ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ───────

******

**Báo cáo Bài tập lớn**

Môn: Hệ điều hành và quản trị mạng Linux theo chuẩn kỹ năng ITSS

Đề tài: **Tìm hiểu IoT, tổng quan về giao thức RPL, mô phỏng và đánh giá.**

Mã lớp học : **126831**

Giáo viên hướng dẫn : TS Nguyễn Đức Toàn

Sinh viên thực hiện: Nhóm 5

| **STT** | **Họ tên** | **Mã sinh viên** |
| --- | --- | --- |
| 1 | Nguyễn Minh Chương | 20184051 |
| 2 | Lại Thế Ngọc | 20176838 |
| 3 | Vũ Minh Công | 20184052 |

***Hà Nội, tháng 2 năm 2022***

Mục lục

[Chương 1: Tìm hiểu về IoT 4](#_heading=h.30j0zll)

[1.1. Định nghĩa IoT 5](#_heading=h.1fob9te)

[1.2. Tầng giao thức 5](#_heading=h.3znysh7)

[1.3. Kiến trúc tham chiếu 6](#_heading=h.2et92p0)

[1.3.1.](#_heading=h.tyjcwt) WSO2 6

[1.3.2. Microsoft Azure 10](#_heading=h.3dy6vkm)

[1.4.](#_heading=h.1t3h5sf) Một số giao thức định tuyến trong IoT 13

[CHƯƠNG 2. TỔNG QUAN VỀ RPL, CONTIKI VÀ HOẠT ĐỘNG CỦA RPL TRÊN CONTIKI 14](#_heading=h.4d34og8)

[2.1. Tổng quan về RPL 14](#_heading=h.2s8eyo1)

[2.1.1. Khái niệm, thuật ngữ sử dụng trong RPL 15](#_heading=h.17dp8vu)

[2.1.2. Các bản tin điều khiển trong RPL 19](#_heading=h.3rdcrjn)

[2.1.3. Bản tin DIS 19](#_heading=h.26in1rg)

[2.1.4. Bản tin DIO 20](#_heading=h.lnxbz9)

[2.1.4.1. Cấu trúc bản tin DIO 20](#_heading=h.35nkun2)

[2.1.4.2. Các luật của bản tin DIO 21](#_heading=h.1ksv4uv)

[2.1.4.4. Bộ định thời DIO 24](#_heading=h.44sinio)

[2.1.4.5. Nhận và xử lý bản tin DIO 25](#_heading=h.2jxsxqh)

[2.1.5. Bản tin DAO 25](#_heading=h.z337ya)

[2.1.5.1. Cấu trúc bản tin DAO 26](#_heading=h.3j2qqm3)

[2.1.5.2. Truyền bản tin DAO 26](#_heading=h.1y810tw)

[2.1.6. Quá trình khởi tạo mạng 27](#_heading=h.4i7ojhp)

[2.1.7. Quá trình định tuyến upward 27](#_heading=h.2xcytpi)

[2.1.7.1. DODAG parent và preferred parent 28](#_heading=h.1ci93xb)

[2.1.7.2. Quản lý neighbor, khám phá DODAG và lựa chọn DODAG parent 28](#_heading=h.3whwml4)

[2.1.7.3 . Tính toán rank và sự di chuyển của các node 29](#_heading=h.2bn6wsx)

[2.1.8. Truyền gói 30](#_heading=h.qsh70q)

[2.2. Tổng quan về Contiki RPL và hoạt động của RPL trên Contiki 30](#_heading=h.3as4poj)

[2.2.1. Hệ điều hành Contiki 31](#_heading=h.1pxezwc)

[2.2.2. Contiki RPL 31](#_heading=h.49x2ik5)

[2.2.3. Định tuyến trong Contiki RPL 34](#_heading=h.2p2csry)

[2.2.3.1. Những yêu cầu cần đạt được khi triển khai Contiki RPL 35](#_heading=h.147n2zr)

[2.2.3.2. Hoạt động của DODAG root / UDP-Server 35](#_heading=h.3o7alnk)

[2.2.3.3. Hoạt động của các node client trong DODAG / UDP-client 37](#_heading=h.23ckvvd)

[2.2.3.4. Cơ chế xử lý bản tin DIO và xây dựng DODAG 39](#_heading=h.ihv636)

[CHƯƠNG 3. MÔ PHỎNG HOẠT ĐỘNG CỦA RPL TRÊN CONTIKI RPL 42](#_heading=h.32hioqz)

[3.1. Mục tiêu mô phỏng 42](#_heading=h.1hmsyys)

[3.2. Tham số cài đặt mô phỏng 42](#_heading=h.41mghml)

[3.3. Kịch bản mô phỏng 42](#_heading=h.2grqrue)

[3.4. Phân tích 43](#_heading=h.vx1227)

[3.4.1. Kịch bản 1: 1 Instance – 1 DODAG root – 15 Node client 43](#_heading=h.3fwokq0)

[3.4.1.1. Quá trình hình thành topo mạng 43](#_heading=h.1v1yuxt)

[3.4.1.2. Tỉ lệ truyền gói thành công 46](#_heading=h.4f1mdlm)

[3.4.1.3. Chi phí truyền gói 47](#_heading=h.2u6wntf)

[3.4.2. Kịch bản 2: 1 Instance – 2 DODAG root – 30 Node client 48](#_heading=h.19c6y18)

[3.4.2.1. Quá trình hình thành topo mạng 48](#_heading=h.3tbugp1)

[3.4.2.2. Tỉ lệ truyền gói thành công 50](#_heading=h.28h4qwu)

[3.4.2.3 Chi phí truyền gói 51](#_heading=h.nmf14n)

[3.5. Đánh giá kết quả mô phỏng 52](#_heading=h.37m2jsg)

[3.6. Kết luận 53](#_heading=h.1mrcu09)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 54](#_heading=h.46r0co2)

[Tài liệu tham khảo 55](#_heading=h.2lwamvv)

# **Chương 1: Tìm hiểu về IoT**

Internet of Things (IoT) có thể định nghĩa là là một mạng cho phép theo dõi và kiểm soát môi trường vật lý bằng cách thu thập, xử lý và phân tích dữ liệu được tạo ra bởi các cảm biến hoặc các đối tượng thông minh. Mặc dù đã xuất hiện được một thời gian dài nhưng Internet of Things chỉ thực sự được sự được chú ý và bùng nổ trong những năm gần đây. Sự tiến bộ nhanh chóng của hàng loạt các lĩnh vực công nghệ khác nhau như phần cứng, hệ thống nhúng, mạng không dây, cảm biến. . . đã cho phép chúng ta gắn mỗi đối tượng vật lý thực với một định danh duy nhất (ID) và kết nối chúng thành một mạng lưới, hay xa hơn cả một tương lai thật sự thông mimh. Mỗi đối tượng này có khả năng thu thập thông tin cảm biến, xử lý thông tin, đáp ứng môi trường một cách thông minh và giao tiếp không dây với những đối tượng khác. Bằng cách kết nối tất cả các đối tượng với Internet chúng ta có thể tạo ra một mạng vạn vật chia sẻ thông tin định danh và trạng thái từ tất cả đối tượng, đó chính là ý tưởng của Internet of Things. Trong chương này nhóm em tìm hiểu về IoT từ bao quát đến chi tiết để xây dựng nền tảng lý thuyết cho việc xây dựng hệ thống IoT.

## **Định nghĩa IoT**

IoT đại diện cho một mạng lưới toàn cầu mà trong đó các đồ vật được kết nối vưới nhau thông các giao thức giao tiếp tiêu chuẩn.

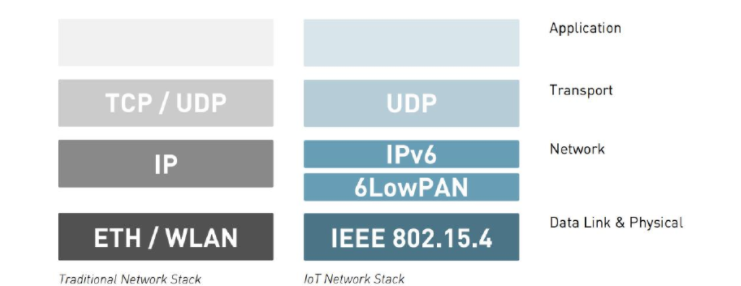
Theo như nhóm nghiên cứu các dự án IoT ở châu Âu, các đồ vật là những đối tượng tham gia chủ động trong công việc trao đổi dữ liệu, thu thập thông tin của môi trường mà chúng được cho phép tương tác khi có hoặc không có sự can thiệp của con người.

Ủy ban Châu Âu lại định nghĩa IoT, Nguồn gốc ngữ nghĩa của sự miêu tả này được gói gọn trong hai từ: Internet và Things. Internet được định nghĩa như một mạng lưới toàn cầu của các mạng lưới máy tính dựa trên giao thức tiêu chuẩn (bộ giao thức internet TCP/IP) được kết nối với nhau, trong khi Things ám chỉ đồ vật được định danh chính xác. Vì vậy theo ngữ nghĩa, Internet of Things nghĩa là mạng lưới toàn cầu của các đồ vật được kết nối với nhau dựa trên các giao thức định tuyến tiêu chuẩn.

Trong nguồn sách do O. Vermesan và các tác giả định nghĩa IoT là, Một loại cơ sở hạ tầng mạng toàn cầu năng động với khả năng tự cấu hình dựa trên các tiêu chuẩn và các giao thức giao tiếp tương thích. Trong đó vật chất thực và vật chất “ảo” được định danh trong mạng lưới thông tin.

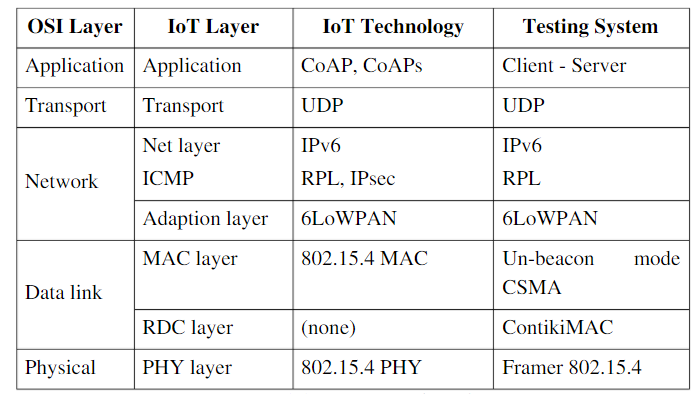
Trong tất cả, định nghĩa được công nhận rộng rãi được đưa ra bởi P. Guillemin và P. Friess là, IoT cho phép con người và đồ vật được sử bất kì phương thức nào và dịch vụ nào để kết nối với nhau mọi lúc, mọi nơi.

## **Tầng giao thức**



So sánh tầng giao thức của mạng internet truyền thống và mạng IoT

Các tổ chức lớn trên thế giới nhƣ IEEE, IETF hay cả W3C đã chuẩn hóa các giao thức của họ nhƣ 6LowPAN, CoAP. . .



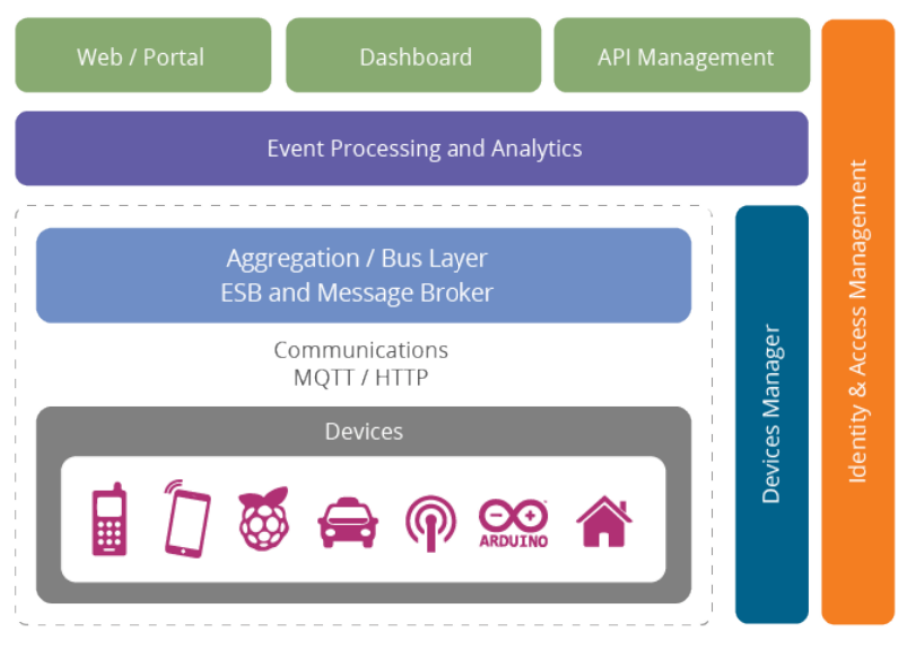
IoT protocol stack

## **Kiến trúc tham chiếu**

Một trong những khó khăn lớn nhất của IoT là việc chưa có một chuẩn thống nhất. Định nghĩa, kiến trúc, các giao thức trong IoT vẫn chưa được chuẩn hóa gây khó khăn cho việc tìm hiểu và phát triển các ứng dụng IoT.

### ***WSO2***

Kiến trúc tham chiếu của IoT theo WSO2 được đưa ra dưới đây



Kiến trúc tham chiếu của IoT

* 5 lớp ngang:

1. Client/External communications - Lớp Giao tiếp người dùng cuối.

2. Event Processing and Analytics - Lớp Xử lý và phân tích.

3. Aggregation/Bus - Lớp Hợp nhất.

4. Communications - Lớp Truyền thông.

5. Devices - Lớp Thiết bị.

* 2 lớp dọc:

1. Device management - Quản lý thiết bị.
2. . Identity and Access management - Định danh và Truy cập.

* Lớp Thiết bị - Devices

Lớp Thiết bị - Communications nằm dưới cùng trong kiến trúc tham chiếu. Các

thiết bị IoT rất đa dạng nhưng thường được chia làm 3 loại dựa theo phần cứng.

Loại thứ nhất có cấu hình thấp nhất, thường dùng chip 8 bit nhúng và không có hệ điều hành (thường là các thiết bị Arduino 8 bit). Ở mức cao hơn, với vi xử lý 32 bit rút gọn, loại thiết bị như Arduino Zeno, Yun có thể chạy Linux rút gọn hoặc hệ điều hành nhúng. Loại thứ 3 là các thiết bị sử dụng nền tảng 32 bit đầy đủ hoặc 64 bit như Raspberry Pi, Beagle Bone, Intel Galileo. . . và cả điện thoại di động. Có rất nhiều loại thiết bị khác nhau, tuy nhiên để được coi là một thiết bị IoT thì cần phải có kết nối trực tiếp hoặc gián tiếp tới Internet.

- Kết nối trực tiếp: Các thiết bị loại này thường đóng vai trò Gateway. Ví dụ: Arduino, Raspberry Pi, Intel Galileo. . . kết nối trực tiếp tới Internet qua Wi-Fi hoặc Ethernet. . .

- Kết nối gián tiếp: qua ZigBee (kết nối tới ZigBee, Gateway) hoặc Bluetooth (kết nối tới Raspberry Pi hoặc điện thoại di động). . .

* Lớp Truyền thông - Communications

Lớp Truyền thông thực hiện nhiệm vụ kết nối các thiết bị. Các giao thức có thể sử dụng để kết nối các thiết bị tới đám mây: MQTT3.1.1, HTTP, CoAP. . .

- HTTP là giao thức lâu đời và phổ biến nhất với nhiều thư viện hỗ trợ. HTTP là giao thức dựa trên hệ ký tự đơn giản nên nhiều thiết bị nhỏ với bộ điều khiển 8 bit có thể hỗ trợ HTTP, các thiết bị mạnh hơn có thể sử dụng các thư viện HTTP đầy đủ. Phiên bản HTTP2 được phát triển nhằm giải quyết vấn đề năng lượng và kết nối cho các thiết bị nhỏ.

- MQTT (Message Queuing Telemetry Transport) ra đời năm 1999 nhằm giải quyết các vấn đề trong hệ thống nhúng và được chuẩn hóa bởi OASIS. MQTT hoạt động theo mô hình hệ phân tán với hai lệnh cơ bản là "publish" và "subscribe". Các client sẽ kết nối tới máy chủ MQTT (MQTT Broker), mỗi client sẽ đăng ký theo dõi (subscribe) các kênh (topic) và tất cả các node đăng ký kênh sẽ nhận được dữ liệu trên kênh khi có bất kì node nào gửi dữ liệu (publish) lên kênh. Được tối ưu hóa việc gửi nhận các bản tin có kích thước nhỏ cho môi trường IoT, có thể đánh giá MQTT cao hơn so với HTTPS cho mục đích kết nối các thiết bị IoT. Trong môi trường không ổn định, băng thông thấp, trễ lớn nhưng MQTT vẫn cho kết quả có độ tin cậy cao và tiết kiệm năng lượng. Với việc được chuẩn hóa bởi OASIS, MQTT hứa hẹn trở thành một trong những giao thức chuẩn cho IoT.

- CoAP (Constrained Application Protocol): Cũng giống như MQTT, CoAP là một giao thức được tối ưu cho môi trường IoT do IETF phát triển dựa trên HTTP nhưng sử dụng mã nhị phân thay cho ký tự nên có kích thước nhỏ gọn hơn HTTP. CoAP cung cấp cấu trúc RESTful cho kết nối và sử dụng mô hình máy chủ - máy trạm như HTTP thay vì mô hình kênh như MQTT.

∙ Lớp Hợp nhất - Aggregation/Bus

Đây là lớp quan trọng để hợp nhất và chuyển đổi các loại bản tin truyền thông với 3 chức năng chính sau:

- Hỗ trợ máy chủ HTTP và/hoặc chuyển đổi MQTT/CoAP để cung cấp các giao tiếp với các thiết bị.

- Hợp nhất nội dung truyền từ các thiết bị khác nhau và thực hiện việc định tuyến tới một thiết bị cụ thể.

- Bắc cầu và chuyển đổi giữa các giao thức khác nhau, ví dụ chuyển đổi API dự trên HTTP ở lớp trên với các bản tin MQTT đến thiết bị.

Ngoài ra lớp này cũng thực hiên chức năng cung cấp một số tính năng tương quan, ánh xạ đơn giản từ các mô hình khác nhau (ví dụ số định danh thiết bị sang số định danh của người sở hữu thiết bị và ngược lại) và bảo mật truy cập dựa

trên chính sách (lớp Hợp nhất yêu cầu lớp Quản lý truy nhập và Định danh thẩm định các yêu cầu truy nhập).

* Lớp Xử lý và Phân tích - Event Processing and Analytics

Lớp này xử lý các sự kiện sau khi dữ liệu từ lớp Hợp nhất được chuyển lên, chức năng chính của lớp là lưu trữ dữ liệu vào cơ sở dữ liệu. Với các mô hình truyền thống, một ứng dụng server-side sẽ thực hiện chức năng trên, tuy nhiên hiện nay có một số cách tiếp cận khác linh hoạt hơn. Cách thứ nhất là sử dụng các Platform dành cho phân tích dữ liệu lớn (Big Data Platform): nền tảng dựa trên điện toán đám mây có khả năng mở rộng, hỗ trợ các công nghệ phân tích dữ liệu với kích thước lớn. Cách tiếp cận thứ hai là sử dụng phương thức Xử lý sự kiện phức tạp (Complex Processing Event) để thực hiện các hoạt động theo thời gian thực và ra quyết định hành động dựa theo kết quả phân tích dữ liệu từ các thiết bị chuyển đến.

* Lớp Giao tiếp người dùng cuối - Client/External Communications

Lớp này tạo ra giao diện giúp quản lý các thiết bị IoT như web/portal, dashboard và hệ thống quản lý API. Với web/portal, kiến trúc cần hỗ trợ các công nghệ Web phía máy chủ như Java Servlets/JSP, PHP, Python, Ruby. . . Dashboard là hệ thống tập trung vào việc trình bày đồ thị, mô tả dữ liệu đến từ các thiết bị và lớp Xử lý sự kiện. Lớp quản lý API có 3 chức năng: cung cấp portal tập trung vào việc hỗ trợ lập trình viên, đóng vai trò Gateway quản lý truy nhập các API; kiểm tra việc điều khiển truy nhập (đối với yêu cầu từ bên ngoài) dựa trên chính sách, định tuyến và cân bằng tải; thực hiện chức năng Gateway đẩy dữ liệu vào lớp phân tích để lưu trữ và xử lý.

* Quản lý thiết bị - Device Management

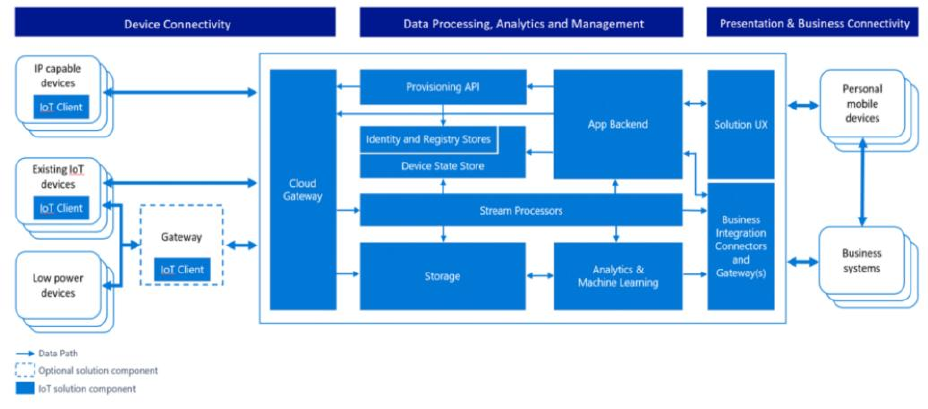
Trong lớp Quản lý thiết bị, hệ thống phía máy chủ quản lý thiết bị giao tiếp với các thiết bị thông qua các giao thức khác nhau và điều khiển một hoặc một nhóm thiết bị (có thể khóa hoặc xóa dữ liệu trên các thiết bị khi cần), quản lý định danh của các thiết bị và ánh xạ từng thiết bị với chủ sở hữu tương ứng. Máy chủ quản lý thiết bị phải phối hợp với lớp Quản lý Định danh và Truy nhập để quản lý việc điều khiển truy nhập vào thiết bị (những người có quyền truy nhập vào thiết bị và quyền hạn tương ứng, quyền của người quản trị hệ thống. . . ).

* Định danh và Truy cập - Identity and Access management

Lớp này cần cung cấp các dịch vụ: Phát hành và thẩm định token định danh Oauth2; dịch vụ định danh khác gồm SAML2 SSO và OpenID Connect; XACML; PDP; danh bạ cho người dùng (ví dụ: LDAP); quản lý chính sách điều khiển truy nhập (PCP).

### ***Microsoft Azure***

Năm 2016, Microsoft đưa ra kiến trúc tham chiếu cho IoT với mục đích đề xuất xây dựng hệ thống IoT cho môi trường doanh nghiệp, các giải pháp thiết bị với khả năng mở rộng và tích hợp với các thiết bị hệ thống back-end.



Kiến trúc tham chiếu bao gồm 3 khối chính:

- Device connectivity: Kết nối các thiết bị.

Các thiết bị có thể được kết nối trực tiếp hoặc gián tiếp thông qua Gateway với Internet. Cloud Gateway là thiết bị đầu cuối kết nối với các Smart Device và cung cấp kết nối hai chiều với hệ thống back-end.

- Data processing, analytics, and management: Xử lý, phân tích và quản lý dữ liệu. Phần back-end bao gồm nhiều thành phần cung cấp các chức năng quản lý thiết bị; lưu trữ, phân tích và xử lý dữ liệu; ảo hóa...

- Presentation and business connectivity: Trình diễn và các kết nối cho kinh doanh. Lớp này chịu trách nhiệm tích hợp các chức năng IoT vào các hoạt động kinh doanh của các doanh nghiệp. Người dùng cuối (End-user) sẽ tương tác với các giải pháp và các thiết bị IoT thông qua lớp này.

* Các tính chất của kiến trúc

- Tính chất không đồng nhất (Heterogeneity). Mô hình của hệ thống phải đáp ứng được yêu cầu của nhiều loại kịch bản, môi trường, thiết bị và các chuẩn khác nhau.

- Tính bảo mật (Security). Kiến trúc tham chiếu này phải cho phép triển khai các dịch vụ bảo mật cho tất cả các vùng, bao gồm quản lý định danh, xác thực và quyền hạn, bảo mật dữ liệu.

- Khả năng mở rộng (Hyper-scale deployments). Hệ thống IoT phải đảm bảo khả năng hỗ trợ đến quy mô hàng triệu thiết bị kết nối. Kiến trúc của hệ thống phải cho phép hệ thống bắt đầu triển khai với một quy mô nhỏ nhưng sẵn sàng mở rộng.

- Tính linh hoạt (Flexibility). Nhu cầu rất đa đạng của thị trường IoT đòi hỏi hệ thống phải có khả năng mở rộng thêm chức năng và cho phép sử dụng nhiều công nghệ của bên thứ 3 cho mỗi thành phần.

* Mô hình dữ liệu

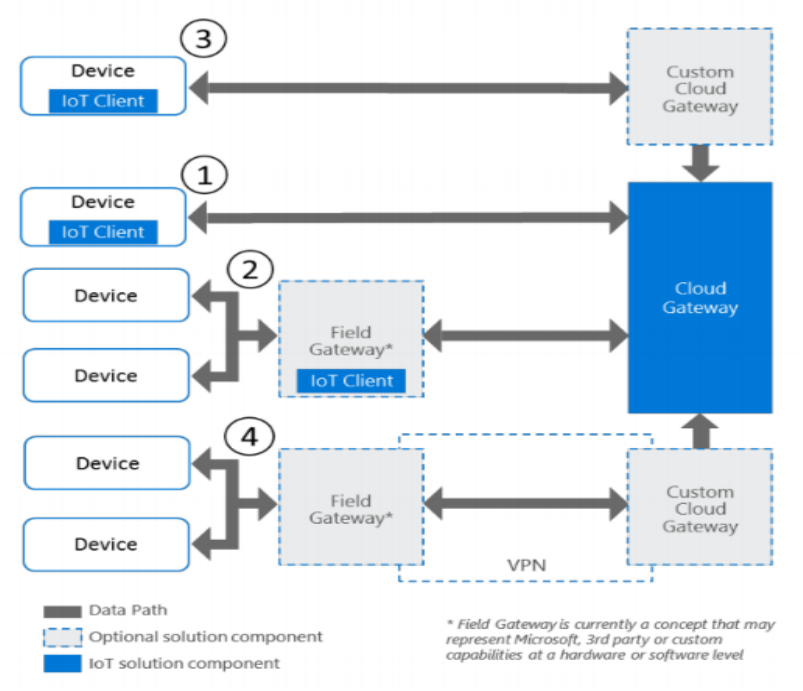
Tìm hiểu về mô hình dữ liệu trong IoT là bước quan trọng đầu tiên để xây dựng trung tâm thu thập, phân tích và kiểm soát dữ liệu.

- Mô hình thiết bị và dữ liệu. Mô hình thiết bị và dữ liệu thường mô tả một giản đồ của hệ thống các thiết bị: cấu trúc, định danh, mô hình dữ liệu được gửi từ các thiết bị. Các ngành khác nhau thường xây dựng một mô hình thiết bị khác nhau nhằm đáp ứng các yêu cầu đặc biệt về chuyên môn. Kiến trúc này tạo ra một mô hình chung, hỗ trợ nhiều mô hình để làm nền tảng cho việc phát triển các hệ thống chuyên biệt.

- Luồng dữ liệu. Kiến trúc tham chiếu đưa ra một khái niệm nền tảng cho luồng dữ liệu bao gồm hệ thống các bản ghi và cách các bản ghi được truyền đi. Luồng dữ liệu này không quy định về định dạng nội dung của các bản ghi, cấu trúc bản ghi sẽ phụ thuộc vào loại dữ liệu. Kiến trúc không quy định về cách đặt tên, ý nghĩa hay bất cứ cấu trúc nào không cần thiết cho các chức năng cơ bản của hệ thống.

* Kết nối thiết bị Smart Device

Các thiết bị có thể được kết nối trực tiếp hoăc gián tiếp thông qua Field Gateway. Hình dưới đây chỉ ra các giải pháp kết nối trong hệ thống.



Các giải pháp kết nối thiết bị trong kiến trúc tham chiếu IoT

1. Kết nối thiết bị trực tiếp đến Cloud Gateway: Kiểu kết nối này dành cho các thiết bị có kết nối trực tiếp đến Internet qua hạ tầng IP, thiết lập các kết nối an toàn đến đám mây.

2. Kết nối thông qua Field Gateway: dành cho các thiết bị sử dụng các kết nối dành cho công nghiệp (như CoAP, OPC), các công nghệ giao tiếp tầm ngắn (như Bluetooth, ZigBee), các thiết bị có phần cứng hạn chế không đủ để cấu hình một kết nối bảo mật TLS/SSL hoặc các thiết bị không có kết nối trực tiếp với Internet.

3. Kết nối thông qua Custom Cloud Gateway. Giải pháp này được sử dụng khi dữ liệu từ các thiết bị được kết nối yêu cầu phải chuyển đổi giao thức, định dạng dữ liệu hoặc thực hiện một tiến trình đặc biệt trước khi chuyển tới Cloud Gateway.

4. Kết nối thông qua một Field Gateway và một Custom Cloud Gateway.

* Field Gateway

Field Gateway là một thiết bị chuyên dụng hoặc một phần mềm có vai trò cung cấp khả năng giao tiếp; hoạt động như một hệ thống điều khiển và trung tâm xử lý dữ liệu cục bộ cho các thiết bị. Việc sử dụng Field Gateway có thể giúp giảm lượng dữ liệu cần phải gửi đến đám mây.

* Cloud Gateway

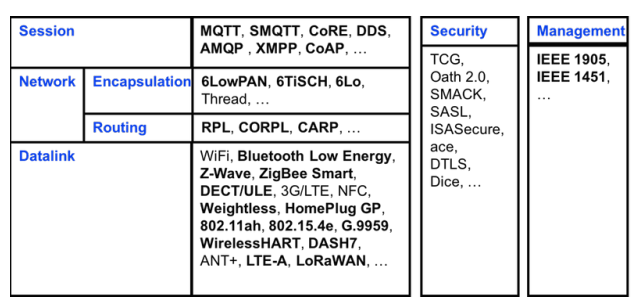
Cloud Gateway là một phần của kiến trúc dựa trên điện toán đám mây, cho phép triển khai các kết nối đến và từ các thiết bị hoặc Field Gateway. Nhiệm vụ chính của Cloud Gateway là quản lý các mặt của kết nối bao gồm bảo vệ đường truyền, xác thực thiết bị, quản lý các giao thức sử dụng ở các tầng kết nối. . .

* Custom Cloud Gateway

Custom Cloud Gateway cho phép thực hiện chuyển đổi giao thức và/hoặc một số dạng xử lý trước khi chuyển dữ liệu tới Cloud Gateway.

## ***Một số giao thức định tuyến trong IoT***

Sau khi được thầy cung cấp tài liệu thì chúng em có tìm hiểu được một số giao thức sử dụng trong IoT theo từng lớp mạng.



Sau quá trình tìm hiểu các kiến thức về IoT thì phần sau nhóm em sẽ tập trung vào tìm hiểu một giao thức IoT. Cụ thể là giao thức RPL. Sau đó mô phỏng và phân tích.

# **CHƯƠNG 2. TỔNG QUAN VỀ RPL, CONTIKI VÀ HOẠT ĐỘNG CỦA RPL TRÊN CONTIKI**

## **2.1. Tổng quan về RPL**

RPL là giao thức định tuyến cho mạng tổn hao năng lượng thấp nói chung và IoT nói riêng. Dự thảo đầu tiên về RPL được IETF đưa ra vào tháng 8/2009. Hiện nay, giao thức RPL đang trong quá trình dân hoàn thiện, với mục tiêu phát triển thành một chuẩn định tuyến trong tương lai. Quá trình phát triển RPL cũng nhận được sự quan tâm, đóng góp của nhiều tổ chức, cá nhân đến từ những tổ chức nghiên cứu khoa học, các trường đại học, viện nghiên cứu trên toàn thế giới.

Theo nguồn [7], [8] và [9] RPL được phát triển trên nền IPv6 nhằm đạt được những yêu cầu định tuyến sau:

- Có khả năng mở rộng (số lượng, quy mô). - Định tuyến dựa trên những thông số bị giới hạn như mức năng lượng,

dung lượng bộ nhớ. - Hỗ trợ tính di động. - Hỗ trợ khả năng tự cấu hình và cấu hình từ bên ngoài. - Hỗ trợ truyền tin multicast và anycast. - Hỗ trợ các luồng thông tin định hướng đến một node trong mạng. - Có cấu trúc mạng động.

Hỗ trợ định tuyến theo nhiều metric khác nhau. - Có khả năng hội tụ về thời gian. Giao thức định tuyến có tính hội tụ về

thời gian khi nó đáp ứng được những mức thời gian trễ cụ thể tương ứng với những điều kiện xác định. Có khả năng quản lý. Khi một node mới tham gia vào mạng, node đó phải tự động cấu hình và tham gia vào mạng mà không cần sự can thiệp của con người.

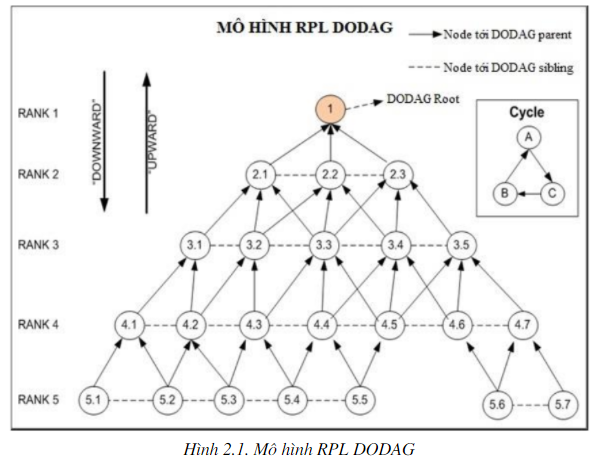
Hỗ trợ truyền gói theo mức độ ưu tiên và có độ tin cậy cao. | Hỗ trợ các phương thức bảo mật.

### ***2.1.1. Khái niệm, thuật ngữ sử dụng trong RPL***

RPL xây dựng và sử dụng các DAG (Directed Acyclic Graph) trong mạng để | thực hiện quá trình định tuyến. Trong đó, DAG là một cấu trúc mạng mà mọi liên | kết giữa các node trong DAG đều có hướng nhất định, hướng về một DAG root

và đảm bảo không tạo ra các vòng lặp trong DAG. | DAG Root: một node trong DAG, có chức năng tập trung và xử lý dữ liệu từ các node khác trong DAG gửi đến. Mọi tuyến liên kết trong DAG đều hướng về và kết thúc tại DAC root.

DODAG (Destination-Oriented DAG) là một DAG chỉ có một node đích, ví dụ như chỉ có một DAG root (trong trường hợp này gọi là DODAG Toot)



DODAG root: một DODAG root là DAG root của một DODAG. DODAG root có thể là router biên cua một DODAG.

Virtual DODAG TOOL: Một DODAC root ảo là kết quả của hai hoặc nhiều bộ định tuyến RPL, điều phối để đồng bộ hóa DODAG state và diễn ra như thế chúng là một DODAC root duy nhất

DODACID (DODAG Identifier): mã nhận dạng của mỗi DODAC root trong mạng. Tất cả các node trong mạng đều lưu DODACID của DODAG mà nó là thành viên.

DODAC Rank: là thông số cho biết vị trí tương đối của node so với DODAG root. Những node càng xa DODAC root thì có rank càng cao. Rank của node có thể được tính thông qua khoảng cách hình học giữa node và DODAC root, hoặc có thể được tính toán thông qua những hàm chức năng khác. Trong RPL, DODAC root luôn có rank bằng 1. Rank được sử dụng để đánh giá mối quan hệ parent-sibling-children giữa các node trong cùng một DODAC, từ đó tránh các vòng lặp có thể xảy ra khi truyền gói đến DODAC root. Hình 2.1 cho thấy rank của các node trong DODAG khi rank được tính bằng số chặng (hop) tối thiểu đến DODAG root.

DODAC Parent: trong cùng một DODAC, node A được gọi là parent của node B khi A có khả năng kết nối trực tiếp đến B và A có rank thấp hơn B. Khi đó A có thể đóng vai trò là next-hop của B trong quá trình truyền gói về DODAG root và B là một node children của A.

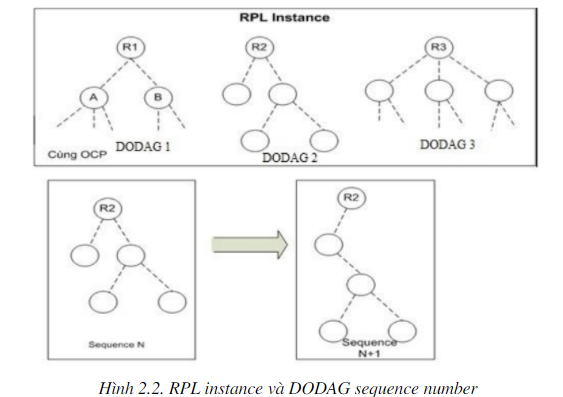
DODAC Sibling: node A là một sibling của node B trong một DODAC nêu chúng có cùng rank trong DODAG đó. Ví dụ, trong hình 2.1, các node (3.2) và (3.4) là sibling của node (3.3); mặt khác, node (3.3) nhận các node (2.1), (2.2), (2.3) là các parent.

Object Function (OF): là hàm chức năng cung cấp các phương thức cho phép một node lựa chọn được DODAG phù hợp, tính toán rank và lựa chọn các parent trong DODA. Các OF được thiết kế theo những quy tắc cụ thể tùy theo mục đích và phương thức định tuyến.

Object Code Point (0CP): là mã nhận dạng tương ứng với một OF cụ thể. OCP được sử dụng trong quá trình kiểm tra các bản tin trong mạng và quá trình tìm kiếm những DODAG phù hợp.

RPL Instance: là một tập hợp gồm một hoặc nhiều DODAG có cơ chế định tuyến giống nhau. Mỗi instance chỉ sử dụng một OF duy nhất để xây dựng cấu trúc mạng. RPL Instance ID: mã nhận dạng của các Instance, tương ứng với các OF cụ thể.

DODAC Sequence Number: là một bộ đếm tuần tự được sử dụng trong quá trình sửa chữa và làm mới DODAG. Khi một DODAC root muốn xây dựng lại một DODAG mới, sequence number được tăng lên một đơn vị và quảng bá tới các node khác trong mạng.



DODAG Interaction: mỗi DODAG ID và DODAG sequence number cho phép xác định một DODAG interaction. Khái niệm này cho phép mỗi node trong mạng phân biệt các DODAG mà node đã tham gia với một DODAG mà node chưa từng là thành viên. Đồng thời khái niệm DODAG interaction cũng cho phép cơ chế tránh các vòng lặp hoạt động hiệu quả hơn. | Trong RPL đề cập đến hai hướng định tuyến: upward và downward. Upward là chiều đi từ các node ở xa DODAC root hướng về DODAC root. Downward là chiều ngược lại, hướng từ DODAC root đến các node ở xa hơn.

RPL Goal: là một host hoặc một tập hợp gồm nhiều host có khả năng đáp ứng được các OF, phục vụ việc tập trung dữ liệu từ các DODAG hoặc tạo kết nối giữa các DODAG với các mạng và ứng dụng ngoài.

DODAG Grounded: một DODAC gọi là grounded nếu DODAG TOọt có khả | năng giao tiếp với một Goal thích hợp.

DODAG Floating: một DODAC gọi là là trôi nổi khi không thể chuyển dữ liệu hoặc kết nối đến một Goal phù hợp.

RPL sử dụng ba loại bản tin điều khiển gồm DODAC Information Solicitation (DIS), DODAG Information Object (DIO), Destination Advertisement Object (DAO) để quảng bá các thông tin định tuyến trong mạng. DIO là bản tin mang thông tin về DODAC, được gửi từ các parent đến các node children và được sử dụng để xây dựng DODAG. DIS chỉ thực hiện nhiệm vụ quảng bá sự xuất hiện của node và yêu cầu những node khác phản hồi bằng các bản tin DIO. DAO là bản tin được gửi từ một children đến các parent nhằm quảng bá khả năng tham gia đình tuyển theo chiều downward của các node trong

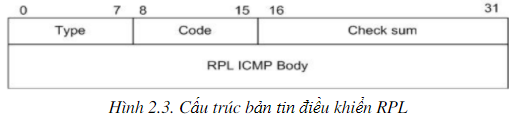
mạng.

### ***2.1.2. Các bản tin điều khiển trong RPL***

Giao thức RPL sử dụng ba loại thông điệp điều khiển để xây dựng và duy trì các DODAG trong mạng, bao gồm: DIS (DODAC Information Solicitation), DIO (DODAC Information Object - bản tin quảng bá DODAC), DAO (Destination Advertisement Object – bản tin quảng bá đích).

Qua quá trình gửi-nhận và xử lý các bản tin ICMP, mỗi node có thể quản lý thông tin các node hàng xóm của nó trong phạm vi kết nối, đưa ra quyết định tham gia vào DODAG phù hợp, lựa chọn các parent, sibling, xác định next-hop

và gửi gói đến DODAG root.



Trường Type của các bản tin RPL-ICMP được quy định bằng 155, sử dụng để phân biệt bản tin RPL-ICMP với bản tin của các giao thức khác. Trường Code được sử dụng để phân biệt các bản tin RPL-ICMP và được định nghĩa như sau:

- Code = 0x01: DODAG Information Solicitation (DIS).

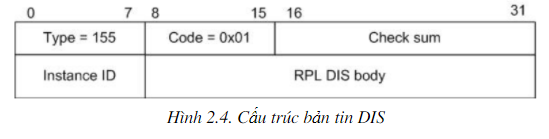
- Code = 0x02: DODAG Information Object (DIO).

- Code = 0x04: Destination Advertisement Object (DAO).

RPL sử dụng các bộ định thời để quản lý tốc độ gửi và số lượng bản tin ICMP trong mạng. Những bộ định thời này hoạt động kết hợp giữa chế độ định thời và chế độ sự kiện. Do đó, không những làm giảm số lượng những bản tin ICMP mà vẫn đảm bảo tính linh động của mạng.

### ***2.1.3. Bản tin DIS***

Bản tin DIS được gửi từ những node tự do trong mạng nhằm quảng bá sự xuất hiện của node, thăm dò sự xuất hiện của các node hàng xóm và yêu cầu những node khác phản hồi bằng các bản tin DIO. Bản tin DIS được gửi multicast khi node ở trạng thái tự do và được gửi unicast đến một parent trong DODAG khi muốn nhận lại một bản tin unicast DIO nhằm cập nhật các thông tin DODAG của parent đó.



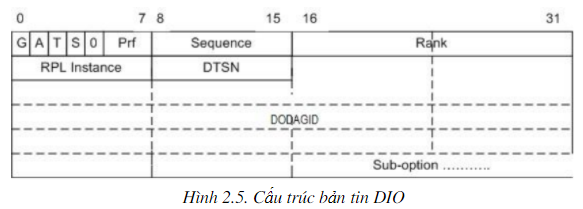
Trường Instance ID cho biết instance mà node gửi DIS tham gia định tuyến. Khi một node trong mạng nhận được một bản tin DIS, nó chỉ gửi lại bản tin DIO phản hồi khi có khả năng định tuyến cho Instance ID đó. Nhờ đó, khi triển khai mạng có nhiều instance, có thể giảm đáng kể số lượng bản tin điều khiển trong mạng, tiết kiệm năng lượng và thời gian xử lý của các node.

### ***2.1.4. Bản tin DIO***

DIO là bản tin được tạo ra tại các DODAC root, mang những thông tin định tuyến của DODAG như instance, rank, metric, ... DIO được sử dụng để quảng bá các thông tin định tuyến của một DODAG xác định trong mạng, phục vụ quá trình xây dựng DODAG và định tuyến upward. Quá trình nhận và xử lý bản tin DIO cho phép một node nhận diện và tham gia vào DODAG phù hợp. Từ đó lựa chọn các parent, xác định các thông số cấu hình và tiếp tục quảng bá thông tin DODAG đến các node khác trong mạng.

#### 2.1.4.1. Cấu trúc bản tin DIO

Cấu trúc của một bản tin DIO gồm hai phần chính: các trường điều khiển và cách sub-option



Các trường điều khiển gồm các cờ trạng thái và các thông số phục vụ cho quá

trình xây dựng DODAG:

- Cờ G (grounded): cho phép xác định khả năng kết nối đến một grounded

- Cờ G (grounded): cho phép xác định khả năng kết nối đến một grounded DODAG của node gửi DIO. Nếu cờ G bằng 0, node gửi DIO là thành viên

của một floating DODAG. Ngược lại, nếu cờ này khác 0 cho biết node gửi

DIO là thành viên của một grounded DODAG.

- Cờ A (Destination Advertisement Supported): cho biết khả năng hỗ trợ cơ

chế quảng bá đích trong quá trình định tuyến downward của DODAG. Nếu A khác 0, DODAC root có khả năng hỗ trợ cơ chế quảng bá đích và các node trong DODAG có thể tham gia quá trình định tuyến downward. Nếu A bằng 0, các node trong DODAG chỉ có thể tham gia định tuyến

upward.

- Cờ T (Destination Advertisement Trigger): được sử dụng để làm mới quá

trình định tuyến downward. Nếu T bằng 0, quá trình định tuyến downward hoạt động bình thường. Khi T khác 0, các node trong DODAG sẽ thực

hiện quá trình làm mới các tuyến downward.

- Cờ S (Destination Advertisement Stored): nếu cờ S bằng 0, chỉ DODAG

Toot được phép lưu các thông tin định tuyến downward từ các bản tin DAO. Nếu S khác 0, node gửi DIO có thể lưu các thông tin định tuyến từ DAO. Trường prf (DODAG preference): sử dụng 3 bit kiều số nguyên dương cho biết độ ưu tiên giữa các DODAC root trong cùng một Instance. Miền giá trị của prf từ 0x00 đến 0x07. Nếu prf bằng 0x07, DODAC root sẽ có độ ưu tiên cao nhất, giá trị mặc định của prf là (). Sequence number: sử dụng 8-bit kiểu số dương là giá trị sequence number được thiết lập tại DODAC root, được sử dụng trong quá trình tái xây dựng DODAG.

Rank: rank của node gửi DIO.

- RPL Instance: cho biết RPL Instance mà DODAG tham gia định tuyến.

Destination Advertisement Trigger Sequence Number (DTSN): được thiết lập tại node phát ra bản tin DIO, sử dụng trong quá trình duy trì các thông số định tuyến theo hướng downward. DODACID: là một địa chỉ IPv6 có độ dài 128 bit và được thiết lập tại DODAG root**.**

#### 2.1.4.2. Các luật của bản tin DIO

- Nếu cơ A của bản tin DIO bằng 0 thì cờ T cũng phải bằng 0 - Đối với cờ G, A, T, trường Prf, Sequence, RPLInstanceID, DODAID, nếu

như node không phải là DODAC root thì cần phải quảng bá các giá trị giống với node parent của nó. Chính vì vậy nếu như DODAG root mà không thay các giá trị trên thì tất cả cá node cũng phải quảng bá các giá trị

như vậy. - Một node có thể cập nhập cờ S, trường Rank và DTSN tại mỗi bước nhảy - Trường DODACID của mỗi DODAC root phải là duy nhất trong một

instance 2.1.4.3. Cấu trúc sub-option trong DIO

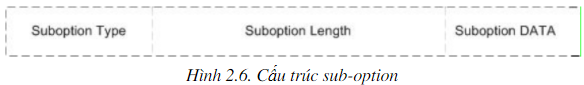
Các sub-option trong DIO bao gồm năm loại: Pad 1, Pad N, DODAG Metric Container, DODAC destination prefix, DODAG Configuration. Tùy theo mục đích sử dụng, các sub-option được chèn vào cấu trúc DIO một cách hợp lý.

Cấu trúc chung của các sub-option, gồm 3 phần:

- Type: 1 byte, dùng để phân biệt các sub\_option.

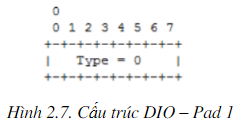
- Length: 2 bytes, cho biết chiều dài sub option.

- DATA: chiều dài thay đổi, lưu các thông tin của sub\_option.

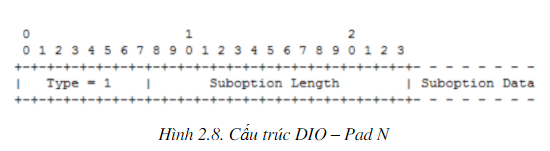


Pad 1: có cấu trúc chỉ gồm 1 byte có type bằng không, được sử dụng nhằm

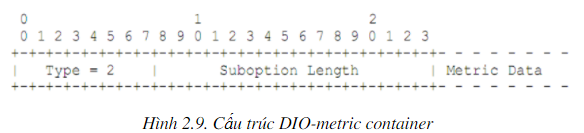
chèn một hoặc hai byte trống vào bản tin DIO.



Pad N: được sử dụng nhằm chèn N byte trống vào bản tin DIO.



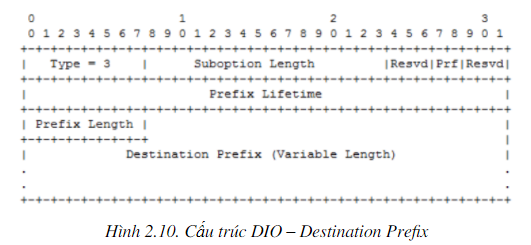
Metric Container được sử dụng để lưu trữ thông tin các Metric được hỗ trợ trong DODAG. Metric container có thể bao gồm một số các metric như thông số liên kết, băng thông, độ trễ, các thông số về tuyến đường, ... Quá trình xử lý và quảng bá các thông tin trong Metric container tuân theo các OF cụ thể được thiết kế trong quá trình triển khai.



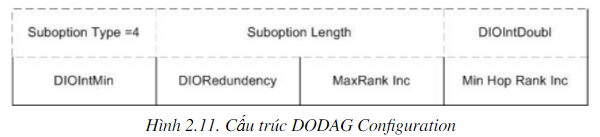
Destination prefix: là sub-option được sử dụng để theo dõi khả năng kết nối đến một nhóm các node trong mạng thông qua DODAC root hoặc qua một parent nằm gần DODAC root của nhóm node mạng đó. Sự theo dõi trở lên hữu ích khi trong mạng có nhiều hơn một DODAC root và đáp ứng các ứng dụng

khác nhau. Khi đó, nhờ quá trình theo dõi các destination prefix được cung cấp | bởi các DODAG, một node có thể đưa ra các quyết định tham gia vào nhiều

DODAG khác nhau, nhằm đáp ứng các mục đích riêng biệt. Nếu trong một bản tin DỊO cần theo dõi khả năng kết nối đến nhiều đích khác nhau, các trường destination prefix có thể được lặp lại để lưu các thông tin đó.



DAD Configuration là sub-option mang các thông tin phục vụ các quá trình cấu hình trong DODAG. Các thông tin này được tạo ra tại DODAC root và không bị thay đổi tại các node khác trong DODAG.



Các trường DIO Interval Min, DỊO Interval Doubling được sử dụng để cấu hình các bộ định thời quản lý việc gửi các bản tin DIO trong DODAG.

Các trường Max Rank Inc và Min Hop Rank Inc được sử dụng để cung cấp các miến giới hạn về rank khi các node trong DODAG tính toán và thiết lập rank.

#### 2.1.4.4. Bộ định thời DIO

Vấn đề quảng bá các bản tin DIO là vấn đề quan trọng nhất trong quá trình triển khai và xây dựng DODAG. Do đó, mỗi node trong mạng luôn duy trì một bộ định thời quản lý tốc độ quảng bá các bản tin DIO đến các node khác trong mạng.

Mỗi bộ định thời DIO luôn duy trì các thông số:

- I (Current Interval): Khoảng thời gian hiện hành được sử dụng trong việc

xác định khoảng thời gian định thời cho DIO.

- T (Timer): khoảng thời gian định thời có giá trị ngẫu nhiên trong khoảng [1/2; I).

- C (Counter): bộ đếm được sử dụng trong cơ chế quản lý các bản tin DIO dư thừa, như các bản tin DỊO được quảng bá từ các node có cùng rank hoặc từ những node có rank cao hơn. Cơ chế này làm giảm số lượng bản

tin phải xử lý và giảm dư thừa thông tin.

- I-min: khoảng thời gian định thời nhỏ nhất được tính bằng mili giây (ms).

Giá trị này được quy định bởi các DODAC root và bằng lũy thừa cơ số

hai của thông số DIO Interval Min trong bản tin DIO gửi từ DODAC root.

- I-doublings: số lần giá trị I được phép tăng gấp đôi trước khi được thiết

lập lại giá trị ban đầu. Khi một node tham gia vào một DODAC, node sẽ thực hiện việc kích hoạt bộ định thời DIO để quản lý tốc độ quảng bá các bản tin DIO tới những node xung quanh.

Các thông số khởi tạo được thiết lập chung cho toàn DODAG như sau:

- I-min và I-doubling bằng các giá trị tạo bởi DODAC root.

- C được thiết lập bằng 0.

- I thiết lập bằng I-min.

- T được chọn ngẫu nhiên trong khoảng [I/2; I].

Sau mỗi lần gửi DIO, bộ định thời DIO tự động nhân đôi khoảng thời gian I và lựa chọn một thời gian định thời T mới thuộc khoảng [I/2; I].

Khi bộ định thời DỊO được khởi động lại, các giá trị định thời được đưa về giá trị khởi tạo mặc định trong DODAG. | Bộ định thời DỊO được reset trong các trường hợp sau: - Khi node tìm thấy một thay đổi có ảnh hưởng đến vị trí của node trong DODAG:

* Khi node tham gia vào một DODAG mới trong mạng.
* Khi node thay đổi rank trong DODAG.
* Khi node nhận được một bản tin DIO từ một parent có sự thay đổi vê rank, DODAG.

• Khi node nhận được một gói dữ liệu từ một parent và các xung đột

trong quá trình truyền gói.

- Khi node nhận một bản tin multicast DIS.

- Khi node di chuyển từ DODAC này sang DODAG khác.

#### 2.1.4.5. Nhận và xử lý bản tin DIO

DIO đầu tiên được tạo ra tại DODAC root, sau đó được quảng bá đến các node khác trong mạng. Các node thành viên trong DODAG sử dụng các bản tin DIO nhận được để cập nhật thông tin DODAC, lựa chọn parent và quảng bá vị trí của nó trong DODAG.

Các phương thức xử lý DIO phải tuân theo những nguyên tắc sau:

- Một node chỉ tạo và gửi bản tin DIO sau khi tham gia một DODAG xác

định. Những node không phải DODAC root, chỉ được phép thay đổi giá trị các có điều khiển S (Destination Advertisements Stored), DTSN và DODAG

rank trong bản tin DIO nhận được từ DODAC root.

- DIO là hợp lệ nếu có Instance ID được đáp ứng.

Khi một node nhận được một bản tin DỊO từ trong mạng, node phải thực hiện kiểm tra tính hợp lệ của DIO đó. Nếu bản tin DỊO được đáp ứng, node mới tiếp tục thực hiện những hành động tiếp theo.

### ***2.1.5. Bản tin DAO***

DAO là bản tin được sử dụng để quảng bá thông tin của các đích, được gửi từ những node có rank cao hơn đến những node có rank thấp hơn dọc theo DODAG. DAO được sử dụng nhằm phục vụ cho những ứng dụng đòi hỏi luồng lưu lượng kiểu P2MP và P2P. Thông qua việc xử lý thông tin của những bản tin DIO nhận được, DODAC root và những node ở gần root có thể quản lý, cập nhật thông tin của những node nằm ở những rank cao hơn trong DODAC. Từ đó, có thể đưa ra những giải pháp định tuyến theo hướng downward. Do nội dung báo cáo chủ yếu tập trung nghiên cứu quá trình xây dựng DODAG và đánh giá khả năng định tuyến upward của giao thức, nên báo cáo chỉ đưa ra những khái niệm cơ bản về DAO và quá trình định tuyến downward.

#### 2.1.5.1. Cấu trúc bản tin DAO

Cấu trúc DAO gồm các trường sau:

- DẠO sequence: số bản tin DAO được một node gửi vào mạng.

- DẠO rank: rank của node tạo bản tin DAO.

- RPL Instance ID: Instance của node tham gia định tuyến.

- Route Tag: được sử dụng để cung cấp thứ tự ưu tiên khi lưu thông tin các

prefix

- Prefix Length: chiều dài prefix.

Rrcount: cho biết số mục trong Reverse Route Stack-ngăn xếp lưu các mục định tuyến theo hướng downward.

- DAO life time: thời gian sống hiệu lực của prefix, phục vụ cho việc xác định khả năng kết nối đến prefix.

- Destination Prefix: là một trường có chiều dài thay đổi, được sử dụng để nhận dạng một địa chỉ đích, một prefix, hoặc một nhóm địa chỉ multicast trong mạng.

- Reverse Route Stack: là trường có chiều dài có thể thay đổi, được sử dụng

để lưu thông tin của những địa chỉ prefix tham gia định tuyến. Khi một node thêm vào Reverse Route Stack, prefix của node được thêm vào danh

sách, đồng thời tăng giá trị RRcount.

- Các sub-option được sử dụng nhằm mở rộng các thành phần của bản tin DAO, tùy theo mục đích nghiên cứu và triển khai.

#### 2.1.5.2. Truyền bản tin DAO

 Các bản tin DAO được truyền từ các node ở rank cao đến các node ở rank thấp theo chiều upward, nhằm quảng bá các trạng thái định tuyến downward cho

những prefix của những nhóm hoặc những Sub-DODAG bên trong DODAG. Cơ chế định tuyến với DAO chỉ có thể hoạt động khi node đã tham gia ít nhất một DODAG trong mạng.

Cơ chế định tuyến và sử dụng DAO có thể chỉ được sử dụng trong từng DODAG, được quyết định bởi DODAG TOOL. Việc nhận dạng cơ chế này được

xác định thông qua một số tham số cấu hình được tạo trong bản tin DIO truyền đi từ DODAC root.

Khi cơ chế này được sử dụng, trong DODAG phải có ít nhất một số các node có khả năng lưu những thông tin từ DAO, bao gồm DODAC root. Khi cơ chế này không được sử dụng, các node trong DODAG không được phép tạo và xử lý các bản tin DAO.

Bản tin DAO được gửi từ node đến một hoặc một nhóm các DODAC parent của node trong DODAG. Những node có khả năng tham gia định tuyến, thực hiện lưu các thông tin trạng thái lấy được từ Reverse Route Stack vào bảng định tuyến. Mỗi mục trong bảng định tuyến cho biết những thông tin trạng thái của

các prefix như: địa chỉ IPv6, địa chỉ Interface, DẠO sequence, DẠO rank, DAO life time, ... Nhờ đó, một node có thể xác định trạng thái của những prefix:

- CONNECTED: trạng thái của bản thân node.

- REACHABLE: trạng thái của một neighbor, gồm 2 trạng thái:

• Confirmed: neighbor được kích hoạt, đã được xác nhận và có khả

năng định tuyến.

• Pending: neighbor được kích hoạt, đang trong quá trình xác nhận, tuy nhiên vẫn có thể sử dụng. Khi đó một bộ đếm Retry Counter được sử dụng để kiểm tra trạng thái của node.

- UNREACHABLE: mục có trạng thái không thể kết nối và bị loại bỏ.

Thông qua việc quản lý trạng thái kết nối của các prefix được cập nhật bởi cơ chế quảng bá đích, các node có rank thấp có thể xác định tuyến đường kiểu downward nhằm phục vụ cho những ứng dụng đòi hỏi giao tiếp kiểu point-tomultipoint hoặc point-to-point.

### ***2.1.6. Quá trình khởi tạo mạng***

Khi triển khai một mạng RPL, mỗi RPL Instance được thiết lập với một hoặc một số DODAC root. Các thông số định tuyến được thiết lập phù hợp với mục đích triển khai. Những DODAC root tự động thiết lập rank bằng 1(Root RANK), sau đó chúng định thời quảng bá các bản tin DỊO đến những node xung quanh để xây dựng DODAG của bản thân.

Trong pha khởi tạo, những node khác trong mạng có thể lựa chọn một trong hai chế độ: hoặc chúng giữ trạng thái silent và không gửi bất kỳ bản tin DỊO nào cho đến khi chúng tham gia vào một DODAG xác định; hoặc ngay lập tức tự thiết lập là DODAC root của một floating DODAC, sau đó gửi multicast các bản tin DIO đến các node khác trong mạng.

Trong quá trình này, mỗi node cũng có thể gửi multicast DIS đến các node xung quanh hoặc chờ nhận những bản tin DỊO được gửi đến. Khi triển khai mạng, cần thiết kế để các node trong mạng có khả năng đáp ứng được những cơ chế trên. Điều đó giúp cho quá trình khởi tạo nhanh chóng và linh động hơn.

### ***2.1.7. Quá trình định tuyến upward***

Trong RPL, quá trình định tuyến upward có vai trò then chốt, quyết định tính chất, hiệu năng hoạt động của mạng. Quá trình này dựa trên việc xử lý các bản tin DIO, xây dựng, xác định và duy trì các DODAC, từ đó mỗi node trong DODAG có thể xác định tuyến đường tối ưu để gửi dữ liệu về DODAC root một cách nhanh chóng và hiệu quả.

#### 2.1.7.1. DODAG parent và preferred parent

Trong RPL sử dụng ba khái niệm thể hiện mối quan hệ logic giữa các node trong mạng, bao gồm: neighbor, DODAC parent và preferred parent. Trong đó preferred parent của một node là một trong số những DODAG parent có các thông số định tuyến tốt nhất. Phương thức lựa chọn preferred parent được quy định bởi các hàm OF. Preferred parent sẽ được chọn là next-hop trong quá trình truyền gói.

#### 2.1.7.2. Quản lý neighbor, khám phá DODAG và lựa chọn DODAG parent

Trong RPL, mỗi node trong mạng tìm kiếm và quản lý các node neighbor trong phạm vi kết nối thông qua cơ chế “neighbor discovery in IPv6” (01. Theo đó, mỗi node trong mạng sẽ quản lý được các thông số trạng thái của những node xung quanh như khả năng kết nối, địa chỉ, tình trạng, ...

Khi một node được khởi tạo với vai trò DODAC root, node sẽ thực hiện việc tạo những bản tin DIO với rank bằng một và quảng bá chúng đến các node xung quanh. Những node gần root nhất khi nhận được DỊO từ DODAC root sẽ tham gia vào DODAG, cập nhật thông tin trong DỊO và xác định rank của bản thân trong DODAG. Sau đó node định thời chuyển tiếp các bản tin DỊO đến những node xung quanh, quảng bá vị trí và DODAG mà nó tham gia. Quá trình này được các thành viên trong DODAG lặp lại liên tục tại những thời điểm định thời. Do đó, quy mô DODAG được xây dựng mở rộng tới những node ở xa DODAG root, những node trong DODAG cập nhật được thông tin của các thành viên xung quanh trong DODAG.

Khi một node ở trạng thái tự do (chưa tham gia DODAG), node sẽ định thời gửi các bản tin DIS tới các neighbor để quảng bá instance mà nó có khả năng | tham gia. Đồng thời yêu cầu những node đáp ứng được phản hồi lại những thông | tin về DODAG mà chúng tham gia. Khi một neighbor nhận được bản tin DIS, nếu đã tham gia một DODAG phù hợp với Instance trong DIS, nó sẽ phản hồi bằng một bản tin DIO tới node gửi DIS . Sự phản hồi này sẽ diễn ra ngay sau thời điểm xử lý bản tin DIS mà không đợi tới thời điểm định thời DIO tiếp theo . Bên cạnh đó , nếu DIS chứa địa chỉ multicast , node sẽ khởi động lại bộ định thời DIO để giảm thời gian gửi DIO tiếp theo . Cơ chế này giúp sự quảng bá DODAG linh động và hiệu quả hơn .

Khi trong phạm vi kết nối của node xuất hiện nhiều hơn một DODAG có khả năng đáp ứng instance mà node tham gia , cơ chế lựa chọn DODAG là sự kết hợp giữa cơ chế so sánh độ ưu tiên của các DODAG và các quy luật được quy định trong OF . Độ ưu tiên của các DODAG được thể hiện thông qua trường DODAG preferred trong bản tin DIO được gửi bởi các DODAG root . Nhờ đó , mỗi node trong mạng đều có thể lựa chọn những DODAG được ưu tiên nhất và thỏa mãn tốt nhất những yêu cầu định tuyến được đặt ra .

Quá trình lựa chọn DODAG phải tuân theo một số quy tắc sau :

- DODAG tham gia phải đáp ứng instance của node .

- DODAG tham gia phải có độ ưu tiên cao nhất và node có rank thấp nhất .

- DODAG tham gia phải thỏa mãn các OF của node .

Sau quá trình khám phá và tham gia vào DODAG trong mạng , mỗi node xử lý những thông tin trong DIO mà chúng nhận được , cập nhật rank và chọn những parent từ những node thành viên xung quanh của DODAG . Các parent của một node phải thỏa mãn những tính chất sau :

- Là một neighbor của node .

- Là thành viên trong cùng DODAG với node .

-Có rank thấp hơn rank của node .

#### 2.1.7.3 . Tính toán rank và sự di chuyển của các node

Lựa chọn rank là quá trình xác định vị trí của node so với DODAG root trong DODAG , đồng thời có ảnh hưởng đến mối quan hệ của node với các node khác trong mạng . Quy tắc tính toán rank được quy định trong các hàm OF tương ứng với mỗi Instance cụ thể . Tuy nhiên , Rank của node phải thỏa mãn những quy tắc sau :

- Rank của node trong mạng luôn lớn hơn 1 và nhỏ hơn giới hạn lớn nhất (Rank Max) được quy định tùy theo quy mô và mục đích triển khai.

- Rank của node phải luôn lớn hơn rank của tất cả các parent của node.

- Node có thể quảng bá rank thấp hơn hoặc cao hơn rank mà nó quảng bá trong các bản tin trước đó. Sự thay đổi đó phụ thuộc sự thay đổi rank của các parent trong DODAG.

- Node có thể quảng bá rank bằng rank max tại mọi thời điểm. Khi một node quảng bá rank bằng rank max, tương đương với sự kiện node không là thành viên của bất kỳ DODAG nào trong mạng.

- Tại mọi thời điểm, node có thể tham gia vào một DODAG mới trong cùng RPL Instance và thay đổi rank phù hợp. Trong thời gian trước khi node quảng bá các bản tin DIO cho DODAG mới, node vẫn tiếp tục gửi các gói đến các parent trong DODAG cũ.

Quá trình di chuyển trong DODAG kéo theo sự thay đổi rank của các node, khi node có khả năng kết nối tới những node gần DODAG root hơn hoặc khi node không thể kết nối đến các parent và buộc phải kết nối với DODAG thông qua những node có rank cao hơn.

Khi một node dịch chuyển theo hướng upward, node phải cập nhật, thay đổi các thông số và mối quan hệ của các parent, sibling cũ, đồng thời thiết lập các mối quan hệ parent và sibling mới. Trong quá trình này, node không cần phải tách ra khỏi DODAG.

Khi một node trong mạng phải di chuyển theo hướng downward, do mất kết nối với tất cả các node parent hiện tại, trước hết node phải tách ra khỏi DODAG mà node đang là thành viên. Sau đó thực hiện quá trình tái tham gia DODAG, cùng với những lựa chọn parent, sibling và rank mới.

### ***2.1.8. Truyền gói***

Sau quá trình tham gia và xây dựng DODAG, các node trong DODAG tạo các gói dữ liệu và bắt đầu gửi gói đến DODAG root. Để gửi gói đến DODAG root, node phải lựa chọn một node

trong route table làm next-hop và gửi gói đến next-hop được chọn. Việc lựa chọn next-hop phải tuân theo những quy luật cụ thể, nhằm mục đích truyền gói hiệu quả, giảm khả năng mất gói, tránh các vòng lặp và các xung đột trong mạng.

Các quy tắc lựa chọn next-hop được quy định như sau:

- Next-hop là một node đã tham gia một DODAG đáp ứng được RPL Instance ID trong header của gói tin đựợc forward.

- Nếu một node hoạt động với giao thức định tuyến đựợc ưu tiên hơn RPL thì chọn node đó làm next-hop.

- Nếu node là thành viên của một DODAG và có một parent là default route thì chọn parent đó là next-hop.

- Nếu node là thành viên của một DODAG và tất cả các parent tạm thời không thể kết nối được, node chọn một trong số các sibling làm next-hop. Nếu không có sibling, gói sẽ bị hủy.

- Node không được chọn các node có rank cao hơn làm next-hop.

Khi truyền gói, tham số Time to live (TTL) được sử dụng để theo dõi và loại bỏ những gói không truyền được tới đích, đồng thời hạn chế các vòng lặp có thể xảy ra. Tại mỗi node gói tin được forward, TTL được giảm 1 đơn vị. Khi TTL bằng 0, gói sẽ bị hủy mà không được truyền đến DODAG root.

## **2.2. Tổng quan về Contiki RPL và hoạt động của RPL trên Contiki**

Hiện nay, giao thức RPL đang trong quá trình hoàn thiện nhằm tạo ra một chuẩn giao thức định tuyến mới, có nhiều đặc tính tối ưu hơn các giao thức định tuyến trước đây. Do đó, việc xây dựng những chương trình mô phỏng, đánh giá các đặc tính, hiệu năng của RPL có ý nghĩa đặc biệt quan trọng.

Với mục đích đó, nhóm phát triển Contiki đã nghiên cứu và triển khai xây dựng module Contiki RPL nhằm đáp ứng nhu cầu nghiên cứu của những người đang sử dụng Contiki làm công cụ tìm hiểu, đánh giá RPL. Phiên bản Contiki RPL đầu tiên được đưa ra vào 5/5/2010, cung cấp những hỗ trợ đầu tiên cho người nghiên cứu RPL và đang tiếp tục nhận được sự đóng góp của cộng đồng sử dụng Contiki trên toàn thế giới.

### ***2.2.1. Hệ điều hành Contiki***

Hệ điều hành Contiki là hệ điều hành mã nguồn mở, được nghiên cứu, thiết kế và phát triển bởi một nhóm các nhà khoa học tại Viện Khoa học máy tính Thụy Điển SICS (Swedish Institute of Computer Science), người đứng đầu là Adam Dunkels. Nhóm phát triển Contiki gồm nhiều thành viên đến từ SICS, CISCO, cùng nhiều tổ chức và các trường đại học khác trên thế giới. Hệ điều hành Contiki được thiết kế cho các vi điều khiển có bộ nhớ nhỏ, với thông số 2KB RAM và 40KB ROM. Nhờ đó, Contiki được sử dụng cho các hệ thống nhúng và các ứng dụng trong mạng cảm biến không dây. Contiki bắt đầu được nghiên cứu từ năm 2001 và phát hành phiên bản đầu tiên Contiki 1.0 năm 2003. Phiên bản hiện nay của Contiki là 3.0, với nhiều thay đổi, bổ sung và phát triển vượt bậc. Trong thực tế, Contiki đã được ứng dụng trong nhiều dự án như giám sát đường hầm xe lửa, theo dõi nước trong biển Baltic, … Nhiều cơ chế, ý tưởng trong Contiki đã được ứng dụng rộng rãi trong công nghiệp. Điển hình như mô hình uIP được phát hành năm 2001 đã được sử dụng trong hệ thống ứng dụng của hàng trăm công ty trong các lĩnh vực hàng hải, thông tin vệ tinh, khai thác dầu mỏ, … mô hình Protothreads được công bố lần đầu tiên năm 2005, đến nay đã được sử dụng trong nhiều ứng dụng như bộ giải mã kỹ thuật số và thiết bị cảm biến rung không dây.

Hệ điều hành Contiki được lập trình bằng ngôn ngữ C, hoạt động dựa trên cơ chế event-driven và có những đặc điểm phù hợp với các hệ thống nhúng và mạng cảm biến không dây:

- Contiki được chia thành nhiều module hoạt động độc lập. Nhờ đó các ứng dụng có thể sử dụng các module một cách linh động và chỉ load những module cần thiết.

- Cơ chế hoạt động điều khiển sự kiện làm giảm năng lượng tiêu hao và hạn chế dung lượng bộ nhớ cần sử dụng.

- Có thể sử dụng IP trong mạng cảm biến thông qua uIP stack được xây dựng dựa trên nền TCP/IP.

- Có những module cho phép ước lượng và quản lý năng lượng một cách hiệu quả.

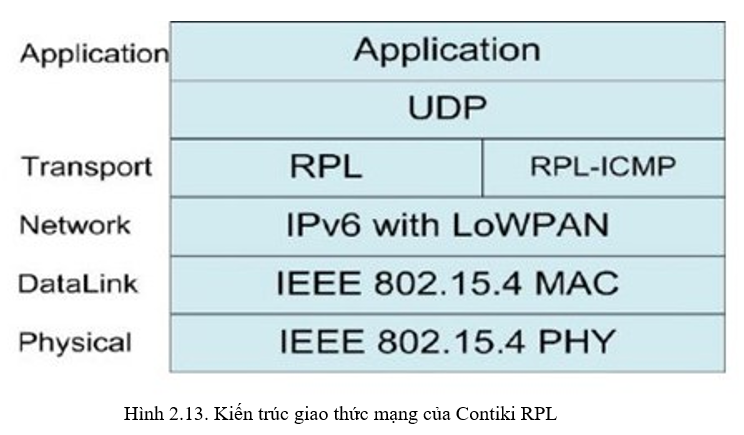
- Các giao thức tương tác giữa các lớp và các node trong mạng dễ dàng hơn.

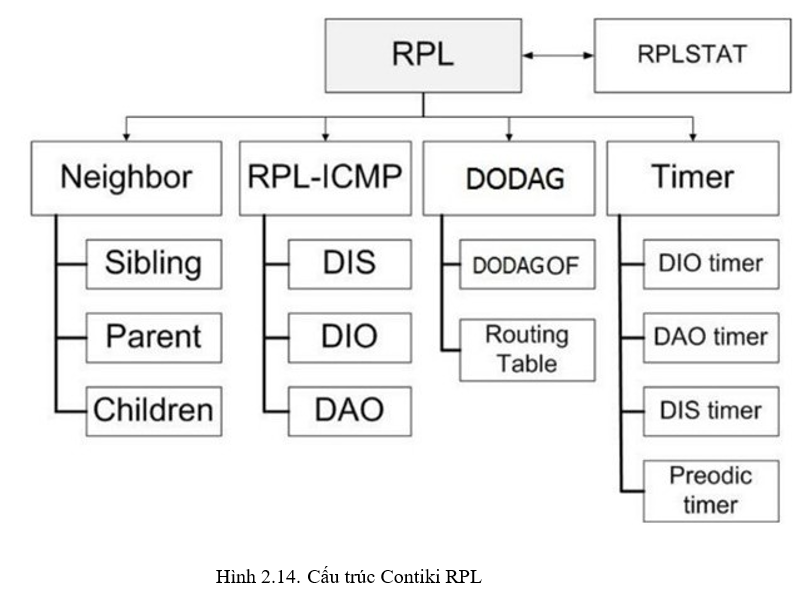
- Sử dụng RIME stack phục vụ các giao thức dành cho mạng năng lượng thấp một cách hiệu quả.

Bên cạnh đó, Contiki còn cung cấp những công cụ hỗ trợ mô phỏng với giao diện đơn giản, dễ sử dụng và hỗ trợ tốt những thiết bị trong thực tế, phục vụ những mục đích nghiên cứu, mô phỏng và triển khai những giao thức mới.

### ***2.2.2. Contiki RPL***

Contiki RPL là một bộ phận trong stack uIP, hoạt động trên IPv6 và sử dụng giao thức truyền tin không tin cậy UDP/IP trên nền 6LoWPAN.





Hình 2.14 cho thấy cấu trúc và các module của Contiki RPL. Contiki RPL bao gồm những module chính:

- Neighbor: quản lý thông tin về các node trong phạm vi kết nối, đồng thời quản lý mối quan hệ trong cùng DODAG của node với các neighbor trong danh sách. Các mối quan hệ cùng DODAG gồm: parent, sibling, chidrent. Những thông tin được theo dõi của mỗi neighbor gồm: DODAG tham gia, DODAG Rank, địa chỉ Ipv6 của mỗi neighbor trong mạng.

- RPL-ICMP: module quản lý các bản tin điều khiển RPL – ICMP gồm 3 loại bản tin chính: DIS, DIO, DAO. Đồng thời quản lý các chức năng kiểm tra xử lý các bản tin ICMP đầu vào, tạo và gửi các bản tin ICMP đầu ra.

∙ uip\_rpl\_input(void): kiểm tra bản tin đầu vào, phân loại bản tin RPL-ICMP và gọi các hàm xử lý tương ứng.

∙ dis\_input(void): hàm được gọi khi bản tin nhận được là DIS, xử lý các thông tin trong DIS nhận được và gọi những hàm xử lý cần thiết.

∙ dis\_output(void): tạo một bản tin DIS, sau đó gửi multicast hoặc unicast tới những node khác trong mạng.

∙ dio\_output(void): tạo mới một bản tin DIO và gửi tới những node khác trong mạng.

∙ dio\_input(void): hàm xử lý và lưu những thông tin trong bản tin DIO nhận được.

∙ dao\_output(void): tạo bản tin DAO, gửi tới những node parent trong DODAG. Bản tin DAO chỉ được gửi sau khi node đã là thành viên của một DODAG cụ thể.

∙ dao\_input(void): hàm kiểm tra và lưu những thông tin trong bản tin DAO nhận được từ những node khác trong mạng.

- Timer: là module quản lý những bộ định thời trong Contiki RPL, bao gồm 4 bộ định thời chính, được xây dựng dựa trên những bộ định thời sẵn có của Contiki

∙ Periodic Timer: có chức năng định thời kiểm tra và làm mới bảng định tuyến, xóa những node trong bảng định tuyến có lifetime bằng 0.

∙ DIS Timer: quản lý gửi các bản tin DIS. Khi hoạt động định thời diễn ra, DIS timer sẽ gọi hàm dis\_output() để tạo một bản tin DIS và gửi vào mạng.

∙ DIO Timer: định thời gửi bản tin DIO vào mạng, nhằm xây dựng và duy trì DODAG trong mạng, xác định mối quan hệ giữa các node. Khi bộ định thời hết hạn sẽ gọi hàm dio\_output() để tạo và gửi DIO tới những node khác trong mạng.

∙ DAO Timer: Định thời tạo và gửi bản tin DAO sau khi node đã tham gia vào một DODAG trong mạng. DODAG: module chính, quản lý các yếu tố ảnh hưởng đến quá trình tham gia, xây dựng, duy trì DODAG. Module này cung cấp những phương thức quản lý DODAG, xử lý thông tin, tìm kiếm, xây dựng bảng định tuyến:

∙ DODAG OF: cung cấp phương thức xác định Object function của DODAG mà node tham gia, cung cấp các phương thức so sánh, xác định vị trí và các thông số của node trong DODAG: rank, OCP, hàm lựa chọn best-parent → next-hop.

∙ Routing Table: cung cấp các phương thức xây dựng bản định tuyến, lựa chọn default route, lọc các route mất kết nối. Ví dụ: rpl\_set\_default\_route(), poison\_routes(), . . .

∙ Các phương thức thao tác với DODAG: rpl\_join\_DODAG(), rpl\_free\_DODAG(), rpl\_find\_DODAG(), rpl\_get\_DODAG(), . . .

∙ Các phương thức sửa chữa, duy trì DODAG: global\_repair(), rpl\_repair\_DODAG(), . . .

∙ Các phương thức xử lý thông tin: rpl\_process\_dio().

∙ Các phương thức quản lý các quan hệ trong neighbor: rpl\_remove\_neighbor(), rpl\_find\_neighbor(), rpl\_ds6\_neighbor\_callback(), rpl\_find\_best\_parent(), . . . - RPLSTAT: module được xây dựng phục vụ mục đích thống kê, lưu các thông số định tuyến như số lượng gói tin, số lượng các bản tin thất lạc. Qua đó, người sử dụng có thể đánh giá được một số những tiêu chí như Packet Delevery Rate (PDR), Packet Delevery Cost (PDC), …

### ***2.2.3. Định tuyến trong Contiki RPL***

Trong một mạng triển khai Contiki RPL định nghĩa hai loại thiết bị: udp-client và udp-server, hoạt động theo kiểu truyền tin không tin cậy UDP. Trong pha đầu tiên khi triển khai mạng, một số node trong mạng được cấu hình là những bộ tập trung dữ liệu - DODAG root cho những RPL Instance cụ thể. Những node khác trong mạng là những node udp-client, không thực hiện chức năng thu thập dữ liệu mà thực hiện chức năng tạo và forward gói về DODAG root. Những node trong mạng Contiki RPL hoạt động tuân theo những ràng buộc sau:

- Tất cả các node trong mạng đều phải có địa chỉ IPv6 riêng biệt, được thiết lập bởi uIP stack trong Contiki. Với 64bit đầu tiên là global prefix được sử dụng để phân biệt các mạng, 64 bit còn lại được dùng định địa chỉ của node (kết hợp với địa chỉ MAC của thiết bị).

- Các udp-server phải là các DODAG root của các DODAG trong mạng. Các udp-server chỉ đáp ứng một RPL Instance theo mục đích triển khai và có một mã nhận dạng DODAGID duy nhất. DODAGID là một địa chỉ IPv6 được thiết lập trước. Các DODAGID có thể được lặp lại ở những RPL Instance khác nhau.

- Tất cả các node trong mạng đều có khả năng tạo, gửi, nhận và xử lý các bản tin ICMP.

- Các node trong mạng có khả năng hoạt động theo cả hai cơ chế: gửi DIS hoặc chờ nhận DIO trong pha khởi động.

- Mỗi udp-client chỉ đáp ứng một loại RPL Instance. Vấn đề triển khai một node có khả năng đáp ứng nhiều RPL Instance khác nhau vẫn đang trong quá trình nghiên cứu và phát triển.

- Các udp-client tìm kiếm và chỉ gia nhập những DODAG tương thích với RPL Instance mà node có khả năng đáp ứng.

- Các node trong mạng tự động cấu hình và xây dựng DODAG. Từ đó định tuyến và truyền gói về DODAG root dọc theo DODAG.

- Các udp-client tạo và truyền các gói multihop trong DODAG. Mỗi gói tin chỉ được gửi đến một DODAG parent duy nhất trong DODAG và không được gửi lại nếu bị mất gói.

- Tất cả các node không bao giờ được gửi gói tin đến những node có rank cao hơn trong cùng một DODAG.

- Các node trong mạng đều có khả năng gửi các bản tin ICMP đến các địa chỉ unicast, multicast trong mạng.

- Các node có khả năng sử dụng cơ chế Neighbor Discovery trong IPv6 để quản lý sự xuất hiện của các node khác trong phạm vi kết nối.

#### 2.2.3.1. Những yêu cầu cần đạt được khi triển khai Contiki RPL

Ngoài những yêu cầu cơ bản trong định tuyến, một mạng Contiki RPL cần đạt được một số yêu cầu sau:

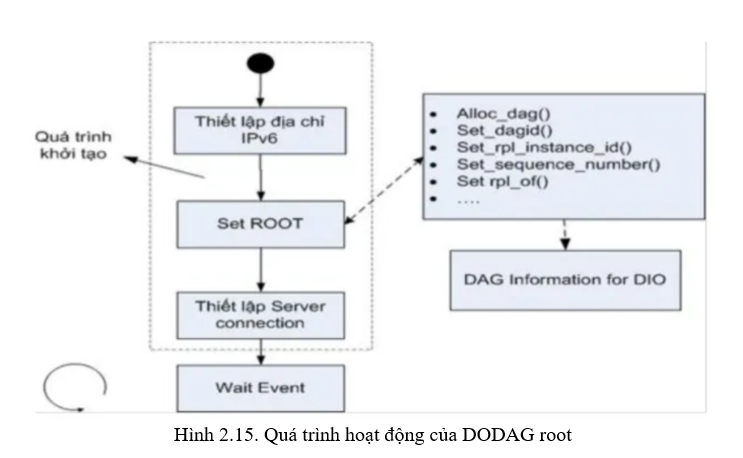
- Mỗi mạng Contiki RPL có thể gồm nhiều RPL Instance, mỗi RPL Instance có thể có nhiều DODAG.

- Mạng phải có khả năng tự cấu hình, xây dựng DODAG.

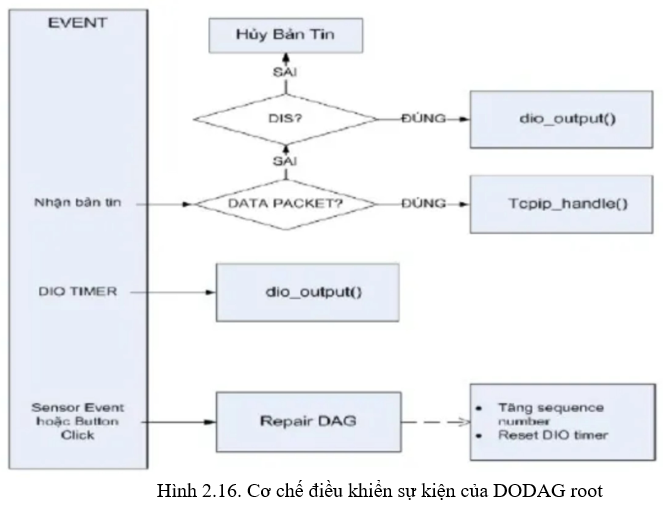
- Các DODAG trong mạng phải có khả năng tự sửa chữa và duy trì DODAG.

#### 2.2.3.2. Hoạt động của DODAG root / UDP-Server

Trong pha đầu tiên khi khởi tạo mạng, những node được lựa chọn làm DODAG root thực hiện quá trình thiết lập địa chỉ IPv6, cổng kết nối và những thông số định tuyến của DODAG như DODAGID, RPLInstanceID, Object function, … (hình 2.15). Những thông tin của DODAG được DODAG root đưa vào bản tin DIO và quảng bá tới những node khác trong mạng. Sau khi xây dựng DODAG, DODAG root nhận các gói tin được gửi về từ những node khác trong DODAG.

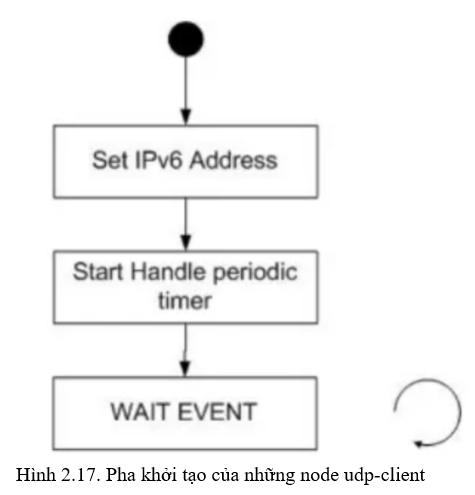


Sau quá trình khởi tạo, DODAG root hoạt động theo cơ chế event-driven, tùy theo những sự kiện đến cụ thể, DODAG root đưa ra những hoạt động tương ứng. Cơ chế hoạt động event-driven của DODAG root như trong hình 2.16.

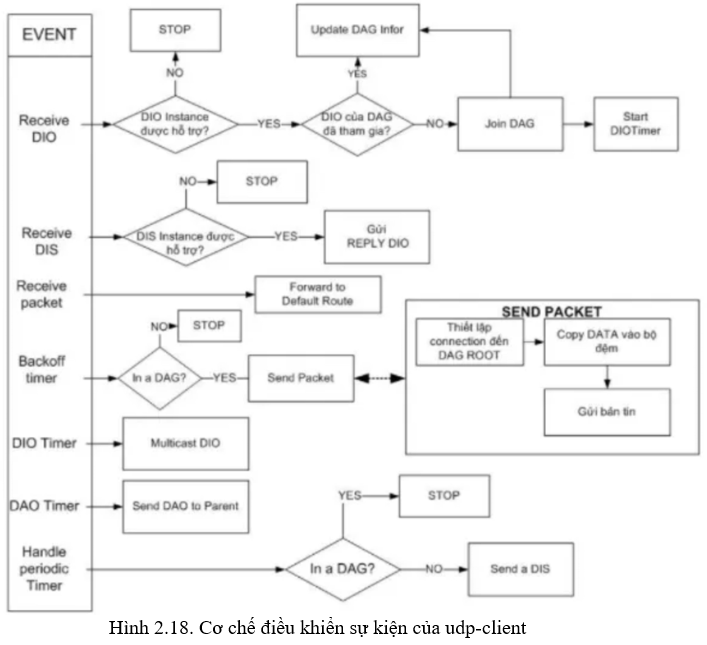


#### 2.2.3.3. Hoạt động của các node client trong DODAG / UDP-client

Trong pha khởi tạo, những udp-client thực hiện quá trình thiết lập địa chỉ IPv6 và khởi động bộ định thời handle-periodic-timer có chức năng định thời gửi các bản tin DIS tới các node khác trong mạng khi node chưa tham gia vào bất kỳ DODAG nào.

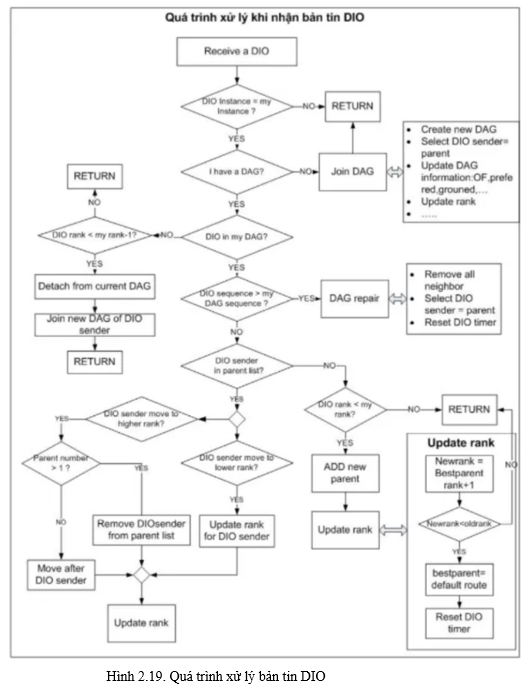


Sau pha khởi tạo, mỗi node đã thiết lập được những thông tin ban đầu và bắt đầu nhận và xử lý những bản tin ICMP trong mạng. Khi đó node tham gia những DODAG hỗ trợ và hoạt động theo cơ chế điều khiển sự kiện. Các hoạt động của node theo những sự kiện đến như: nhận các bản tin ICMP, nhận gói từ node khác trong mạng, các sự kiện định thời. Các hoạt động tương ứng của node được chỉ ra trong hình 2.18:



#### 2.2.3.4. Cơ chế xử lý bản tin DIO và xây dựng DODAG

Hoạt động gửi, nhận, xử lý bản tin DIO là hoạt động quan trọng nhất, quyết định đến khả năng xây dựng DODAG và định tuyến của một mạng RPL. Trong Contiki RPL, quá trình tạo và gửi DIO được bắt đầu tại những DODAG root. Quá trình nhận và xử lý DIO giúp cho các node trong mạng nhận dạng những DODAG phù hợp, tham gia DODAG và lựa chọn parent trong DODAG



Một số nguyên tắc khi xử lý bản tin DIO:

- Bản tin DIO phải có Instance ID được node hỗ trợ thì mới được tiếp tục xử lý.

- Khi node chưa tham gia bất kỳ DODAG nào thì tham gia vào DODAG của node gửi DIO.

- Nếu bản tin DIO của một DODAG mà node không phải thành viên và nếu việc tham gia vào DODAG mới khiến node có rank thấp hơn thì node sẽ gia nhập DODAG mới.

- Nếu DIO đến từ một node trong cùng một DODAG và sequence number đã được tăng lên thì node tham gia quá trình sửa chữa DODAG.

- Nếu DIO trong cùng DODAG, quảng bá rank thấp hơn rank của node nhưng chưa có trong danh sách parent thì thêm node gửi DIO làm parent, sau đó cập nhật vị trí của node trong DODAG.

- Nếu DIO từ một parent trong DODAG, trước hết cần cập nhật thông tin của parent, sau đó tùy thuộc vào sự thay đổi của các parent đưa ra hành động phù hợp.

- Khi rank thay đổi, phải reset bộ định thời DIO.

- Chỉ chuyển lên rank cao hơn theo một parent khi không còn parent thay thế.

- Best parent là node thuộc DODAG có thứ tự ưu tiên cao nhất và có rank thấp nhất.

# **CHƯƠNG 3. MÔ PHỎNG HOẠT ĐỘNG CỦA RPL TRÊN CONTIKI RPL**

## **3.1. Mục tiêu mô phỏng**

Trong quá trình thực hiện mô phỏng, báo cáo hướng tới các mục tiêu sau:

- Phân tích quá trình gửi nhận các bản tin (DIS, DIO) qua đó đánh giá quá trình hình thành topo mạng

- Đánh giá độ tin cậy của giao thức RPL thông qua tỉ lệ truyền gói thành công – Packet Delivery Ratio (PDR)

- Đánh giá mức tiêu thụ năng lượng thông qua chi phí truyền gói – Packet Delivery Cost (PDC)

## **3.2. Tham số cài đặt mô phỏng**

| STT | Nội dung | Tham số |
| --- | --- | --- |
| 1 | Thời gian mô phỏng | 5400s |
| 2 | Diện tích mô phỏng | 60000 m2 |
| 3 | Cự ly giữa các node | Nhỏ hơn 50m so với node parent của nó |
| 4 | Node mô phỏng | Thiết bị Zolertia z1 |
| 5 | Số lượng node | 1 node đóng vai trò root thì ít nhất 15 node đóng vai trò client |
| 6 | Vị trí node | Phân bố ngẫu nhiên nhưng đảm bảo khoảng cách các node nhỏ hơn 50m so với parent của nó |
| 7 | Định tuyến | Tối thiểu hóa số chặng truyền gói tin từ các node tới DODAG root |
| 8 | Tham gia DODAG của node | Tại một thời điểm, trong một instance, mỗi node chỉ tham gia một DODAG trong instance đó |
| 9 | Thời gian gửi bản tin DATA | 60s+Random(1:60) |
| 10 | Thời gian gửi bản tin điều khiển | Mặc định |
| 11 | Kích thước gói tin | Mặc định |

## **3.3. Kịch bản mô phỏng**

Trong phần này báo cáo sẽ đưa ra 3 kịch bản mô phỏng

- Kịch bản 1: 1 Instance – 1 DODAG root – 15 node client

- Kịch bản 2: 1 Instance – 2 DODAG root – 30 node client

Từ 2 kịch bản mô phỏng, báo cáo sẽ đưa ra các kết luận

- RPL là giao thức có các thông số hiệu năng tốt và đáng tin cậy thông qua việc phân tỉ lệ truyền gói thành công PDR và chi phí truyền gói PDC.

- Khi trong mạng có nhiều Instance, các node trong mạng có khả năng xác định DODAG nó sẽ tham gia. Nhờ đó, quá trình định tuyến trở nên hiệu quả, giảm các bản tin dư thừa trong mạng.

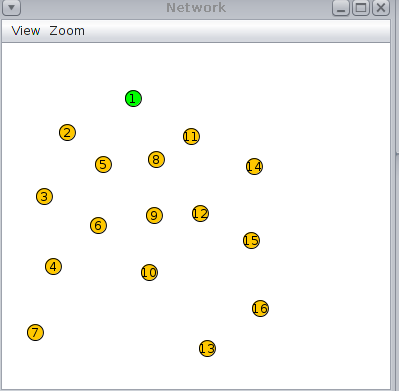
## **3.4. Phân tích**

### ***3.4.1. Kịch bản 1: 1 Instance – 1 DODAG root – 15 Node client***

#### 3.4.1.1. Quá trình hình thành topo mạng

a. Quá trình gửi nhận bản tin DIS

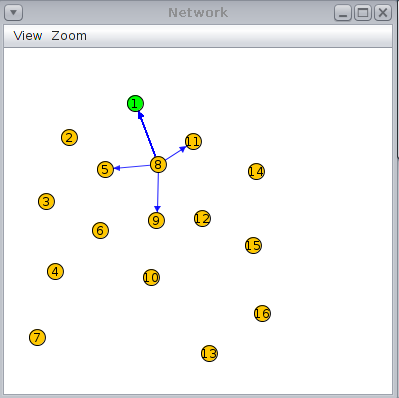
Mô phỏng với một mạng gồm 1 DODAG root và 15 node đóng vai trò là những client tham gia quá trình định tuyến. Vị trí các node trong mạng được phân bố như hình dưới.



Sau quá trình khởi tạo mạng các node gửi bản tin multicast DIS tới các node

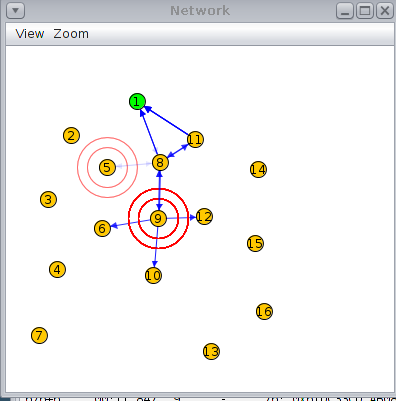
hàng xóm để được xin tham gia vào DODAG. Các bản tin này sẽ bị các node từ

chối do các node chưa tham gia vào một DODAG nào cụ thể.

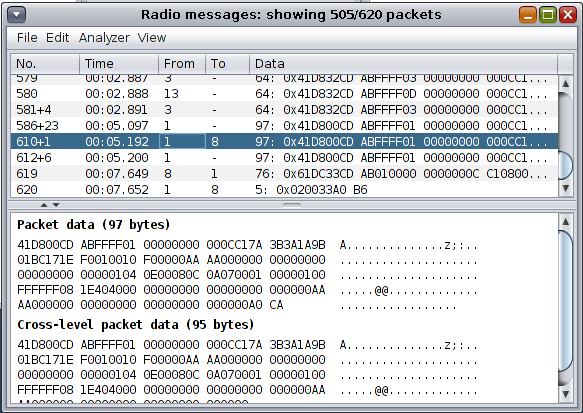


B, Quá trình downward

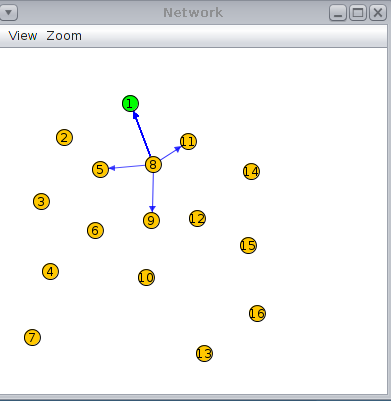
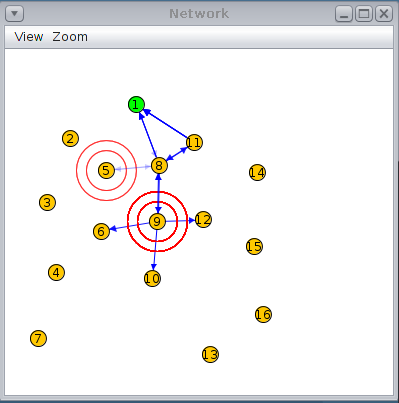
Đầu tiên, DODAG root sẽ gửi bản tin multicast DIO tới các node hàng xóm của nó để quảng bá. Cụ thể ở ví dụ này là từ node 1 tới node 5, 8, 11

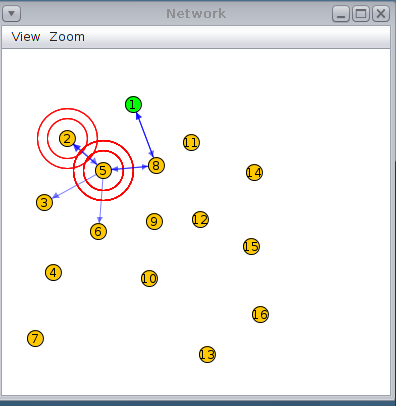
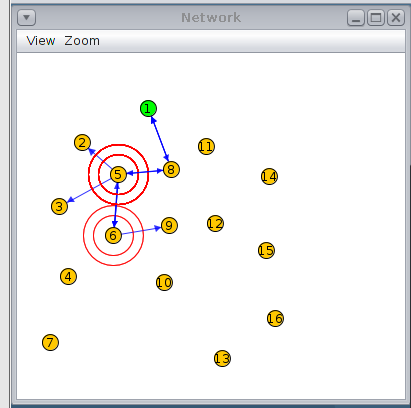


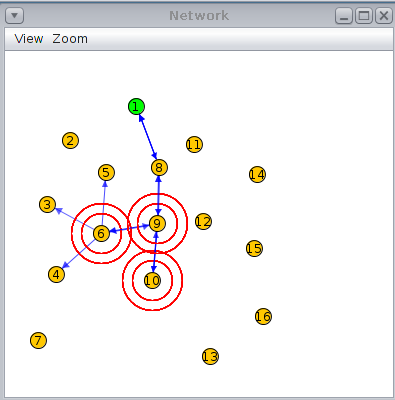
Note 1 gửi bản tin DIO tới node 5, 8, 11 với các giá trị



Tương tự, sau khi tham gia vào DODAG node 5, node 8, node 11 sẽ giửi multicast tới các node hàng xóm của nó là các node 2, 3, 6, 9, 12, 14, 15. Cứ như vậy DODAG được hình thành.



#### 3.4.1.2. Tỉ lệ truyền gói thành công

Tỉ lệ truyền gói (PDR) được tính như sau:

PDR=

Các số liệu được thống kê từ chương trình mô phỏng với 1 DODAG root và 15 client trong thời gian 5400 giây. Các bảng thông kê số lượng gói truyền tại các node và số gói nhận tương ứng tại DAG root như sau:

| STT | Rank | Node ID | Số bản tin gửi | Số bản tin đến root | PDR |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | 2 | 89 | 89 | 1.000 |
| 2 | 1 | 5 | 89 | 89 | 1.000 |
| 3 | 1 | 8 | 89 | 89 | 1.000 |
| 4 | 1 | 11 | 89 | 89 | 1.000 |
| 5 | 1 | 14 | 89 | 89 | 1.000 |
| 6 | 2 | 3 | 89 | 89 | 1.000 |
| 7 | 2 | 6 | 89 | 88 | 0.989 |
| 8 | 2 | 9 | 89 | 89 | 1.000 |
| 9 | 2 | 12 | 89 | 89 | 1.000 |
| 10 | 2 | 15 | 89 | 88 | 0.989 |
| 11 | 3 | 4 | 89 | 87 | 0.978 |
| 12 | 3 | 10 | 89 | 88 | 0.989 |
| 13 | 3 | 16 | 89 | 89 | 1.000 |
| 14 | 4 | 7 | 89 | 88 | 0.989 |
| 15 | 4 | 13 | 89 | 89 | 1.000 |

Từ bảng trên ta tổng hợp được bảng dưới:

| STT | Rank | Tổng số bản tin gửi | Tổng số bản tin đến Root | PDR |
| --- | --- | --- | --- | --- |
| 1 | 1 | 445 | 445 | 1.000 |
| 2 | 2 | 445 | 444 | 0.997 |
| 3 | 3 | 267 | 264 | 0.989 |
| 4 | 4 | 178 | 177 | 0.994 |

Tỉ lệ truyền gói thành công của toàn mạng:

PDR==0.996

Từ bảng các bảng trên ta có thể thấy tỉ lệ truyền gói thành công là khá cao và tương đối đồng đều giữa các mức rank 1,2,3,4 trong DODAG. Tức là mức rank không ảnh hướng đến tỉ lệ truyền gói thành công. Điều đó đồng nghĩa độ tin cậy khi gửi gói trong mạng RPL với kịch này là khá cao và ổn định.

#### 3.4.1.3. Chi phí truyền gói

Chi phí truyền gói (PDC) được tính như sau:

PDC=

| Chức năng | Node ID | Bản tin DATA | Bản tin ICMP | Tổng số bản tin |
| --- | --- | --- | --- | --- |
| DAG root | 1 | 0 | 17 | 17 |
| Client | 2 | 89 | 124 | 213 |
| Client | 3 | 89 | 122 | 211 |
| Client | 4 | 89 | 146 | 235 |
| Client | 5 | 89 | 130 | 219 |
| Client | 6 | 89 | 115 | 204 |
| Client | 7 | 89 | 115 | 204 |
| Client | 8 | 89 | 116 | 205 |
| Client | 9 | 89 | 132 | 221 |
| Client | 10 | 89 | 131 | 220 |
| Client | 11 | 89 | 133 | 222 |
| Client | 12 | 89 | 135 | 224 |
| Client | 13 | 89 | 124 | 213 |
| Client | 14 | 89 | 127 | 216 |
| Client | 15 | 89 | 113 | 202 |
| Client | 16 | 89 | 124 | 213 |
| Tổng số bản tin | | 1335 | 1904 | 3239 |

Bảng trên thống kê số bản tin DATA được gửi từ các node trong mạng đến DODAG root trong thời gian mô phỏng 5400.

-Số bản tin DATA nhận được tại DODAG root trong thời gian mô phỏng 5400s là 1330 tin.

-Tổng tất cả các loại bản tin được gửi trong mạng là 3239 bản tin.

Chi phí truyền gói

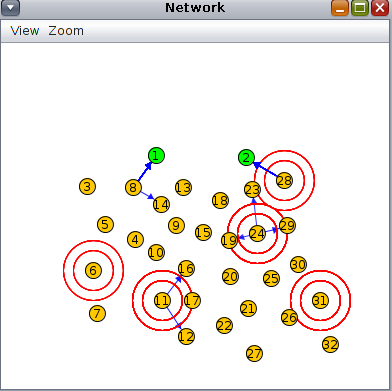
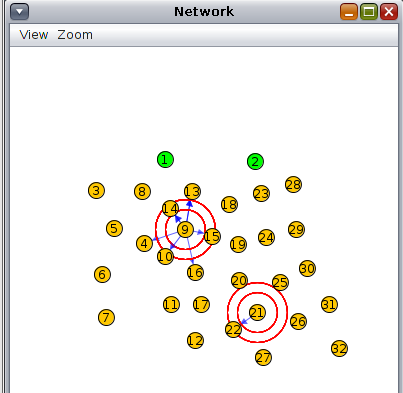
PDC==2,435

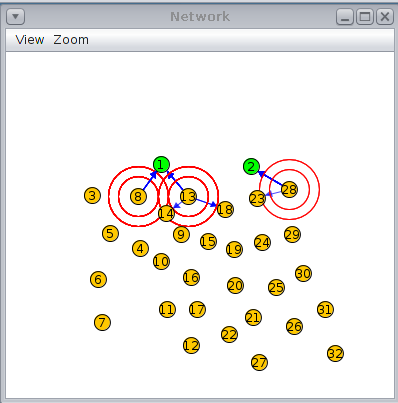
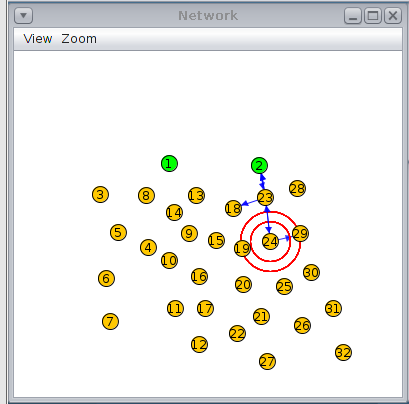
Có thể nhận thấy, PDC cho toàn mạng RPL là khá thấp. Do đặc điểm định thời gửi các bản tin ICMP đã làm giảm số lượng bản tin điều khiển trong mạng. Điều này cho thấy RPL là giao thức có các thông số hiệu năng tốt và đáng tin cậy.

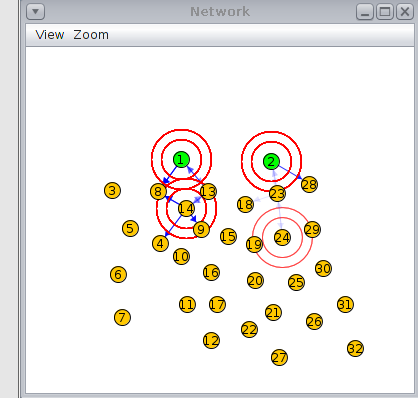
### ***3.4.2. Kịch bản 2: 1 Instance – 2 DODAG root – 30 Node client***

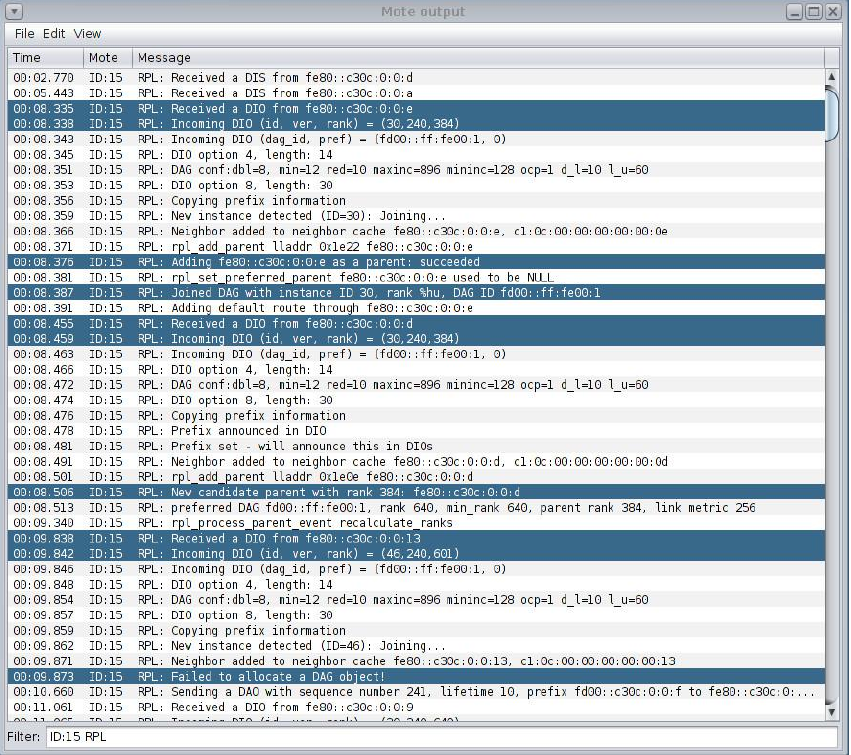
#### 3.4.2.1. Quá trình hình thành topo mạng

Tương tự như kịch bản số 1, các node sẽ lần lượt gửi các bản tin DIS, DIO để xây dựng topo mạng.





Dựa vào hình trên có thể thấy được quá trình tham gia vào DODAG của node 15 trong kịch bản 1 Instance – 2 DODAG root

- Node 15 nhận bản tin DIO từ node 14

* Kiểm tra thông tin tham gia vào DODAG của node 14. Cụ thể với Instance ID = 30, DODAG root là node 1.
* Nhận node 14 làm DODAG parent.
* Chỉ Default Route về node 14

- Node 15 nhận bản tin DIO từ node 13

* Kiểm tra thông tin thấy node 13 cùng DODAG và có rank thấp hơn.
* Nhận node 13 làm Candidate parent

- Node 15 nhận bản tin DIO từ node 19

* Kiểm tra thông tin thấy node 19 không cùng Instance ID (30 # 46)
* Từ chối tham gia vào DODAG mới

Tương tự như node 15, 29 node còn lại sẽ có các quá trình thực hiện tương tự để xây dựng DODAG.

Thông qua quá trình hình thành topo và truyền dữ liệu, có thể kết luận như sau

- Trong cùng một Instance khi một node đã tham gia vào một DODAG nếu như quá trình kết nối tới các node hàng xóm hoặc node root không bị gián đoạn thì node đó sẽ không tham gia vào DODAG khác

- Tất cả các node trong mạng chỉ truyền gói tin về root mà nó đang là thành viên.

#### 3.4.2.2. Tỉ lệ truyền gói thành công

Các số liệu được thống kê từ chương trình mô phỏng với 1 Instance 2 DODAG root và 30 client trong thời gian 5400 giây. Các bảng thông kê số lượng gói truyền tại các node và số gói nhận tương ứng tại DAG root như sau:

| STT | Rank | Node ID | Số bản tin gửi | Số bản tin đến root | PDR |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | 3 | 89 | 89 | 1.000 |
| 2 | 1 | 8 | 89 | 89 | 1.000 |
| 3 | 1 | 14 | 89 | 89 | 1.000 |
| 4 | 1 | 13 | 89 | 89 | 1.000 |
| 5 | 1 | 18 | 89 | 89 | 1.000 |
| 6 | 1 | 23 | 89 | 89 | 1.000 |
| 7 | 1 | 28 | 89 | 88 | 0.989 |
| 8 | 2 | 5 | 89 | 89 | 1.000 |
| 9 | 2 | 4 | 89 | 89 | 1.000 |
| 10 | 2 | 10 | 89 | 88 | 0.989 |
| 11 | 2 | 9 | 89 | 87 | 0.978 |
| 12 | 2 | 15 | 89 | 88 | 0.989 |
| 13 | 2 | 19 | 89 | 89 | 1.000 |
| 14 | 2 | 24 | 89 | 88 | 0.989 |
| 15 | 2 | 29 | 89 | 89 | 1.000 |
| 16 | 3 | 6 | 89 | 88 | 0.989 |
| 17 | 3 | 11 | 89 | 89 | 1.000 |
| 18 | 3 | 17 | 89 | 89 | 1.000 |
| 19 | 3 | 16 | 89 | 89 | 1.000 |
| 20 | 3 | 20 | 89 | 87 | 0.978 |
| 21 | 3 | 25 | 89 | 89 | 1.000 |
| 22 | 3 | 30 | 89 | 84 | 0.944 |
| 23 | 4 | 7 | 89 | 89 | 1.000 |
| 24 | 4 | 12 | 89 | 89 | 1.000 |
| 25 | 4 | 22 | 89 | 89 | 1.000 |
| 26 | 4 | 21 | 89 | 89 | 1.000 |
| 27 | 4 | 26 | 89 | 88 | 0.989 |
| 28 | 4 | 31 | 89 | 85 | 0.955 |
| 29 | 5 | 27 | 89 | 89 | 1.000 |
| 30 | 5 | 32 | 89 | 89 | 1.000 |

Từ bảng trên ta tổng hợp được bảng sau

| STT | Rank | Tổng số bản tin gửi | Tổng số bản tin đến Root | PDR |
| --- | --- | --- | --- | --- |
| 1 | 1 | 623 | 619 | 0.994 |
| 2 | 2 | 712 | 703 | 0.987 |
| 3 | 3 | 623 | 615 | 0.987 |
| 4 | 4 | 534 | 529 | 0.990 |
| 5 | 5 | 178 | 178 | 1.000 |

PDR==0.990

Từ bảng trên và kết quả PDR, ta có thể thấy cũng như nhận định của kịch bản 1, PDR là khá cao và đồng đều giữa các Rank trong DODAG. Qua đó có thể thấy độ tin khi gửi gói trong mạng RPL là khá cao và ổn định.

#### 3.4.2.3 Chi phí truyền gói

| Chức năng | Node ID | Bản tin DATA | Bản tin ICMP | Tổng số bản tin |
| --- | --- | --- | --- | --- |
| DAG root | 1 | 0 | 20 | 20 |
| DAG root | 2 | 0 | 20 | 20 |
| Client | 3 | 89 | 117 | 182 |
| Client | 4 | 89 | 121 | 188 |
| Client | 5 | 89 | 115 | 179 |
| Client | 6 | 89 | 113 | 176 |
| Client | 7 | 89 | 121 | 188 |
| Client | 8 | 89 | 112 | 174 |
| Client | 9 | 89 | 117 | 182 |
| Client | 10 | 89 | 128 | 199 |
| Client | 11 | 89 | 115 | 179 |
| Client | 12 | 89 | 115 | 179 |
| Client | 13 | 89 | 115 | 179 |
| Client | 14 | 89 | 126 | 196 |
| Client | 15 | 89 | 124 | 193 |
| Client | 16 | 89 | 113 | 176 |
| Client | 17 | 89 | 113 | 176 |
| Client | 18 | 89 | 117 | 182 |
| Client | 19 | 89 | 131 | 204 |
| Client | 20 | 89 | 126 | 196 |
| Client | 21 | 89 | 139 | 216 |
| Client | 22 | 89 | 130 | 202 |
| Client | 23 | 89 | 106 | 165 |
| Client | 24 | 89 | 119 | 185 |
| Client | 25 | 89 | 128 | 199 |
| Client | 26 | 89 | 133 | 207 |
| Client | 27 | 89 | 124 | 193 |
| Client | 28 | 89 | 106 | 165 |
| Client | 29 | 89 | 121 | 188 |
| Client | 30 | 89 | 128 | 199 |
| Client | 31 | 89 | 121 | 188 |
| Client | 32 | 89 | 126 | 196 |
| Tổng số bản tin | | 2670 | 3659 | 5692 |

Bảng trên thống kê số bản tin DATA được gửi từ các node trong mạng đến DODAG root trong thời gian mô phỏng 5400 giây.

- Số bản tin DATA nhận được tại DODAG root trong thời gian mô phỏng 5400 giây: 2644 (gói tin).

- Tổng tất cả các loại bản tin được gửi trong mạng: 5692 (bản tin).

Chi phí truyền gói:

PDC==2,137

Giống như kịch bản số 1, dựa vào thông số PDC bằng 2.137 ta có thể thấy được trong kịch bản này chi phí truyền gói tương đối thấp, giao thức RPL có hiệu năng tốt.

## ***3.5. Đánh giá kết quả mô phỏng***

Về Topo mạng

- Cả 2 kịch bản đều có kết quả hình thành topo mạng trong thời gian ngắn từ 2-3 phút qua đó thấy được giao thức RPL có hiệu năng tốt trong việc hình thành topo mạng.

- Cả 2 kịch bản đưa ra đều thỏa mãn yêu cầu của mục tiêu mô phỏng: tối thiểu hóa số chặng truyền gói tin từ các node tới DODAG root

- Riêng kịch bản 2 còn chỉ rõ hơn kết quả tại một thời điểm, trong một instance, mỗi node chi tham gia vào một DODAG trong instance đó và chỉ gửi bản tin về DODAG root nó tham gia

Về chỉ số PDR và PDC

- Tỉ số PDR kịch bản 1 cao hơn kịch bản còn lại, bên cạnh đó chi phí truyền gói PDR của kịch bản 1 thấp hơn kịch bản còn lại. Qua đó thấy được PDR và PDC phụ thuộc vào số lượng node trong mạng, số lượng node càng ít thì PDR càng tăng và PDC càng giảm. Có nhiều nguyên nhân để dẫn đến kết quả này nhưng một trong những nguyên nhân chính là do lượng bản tin ICMP ít hơn.

## ***3.6. Kết luận***

Trong chương này nhóm em đã mô phỏng một số kịch bản và kết quả về hoạt động của giao thức RPL dựa vào Contiki RPL. Kết quả mô phỏng cho thấy: thông qua việc xây dựng các DODAG, mỗi node trong mạng có khả năng xác định vị trí gần Root nhất, tối ưu khoảng cách truyền gói về bộ tập trung dữ liệu với số hop nhỏ nhất. Hơn nữa, cơ chế tự cấu hình mạng linh động và có độ tin cậy cao khi truyền gói.

Tuy nhiên, do thời gian hạn hẹp, trong chương trình mô phỏng chưa xây dựng được chương trình có khả năng hỗ trợ nhiều Instance cho mỗi node trong mạng. Vì vậy, quá trình nghiên cứu tiếp theo sẽ thực hiện một số vấn đề sau:

- Triển khai thử trên node thật, đánh giá các kết quả thu được.

# **KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

IOT đang ngày càng trở lên phổ biến và quen thuộc hơn với cuộc sống hiện đại, khi những ứng dụng ngày càng trở nên phong phú, đa dạng, đòi hỏi tính linh động cao hơn. Với những tính năng ưu việt và ứng dụng đa dạng, mạng IOT sẽ không ngừng được nghiên cứu và phát triển trong tương lai. Trong báo cáo này, nhóm em đã tìm hiểu và trình bày những khái niệm tổng quan về một giao thức định tuyến mới và hiện đang trong quá trình nghiên cứu, phát triển: RPL – Routing Protocol for Low Power and Lossy Network. Đồng thời, nhóm em cũng đã tiến hành mô phỏng RPL trên Contiki nhằm mục đích đánh giá hiệu năng của mạng khi sử dụng giao thức này. Dưới đây là một số công việc nhóm em đã thực hiện trong quá trình hoàn thành báo cáo:

- Tìm hiểu được cơ bản về IoT, các tầng, cấu trúc tham chiếu, các giao thức.

- Nghiên cứu, tìm hiểu các khái niệm, nguyên lý, hoạt động trong giao thức RPL.

- Tìm hiểu, cài đặt Contiki, Cooja, . . .

- Xây dựng chương trình mô phỏng với những chức năng: xây dựng topo mạng, nhận dạng DODAG root, multi DODAG root, phân biệt Instance.

- Thống kê số liệu, tổng hợp, đánh giá một số thông số định tuyến của giao thức RPL: Packet delivery ratio, Packet Delivery Cost.

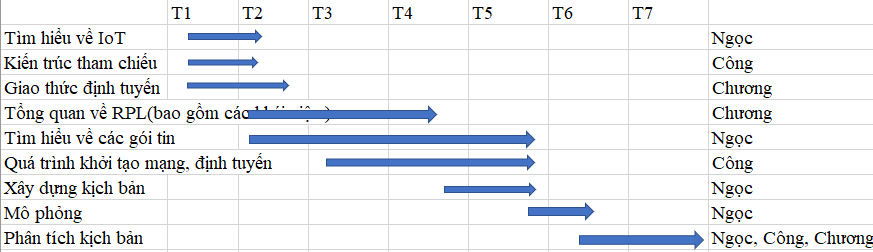
Trong tương lai cần nghiên cứu, phát triển:

- Đánh giá nguyên nhân mất gói và cách khắc phục

- Triển khai thử trên node thật, đánh giá các kết quả thu được.

- Cố gắng ứng dụng những kiến thức tìm hiểu được trong báo cáo để làm đồ án tốt nghiệp.

# Phân chia công việc



# **Tài liệu tham khảo**

-INFSO D.4 Networked Enterprise RFID INFSO G.2 Micro Nanosystems in Co-operation with the Working Group RFID of the ETP EPOSS, “Internet of Things in 2020, Roadmap for future, Version 1.1”, European Commission, Information Society and Media, Tech. Rep., May 2008.

-H. Sundmaeker, P. Guillemin, P. Friess, S. Woelffle, “Vision and Challenges for Realizing the Internet of Things”, The Cluster of European Research Projects on the Internet of Things-CERP IoT, 2010.

-O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, et al., “Internet of Things Strategic Research Agenda”, Chapter 2 in Internet of Things Global Technological and Societal Trends, River Publishers, 2011.

-European Commission, “Internet of Things in 2020 Road Map for The Future”, Working Group RFID of the ETP EPOSS, Tech. Rep., May2008,