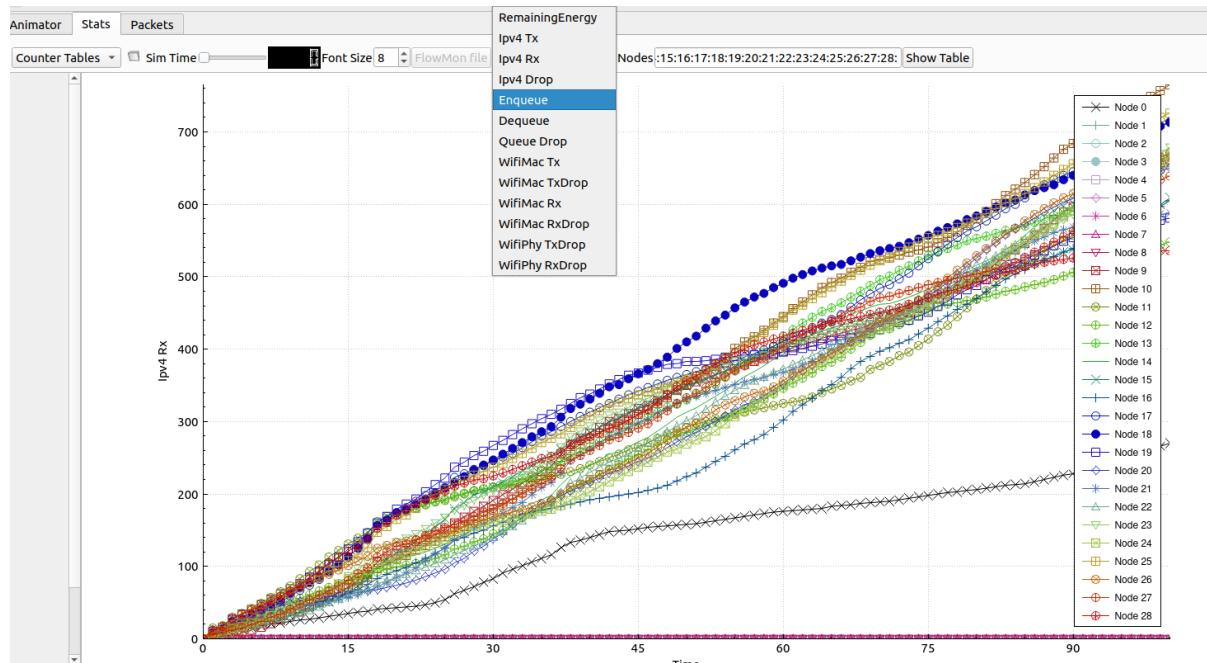


Hình 83: Chart(biểu đồ ghi lại lịch sử đường bay của từng Drone)



Hình 84: Chart(biểu đồ ghi lại luồng dữ liệu mà trạm mặt đất nhận được từ Drone)

4.1.7 Các thông tin khác:



Hình 85: Thông tin khác

4.2 Kịch bản 1: Sự khác nhau về số lượng UAV

Số lượng UAV	10	25	50
Tốc độ UAV	10m/s-20m/s	10m/s-20m/s	10m/s-20m/s
Kích thước map	500x500(m)	500x500(m)	500x500(m)
Công suất phát	30dBm	30dBm	30dBm

Bảng 6: Bảng thông tin kịch bản 1

4.2.1 Số lượng 10 UAV:

```
phuc@phuc-virtual-machine:~/ns-allinone-3.38/ns-3.38$ ./ns3 run scratch/UAV-1.cc
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ..../build/scratch/ns3.38-UAV-1-default
Bắt đầu tìm kiếm cứu hộ !!!
Drone 7 (IP: 10.0.0.8) phát hiện người thứ 1 tại (158.785657, 25.591753, 0.000000) lúc 24100ms
Server đã nhận được thông tin: Drone 7 (IP: 10.0.0.8) phát hiện người thứ 1 tại (158.785657, 25.591753, 0.000000) lúc 24100ms
Drone 5 (IP: 10.0.0.6) phát hiện người thứ 2 tại (479.206158, 177.523766, 0.000000) lúc 45100ms
Drone 3 (IP: 10.0.0.4) phát hiện người thứ 3 tại (74.792043, 334.492799, 0.000000) lúc 108100ms
Kết thúc thời gian tìm kiếm !!!
phuc@phuc-virtual-machine:~/ns-allinone-3.38/ns-3.38$
```

Hình 86: Kết quả thu được 10 UAV

⇒ Kết quả thu được: Drone 7, 5 và 3 phát hiện được người nhưng chỉ có 1 gói tin đầu tiên Ground Station nhận được.

Flow Id:1	Flow Id:2	Flow Id:3
=====	=====	=====
UDP 10.0.0.8/49153--->10.0.0.1/8080	UDP 10.0.0.6/49153--->10.0.0.1/8080	UDP 10.0.0.4/49153--->10.0.0.1/8080
Tx bitrate:-1.064e+09kbps	Tx bitrate:-1.072e+09kbps	Tx bitrate:-1.072e+09kbps
Rx bitrate:-1.064e+09kbps	Rx bitrate:0kbps	Rx bitrate:0kbps
Mean delay:5.05196ms	Mean delay:-1ms	Mean delay:-1ms
Packet Loss ratio:0%	Packet Loss ratio:100%	Packet Loss ratio:100%
timeFirstTxPacket= 2.41e+10ns	timeFirstTxPacket= 4.51e+10ns	timeFirstTxPacket= 1.081e+11ns
timeFirstRxPacket= 2.41051e+10ns	timeFirstRxPacket= 0ns	timeFirstRxPacket= 0ns
timeLastTxPacket= 2.41e+10ns	timeLastTxPacket= 4.51e+10ns	timeLastTxPacket= 1.081e+11ns
timeLastRxPacket= 2.41051e+10ns	timeLastRxPacket= 0ns	timeLastRxPacket= 0ns
delaySum= 5.05196e+06ns	delaySum= 0ns	delaySum= 0ns
jitterSum= 0ns	jitterSum= 0ns	jitterSum= 0ns
lastDelay= 5.05196e+06ns	lastDelay= 0ns	lastDelay= 0ns
txBytes= 133	txBytes= 134	txBytes= 134
rxBytes= 133	rxBytes= 0	rxBytes= 0
txPackets= 1	txPackets= 1	txPackets= 1
rxPackets= 1	rxPackets= 0	rxPackets= 0
lostPackets= 0	lostPackets= 1	lostPackets= 1
timesForwarded= 0	timesForwarded= 0	timesForwarded= 0
delayHistogram nBins:6	Packets Dropped:	Packets Dropped:
Index:5 Start:0.005 Width:0.001 Count:1	No Route:0	No Route:0
jitterHistogram nBins:0	TTL Expire:0	TTL Expire:0
packetSizeHistogram nBins:7	Bad Checksum:0	Bad Checksum:0
Index:6 Start:120 Width:20 Count:1	Queue:0	Queue:0
flowInterruptionsHistogram nBins:0	Interface Down:0	Interface Down:0
	Route error:0	Route error:0
	Fragment timeout:1	Fragment timeout:1

Hình 87: Flow Monitor 10 UAV

⇒ Phân tích luồng dữ liệu có thể 2 gói tin có thông số lostPackets = 1.

- **Nhận xét:** Kết quả trên là do số lượng Drone quá ít dẫn đến gói tin không tìm được đường đi để đi về Ground Station. Bên cạnh đó, thời gian phát hiện đủ 3 người tương đối lâu (108.1s).

4.2.2 Số lượng 25 UAV:

```
phuc@phuc-virtual-machine:~/ns-allinone-3.38/ns-3.38$ ./ns3 run scratch/UAV-1.cc
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ..../build/scratch/ns-3.38-UAV-1-default
Bắt đầu tìm kiếm cứu hộ !!!
Drone 2 (IP: 10.0.0.3) phát hiện người thứ 1 tại (27.062139, 148.745664, 0.000000) lúc 100ms
Server đã nhận được thông tin: Drone 2 (IP: 10.0.0.3) phát hiện người thứ 1 tại (27.062139, 148.745664, 0.000000) lúc 100ms
Drone 7 (IP: 10.0.0.8) phát hiện người thứ 2 tại (369.423980, 209.494317, 0.000000) lúc 6100ms
Server đã nhận được thông tin: Drone 7 (IP: 10.0.0.8) phát hiện người thứ 2 tại (369.423980, 209.494317, 0.000000) lúc 6100ms
Drone 20 (IP: 10.0.0.21) phát hiện người thứ 3 tại (415.096529, 39.339081, 0.000000) lúc 24100ms
Server đã nhận được thông tin: Drone 20 (IP: 10.0.0.21) phát hiện người thứ 3 tại (415.096529, 39.339081, 0.000000) lúc 24100ms
Kết thúc thời gian tìm kiếm !!!
```

Hình 88: Kết quả thu được 25 UAV

⇒ Kết quả thu được: Các drone tìm được cả 3 người và 3 gói tin đều truyền được về cho Ground Station.

Flow Id:1	Flow Id:2	Flow Id:3	Flow Id:4
==	=====	=====	=====
10.0.0.3/49153--->10.0.0.1/8080	UDP 10.0.0.8/49153--->10.0.0.1/8080	UDP 10.0.0.3/654--->10.0.0.19/654	UDP 10.0.0.3/654--->10.0.0.1/654
rate:-1.048e+09kbps	Tx bitrate:-1.064e+09kbps	Rx bitrate:-0.07255kbps	Tx bitrate:0.189862kbps
rate:-1.048e+09kbps	Rx bitrate:-1.064e+09kbps	Rx bitrate:0.072566kbps	Rx bitrate:0.189858kbps
\delay:6.00342ms	Mean delay:279.699ms	Mean delay:1.35064ms	Mean delay:0.336365ms
Packet Loss ratio:0%	Packet Loss ratio:0%	Packet Loss ratio:0%	Packet Loss ratio:0%
FirstTxPacket= 1e+08ns	timeFirstTxPacket= 6.1e+09ns	timeFirstTxPacket= 6.35545e+09ns	timeFirstTxPacket= 6.35557e+09ns
FirstRxPacket= 1.06003e+08ns	timeFirstRxPacket= 6.3797e+09ns	timeFirstRxPacket= 6.35789e+09ns	timeFirstRxPacket= 6.35576e+09ns
LastTxPacket= 1e+08ns	timeLastTxPacket= 6.1e+09ns	timeLastTxPacket= 1.06591e+10ns	timeLastTxPacket= 1.00634e+10ns
LastRxPacket= 1.06003e+08ns	timeLastRxPacket= 6.3797e+09ns	timeLastRxPacket= 1.60594e+10ns	timeLastRxPacket= 1.00638e+10ns
rSum= 6.00342e+06ns	delaySum= 2.79699e+08ns	delaySum= 2.70127e+06ns	delaySum= 672730ns
Sum= 0ns	jitterSum= 0ns	jitterSum= 2.17874e+06ns	jitterSum= 48090ns
\elay= 6.00342e+06ns	lastDelay= 2.79699e+08ns	lastDelay= 2.70127e+06ns	lastDelay= 672730ns
txBytes= 131	txBytes= 133	txBytes= 88	txBytes= 88
rxBytes= 131	rxBytes= 133	rxBytes= 88	rxBytes= 88
txPackets= 1	txPackets= 1	txPackets= 2	txPackets= 2
rxPackets= 1	rxPackets= 1	rxPackets= 2	rxPackets= 2
ackets= 0	lostPackets= 0	lostPackets= 0	lostPackets= 0
timesForwarded= 0	timesForwarded= 4	timesForwarded= 0	timesForwarded= 0
Histogram nBins:7	delayHistogram nBins:280	delayHistogram nBins:3	delayHistogram nBins:1
c:6 Start:0.006 Width:0.001 Count:1	Index:279 Start:0.279 Width:0.001 Count:1	Index:0 Start:0 Width:0.001 Count:1	Index:0 Start:0 Width:0.001 Count:2
Histogram nBins:0	jitterHistogram nBins:0	Index:2 Start:0.002 Width:0.001 Count:1	jitterHistogram nBins:1
etSizeHistogram nBins:7	packetSizeHistogram nBins:7	packetSizeHistogram nBins:3	packetSizeHistogram nBins:3
c:6 Start:120 Width:20 Count:1	Index:6 Start:120 Width:20 Count:1	Index:2 Start:40 Width:20 Count:1	Index:2 Start:40 Width:20 Count:2
nterruptionsHistogram nBins:0	flowInterruptionsHistogram nBins:0	flowInterruptionsHistogram nBins:39	flowInterruptionsHistogram nBins:15
		Index:38 Start:9.5 Width:0.25 Count:1	Index:14 Start:3.5 Width:0.25 Count:1

Hình 89: Flow Monitor 25 UAV

⇒ Luồng dữ liệu cũng cho thấy không bị mất gói tin nào.

- **Nhận xét:** Do số lượng Drone phù hợp nên quá trình mô phỏng cứu hộ hoàn thành mà không bị mất gói tin. Thời gian để Drone phát hiện cả 3 người ở mức trung bình (24,1s).

4.2.3 Số lượng 50 UAV:

```
phuc@phuc-virtual-machine:~/ns-allinone-3.38/ns-3.38$ ./ns3 run scratch/UAV-1.cc
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ../build/scratch/ns3.38-UAV-1-default
Bắt đầu tìm kiếm cứu hộ !!!
Drone 12 (IP: 10.0.0.13) phát hiện người thứ 1 tại (169.154770, 333.373043, 0.000000) lúc 100ms
Drone 15 (IP: 10.0.0.16) phát hiện người thứ 2 tại (126.486646, 385.278640, 0.000000) lúc 100ms
Drone 31 (IP: 10.0.0.32) phát hiện người thứ 3 tại (255.564464, 471.607639, 0.000000) lúc 100ms
Server đã nhận được thông tin: Drone 15 (IP: 10.0.0.16) phát hiện người thứ 2 tại (126.486646, 385.278640, 0.000000) lúc 100ms
Server đã nhận được thông tin: Drone 12 (IP: 10.0.0.13) phát hiện người thứ 1 tại (169.154770, 333.373043, 0.000000) lúc 100ms
Server đã nhận được thông tin: Drone 31 (IP: 10.0.0.32) phát hiện người thứ 3 tại (255.564464, 471.607639, 0.000000) lúc 100ms
Max Packets per trace file exceeded
Kết thúc thời gian tìm kiếm !!!
```

Hình 90: Kết quả thu được 50 UAV

⇒ Kết quả thu được: Cả 3 người đều được phát hiện và số gói tin truyền được thành công là 3/3.

Flow Id:1 ===== UDP 10.0.0.13/49153--->10.0.0.1/8080 Tx bitrate:-1.072e+09kbps Rx bitrate:-1.072e+09kbps Mean delay:311.87ms Packet Loss ratio:0% timeFirstTxPacket= 1e+08ns timeFirstRxPacket= 4.1187e+08ns timeLastTxPacket= 1e+08ns timeLastRxPacket= 4.1187e+08ns delaySum= 3.1187e+08ns jitterSum= 0ns lastDelay= 3.1187e+08ns txBytes= 134 rxBytes= 134 txPackets= 1 rxPackets= 1 lostPackets= 0 timesForwarded= 4 delayHistogram nBins:312 Index:311 Start:0.311 Width:0.001 Count:1 jitterHistogram nBins:0 packetSizeHistogram nBins:7 Index:6 Start:120 Width:20 Count:1 flowInterruptionsHistogram nBins:0	Flow Id:2 ===== UDP 10.0.0.16/49153--->10.0.0.1/8080 Tx bitrate:-1.072e+09kbps Rx bitrate:-1.072e+09kbps Mean delay:294.471ms Packet Loss ratio:0% timeFirstTxPacket= 1e+08ns timeFirstRxPacket= 3.94471e+08ns timeLastTxPacket= 1e+08ns timeLastRxPacket= 3.94471e+08ns delaySum= 2.94471e+08ns jitterSum= 0ns lastDelay= 2.94471e+08ns txBytes= 134 rxBytes= 134 txPackets= 1 rxPackets= 1 lostPackets= 0 timesForwarded= 4 delayHistogram nBins:295 Index:294 Start:0.294 Width:0.001 Count:1 jitterHistogram nBins:0 packetSizeHistogram nBins:7 Index:6 Start:120 Width:20 Count:1 flowInterruptionsHistogram nBins:0	Flow Id:3 ===== UDP 10.0.0.32/49153--->10.0.0.1/8080 Tx bitrate:-1.072e+09kbps Rx bitrate:-1.072e+09kbps Mean delay:685.842ms Packet Loss ratio:0% timeFirstTxPacket= 1e+08ns timeFirstRxPacket= 7.85842e+08ns timeLastTxPacket= 1e+08ns timeLastRxPacket= 7.85842e+08ns delaySum= 6.85842e+08ns jitterSum= 0ns lastDelay= 6.85842e+08ns txBytes= 134 rxBytes= 134 txPackets= 1 rxPackets= 1 lostPackets= 0 timesForwarded= 6 delayHistogram nBins:686 Index:685 Start:0.685 Width:0.001 Count:1 jitterHistogram nBins:0 packetSizeHistogram nBins:7 Index:6 Start:120 Width:20 Count:1 flowInterruptionsHistogram nBins:0	Flow Id:4 ===== UDP 10.0.0.51/654--->10.0.0.26/654 Tx bitrate:-3.84e+08kbps Rx bitrate:-3.84e+08kbps Mean delay:8.1ms Packet Loss ratio:0% timeFirstTxPacket= 3.52668e+08ns timeFirstRxPacket= 3.60768e+08ns timeLastTxPacket= 3.52668e+08ns timeLastRxPacket= 3.60768e+08ns delaySum= 8.1e+06ns jitterSum= 0ns lastDelay= 8.1e+06ns txBytes= 48 rxBytes= 48 txPackets= 1 rxPackets= 1 lostPackets= 0 timesForwarded= 0 delayHistogram nBins:9 Index:8 Start:0.008 Width:0.001 Count:1 jitterHistogram nBins:0 packetSizeHistogram nBins:3 Index:2 Start:40 Width:20 Count:1 flowInterruptionsHistogram nBins:0
--	---	---	--

Hình 91: Flow Monitor 50 UAV

⇒ Luồng dữ liệu: Không bị mất gói tin.

- **Nhận xét:** Với số lượng drone lớn thì thời gian phát hiện người (0.1s) và thời gian truyền gói tin cũng gần như ngay lập tức. Mặc dù số lượng drone giúp việc tìm kiếm diễn ra nhanh hơn và chính xác hơn nhưng dẫn đến chi phí cao hơn, tốn tài nguyên hơn.

4.2.4 So sánh

Số lượng UAV	10	25	50
Thời gian phát hiện đủ 3 người	108.1s	24.1s	0.1s
Số người phát hiện	3 người	3 người	3 người
Số gói tin truyền về thành công	1 gói tin	3 gói tin	3 gói tin
Chi phí cho thiết bị	Thấp	Trung bình	Cao

Bảng 7: Bảng so sánh số lượng UAV

4.3 Kích bản 2: Sự khác nhau về kích thước map

Số lượng UAV	100	100	100
--------------	-----	-----	-----

Tốc độ UAV	10m/s-20m/s	10m/s-20m/s	10m/s-20m/s
Kích thước map	1250 x 1250(m)	1500 x 1500(m)	2000 x 2000(m)
Công suất phát	40dBm	40dBm	40dBm

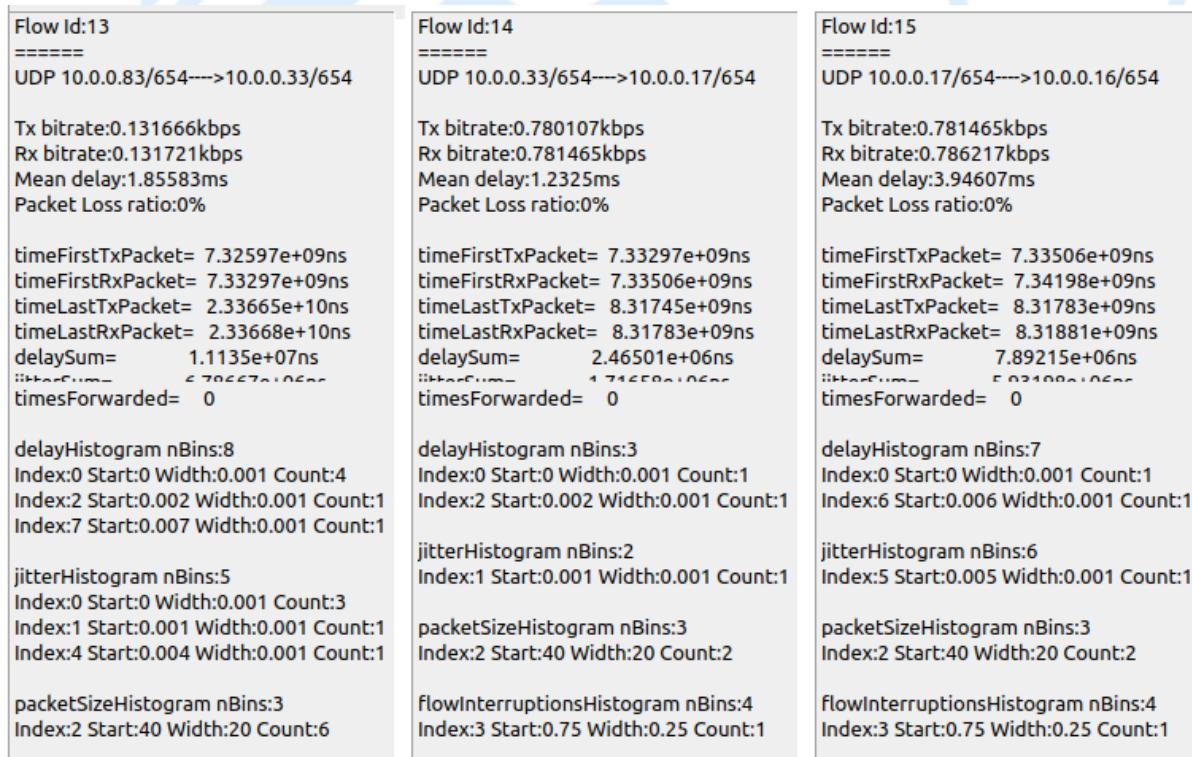
Bảng 8: Bảng thông tin kích bản 2

4.3.1 Kích thước 1250 x 1250 m

```
phuc@phuc-virtual-machine:~/ns-allinone-3.38/ns-3.38$ ./ns3 run scratch/UAV-2.cc
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ../build/scratch/ns3.38-UAV-2-default
Bắt đầu tìm kiếm cứu hộ !!!
Drone 37 (IP: 10.0.0.38) phát hiện người thứ 1 tại (217.659039, 579.748229, 0.000000) lúc 100ms
Server đã nhận được thông tin: Drone 37 (IP: 10.0.0.38) phát hiện người thứ 1 tại (217.659039, 579.748229, 0.000000) lúc 100ms
Drone 91 (IP: 10.0.0.92) phát hiện người thứ 2 tại (1076.616222, 977.946756, 0.000000) lúc 6100ms
Server đã nhận được thông tin: Drone 91 (IP: 10.0.0.92) phát hiện người thứ 2 tại (1076.616222, 977.946756, 0.000000) lúc 6100ms
Drone 75 (IP: 10.0.0.76) phát hiện người thứ 3 tại (588.199399, 575.820727, 0.000000) lúc 12100ms
Server đã nhận được thông tin: Drone 75 (IP: 10.0.0.76) phát hiện người thứ 3 tại (588.199399, 575.820727, 0.000000) lúc 12100ms
Max Packets per trace file exceeded
Kết thúc thời gian tìm kiếm !!!
```

Hình 92: Kết quả thu được 1250 x 1250 m

⇒ Kết quả thu được: Tất cả người đều được tìm thấy, Ground Station nhận được cả 3 gói tin.



Hình 93: Flow Monitor 1250 x 1250 m

⇒ Luồng dữ liệu: Không bị mất gói tin.

- **Nhận xét:** Thời gian tìm đủ 3 người tương đối nhanh (12.1s), quá trình phát hiện người để cứu hộ thành công.

4.3.2 Kích thước 1500 x 1500 m

```
phuc@phuc-virtual-machine:~/ns-allinone-3.38/ns-3.38$ ./ns3 run scratch/UAV-2.cc
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ..../build/scratch/ns3.38-UAV-2-default
Bắt đầu tìm kiếm cứu hộ !!!
Drone 37 (IP: 10.0.0.38) phát hiện người thứ 1 tại (261.190847, 695.697875, 0.000000) lúc 100ms
Server đã nhận được thông tin: Drone 37 (IP: 10.0.0.38) phát hiện người thứ 1 tại (261.190847, 695.697875, 0.000000) lúc 100ms
Drone 91 (IP: 10.0.0.92) phát hiện người thứ 2 tại (1291.939467, 1173.536107, 0.000000) lúc 9100ms
Server đã nhận được thông tin: Drone 91 (IP: 10.0.0.92) phát hiện người thứ 2 tại (1291.939467, 1173.536107, 0.000000) lúc 9100ms
Drone 75 (IP: 10.0.0.76) phát hiện người thứ 3 tại (705.839279, 690.984872, 0.000000) lúc 18100ms
Server đã nhận được thông tin: Drone 75 (IP: 10.0.0.76) phát hiện người thứ 3 tại (705.839279, 690.984872, 0.000000) lúc 18100ms
Max Packets per trace file exceeded
Kết thúc thời gian tìm kiếm !!!
phuc@phuc-virtual-machine:~/ns-allinone-3.38/ns-3.38$
```

Hình 94: Kết quả thu được 1500 x 1500 m

⇒ Kết quả thu được: Tìm được cả 3 người và gửi thành công cả 3 gói tin.

Flow Id:1	Flow Id:2	Flow Id:3	Flow Id:4
<pre>===== UDP 10.0.0.38/49153--->10.0.0.1/8080 Tx bitrate:-1.072e+09bps Rx bitrate:-1.072e+09bps Mean delay:273.8ms Packet Loss ratio:0% timeFirstTxPacket= 1e+08ns timeFirstRxPacket= 3.738e+08ns timeLastTxPacket= 1e+08ns timeLastRxPacket= 3.738e+08ns delaySum= 2.738e+08ns jitterSum= 0ns lastDelay= 2.738e+08ns txBytes= 134 rxBytes= 134 txPackets= 1 rxPackets= 1 lostPackets= 0 timesForwarded= 3 delayHistogram nBins:274 Index:273 Start:0.273 Width:0.001 Count:1 jitterHistogram nBins:0 packetSizeHistogram nBins:7 Index:6 Start:120 Width:20 Count:1 flowInterruptionsHistogram nBins:0</pre>	<pre>===== UDP 10.0.0.55/654--->10.0.0.87/654 Tx bitrate:0.0717704kbps Rx bitrate:0.0718388kbps Mean delay:5.4575ms Packet Loss ratio:0% timeFirstTxPacket= 3.47303e+08ns timeFirstRxPacket= 3.57905e+08ns timeLastTxPacket= 1.10481e+10ns timeLastRxPacket= 1.10485e+10ns delaySum= 1.0915e+07ns jitterSum= 1.02879e+07ns lastDelay= 1.0915e+07ns txBytes= 96 rxBytes= 96 txPackets= 2 rxPackets= 2 lostPackets= 0 timesForwarded= 0 delayHistogram nBins:11 Index:0 Start:0 Width:0.001 Count:1 Index:10 Start:0.01 Width:0.001 Count:1 jitterHistogram nBins:11 Index:10 Start:0.01 Width:0.001 Count:1 packetSizeHistogram nBins:3 Index:2 Start:40 Width:20 Count:2 flowInterruptionsHistogram nBins:43 Index:42 Start:10.5 Width:0.25 Count:1</pre>	<pre>===== UDP 10.0.0.55/654--->10.0.0.1/654 Tx bitrate:0.134124kbps Rx bitrate:0.13416kbps Mean delay:1.12347ms Packet Loss ratio:0% timeFirstTxPacket= 3.47303e+08ns timeFirstRxPacket= 3.56168e+08ns timeLastTxPacket= 3.23177e+10ns timeLastRxPacket= 3.23181e+10ns delaySum= 1.34816e+07ns jitterSum= 9.50049e+06ns lastDelay= 1.34816e+07ns txBytes= 536 rxBytes= 536 txPackets= 12 rxPackets= 12 lostPackets= 0 timesForwarded= 0 delayHistogram nBins:9 Index:0 Start:0 Width:0.001 Count:10 Index:1 Start:0.001 Width:0.001 Count:1 Index:8 Start:0.008 Width:0.001 Count:1 jitterHistogram nBins:8 Index:0 Start:0 Width:0.001 Count:9 Index:1 Start:0.001 Width:0.001 Count:1 Index:7 Start:0.007 Width:0.001 Count:1 packetSizeHistogram nBins:3 Index:2 Start:40 Width:20 Count:12 flowInterruptionsHistogram nBins:43 Index:2 Start:0.5 Width:0.25 Count:1 Index:4 Start:1 Width:0.25 Count:2 Index:7 Start:1.75 Width:0.25 Count:2 Index:8 Start:2 Width:0.25 Count:1</pre>	<pre>===== UDP 10.0.0.87/654--->10.0.0.55/654 Tx bitrate:0.0603899kbps Rx bitrate:0.0603936kbps Mean delay:0.626716ms Packet Loss ratio:0% timeFirstTxPacket= 3.57905e+08ns timeFirstRxPacket= 3.59767e+08ns timeLastTxPacket= 2.63225e+10ns timeLastRxPacket= 2.63228e+10ns delaySum= 3.13358e+06ns jitterSum= 1.61592e+06ns lastDelay= 3.13358e+06ns txBytes= 196 rxBytes= 196 txPackets= 5 rxPackets= 5 lostPackets= 0 timesForwarded= 0 delayHistogram nBins:2 Index:0 Start:0 Width:0.001 Count:4 Index:1 Start:0.001 Width:0.001 Count:1 jitterHistogram nBins:2 Index:0 Start:0 Width:0.001 Count:3 Index:1 Start:0.001 Width:0.001 Count:1 packetSizeHistogram nBins:3 Index:1 Start:20 Width:20 Count:2 Index:2 Start:40 Width:20 Count:3 flowInterruptionsHistogram nBins:43 Index:15 Start:3.75 Width:0.25 Count:1 Index:17 Start:4.25 Width:0.25 Count:1 Index:27 Start:6.75 Width:0.25 Count:1 Index:42 Start:10.5 Width:0.25 Count:1</pre>

Hình 95: Flow Monitor 1500 x 1500 m

⇒ Luồng dữ liệu: Không bị mất gói tin.

- **Nhận xét:** Tương tự kích thước 1250x1250 (m), các drone vẫn hoạt động tốt, truyền gói tin về tới đích mặc dù chậm hơn trong việc phát hiện người (18.1s>12.1s).

4.3.3 Kích thước 2000 x 2000 m

```
phuc@phuc-virtual-machine:~/ns-allinone-3.38/ns-3.38$ ./ns3 run scratch/UAV-2.cc
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ..../build/scratch/ns3.38-UAV-2-default
Bắt đầu tìm kiếm cứu hộ !!!
Drone 37 (IP: 10.0.0.38) phát hiện người thứ 1 tại (348.254462, 927.597167, 0.000000) lúc 100ms
Drone 75 (IP: 10.0.0.76) phát hiện người thứ 2 tại (941.119039, 921.313163, 0.000000) lúc 30100ms
Server đã nhận được thông tin: Drone 75 (IP: 10.0.0.76) phát hiện người thứ 2 tại (941.119039, 921.313163, 0.000000) lúc 30100ms
Drone 20 (IP: 10.0.0.21) phát hiện người thứ 3 tại (1722.585956, 1564.714809, 0.000000) lúc 81100ms
Server đã nhận được thông tin: Drone 20 (IP: 10.0.0.21) phát hiện người thứ 3 tại (1722.585956, 1564.714809, 0.000000) lúc 81100ms
Max Packets per trace file exceeded
Kết thúc thời gian tìm kiếm !!!
phuc@phuc-virtual-machine:~/ns-allinone-3.38/ns-3.38$
```

Hình 96: Kết quả thu được 2000 x 2000 m

⇒ Kết quả thu được: Phát hiện được 3 người, 2 gói tin truyền thành công.

Flow Id:1 =====	Flow Id:2 =====	Flow Id:3 =====	Flow Id:4 =====
UDP 10.0.0.38/49153--->10.0.0.1/8080 Tx bitrate:-1.072e+09kbps Rx bitrate:0kbps Mean delay:1ms Packet Loss ratio:100% timeFirstTxPacket= 1e+08ns timeFirstRxPacket= 0ns timeLastTxPacket= 1e+08ns timeLastRxPacket= 0ns delaySum= 0ns jitterSum= 0ns lastDelay= 0ns txBytes= 134 rxBytes= 0 txPackets= 1 rxPackets= 0 lostPackets= 1 timesForwarded= 0 Packets Dropped: No Route:0 TTL Expire:0 Bad Checksum:0 Queue:0 Interface Down:0 Route error:0 Fragment timeout:1 Bytes Dropped:	UDP 10.0.0.76/49153--->10.0.0.1/8080 Tx bitrate:-1.088e+09kbps Rx bitrate:1.088e+09kbps Mean delay:1294.28ms Packet Loss ratio:0% timeFirstTxPacket= 3.01e+10ns timeFirstRxPacket= 3.13943e+10ns timeLastTxPacket= 3.01e+10ns timeLastRxPacket= 3.13943e+10ns delaySum= 1.29428e+09ns jitterSum= 0ns lastDelay= 1.29428e+09ns txBytes= 136 rxBytes= 136 txPackets= 1 rxPackets= 1 lostPackets= 0 timesForwarded= 8 delayHistogram nBins:1295 Index:1294 Start:1.294 Width:0.001 Count:1 jitterHistogram nBins:0 packetSizeHistogram nBins:7 Index:6 Start:120 Width:20 Count:1 flowInterruptionsHistogram nBins:0	UDP 10.0.0.55/654--->10.0.0.1/654 Tx bitrate:-3.84e+08kbps Rx bitrate:-3.84e+08kbps Mean delay:9.04251ms Packet Loss ratio:0% timeFirstTxPacket= 3.13102e+10ns timeFirstRxPacket= 3.13206e+10ns timeLastTxPacket= 3.53179e+10ns timeLastRxPacket= 3.53183e+10ns delaySum= 1.0769e+07ns jitterSum= 1.00558e+07ns lastDelay= 1.0769e+07ns txBytes= 88 rxBytes= 88 txPackets= 2 rxPackets= 2 lostPackets= 0 timesForwarded= 0 delayHistogram nBins:11 Index:0 Start:0 Width:0.001 Count:1 Index:10 Start:0.01 Width:0.001 Count:1 jitterHistogram nBins:11 Index:10 Start:0.01 Width:0.001 Count:1 packetSizeHistogram nBins:3 Index:2 Start:40 Width:20 Count:2 flowInterruptionsHistogram nBins:16 Index:15 Start:3.75 Width:0.25 Count:1	timeFirstTxPacket= 3.13102e+10ns timeFirstRxPacket= 3.13192e+10ns timeLastTxPacket= 3.13102e+10ns timeLastRxPacket= 3.13192e+10ns delaySum= 9.04251e+06ns jitterSum= 0ns lastDelay= 9.04251e+06ns txBytes= 48 rxBytes= 48 txPackets= 1 rxPackets= 1 lostPackets= 0 timesForwarded= 0 delayHistogram nBins:10 Index:9 Start:0.009 Width:0.001 Count:1 jitterHistogram nBins:0 packetSizeHistogram nBins:3 Index:2 Start:40 Width:20 Count:1 flowInterruptionsHistogram nBins:0

Hình 97: Flow Monitor 2000 x 2000 m

⇒ Luồng dữ liệu: 1 gói tin bị mất trong quá trình truyền.

- **Nhận xét:** Do kích thước map lớn dẫn đến quá trình tìm kiếm và trao đổi thông tin trở nên khó khăn. Điều này khiến cho gói tin đầu tiên không về được Ground Station và thời gian tìm kiếm đủ 3 người tương đối lâu (81.1s).

4.3.4 So sánh

Kích thước map	1250 x 1250(m)	1500 x 1500(m)	2000 x 2000(m)
Thời gian phát hiện đủ 3 người	12.1s	18.1s	81.1s

Số người phát hiện	3 người	3 người	3 người
Số gói tin truyền về thành công	3 gói tin	3 gói tin	2 gói tin

Bảng 9: Bảng so sánh kích thước map

4.4 Kịch bản 3: Sự khác nhau về công suất phát

Số lượng UAV	25	25	25
Tốc độ UAV	10m/s-20m/s	10m/s-20m/s	10m/s-20m/s
Kích thước map	750x750(m)	750x750(m)	750x750(m)
Công suất phát	20dBm	45dBm	75dBm

Bảng 10: Bảng thông tin kịch bản 3

4.4.1 Công suất phát 20 dBm

```
phuc@phuc-virtual-machine:~/ns-allinone-3.38/ns-3.38$ ./ns3 run scratch/UAV-3.cc
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ../../build/scratch/ns3.38-UAV-3-default
Bắt đầu tìm kiếm cứu hộ !!!
Drone 2 (IP: 10.0.0.3) phát hiện người thứ 1 tại (40.593209, 223.118496, 0.000000) lúc 3100ms
Drone 7 (IP: 10.0.0.8) phát hiện người thứ 2 tại (554.135969, 314.241476, 0.000000) lúc 18100ms
Drone 20 (IP: 10.0.0.21) phát hiện người thứ 3 tại (622.644794, 59.008622, 0.000000) lúc 78100ms
Kết thúc thời gian tìm kiếm !!!
phuc@phuc-virtual-machine:~/ns-allinone-3.38/ns-3.38$
```

Hình 98: Kết quả thu được 20 dBm

⇒ Kết quả thu được: Tìm kiếm được 3 người nhưng không gói tin nào được gửi về.

Flow Id:1	Flow Id:2	Flow Id:3	Flow Probes:
=====	=====	=====	Index:0
UDP 10.0.0.3/49153--->10.0.0.1/8080	UDP 10.0.0.8/49153--->10.0.0.1/8080	UDP 10.0.0.21/49153--->10.0.0.1/8080	
Tx bitrate:-1.056e+09 kbps	Tx bitrate:-1.072e+09 kbps	Tx bitrate:-1.08e+09 kbps	Index:1
Rx bitrate:0 kbps	Rx bitrate:0 kbps	Rx bitrate:0 kbps	Index:2
Mean delay:-1ms	Mean delay:-1ms	Mean delay:-1ms	Index:3
Packet Loss ratio:100%	Packet Loss ratio:100%	Packet Loss ratio:100%	Index:4
timeFirstTxPacket= 3.1e+09ns	timeFirstTxPacket= 1.81e+10ns	timeFirstTxPacket= 7.81e+10ns	FlowId:1 Packets:1 Bytes:132 DelayFromFirstProbeSum:0ns
timeFirstRxPacket= 0ns	timeFirstRxPacket= 0ns	timeFirstRxPacket= 0ns	
timeLastTxPacket= 3.1e+09ns	timeLastTxPacket= 1.81e+10ns	timeLastTxPacket= 7.81e+10ns	
timeLastRxPacket= 0ns	timeLastRxPacket= 0ns	timeLastRxPacket= 0ns	
delaySum= 0ns	delaySum= 0ns	delaySum= 0ns	Index:5
jitterSum= 0ns	jitterSum= 0ns	jitterSum= 0ns	Index:6
lastDelay= 0ns	lastDelay= 0ns	lastDelay= 0ns	Index:7
txBytes= 132	txBytes= 134	txBytes= 135	Index:8
rxBytes= 0	rxBytes= 0	rxBytes= 0	Index:9
txPackets= 1	txPackets= 1	txPackets= 1	Index:10
rxPackets= 0	rxPackets= 0	rxPackets= 0	
lostPackets= 1	lostPackets= 1	lostPackets= 1	
timesForwarded= 0	timesForwarded= 0	timesForwarded= 0	
Packets Dropped:	Packets Dropped:	Packets Dropped:	
No Route:0	No Route:0	No Route:0	Index:11
TTL Expire:0	TTL Expire:0	TTL Expire:0	Index:12
Bad Checksum:0	Bad Checksum:0	Bad Checksum:0	Index:13
Queue:0	Queue:0	Queue:0	Index:14
Interface Down:0	Interface Down:0	Interface Down:0	FlowId:2 Packets:1 Bytes:134 DelayFromFirstProbeSum:0ns
Route error:0	Route error:0	Route error:0	
Fragment timeout:1	Fragment timeout:1	Fragment timeout:1	
Bytes Dropped:	Bytes Dropped:	Bytes Dropped:	
No Route:0	No Route:0	No Route:0	Index:15
TTL Expire:0	TTL Expire:0	TTL Expire:0	Index:16
Bad Checksum:0	Bad Checksum:0	Bad Checksum:0	Index:17
Queue:0	Queue:0	Queue:0	Index:18
Interface Down:0	Interface Down:0	Interface Down:0	
Route error:0	Route error:0	Route error:0	
Fragment timeout:132	Fragment timeout:134	Fragment timeout:135	Index:19

Hình 99: Flow Monitor 20 dBm

⇒ Luồng dữ liệu: Cả 3 gói tin đều bị mất.

- **Nhận xét:** Do công suất phát quá thấp dẫn đến phạm vi phát sóng của thiết bị thấp và không xây dựng được đường đi để gói tin về Ground Station.

4.4.2 Công suất phát 45 dBm

```
phuc@phuc-virtual-machine:~/ns-allinone-3.38/ns-3.38$ ./ns3 run scratch/UAV-3.cc
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ../build/scratch/ns3.38-UAV-3-default
Bắt đầu tìm kiếm cứu hộ !!!
Drone 2 (IP: 10.0.0.3) phát hiện người thứ 1 tại (40.593209, 223.118496, 0.000000) lúc 3100ms
Server đã nhận được thông tin: Drone 2 (IP: 10.0.0.3) phát hiện người thứ 1 tại (40.593209, 223.118496, 0.000000) lúc 3100ms
Drone 7 (IP: 10.0.0.8) phát hiện người thứ 2 tại (554.135969, 314.241476, 0.000000) lúc 18100ms
Server đã nhận được thông tin: Drone 7 (IP: 10.0.0.8) phát hiện người thứ 2 tại (554.135969, 314.241476, 0.000000) lúc 18100ms
Drone 20 (IP: 10.0.0.21) phát hiện người thứ 3 tại (622.644794, 59.008622, 0.000000) lúc 78100ms
Server đã nhận được thông tin: Drone 20 (IP: 10.0.0.21) phát hiện người thứ 3 tại (622.644794, 59.008622, 0.000000) lúc 78100ms
Max Packets per trace file exceeded
Kết thúc thời gian tìm kiếm !!!
phuc@phuc-virtual-machine:~/ns-allinone-3.38/ns-3.38$
```

Hình 100: Kết quả thu được 45 dBm

⇒ Kết quả: Tìm được 3 người và cả 3 gói tin Ground Station đều nhận được.

Flow Id:1 ===== UDP 10.0.0.3/49153--->10.0.0.1/8080 Tx bitrate:-1.056e+09kbps Rx bitrate:-1.056e+09kbps Mean delay:5.92479ms Packet Loss ratio:0% timeFirstTxPacket= 3.1e+09ns timeFirstRxPacket= 3.10592e+09ns timeLastTxPacket= 3.1e+09ns timeLastRxPacket= 3.10592e+09ns delaySum= 5.92479e+06ns jitterSum= 0ns lastDelay= 5.92479e+06ns txBytes= 132 rxBytes= 132 txPackets= 1 rxPackets= 1 lostPackets= 0 timesForwarded= 0 delayHistogram nBins:6 Index:5 Start:0.005 Width:0.001 Count:1 jitterHistogram nBins:0 packetSizeHistogram nBins:7 Index:6 Start:120 Width:20 Count:1 flowInterruptionsHistogram nBins:0	Flow Id:2 ===== UDP 10.0.0.8/49153--->10.0.0.1/8080 Tx bitrate:-1.072e+09kbps Rx bitrate:-1.072e+09kbps Mean delay:1012.83ms Packet Loss ratio:0% timeFirstTxPacket= 1.81e+10ns timeFirstRxPacket= 1.91128e+10ns timeLastTxPacket= 1.81e+10ns timeLastRxPacket= 1.91128e+10ns delaySum= 1.01283e+09ns jitterSum= 0ns lastDelay= 1.01283e+09ns txBytes= 134 rxBytes= 134 txPackets= 1 rxPackets= 1 lostPackets= 0 timesForwarded= 2 delayHistogram nBins:1013 Index:1012 Start:1.012 Width:0.001 Count:1 jitterHistogram nBins:0 packetSizeHistogram nBins:7 Index:6 Start:120 Width:20 Count:1 flowInterruptionsHistogram nBins:0	Flow Id:3 ===== UDP 10.0.0.19/654--->10.0.0.8/654 Tx bitrate:0.012818kbps Rx bitrate:0kbps Mean delay-1ms Packet Loss ratio:100% timeFirstTxPacket= 1.81052e+10ns timeFirstRxPacket= 0ns timeLastTxPacket= 7.30281e+10ns timeLastRxPacket= 0ns delaySum= 0ns jitterSum= 0ns lastDelay= 0ns txBytes= 88 rxBytes= 0 txPackets= 2 rxPackets= 0 lostPackets= 2 timesForwarded= 0 delayHistogram nBins:0 jitterHistogram nBins:0 packetSizeHistogram nBins:0 flowInterruptionsHistogram nBins:0	Flow Id:4 ===== UDP 10.0.0.19/654--->10.0.0.1/654 Tx bitrate:-3.84e+08kbps Rx bitrate:-3.84e+08kbps Mean delay:14.469ms Packet Loss ratio:0% timeFirstTxPacket= 1.81052e+10ns timeFirstRxPacket= 1.81196e+10ns timeLastTxPacket= 1.81052e+10ns timeLastRxPacket= 1.81196e+10ns delaySum= 1.4469e+07ns jitterSum= 0ns lastDelay= 1.4469e+07ns txBytes= 48 rxBytes= 48 txPackets= 1 rxPackets= 1 lostPackets= 0 timesForwarded= 0 delayHistogram nBins:15 Index:14 Start:0.014 Width:0.001 Count:1 jitterHistogram nBins:0 packetSizeHistogram nBins:3 Index:2 Start:40 Width:20 Count:1 flowInterruptionsHistogram nBins:0
--	--	---	--

Hình 101: Flow Monitor 45 dBm

⇒ Luồng dữ liệu: Không bị mất gói tin nào.

- **Nhận xét:** Công suất phát thích hợp giúp việc giao tiếp trở nên dễ dàng hơn nên các gói tin có thể tìm ra đường đi để về Ground Station.

4.4.3 Công suất phát 75 dBm

```
phuc@phuc-virtual-machine:~/ns-allinone-3.38/ns-3.38$ ./ns3 run scratch/UAV-3.cc
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ..../build/scratch/ns-3.38-UAV-3-default
Bắt đầu tìm kiếm cứu hộ !!!
Drone 2 (IP: 10.0.0.3) phát hiện người thứ 1 tại (40.593209, 223.118496, 0.000000) lúc 3100ms
Server đã nhận được thông tin: Drone 2 (IP: 10.0.0.3) phát hiện người thứ 1 tại (40.593209, 223.118496, 0.000000) lúc 3100ms
Drone 7 (IP: 10.0.0.8) phát hiện người thứ 2 tại (554.135969, 314.241476, 0.000000) lúc 18100ms
Server đã nhận được thông tin: Drone 7 (IP: 10.0.0.8) phát hiện người thứ 2 tại (554.135969, 314.241476, 0.000000) lúc 18100ms
Drone 20 (IP: 10.0.0.21) phát hiện người thứ 3 tại (622.644794, 59.008622, 0.000000) lúc 78100ms
Server đã nhận được thông tin: Drone 20 (IP: 10.0.0.21) phát hiện người thứ 3 tại (622.644794, 59.008622, 0.000000) lúc 78100ms
Max Packets per trace file exceeded
Kết thúc thời gian tìm kiếm !!!
phuc@phuc-virtual-machine:~/ns-allinone-3.38/ns-3.38$
```

Hình 102: Kết quả thu được 75 dBm

⇒ Kết quả thu được: Tìm kiếm được 3 người và cả 3 gói tin đều truyền thành công

Flow Id:1 ===== UDP 10.0.0.3/49153--->10.0.0.1/8080 Tx bitrate:-1.056e+09 kbps Rx bitrate:-1.056e+09 kbps Mean delay:5.9294ms Packet Loss ratio:0% timeFirstTxPacket= 3.1e+09ns timeFirstRxPacket= 3.10593e+09ns timeLastTxPacket= 3.1e+09ns timeLastRxPacket= 3.10593e+09ns delaySum= 5.9294e+06ns jitterSum= 0ns lastDelay= 5.9294e+06ns txBytes= 132 rxBytes= 132 txPackets= 1 rxPackets= 1 lostPackets= 0 timesForwarded= 0 delayHistogram nBins:6 Index:5 Start:0.005 Width:0.001 Count:1 jitterHistogram nBins:0 packetSizeHistogram nBins:7 Index:6 Start:120 Width:20 Count:1 flowInterruptionsHistogram nBins:0	Flow Id:2 ===== UDP 10.0.0.8/49153--->10.0.0.1/8080 Tx bitrate:-1.072e+09 kbps Rx bitrate:-1.072e+09 kbps Mean delay:9.08118ms Packet Loss ratio:0% timeFirstTxPacket= 1.81e+10ns timeFirstRxPacket= 1.81091e+10ns timeLastTxPacket= 1.81e+10ns timeLastRxPacket= 1.81091e+10ns delaySum= 9.08118e+06ns jitterSum= 0ns lastDelay= 9.08118e+06ns txBytes= 134 rxBytes= 134 txPackets= 1 rxPackets= 1 lostPackets= 0 timesForwarded= 0 delayHistogram nBins:10 Index:9 Start:0.009 Width:0.001 Count:1 jitterHistogram nBins:0 packetSizeHistogram nBins:7 Index:6 Start:120 Width:20 Count:1 flowInterruptionsHistogram nBins:0	Flow Id:3 ===== UDP 10.0.0.21/49153--->10.0.0.1/8080 Tx bitrate:-1.08e+09 kbps Rx bitrate:-1.08e+09 kbps Mean delay:3.59828ms Packet Loss ratio:0% timeFirstTxPacket= 7.81e+10ns timeFirstRxPacket= 7.81036e+10ns timeLastTxPacket= 7.81e+10ns timeLastRxPacket= 7.81036e+10ns delaySum= 3.59828e+06ns jitterSum= 0ns lastDelay= 3.59828e+06ns txBytes= 135 rxBytes= 135 txPackets= 1 rxPackets= 1 lostPackets= 0 timesForwarded= 0 delayHistogram nBins:4 Index:3 Start:0.003 Width:0.001 Count:1 jitterHistogram nBins:0 packetSizeHistogram nBins:7 Index:6 Start:120 Width:20 Count:1 flowInterruptionsHistogram nBins:0
---	---	---

Hình 103: Flow Monitor 75 dBm

⇒ Luồng dữ liệu: Không mất gói tin nào.

➤ **Nhận xét:** Độ hiệu quả gần như tương tự với mức công suất 45 dBm.

4.4.4 So sánh

- So sánh kĩ hơn sự khác nhau thời gian gói tin đến Ground Station của 2 mức công suất 45dBm và 75 dBm:

udp&&udp.length>100						
No.	Time	Source	Destination	Protocol	Length	Info
56	3.072775	10.0.0.3	10.0.0.1	UDP	170	49153 → 8080 Len=104
399	19.079677	10.0.0.8	10.0.0.1	UDP	172	49153 → 8080 Len=106
960	79.080849	10.0.0.21	10.0.0.1	UDP	173	49153 → 8080 Len=107

Hình 104: Thời gian đến của mức công suất 45 dBm

udp&&udp.length>100						
No.	Time	Source	Destination	Protocol	Length	Info
108	3.103782	10.0.0.3	10.0.0.1	UDP	170	49153 → 8080 Len=104
459	18.106934	10.0.0.8	10.0.0.1	UDP	172	49153 → 8080 Len=106
1827	78.101451	10.0.0.21	10.0.0.1	UDP	173	49153 → 8080 Len=107

Hình 105: Thời gian đến của mức công suất 75 dBm

- **Nhận xét:** Các gói tin của mức công suất 75 dBm có thời gian đến nhanh hơn so với mức công suất 45 dBm. Gói tin 1 của mức công suất 75 dBm chậm hơn là do thời điểm 3s là thời điểm bảng định tuyến reset nên cần thời gian để xây dựng lại bảng định tuyến dẫn đến gói tin tới chậm hơn. Nhìn chung, công suất cao khiếu cho việc các thiết bị giao tiếp được xa hơn, tìm được đi ngắn hơn đến đích nên tốc độ tới đích của gói tin nhanh hơn.

Công suất phát	20 dBm	45 dBm	75 dBm
Tính thực tế	Cao	Trung bình	Thấp
Số người phát hiện	3 người	3 người	3 người
Số gói tin truyền về thành công	0 gói tin	3 gói tin	2 gói tin

Bảng 11: Bảng so sánh công suất phát

CHƯƠNG V. KẾT LUẬN

5.1 Tóm tắt kết quả đạt được

- Đồ án "Sử dụng NetSimulyzer để mô phỏng lại UAV-based network" đã hoàn thành các mục tiêu nghiên cứu đề ra, tạo ra một nền tảng mô phỏng mạnh mẽ và linh hoạt để nghiên cứu về mạng lưới UAV. Các kết quả nổi bật đạt được trong quá trình thực hiện đồ án bao gồm:

- **Xây dựng môi trường mô phỏng:** Đồ án đã thành công trong việc thiết lập một môi trường mô phỏng mạng UAV hoàn chỉnh trên nền tảng ns-3, tích hợp một cách chặt chẽ với NetSimulyzer để trực quan hóa và phân tích dữ liệu. Các thiết bị mạng như UAV, trạm điều khiển và các yếu tố môi trường được thiết lập một cách chi tiết và có tính tùy biến cao.

- **Triển khai thành công các kịch bản:** Đồ án đã thiết kế và triển khai ba kịch bản mô phỏng chính, bao gồm việc đánh giá ảnh hưởng của số lượng UAV, kích thước map, và công suất phát đến hiệu năng của mạng. Các kịch bản này giúp chúng em nghiên cứu các khía cạnh khác nhau của mạng UAV và có được các dữ liệu cần thiết để phân tích và đánh giá.

- Thu thập và phân tích dữ liệu:

+ Chúng em đã khai thác và thu thập được các dữ liệu về hiệu năng mạng như throughput, delay, jitter, packet loss rate, từ FlowMonitor của ns-3, các số liệu này là cơ sở để đánh giá hiệu năng mạng một cách khách quan.

+ Các kết quả này đã được NetSimulyzer trực quan hóa, giúp chúng ta dễ dàng hình dung và nắm bắt được cách hoạt động của mạng.

+ Đồng thời chúng em đã dùng các đồ thị và số liệu từ NetAnim để đánh giá thêm về vị trí và sự di chuyển của UAV, cũng như khả năng kết nối, giao tiếp giữa các node trong mạng.

- Phân tích hiệu năng mạng UAV:

+ Đã tiến hành phân tích các số liệu thu được để đưa ra nhận xét về ảnh hưởng của số lượng UAV, kích thước bản đồ, công suất phát đến hiệu năng của mạng.

+ Đánh giá được các yếu tố ảnh hưởng chính đến hiệu năng mạng.

- **Đề xuất các hướng phát triển:** Đồ án đã đề xuất việc tích hợp học tăng cường (Reinforcement Learning - RL) như một hướng đi đầy tiềm năng, giúp tối ưu hóa các khía cạnh như định tuyến, quản lý tài nguyên và điều khiển UAV.

5.2 Kết luận về vai trò của NetSimulyzer trong mô phỏng mạng UAV

- **Tính trực quan:** NetSimulyzer cung cấp khả năng trực quan hóa 3D, tạo ra một môi trường mô phỏng chân thực, giúp người dùng dễ dàng theo dõi sự di chuyển của UAV, các luồng dữ liệu, và các tương tác giữa các thiết bị. Điều này giúp đơn giản hóa việc phân tích các hệ thống mạng phức tạp như UAV-based network.
- **Khả năng tùy biến:** NetSimulyzer có khả năng tùy chỉnh cao, cho phép người dùng thay đổi các mô hình 3D của thiết bị, tạo ra các môi trường mô phỏng khác nhau, và thêm các yếu tố trang trí. Tính năng này làm cho NetSimulyzer trở thành một công cụ linh hoạt, đáp ứng được nhiều yêu cầu nghiên cứu và mô phỏng đa dạng.
- **Tích hợp với ns-3:** Việc NetSimulyzer có thể tích hợp với ns-3 giúp tạo ra một quy trình nghiên cứu hiệu quả, có thể tận dụng được các khả năng của cả hai công cụ. NetSimulyzer có thể nhận dữ liệu từ ns-3 để hiển thị và phân tích một cách trực quan.

5.3 Đóng góp của đồ án và giá trị thực tiễn

- **Nền tảng nghiên cứu:** Đồ án xây dựng một nền tảng mô phỏng mạng UAV hữu ích, có thể sử dụng trong các nghiên cứu về mạng không dây, các giao thức định tuyến, các kỹ thuật truyền dẫn và quản lý tài nguyên.
- **Ứng dụng thực tiễn:** Đồ án minh họa khả năng áp dụng NetSimulyzer và ns-3 trong việc thiết kế, tối ưu hóa và quản lý mạng UAV. Các kết quả từ mô phỏng có thể được sử dụng để đưa ra các quyết định sáng suốt trong việc triển khai mạng UAV trong thực tế.
- **Tài liệu tham khảo:** Đồ án cung cấp một tài liệu tham khảo chi tiết về các bước thực hiện, các kiến thức cơ bản, cách cấu hình và đánh giá hiệu năng, từ đó có thể giúp những người mới bắt đầu dễ dàng tiếp cận và làm quen với công nghệ mô phỏng.
- **Tính linh hoạt:** Đề tài này có thể là cơ sở để xây dựng một hệ thống mô phỏng phức tạp hơn và phù hợp với nhiều ứng dụng khác nhau.
- **Tối ưu hóa:** Các thông số và các kịch bản thực nghiệm có thể giúp tối ưu hóa các tham số để xây dựng một mạng lưới UAV phù hợp với yêu cầu thực tế.

5.4 Hướng phát triển tiếp theo của đề tài đồ án

- **Tích hợp học tăng cường (RL):** Đồ án đề xuất tích hợp các thuật toán RL vào việc điều khiển UAV, từ đó có thể tự động tối ưu hóa việc truyền dữ liệu và các chiến lược định tuyến.
- **Mở rộng các kịch bản:** Xây dựng các kịch bản mô phỏng phức tạp hơn, bao gồm các tình huống di chuyển thực tế, các chướng ngại vật, và điều kiện thời tiết khác nhau.

- **Tích hợp thêm các công cụ:** Tích hợp NetSimulyzer với các công cụ khác để có thể phân tích và đánh giá hiệu quả hơn, như:

+ Sử dụng các tool phân tích như pandas, numpy của Python để xử lý dữ liệu, tạo đồ thị.

+ Tích hợp với các công cụ lưu trữ dữ liệu (database, time-series database).

- **Mô hình hóa các loại UAV khác nhau:** Thủ nghiệm với các loại UAV khác nhau để mô phỏng các tình huống phức tạp hơn.

- **Khảo sát các giao thức định tuyến khác:** So sánh các giao thức định tuyến khác nhau trong mạng MANET như DSR, OLSR và các giao thức định tuyến dành cho mạng UAV chuyên biệt, để tìm ra giao thức tốt nhất.

- **Mô phỏng các giao thức khác:** Trong quá trình mô phỏng, nếu có thể nên kết hợp các giao thức truyền tải khác như TCP.

- **Ứng dụng vào thực tiễn:** Liên hệ các kết quả mô phỏng với các ứng dụng thực tế như giám sát môi trường, cứu hộ, giao thông, và thử nghiệm trực tiếp (nếu có cơ hội).

Bảng phân công công việc

Họ và tên	Công việc	Hoàn thành (%)
Nguyễn Dương Hoàng Phúc	<ul style="list-style-type: none"> - Nghiên cứu tổng quan về UAV và ứng dụng thực tế của chúng; Tìm hiểu về các công cụ NetSimulyzer, ns-3; Thiết kế các kịch bản mô phỏng; Trực tiếp chạy các kịch bản mô phỏng trên ns-3 và sử dụng NetSimulyzer. - Viết report phần tổng quan, phương pháp nghiên cứu, tóm tắt, kết luận và đưa ra hướng phát triển. - Tổng hợp phần kết quả giám sát của báo cáo - Làm slide thuyết trình và thuyết trình về tổng quan đề tài và kết luận. 	100%
Phan Văn Tài	<ul style="list-style-type: none"> - Nghiên cứu về mạng MANET và các giao thức định tuyến; Tìm hiểu về ns-3; Xây dựng mô hình mạng MANET cho UAV trên ns-3. - Viết report phần cơ sở lý thuyết liên quan đến MANET, ns-3, mô hình mạng và luồng dữ liệu. - Làm slide thuyết trình về phần cơ sở lý thuyết liên quan đến kiến trúc mạng MANET và luồng dữ liệu. - Thuyết trình về các kịch bản của UAV 	100%
Phạm Ngọc Sơn	<ul style="list-style-type: none"> - Nghiên cứu chi tiết về NetSimulyzer và cách tích hợp với ns-3; Tìm hiểu về các ứng dụng của NetSimulyzer; Phân tích dữ liệu thu thập được từ NetSimulyzer. - Viết report phần cơ sở lý thuyết liên quan đến NetSimulyzer và phần triển khai mô phỏng, các bước thu thập và xử lý dữ liệu. - Làm slide và thuyết trình về cách NetSimulyzer hoạt động và phân tích kết quả. 	100%

Bảng 12: Bảng phân công công việc

CHƯƠNG VI. TÀI LIỆU THAM KHẢO

- [1] <https://github.com/usnistgov/NetSimulyzer>
- [2] <https://apps.nsnam.org/app/netsimulyzer/>
- [3] <https://github.com/usnistgov/NetSimulyzer-ns3-module>
- [4] https://www.nsnam.org/wp-content/uploads/2021/wns3-session-4/wns3_2021_black.pdf
- [5] <https://www.nsnam.com/2023/07/netsimulyzer-3d-visualizer-for-network.html>
- [6] <https://www.projectguideline.com/simulation-of-uav-based-search-and-rescue-scenario-with-ns-3/>
- [7] <https://luanvan.net.vn/luan-van/mang-di-dong-va-khong-day-tim-hieu-ve-mang-manet-46149/>
- [8] <https://www.nsnam.org/>
- [9] http://tailieuso.udn.vn/bitstream/TTHL_125/10148/2/NguyenBo.TT.pdf
- [10] <https://luanvan.net.vn/luan-van/do-an-tim-hieu-va-thuc-hien-mo-phong-mang-manet-30721/>
- [11] <https://123docz.net/trich-doan/866173-tong-quan-ve-mang-manet-va-ung-dung.htm>
- [12] <https://www.youtube.com/watch?v=OgyNoCgPpYc>
- [13] <https://www.youtube.com/watch?v=ZbuZSWH68CI>
- [14] <https://www.youtube.com/watch?v=QbQyjY7R4-Q&t=2133s>
- [15] <https://www.youtube.com/watch?v=I6YoChHKN-Y>
- [16] https://www.youtube.com/watch?v=_CuseF1JWE&t=3167s
- [17] <https://www.projectguideline.com/visualizing-ns-3-simulation-using-netanim/>
- [18] <https://www.projectguideline.com/public-safety-communication-modeling-tools-based-on-ns-3/>
- [19] <https://www.youtube.com/watch?v=MBSyuT931iY>