TRẦN TRUNG HIẾU

KỸ THUẬT VIỄN THÔNG

KHOÁ 2017B

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Trần Trung Hiếu

NGHIÊN CỨU PHÁT TRIỂN NỀN TẢNG ONEM2M CHO ỨNG DỤNG IOT/M2M

LUẬN VĂN THẠC SĨ KHOA HỌC
….....

Hà Nội – Năm 2019

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Trần Trung Hiếu

NGHIÊN CỨU PHÁT TRIỂN NỀN TẢNG ONEM2M CHO ỨNG DỤNG IOT/M2M

Chuyên ngành : Kỹ thuật viễn thông

LUẬN VĂN THẠC SĨ KHOA HỌC
…….....

NGƯỜI HƯỚNG DẪN KHOA HỌC:
TS. Phùng Thị Kiều Hà

Hà Nội – Năm 2019

LỜI NÓI ĐẦU

Internet of Things (IoT) được kì vọng sẽ tạo ra một kỷ nguyên mới, khi mỗi vật đều được định danh và có khả năng trao đổi thông tin với nhau mà không cần tương tác trực tiếp giữa người với người hoặc người với máy. IoT đã phát triển nhờ vào sự hội tụ của công nghệ không dây, công nghệ vi cơ điện tử và nền tảng kết nối mạng Internet toàn cầu. Theo khái niệm, IoT là vạn vật kết nối qua Internet. Ở cấp độ người dùng cá nhân, chúng ta có thể điều khiển mọi đồ vật như tivi, xe hơi,… thông qua các thiết bị thông minh như máy tính, điện thoại. Đối với quy mô lớn hơn, IoT xuất hiện trong các lĩnh vực nông nghiệp thông minh, giám sát môi trường, tự động hóa trong công nghiệp…

Cùng với sự bùng nổ của IoT, có rất nhiều các công nghệ truyền thông không dây được đưa vào sử dụng như Bluetooth Low Energy, WiFi và ZigBee. Mỗi chuẩn kết nối đều có những đặc điểm về bảo mật, độ trễ, tính di động, khoảng cách truyền, năng lượng tiêu thụ, tuổi tho pin, tốc độ dữ liệu tối đa rất khác nhau. Đi kèm theo đó là những giao thức mới phù hợp cho mạng cảm biến không dây hoạt động dưới điều kiện năng lượng khiêm tốn và khoảng cách truyền thông xa trong khi lượng dữ liệu trao đổi không quá lớn như MQTT, CoAP. Tuy nhiên khi càng nhiều công nghệ và giao thức ra đời, sự đa dạng hóa dẫn đến phân mảnh hệ sinh thái IoT bởi mỗi doanh nghiệp, tổ chức đều muốn phát triển những giải pháp, nền tảng độc lập của riêng mình, từ đó chọn lựa chuẩn kết nối và giao thức riêng, thậm chí độc quyền. Ví dụ như các mạng cảm biến công suất thấp sẽ sử dụng những tiêu chuẩn kết nối tiết kiệm năng lượng như ZigBee, 6LowPAN và các giao thức như CoAP, MQTT. Mặt khác, các ứng dụng yêu cầu tốc độ cao lại thường sử dụng Wi-Fi, 3GPP và các giao thức như HTTP. Do đó, với nỗ lực thúc đẩy sự hội tụ của các công nghệ, phát triển một nền tảng có thể kết nối và tương thích diện rộng giúp các ứng dụng IoT có thể phát triển theo chiều ngang, oneM2M ra đời.

Việc nghiên cứu nền tảng oneM2M đang là một đề tài vô cùng rộng mở, thu hút nhiều kỹ sư cùng các chuyên gia trên toàn thế giới. Tiêu chuẩn chung cho oneM2M vẫn đang được dần hoàn thiện theo thời gian, nhiều tính năng và công nghệ mới cần được phát triển. Trên thực thế đó, em thực hiện luận văn này nhằm tìm hiểu chi tiết về tiêu chuẩn oneM2M hiện có, thực hiện xây dựng mô hình thử nghiệm các tính năng nổi bật. Ngoài ra, một số cải tiền và đề xuất phát triển điện toán biên, điện toán sương mù (Edge/Fog Computing) cũng được xem xét bổ sung vào tiêu chuẩn hiện hành.

Để có được kết quả này, em xin được cảm ơn sự hướng dẫn và chỉ bảo tận tình của cô giáo TS. Phùng Thị Kiều Hà. Cô đã cung cấp cho em những tài liệu, kiến thức quan trọng trong quá trình làm Luận văn tốt nghiệp.

Hà Nội, ngày 30 tháng 10 năm 2019

Trần Trung Hiếu

LỜI CAM ĐOAN

Tôi là Trần Trung Hiếu, mã số học viên CB170233, sinh viên lớp 17BKTVT.KH, khóa 2017B. Người hướng dẫn là TS. Phùng Thị Kiều Hà. Tôi xin cam đoan toàn bộ nội dung được trình bày trong đồ án *Nghiên cứu phát triển nền tảng oneM2M cho ứng dụng IoT/M2M* là kết quả quá trình tìm hiểu và nghiên cứu của tôi. Các dữ liệu được nêu trong đồ án là hoàn toàn trung thực, phản ánh đúng kết quả đo đạc thực tế. Mọi thông tin trích dẫn đều tuân thủ các quy định về sở hữu trí tuệ; các tài liệu tham khảo được liệt kê rõ ràng. Tôi xin chịu hoàn toàn trách nhiệm với những nội dung được viết trong luận văn này.

Hà nội, ngày 30 tháng 10 năm 2019

Người cam đoan

Trần Trung Hiếu

MỤC LỤC

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT ..... **i**

DANH MỤC HÌNH VẼ ..... **iii**

DANH MỤC BẢNG BIỂU ..... **v**

TÓM TẮT LUẬN VĂN ..... **vi**

MỞ ĐẦU ..... **vii**

1.1 Bối cảnh nghiên cứu ..... ***vii***

1.2 oneM2M và những nghiên cứu hiện tại ..... ***viii***

1.3 Mục tiêu nghiên cứu của luận văn ..... ***x***

1.4 Các vấn đề giải quyết của luận văn ..... ***x***

1.5 Những giới hạn trong nghiên cứu của luận văn ..... ***xi***

1.6 Phương pháp nghiên cứu ..... ***xi***

1.7 Bố cục luận văn ..... ***xi***

CHƯƠNG 1. TỔNG QUAN VỀ IoT **1**

1.1 Định nghĩa IoT ***1***

1.2 Kiến trúc phân tầng IoT ***2***

1.2.1 Tầng Object 3

1.2.2 Tầng Object Astraction 3

1.2.3 Tầng Service Management 3

1.2.4 Tầng Application 4

1.2.5 Tầng Bussiness 4

1.3 Các yếu tố trong IoT ***4***

1.3.1 Nhận dạng 5

1.3.2 Cảm biến 6

1.3.3 Truyền thông 6

1.3.4 Tính toán 6

1.3.5 Dịch vụ 7

1.3.6 Ngữ nghĩa 7

1.4 Tiêu chuẩn chung trong IoT *7*
1.4.1 Một số giao thức tầng ứng dụng 8
1.4.1.1 HTTP *9*
1.4.1.2 CoAP *10*
1.4.1.3 MQTT *11*
1.4.2 Một số công nghệ truyền thông không dây 12
1.4.2.1 Wi-Fi *13*
1.4.2.2 Zigbee *13*
1.4.2.3 Bluetooth *13*
1.5 Kết luận chương ***14***
CHƯƠNG 2. TỔNG QUAN VỀ ONEM2M **15**
2.1 Lịch sử phát triển ***15***
2.1.1 Sự ra đời của oneM2M 15
2.1.2 Sự phát triển của oneM2M 17
2.2 Kiến trúc hệ thống ***18***
2.2.1 Kiến trúc mức chức năng 19
2.2.2 Kiến trúc mức vật lý 22
2.3 Giao thức oneM2M ***23***
2.3.1 Kiến trúc REST 24
2.3.2 Giao diện lập trình ứng dụng oneM2M 25
2.3.3 Phương thức oneM2M 26
2.3.4 Tài nguyên oneM2M 27
2.3.5 Bản tin oneM2M Primitive 30
2.3.5.1 Bản tin request primitive *31*
2.3.5.2 Bản tin response primitive *33*
2.4 Tổ chức và truy cập dữ liệu trong oneM2M ***34***
2.4.1 Container 34
2.4.2 Cơ chế đăng ký và thông báo 35
2.4.3 Cơ chế khám phá 36
2.5 oneM2M Binding ***36***
2.5.1 HTTP Binding 37
2.5.2 CoAP Binding 39

2.5.3 MQTT Binding 39

2.6 Các dự án triển khai oneM2M 41

2.7 Kết luận chương 42

CHƯƠNG 3. XÂY DỰNG MẠNG ĐA CÔNG NGHỆ TRÊN NỀN TẢNG OM2M THEO TIÊU CHUẨN ONEM2M 44

3.1 Dự án OM2M 44

3.1.1 Sự ra đời 44

3.1.2 Các đặc điểm và tính năng 44

3.1.3 Kiến trúc OM2M 45

3.1.3.1 CORE Module 46

3.1.3.2 Binding Module 48

3.1.3.3 IPE Module 49

3.1.3.4 EclipseLink Persistence Module 50

3.1.4 Mã nguồn OM2M 51

3.2 Kiến trúc testbed 51

3.3 Tích hợp cảm biến IoT 53

3.3.1 Cảm biến IoT đo cường độ ánh sáng 53

3.3.2 Cấu hình tích hợp cảm biến IoT sử dụng giao thức MQTT 54

3.3.3 Lập trình oneM2M ADN 54

3.3.4 Kết quả 56

3.4 Tích hợp mạng Zigbee 57

3.4.1 Mạng Zigbee 58

3.4.2 Cấu hình tích hợp mạng Zigbee 59

3.4.3 Phát triển ZigbeeModule 59

3.4.4 Kết quả 61

3.5 Tích hợp thiết bị Bluetooth 61

3.5.1 Cấu hình tích hợp thiết bị Bluetooth 61

3.5.2 Phát triển Bluetooth IPE 62

3.5.3 Kết quả 64

3.6 Mô tả và đánh giá hoạt động 65

3.7 Kết luận chương 67

CHƯƠNG 4. TRIỂN KHAI ONEM2M VỚI ĐỊNH HƯỚNG ĐIỆN TOÁN BIÊN 68

4.1 Điện toán biên *68*
4.1.1 Giới thiệu chung 68
4.1.2 Lợi ích 69
4.2 Điện toán biên trong oneM2M *70*
4.2.1 Kiến trúc mô hình 70
4.2.2 Những khó khăn tồn tại 72
4.2.2.1 Giới hạn kết nối với MN-CSE *72*
4.2.2.2 Cơ chế retargeting *73*
4.2.3 Hướng giải quyết 74
4.2.3.1 Mở rộng kết nối cho MN-CSE *74*
4.2.3.2 Tích hợp giao thức định tuyến cho oneM2M *76*
4.3 Kết luận chương *78*
KẾT LUẬN **80**
TÀI LIỆU THAM KHẢO **81**
PHỤ LỤC **1**
Phụ lục 1. Bảng danh sách các loại tài nguyên *1*
Phụ lục 2. Bản tin primitive *2*
Phụ lục 3. Ánh xạ bản tin *2*

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

6LoWPAN	IPv6 protocol over low-power wireless PANs	Giao thức IPv6 cho mạng cá nhân công suất thấp
ACK	Acknowledgment	Bản tin thông báo nhận
ADN	Application Dedicated Node	Đại diện đặc trưng cho thiết bị trong oneM2M
AE	Application Enity	Thực thể ứng dụng
AMQP	Advanced Message Queue Protocol	Giao thức giao nhận tin nhắn sử dụng hàng đợi
API	Application Programming Interface	Giao diện lập trình ứng dụng
CoAP	Constrained Application Protocol	Giao thức cho ứng dụng tài nguyên hạn chế
CSE	Common Service Entity	Thực thể dịch vụ mạng ngang hàng
DTLS	Datagram Transport Layer Security	Bảo mật tầng truyền dẫn datagram
EPC	Electronic Product Code	Mã sản phẩm điện tử
ETSI	European Telecommunications Standards	Viện tiêu chuẩn Viễn thông châu Âu
FPGA	Field-programmable gate array	Mảng cổng lập trình được dạng trường
GSM	Global System for Mobile Communication	Hệ thống thông tin di động toàn cầu thế hệ 2
IEEE	Institute of Electrical and Electronics Engineers	Viện kỹ thuật Điện và Điện tử
IETF	Internet Engineering Task Force	Nhóm đặc trách kỹ thuật Internet
IN	Infrastructure Node	Máy chủ oneM2M
IP	Internet Protocol	Giao thức Internet
IPE	Interworking Proxy Entity	Module kết nối đa công nghệ
ISM	Industrial, Scientific and Medical	Băng tần sử dụng cho công nghiệp, khoa học và y tế

JSON	JavaScript Object Notation	Dạng dữ liệu json
LAN	Local Area Network	Mạng cục bộ
LPWA	A low-power wide-area network	Mạng công suất thấp vùng phủ lớn
MAC	Medium Access Control	Lớp điều khiển truy nhập môi trường
MCU	Micro Controller Unit	Khối vi điền khiển
MN	Middle Node	Gateway oneM2M
MQTT	MQ Telemetry Transport	Giao thức MQTT
NFC	Near Field Communication	Công nghệ giao tiếp trường gần
NSE	Network Services Entity	Thực thể mạng cơ bản
OS	Operating System	Hệ điều hành
OWL	Ontology Web Language	Ngôn ngữ web logic
QoS	Quality of Service	Chất lượng dịch vụ
RDF	Resource Description Framework	Nền tảng mô tả dữ liệu
RFID	Radio Frequency Identification	Công nghệ nhận dạng đối tượng bằng sóng vô tuyến
RPL	Routing Protocol for Low Power	Giao thức định tuyến cho mạng công suất thấp
SOA	Service-oriented architecture	Kiến trúc hướng dịch vụ
SOCs	System on a chip	Hệ thống vi mạch tích hợp trên chip
SSL	Secure Sockets Layer	Giao thức mã hóa dữ liệu
TCP	Transmission Control Protocol	Giao thức TCP
UDP	User Datagram Protocol	Giao thức UDP
UWB	Ultra Wide Band	Băng siêu rộng
W3C	World Wide Web Consortium	Tổ chức web toàn cầu
Wi-Fi	Wireless Fidelity	Công nghệ Wifi
XML	Extensible Markup Language	Ngôn ngữ đánh dấu mở rộng
XMPP	Xtensible Messaging and Presence Protocol	Giao thức XMPP

DANH MỤC HÌNH VẼ

Hình 1.1 Sự hội tụ tầm nhìn trong IoT [14] 1
Hình 1.2 Kiến trúc phân tầng IoT [1] 3
Hình 1.3 Cấu trúc bản tin HTTP 9
Hình 1.4 Cấu trúc bản tin CoAP 11
Hình 1.5 Cấu trúc bản tin MQTT 12
Hình 2.1 Sự phát triển của oneM2M 17
Hình 2.2 Kiến trúc ba tầng hệ thống IoT 18
Hình 2.3 Kiến trúc chức năng oneM2M 19
Hình 2.4 Kiến trúc mức vật lý 22
Hình 2.5 Interworking Proxy 23
Hình 2.6 Tài nguyên <*CSEBase*> 29
Hình 2.7 Ví dụ về cây tài nguyên 34
Hình 2.8 Ví dụ về cơ chế đăng ký và thông báo 35
Hình 2.9 Ví dụ về quá trình khám phá 36
Hình 2.10 Kiến trúc oneM2M Binding 37
Hình 2.11 Cấu hình HTTP Binding 38
Hình 2.12 Cấu hình MQTT Binding 40
Hình 3.1 Kiến trúc OM2M 45
Hình 3.2 Kiến trúc CORE module 46
Hình 3.3 Lưu đồ hoạt động của khối Router 47
Hình 3.4 Kiến trúc HTTP Binding Module 48
Hình 3.5 Kiến trúc IPE Module 49
Hình 3.6 Kiến trúc EclipseLink Persistence Module 50
Hình 3.7 Kiến trúc hệ thống triển khai thử nghiệm 52
Hình 3.8 Sơ đồ mạch đo cường độ ánh sáng 53
Hình 3.9 Cấu hình tích hợp Wi-Fi sensor 54
Hình 3.10 Lưu đồ trao đổi bản tin giữa ESP8266 và MN 55

Hình 3.11 Cây tài nguyên ứng với Wi-Fi sensor 57
Hình 3.12 Tài nguyên <*ContentInstance*> lưu giá trị cảm biến ánh sáng 57
Hình 3.13 Kit Z1 Zolertia..... 58
Hình 3.14 Cấu hình kết nối với Zigbee Network..... 59
Hình 3.15 Lưu đồ trao đổi bản tin giữa các khối và Zigbee Device..... 60
Hình 3.16 Cây tài nguyên của ZIGBEE_DEVICE..... 61
Hình 3.17 Cấu hình tích hợp thiết bị Bluetooth..... 62
Hình 3.18 Luồng dữ liệu trao đổi với Bluetooth IPE..... 63
Hình 3.19 Kiến trúc thư viện BlueCove 64
Hình 3.20 Cây tài nguyên của Bluetooh Device..... 65
Hình 3.21 Mô hình testbed trong thực tế 65
Hình 4.1 Edge/Fog Layer [25] 69
Hình 4.2 Kiến trúc oneM2M tích hợp Edge/Fog computing [26] 71
Hình 4.3 Topo mạng thể hiện hạn chế của oneM2M 74
Hình 4.4 Topo mạng thử nghiệm 76
Hình 4.5 Cây tài nguyên trên MN 3..... 76
Hình PL.1 Ánh xạ bản tin HTTP request và Primitive request..... 3
Hình PL.2 Ánh xạ bản tin HTTP response và Primitive response..... 4
Hình PL.3 Ánh xạ bản tin CoAP request và Primitive request..... 5
Hình PL.4 Ánh xạ bản tin CoAP response và Primitive response..... 5
Hình PL.5 Ánh xạ bản tin MQTT và Primitive request..... 5
Hình PL.6 Ánh xạ bản tin MQTT và Primitive response 6
Hình PL.7 Ví dụ về MQTT payload 6

DANH MỤC BẢNG BIỂU

Bảng 1.1 Các nhân tố trong IoT 5
Bảng 1.2 Các giao thức trong IoT 8
Bảng 1.3 Một số giao thức tầng ứng dụng 9
Bảng 1.4 Các công nghệ truyền dẫn trong IoT [17].....12
Bảng 2.1 Một số loại tài nguyên.....28
Bảng 2.2 Universal Attribute.....29
Bảng 2.3 Các tham số của Request Message 31
Bảng 2.4 Các tham số của Response Message.....33
Bảng 3.1 Z1 Protocol Stack.....58
Bảng 4.1 Các tham số cấu hình remoteCSE.....75
Bảng 4.2 Thuộc tính descendantCSEs.....77
Bảng PL.1 Danh sách các tài nguyên oneM2M 1
Bảng PL.2 Ánh xạ trường HTTP Method 3

TÓM TẮT LUẬN VĂN

Luận văn trình bày các khái niệm cơ bản dùng để phát triển một hệ thống tuân theo tiêu chuẩn oneM2M dựa trên nền tảng OM2M, bao gồm mô hình hệ thống, giao thức, phương pháp tích hợp đa công nghệ. Trên cơ sở đó đó, luận văn trìn bày về xây dựng testbed thử nghiệm những đặc tính nổi trội có trong oneM2M giúp kết nối đa công nghệ (Wi-Fi, Blueooth, Zigbee) trên đa giao thức (HTTP, MQTT, CoAP). Ngoài ra, luận văn cũng đề xuất một số cải tiến cho tiêu chuẩn oneM2M giúp phát triển điện toán biên trong hệ thống.

PROJECT ABSTRACT

This thesis provides basic concepts to develop a oneM2M-based system using OM2M, including: architecture model, oneM2M protocol, integrated multi-technology method. Based on the study, several actual implementations on testbed that enable the interoperation of the disparated application layer protocols (HTTP, MQTT, CoAP) and multiple technologies (Wi-Fi, Bluetooth, Zigbee) are deployed. Moreover, the proposed improvements of oneM2M supporting Edge Computing are shown.

MỞ ĐẦU

1.1 Bối cảnh nghiên cứu

Sự bùng nổ của cuộc các mạng công nghệp 4.0, Internet vạn vật (IoT) và tương tác máy-máy (M2M) sẽ tạo ra một thế giới siêu kết nối. Hàng tỷ thiết bị khác nhau, từ những thiết bị đầu cuối thông minh đến các cảm biến tài nguyên hạn chế đều tham gia vào mạng. Sự khác biệt về giao thức, công nghệ và phần cứng của các thiết bị này dẫn đến thị trường IoT bị phân mảnh rất mạnh, khi các giải pháp khác nhau được phát triển độc lập theo chiều dọc thay vì theo chiều ngang. Đặc biệt, khả năng tương tác của các thiết bị là một trong những thách thức hàng đầu cần được quan tâm khi tích hợp và phát triển nhiều loại dịch vụ, ứng dụng IoT [1]. Thật vậy, 40% lợi ích tiềm năng của IoT phụ thuộc vào khả năng tương tác (công nghệ, giao thức, định dạng dữ liệu, nội dung) [2]. Tuy nhiên, việc san bằng trở ngại về khả năng tương tác không đơn giản. Do vậy, các tổ chức phát triển tiêu chuẩn (ETSI, ITU,…) cùng hợp tác với các tập đoàn công nghiệp (AllJoyn,…) tiến hành các hoạt động nghiên cứu nhằm tăng cường khả năng tương tác giữa các thiết bị không đồng nhất trên phạm vi toàn cầu.

Sáng kiến Toàn cầu oneM2M (oneM2M Global Initiative) [3] là một dự án quốc tế, được thành lập với mục tiêu phát triển bộ thông số kỹ thuật tiêu chuẩn cho lớp dịch vụ dùng chung giúp các nền tảng M2M độc lập có thể tương tác lẫn nhau trong mạng IoT toàn cầu. Lớp dịch vụ dùng chung này dự định được nhúng trực tiếp vào nhiều loại phần cứng cũng như phần mềm khác nhau. Để cùng hỗ trợ hợp tác trong nghiên cứu, nhóm phát triển dự án oneM2M đã thông qua tư cách thành viên cho nhiều tổ chức tiêu chuẩn. Ví dụ, kiến trúc trong oneM2M có nhiều điểm tương đồng trong thuật ngữ, cách phân tầng với kiến trúc IoT-A [4], một dự án phát triển mô hình tham chiếu thống nhất cho IoT của châu Âu. Ngoài ra, oneM2M có hợp tác với dự án đối tác thế hệ thứ ba (3GPP), cùng trao đổi và đưa ra giải pháp chung nhằm đảm bảo tương tác giữa oneM2M

với các hệ thống mạng di động hiện có. Trong phiên bản thứ ba của mình, oneM2M đã đưa ra được bộ quy tắc giúp tương tác với 3GPP [5].

1.2 oneM2M và những nghiên cứu hiện tại

oneM2M cung cấp bộ các tiêu chuẩn kỹ thuật giúp người dùng có thể xây dựng hệ thống IoT tuân theo tiêu chuẩn oneM2M cũng như liên kết những thiết bị, hệ thống khác đang tồn tại vào trong hệ sinh thái oneM2M. Thành phần quan trọng nhất trong tiêu chuẩn oneM2M là lớp dịch vụ oneM2M. Lớp dịch vụ này thường được triển khai như một firmware đảm bảo kết nối giữa các ứng dụng IoT và phần cứng. Nó cung cấp giải pháp lưu trữ, xử lý dữ liệu, trao đổi bản tin. Các thực thể trong hệ thống oneM2M trao đổi với nhau qua giao thức oneM2M, được xây dựng dựa trên kiến trúc RESTful API. Đối với những thực thể/thiết bị khác muốn kết nối, tích hợp với hệ thống, oneM2M cung cấp hai chức năng:

- Binding Protocol: Giúp các thiết bị sử dụng giao thức trên nền mạng IP có thể giao tiếp với hệ thống lõi của oneM2M. Hiện tại oneM2M đã hỗ trợ HTTP, MQTT và CoAP Binding.
- Interworking Proxy: Sử dụng thực thể Interworking Proxy Entity (IPE) để giao tiếp với thiết bị sử dụng các giao thức non-IP. Ngoài ra, chúng ta còn có thể dùng thực thể này tích hợp hệ thống oneM2M với hệ thống khác.

Đối với giải pháp dùng Binding Protocol, [6] đã thực hiện được việc tích hợp giao thức MQTT và CoAP lên trên nền công nghệ Lora. Quá trình này đòi hỏi hiểu biết sâu sắc về công nghệ truyền dẫn LoRa và khi đó oneM2M chỉ được xem như phương tiện giúp thử nghiệm và thực hiện đo đạc, đánh giá hiệu năng. Bài báo [7] đưa thêm một số cải tiến về nội dung bản tin MQTT trong MQTT Binding giúp triển khai một ứng dụng tự động cấu hình có kết hợp phân tích ngữ nghĩa. Về cơ bản, khi sử dụng "Binding

Protocol", chúng ta không cần thay đổi quá nhiều hệ thống oneM2M mà tập trung vào việc phát triển các thiết bị, giải pháp thực tế của hệ thống IoT.

Đối với giải pháp dùng IPE, [8] đã giới thiệu về IPE cho phép hệ thống oneM2M tương tác với các hệ thống IoT cũ từ Nest, Jawbone, và Withings. IPE được đề xuất thực hiện dịch các bản tin nội bộ trong oneM2M sang các bản tin thuộc giao thức trong máy chủ tập trung (nơi lưu trữ và quản lý toàn bộ dữ liệu) của hệ thống cũ thay vì dịch trực tiếp sang giao thức của các thiết bị. Do đó, giải pháp này vẫn cần các máy chủ hoạt động trung gian chứ chưa tích hợp được đến mức thấp hơn. Tương tự, nghiên cứu được đề xuất trong [9] đã thiết kế một kiến trúc tích hợp dựa trên IPE để giải quyết khả năng tương tác của các nền tảng IoT cụ thể (AllJoyn/IoTivity) với hệ thống oneM2M. IPE này hoạt động như phần mềm trung gian, hỗ trợ ánh xạ chức năng quản lý thiết bị giữa các nền tảng này. Mặc dù thiết kế của IPE đã được mô tả, nhưng không có bằng chứng nào cho thấy liên kết thông qua đó là hai chiều. Cuối cùng, [10] đã làm rõ quy trình trao đổi thông tin của IPE và áp dụng trong điều kiện thực tế như thành phố thông minh để liên kết nhiều nền tảng dịch vụ IoT.

Đã có rất nhiều những giải pháp cụ thể sử dụng oneM2M được đưa vào nghiên cứu như quản lý thiết bị bay [11], [12], giám sát trạm di động [13],… Tuy nhiên đa số các thử nghiệm đều được giả định và mô phỏng.

Trước thực trạng về việc nghiên cứu oneM2M đã nêu, luận văn sẽ đề xuất một mô hình IoT cơ bản gồm: cảm biến, thiết bị chấp hành, hệ thống oneM2M và các ứng dụng dịch vụ để thử nghiệm, đưa ra quy trình chung giúp thiết kế các giải pháp cụ thể khác. Để thực hiện triển khai được mô hình nêu trên, luận văn có trình bày cơ chế hoạt động và cách sử dụng Binding Protocol, đồng thời, nghiên cứu hai cách sử dụng IPE:

- Đối với kết nối công nghệ Zigbee, IPE kết nối đến máy chủ tương tự bài báo [9], cách kết nối này không làm biến đổi hệ thống IoT có sẵn.

- Đối với kết nối công nghệ Bluetooth, IPE kết nối trực tiếp đến các thiết bị cuối dựa trên những bài học từ [10], cách kết nối này giúp chúng ta làm chủ hoàn toàn việc tích hợp.

1.3 Mục tiêu nghiên cứu của luận văn

Luận văn xây dựng một hệ thống IoT có thể phát triển theo chiều ngang, tính linh hoạt cao, kết hợp được nhiều tính năng mới, ở đó không còn ranh giới và phân biệt giữa các nền tảng, công nghệ, giao thức. Dựa trên đó, một số hướng tích hợp Edge computing vào oneM2M được đề xuất. Đây là kết quả có ý nghĩa không chỉ giúp chuẩn hóa và đẩy nhanh quá trình triển khai, mở rộng các ứng dụng IoT theo tiêu chuẩn oneM2M mà còn giúp các nhà nghiên cứu tìm hiểu sâu hơn về nền tảng, đưa ra được các cải tiến thích hợp.

1.4 Các vấn đề giải quyết của luận văn

Đóng góp 1: Trình bày hệ thống các kiến thức lý thuyết thiết yếu cần có để triển khai được hệ thống IoT theo tiêu chuẩn oneM2M.

Đóng góp 2: Thông qua việc tìm hiểu và phát triển OM2M, đưa ra được sơ đồ kiến trúc các thành phần của dự án, cấu trúc chức năng chung của các Module phục vụ giao tiếp đa công nghệ, đa giao thức.

Đóng góp 3: Trong quá trình xây dựng testbed thử nghiệm, luận văn đã: sửa lỗi Module CoAP binding, chuẩn hóa dạng bản tin sử dụng giao thức MQTT, phát triển Module Bluetooth IPE trên nền tảng OM2M.

Đóng góp 4: Khắc phục hạn chế về số kết nối đến MN-CSE và đưa ra giải pháp cho định tuyến trong oneM2M giúp tích hợp điện toán biên.

1.5 Những giới hạn trong nghiên cứu của luận văn

Hạn chế: Mô hình thực nghiệm sử dụng các Module ảo hóa chạy trong hệ điều hành trên laptop, tuy nhiên, thực tế nó cần được lập trình tích hợp vào ngay trong các thiết bị phần cứng Gateway.

1.6 Phương pháp nghiên cứu

Phương pháp phân tích và tổng hợp lý thuyết được dùng để phân tích và chắt lọc các nội dung từ các tài liệu kỹ thuật của oneM2M, giúp đưa ra hệ thống lý thuyết đầy đủ và sâu sắc về cách xây dựng giải pháp IoT đa công nghệ, đa giao thức. Sử dụng dự án nguồn mở OM2M để mô phỏng thực nghiệm và triển khai testbed theo tiêu chuẩn oneM2M, đồng thời phân tích, tổng kết đưa ra những cải tiến.

1.7 Bố cục luận văn

Luận văn bao gồm 4 chương:

Nội dung chương 1: Giới thiệu chung về quá trình phát triển IoT và vấn đề phát triển nền tảng giao thức chung cho IoT.

Nội dung chương 2: Tổng quan về chuẩn giao thức truyền thông oneM2M.

Nội dung chương 3: Tổng quan về nền tảng OM2M triển khai oneM2M và thực nghiệm xây dựng testbed hệ thống IoT đa nền tảng, đa giao thức dựa trên OM2M.

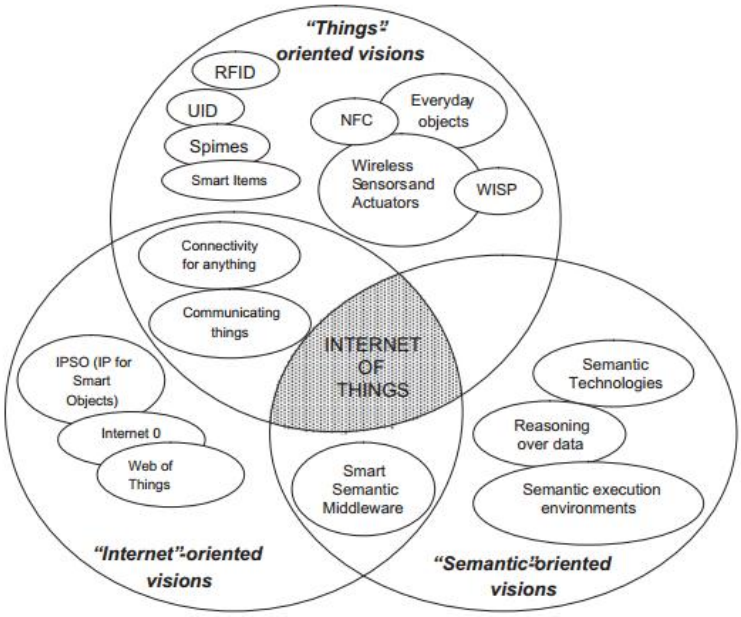
Nội dung chương 4: Hướng phát triển OM2M với định hướng tích hợp tính toán biên trên các gateway/MN.

CHƯƠNG 1. TỔNG QUAN VỀ IoT

Chương đầu tiên này sẽ trình bày về IoT, các hướng tiếp cận, kiến trúc phân tầng cũng như các công nghệ kết nối, giao thức truyền thông phổ biến đang được áp dụng. Từ đó, bức tranh tổng quan về sự phát triển của Internet vạn vật cùng những thách thức và hướng nghiên cứu được đưa ra. Nhằm phục vụ mô tả hệ thống triển khai thử nghiệm ở chương tiếp theo, các nội dung liên quan đến công nghệ Wi-Fi, Bluetooth, Zigbee và giao thức HTTP, CoAP, MQTT được tập trung trình bày chi tiết ngay trong chương này.

1.1 Định nghĩa IoT

Cho đến nay, khi khái niệm IoT (Internet of Thing) hay vạn vật kết nối đã trở nên rất quen thuộc, tuy nhiên chưa có một định nghĩa rõ ràng nào cho thuật ngữ này. Xét về mặt ngữ nghĩa, IoT được hiểu là mọi *"vật"* có khả năng kết nối internet tại bất kì đâu và bất kì khi nào thông qua các công nghệ truyền thông, và ở đó, mỗi *"vật"* đều được định danh trong mạng.



Hình 1.1 Sự hội tụ tầm nhìn trong IoT [14]

Để hiểu rõ hơn về IoT, chúng ta có thể tiếp cận khái niệm này theo ba định hướng cốt lõi được thể hiện như Hình 1.1, bao gồm:

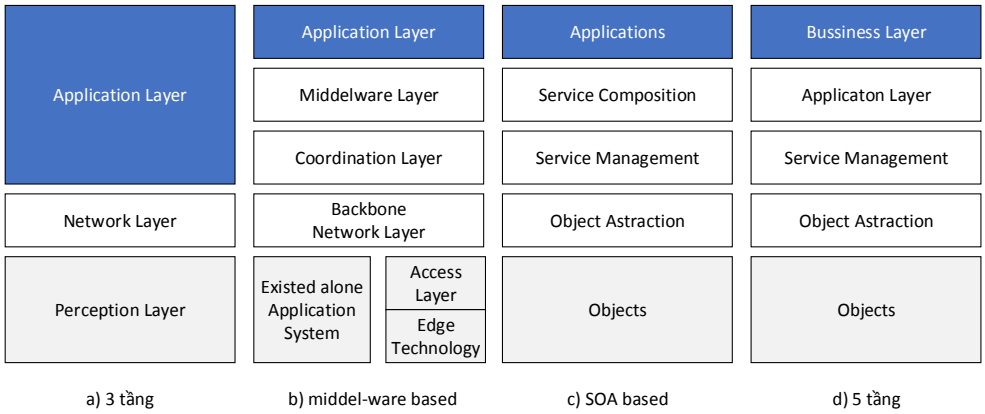
"Things oriented" hướng đến các "vật", đại diện cho những thiết bị vật lý mang trong mình phần cứng được tích hợp các công nghệ truyền dẫn như RFID, NFC,… Các thiết bị này thường có hiệu năng và năng lượng hạn chế, tập hợp của chúng tạo thành mạng cảm biến không dây.

"Internet oriented" hướng đến khả năng kết nối các thiết bị vào trong mạng internet. Vấn đề đặt ra là giao thức, phương thức truyền dữ liệu trong mạng bao gồm những thiết bị nhúng có hiệu năng và năng lượng hạn chế.

"Semantics oriented". Trên thực tế, chỉ cần kết hợp hai định hướng kể trên chúng ta đã có thể tạo ra một mạng IoT. Chẳng hạn kết hợp công nghệ truyền dẫn Zigbee (IEEE 802.15.4) trên kiến trúc IP (sử dụng Ipv6), truyền dữ liệu qua 6LowPAN. Tuy nhiên khi số lượng các thiết bị tăng lên, những vấn đề liên quan đến cách trình bày, lưu trữ, tương tác, tìm kiếm và tổ chức thông tin trở thành thách thức không nhỏ. **"Semantics oriented"** có ý đề cập đến những giải pháp, cấu trúc có thể tổ chức, khai phá và mở rộng thông tin thu thập được.

1.2 Kiến trúc phân tầng IoT

IoT có khả năng kết nối hàng tỷ các thiết bị không đồng nhất thông qua Internet, do đó, nó cần có kiến trúc tham chiếu linh hoạt. Số lượng mô hình tham chiếu tăng nhưng chưa đi đến một sự hội tụ nào cả [1]. Dự án IoT-A [4] cố gắng thiết kế một kiến trúc chung dựa trên phân ích nhu cầu của các nhà nghiên cứu và các ngành công nghiệp. Trong các mô hình được đề xuất, mô hình cơ bản gồm ba tầng gồm: *Application*, *Network* và *Perception*. Tuy nhiên, trong các nghiên cứu gần đây, một số mô hình đề xuất thêm một vài lớp trừu tượng vào kiến trúc chung. Hình 1.2 minh họa một số kiến trúc phổ biến, mô hình 5 lớp sẽ được xem xét và phân tích kỹ trong luận văn.



Hình 1.2 Kiến trúc phân tầng IoT [1]

1.2.1 Tầng Object

Tầng đầu tiên, Objects (device/perception), đại diện cho các cảm biến vật lý trong mạng IoT có nhiệm vụ thu thập và xử lý thông tin. Tầng này bao gồm các cảm biến, thiết bị chấp hành có thể thực hiện đa dạng các chức năng như truy vấn vị trí, nhiệt độ, trọng lượng, chuyển động, độ rung, gia tốc, độ ẩm,… Để có thể cấu hình nhiều đối tượng không đồng nhất, các nhà phát triển cần quan tâm đến việc phát triển cơ chế cắm và chạy (plug-and-play). Tất cả dữ liệu lớn được tạo ra trong cả hệ thống IoT đều khởi nguồn từ đây, chúng được số hóa và truyền tiếp lên các tầng cao hơn.

1.2.2 Tầng Object Astraction

Tầng thứ hai làm nhiệm vụ chuyển dữ liệu từ tầng Object lên cho tầng cao hơn thông các các kênh truyền bảo mật. Dữ liệu có thể được truyền qua rất nhiều công nghệ khác nhau như RFID, 3G, GSM, Wi-Fi, Bluetooth Low Energy, hồng ngoại, Zigbee,… Hơn nữa, một số chức năng liên quan đến điện toán, quy trình quản lý dữ liệu cũng có thể được xử lý ở đây.

1.2.3 Tầng Service Management

Tầng Service Management hoặc Middleware tương ứng với một dịch vụ được yêu cầu dựa trên địa chỉ và tên. Các lập trình viên có thể làm việc với các thiết bị không đồng nhất mà chẳng cần quan tâm đến nền tảng phần cứng cụ thể. Tầng này cũng xử

lý dữ liệu nhận được, đưa ra quyết định và cung cấp các dịch vụ được yêu cầu thong qua giao thức mạng có dây.

1.2.4 Tầng Application

Tầng này cung cấp các dịch vụ theo yêu cầu của người dùng. Chẳng hạn, nó có thể cung cấp các dữ liệu về nhiệt độ, độ ẩm không khí tới khách hàng yêu cầu. Tầm quan trọng của tầng Application trong hệ thống IoT là nó có khả năng cung cấp dịch vụ thông minh phù hợp nhu cầu người dùng. Tầng Application bao gồm rất nhiều các thị trường ngành dọc như nhà thông minh, giao thông, tự động hóa, chăm sóc sức khỏe,…

1.2.5 Tầng Bussiness

Tầng Bussiness quản lý các hoạt động và dịch vụ toàn hệ thống IoT. Nhiệm vụ của tầng này là xây dựng mô hình kinh doanh, lập biểu đồ,… dựa trên dữ liệu nhận được từ tầng ứng dụng. Về phía người dùng, tầng này có khả năng phân tích dữ liệu lớn, đưa ra những phân tích, đánh giá và hành động phù hợp. Đối với các tầng dưới, tầng Business có thể giám sát và quản lý, so sánh kết quả đầu ra của mỗi tầng với dự kiến để tăng cường dịch vụ và tối ưu hệ thống.

1.3 Các yếu tố trong IoT

Để có cái nhìn sâu sắc và giá trị về các chức năng của hệ thống IoT, trong phần này, chúng ta cùng tập trung đi vào tìm hiểu 6 yếu tố quan trọng với IoT bao gồm: nhận dạng (identification), cảm biến (sensing), truyền thông (communication), tính toán (computaion), dịch vụ (service) và ngữ nghĩa (semantic). Mỗi yếu tố đều được phân loại và đặc tả dựa trên các công nghệ, giao thức, loại phần cứng, phần mềm khác nhau. Một số ví dụ về các công nghệ phổ biến sử dụng trong IoT được mô tả ở Bảng 1.1 dưới đây.

Bảng 1.1 Các nhân tố trong IoT

IoT Elements		Ví dụ
Identification	**Naming**	EPC, uCode
	Addressing	IPv4, IPv6
Sensing		Cảm biến thông minh, thiết bị đeo thông minh, cảm biến nhúng, thiết bị chấp hành, RFID tag.
Communication		RFID, NFC, UWB, Bluetooth, BLE, IEEE 802.15.4, Z-Wave, WiFi, WiFiDirect, LTE-A
Computation	**Hardware**	SmartThings, Arduino, Phidgets, Intel Galileo, Raspberry Pi, Gadgeteer, BeagleBone, Cubieboard, Smart Phones
	Software	OS (Contiki, TinyOS, LiteOS, Riot OS, Android); Cloud (Nimbits, Hadoop, etc.)
Service		Identity-related (shipping), Information Aggregation (smart grid), Collaborative-Aware (smart home), Ubiquitous (smart city)
Semantic		RDF, OWL, EXI

1.3.1 Nhận dạng

Nhận dạng rất quan trọng trong IoT, giúp đặt tên cho các thực thể và khớp các dịch vụ được cung cấp với đúng nhu cầu của người dùng. Có một số phương pháp nhận dạng thường được sử dụng là mã sản phẩm điện tử (EPC) và mã phổ biến (uCode). Nhận dạng một đối tượng được xem xét bởi hai tham số ID và địa chỉ. ID thường liên quan đến tên (ví dụ "T1" là tên gọi đại diện cảm biến nhiệt độ), mặt khác, cảm biến đó cũng được đánh một địa chỉ cố định phục vụ việc truyền thông. Các phương pháp đánh địa chỉ bao gồm IPv6 và IPv4. 6LoWPAN cho phép sử dụng địa chỉ IPv6 trong các mạng không dây công suất thấp. Việc nhận dạng có thể được thực hiện trong quy mô nhỏ, với các mạng riêng tư chứ không nhất thiết phải là phương thức áp dụng cho mạng toàn cầu. Tóm lại, phương thức nhận dạng được sử dụng để xác định rõ ràng từng đối tượng trong mạng.

1.3.2 Cảm biến

Cảm biến được hiểu là thu thập dữ liệu từ các đối tượng và gửi cho kho dữ liệu, cơ sở dữ liệu hoặc đám mây. Dữ liệu lấy được dùng để phân tích rồi đưa ra các hành động cụ thể dựa trên dịch vụ được yêu cầu. Các thiết bị cảm biến rất đa dạng, có thể là những thiết bị chuyên dụng dùng để đo thông số của môi trường hoặc các thiết bị vòng đeo tay thông minh, thậm chí mỗi chiếc điện thoại thông minh cũng là nguồn cung cấp nhiều dữ liệu cảm biến như ánh sáng, nhịp tim, la bàn,…

1.3.3 Truyền thông

Các công nghệ truyền thông IoT kết nối nhiều đối tượng không đồng nhất với nhau để cung cấp những dịch vụ thông minh cụ thể. Mỗi công nghệ được thiết kế ra sẽ có khoảng cách truyền dẫn, mức tiêu thụ năng lượng, thời gian trễ, dung lượng bản tin khác nhau để phù hợp với từng ứng dụng riêng. Những thiết bị cần vùng phủ lớn, sử dụng năng lượng pin rất hạn chế sẽ dùng LPWA (Lora, Sigfox,…) hoặc NB-IoT. Những giải pháp khép kín như nhà thông minh thường dụng mạng Private Network gồm Zigbee, IEEE 802.15.4,… Các thiết bị được cung cấp nguồn điện dài hạn và cần truyền dẫn với khoảng cách lớn có thể dùng mạng di động có sẵn.

1.3.4 Tính toán

Việc tính toán được thực hiện dựa trên các đơn vị tính toán như MCU, SOCs, FPGAs,.. và phần mềm ứng dụng chạy trên đó. Có rất nhiều các nền tảng phần cứng được phát triển để chạy các ứng dụng IoT như Arduino, UDOO, FriendlyARM, Intel Galileo, Raspberry PI, Gadgeteer, BeagleBone, Cubieboard, Z1, WiSense, Mulle, và T-Mote Sky. Đi kèm theo đó có một số hệ điều hành như TinyOS, Contiki, LiteOS, Riot OS và Android.

Điện toán đám mây cũng là một trong những phần quan trọng khi nhắc đến yếu tố tính toán trong IoT. Mọi thiết bị gửi dữ liệu lên đám mây, tại đây có máy chủ cấu hình

mạnh và kho dữ liệu lớn để xử lý, tính toán dữ liệu thô rồi gửi trả thông tin hữu ích cần thiết cho các thiết bị.

1.3.5 Dịch vụ

Dịch vụ là những yếu tố liên quan đến ứng dụng thực tế của IoT vào cuộc sống. Đó là: Tự động hóa trong công nghiệp giúp máy móc, robot có thể thực hiện các quy trình sản xuất nhanh, chính xác và tự động. Chăm sóc sửa khỏe thông minh được phát triển nhờ vào những cảm biến có thể đo nhiều thông số sức khỏe của người dùng. IoT cũng cung cấp các giải pháp kết nối giữa bệnh nhân và phòng khám. Mạng lưới điện thông minh sử dụng IoT để cải thiện và nâng cao mức tiêu thụ điện cho các hộ gia đình và tòa nhà thông qua việc giảm sát, điều khiển và phân phối điện tối ưu dựa trên mật độ dân số. Ngoài ra còn có các ứng dụng về nhà thông minh, thành phố thông minh, giao thông thông minh,…

1.3.6 Ngữ nghĩa

Ngữ nghĩa trong IoT đề cập đến khả năng trích xuất thông tin có ích từ những dữ liệu đã thu thập được để cung cấp cho các dịch vụ cần thiết. Vấn đề này liên quan đến việc khai phá và sử dụng tài nguyên cũng như mô hình thông tin. Ngữ nghĩa giúp ứng dụng gửi yêu cầu đến đúng tài nguyên, các yêu cầu này được định dạng tuân theo một số công nghệ Semantic Web như RDF, OWL. Sử dụng ngữ nghĩa trong IoT, chúng ta có thể tương tác với hệ thống qua ngôn ngữ tự nhiên, đồng thời giảm băng thông mà không ảnh hưởng đến tài nguyên, tuổi thọ pin.

1.4 Tiêu chuẩn chung trong IoT

Nhiều tiêu chuẩn chung cho IoT được đề xuất nhằm đơn giản hóa công việc cho các nhà phát triển cũng như nhà cung cấp dịch vụ. Nhiều tổ chức đã được tạo ra nhằm cung cấp các giao thức hỗ trợ IoT, các tổ chức đi đầu có thể kể đến là: W3C, IETF,

IEEE và ETSI. Bảng 1.2 tổng hợp những giao thức nổi tiếng được định nghĩa bởi các tổ chức này.

Bảng 1.2 Các giao thức trong IoT

<table>
<tr><td colspan="2">Application Protocol</td><td>CoAP</td><td>AMQP</td><td>MQTT</td><td>MQTT -NS</td><td>XMPP</td><td>HTTP REST</td></tr>
<tr><td colspan="2">Service Discovery</td><td colspan="3">mDNS</td><td colspan="3">DNS-SD</td></tr>
<tr><td rowspan="4">Infrastructure Protocols</td><td>Routing Protocol</td><td colspan="6">RPL</td></tr>
<tr><td>Network Layer</td><td colspan="3">6LoWPAN</td><td colspan="3">IPv4/IPv6</td></tr>
<tr><td>Link Layer</td><td colspan="6">IEEE 802.15.4</td></tr>
<tr><td>Physical/ Device Layer</td><td>LTE-A</td><td colspan="2">EPCglobal</td><td>IEEE 802.15.4</td><td colspan="2">Z-Wave</td></tr>
<tr><td colspan="2">Influential Protocols</td><td colspan="3">IEEE 1888.3, IPSec</td><td colspan="3">IEEE 1905.1</td></tr>
</table>

Phần sau của luận văn sẽ tập trung vào trình bày một số giao thức và công nghệ được sử dụng trực tiếp để xây dựng hệ thống IoT trong chương ba.

1.4.1 Một số giao thức tầng ứng dụng

Đi cùng với sự đa dạng của các giao thức tầng dưới, hiện nay, khi nhắc đến các ứng dụng IoT, chúng ta có thể kể đến một số giao thức tầng ứng dụng được liệt kê và so sánh trong Bảng 1.3. Trong đó, HTTP là giao thức phổ biến nhất, nó hiện đang được sử dụng cho hầu hết các server và cũng là tiêu chuẩn ít gặp vấn đề về tương thích. Tuy nhiên, giới hạn của giao thức HTTP nằm ở dung lượng lớn và quá trình giao tiếp phức tạp. CoAP ra đời như một phiên bản thu gọn và cải tiến của HTTP chạy trên nền UDP. Hai giao thức này đều thuộc dạng Request/Response và hỗ trợ kiến trúc RESTful. MQTT có kiến trúc dạng Publish/Subcribe giúp đơn giản hóa quá việc trao đổi dữ liệu thông qua các kênh (topic) được quản lý bởi một Broker trung gian. AMQP thường được dùng cho kết nối từ gateway đến server, nó hoạt động như một hàng đợi

lưu trữ và đảm bảo gói không bị mất. XMPP là giao thức thường được dùng trong các ứng dụng trò chuyện thời gian thực. Tuy nhiên do các đặc điểm kỹ thuật phức tạp và quy trình trao đổi dữ liệu tuân theo chuẩn XML nên khó tương thích với các ứng dụng IoT.

Bảng 1.3 Một số giao thức tầng ứng dụng

Application Protocol	RESTful	Transport	Publish/ Subscribe	Request/ Response	Security	QoS
CoAP	✓	UDP	✓	✓	DTLS	✓
MQTT	×	TCP	✓	×	SSL	✓
MQTT-NS	×	TCP	✓	×	SSL	✓
XMPP	×	TCP	✓	✓	SSL	×
AMQP	×	TCP	✓	×	SSL	✓
HTTP	✓	TCP	×	✓	SSL	×

1.4.1.1 HTTP

Hypertext Transfer Protocol (HTTP) là một giao thức thuộc tầng ứng dụng, có thể truyền tải các tệp tin đa phương tiện. HTTP được thiết kế theo mô hình client-server, trong đó, client mở một kết nối để tạo ra một yêu cầu (request), sau đấy chờ đợi cho đến khi nó nhận được phản một phản hồi (response) từ server. Cấu trúc bản tin HTTP được mô tả như trong hình dưới đây:

<table>
<tr><td>Request Line</td><td>Status Line</td></tr>
<tr><td colspan="2">General Header</td></tr>
<tr><td>Request Header</td><td>Response Header</td></tr>
<tr><td colspan="2">Entity Header</td></tr>
<tr><td colspan="2">Empty Line</td></tr>
<tr><td colspan="2">Message Body</td></tr>
</table>

Hình 1.3 Cấu trúc bản tin HTTP

a) HTTP request

HTTP request gồm hai phần chính là header (bắt buộc) và Message Body (có thể trống tùy vào loại bản tin). Header bao gồm:

- HTTP Request-Line gồm một phương thức (POST, GET,…), một địa chỉ định danh tài nguyên (URI), và phiên bản HTTP. Ví dụ: *GET / HTTP/1.1* mô tả phương thức GET tới URI là / (ở đây nó đại diện cho tài nguyên gốc) và dùng phiên bản HTTP/1.1.
- Tiếp sau Request-Line, một HTTP request có thể bao gồm một hay nhiều dòng message header. Một message header có thể chứa các loại General Header, Request Header, hoặc Entity Header. General Header áp dụng trong truyền dữ liệu; Request Header áp dụng cho các request cụ thể, và Entity Header phục vụ cho Message Body trong request.

HTTP request luôn chứa một dòng trống sau Request Line với bất kỳ header nào. Nếu request bao gồm một Message Body, phần body đi sau một dòng trống. Dòng trống (blank line) rất quan trọng vì server xác định được phần kết thúc của request, hoặc phần kết của header. Không có dòng trống, server nhận các message sẽ không biết được các header khác nữa có tiếp tục được truyền không.

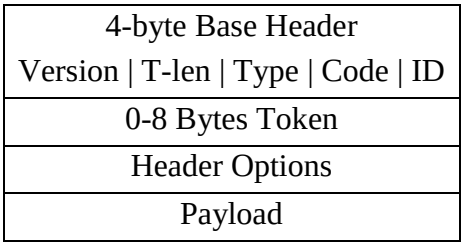
b) HTTP response

HTTP Response khá giống với HTTP Request. Dấu hiệu khác biệt duy nhất là response bắt đầu với Status Line, trong khi request bắt đầu bằng Request Line. Status Line, cũng giống như Request Line, chứa ba mục ngăn cách bởi các khoảng trống. HTTP Status Line bắt đầu với phiên bản cao nhất mà server hỗ trợ, tiếp sau là mã trạng thái (Status Code), và một đoạn text mô tả response (Reason Phrase. Status Code). Ví dụ: *HTTP/1.1 200 OK* cho biết HTTP/1.1 là phiên bản HTTP, mã 200 ứng với mô tả thực hiện thành công (OK).

1.4.1.2 CoAP

The Constrained Application Protocol (CoAP) [15] là một giao thức tầng ứng dụng chuyên biệt phù hợp với những thiết bị và mạng bị giới hạn cấu hình phần cứng hoạt

động trong môi trường IoT. Nó dựa trên kiến trúc REST và hỗ trợ IPv6 thông qua Low-Power Wireless Personal Area Networks (6LoWPANs). Mặc dù mô hình kiến trúc của CoAP rất giống với HTTP (chúng đều sử dụng cặp bản tin request/response và các response code), CoAP có kích thước bản tin nhỏ hơn và phương pháp mã hóa hiệu quả hơn giúp giảm dung lượng bộ nhớ thiết bị, băng thông mạng. Cấu trúc của bản tin CoAP được mô tả trong Hình 1.4 sau đây:



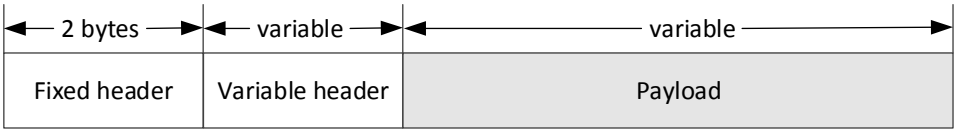
Hình 1.4 Cấu trúc bản tin CoAP

Phần header của một bản tin CoAP gồm 4-byte cố định xác định Version (phiên bản CoAP sử dụng), T-len (xác định chiều dài trường Token Length), Type (loại bản tin sử dụng Confirmbale 0, non-confirmable 1, Acknowledgement 2, Reset 3), Code và ID (sử dụng để phát hiện các bản tin trùng lặp). Theo sao đó là trường Token có độ dài từ 0 đến 8 bytes, và trường Options chứa những tham số có thể cấu hình thêm (Location, Max-Age, Etag,…). Phần Payload chứa nội dung của bản tin.

1.4.1.3 MQTT

MQTT (MQ Telemetry Transport) [16] là một giao thức truyền tin nhỏ gọn dạng publish/subcribe. Có hai loại thực thể trong mô hình MQTT là client và broker/server. Tất cả các client đều kết nói đến broker trên nền giao thức TCP. Broker chịu trách nhiệm tiếp nhận và điều phối bản tin từ nơi gửi đến đúng nơi nhận. MQTT có tính hướng bản tin, mỗi bản tin là một đoạn rời rạc của tín hiệu không được đọc tại broker. Bản tin được publish đến một địa chỉ xác định (tương ứng với một kênh xác định). Khi client gửi bản tin đến kênh gọi là quá trình publish. Client đăng ký (subcribe) kênh để nhận dữ liệu và một client có thể đăng nhiều kênh. Một kênh nào đó nhận dữ

liệu publish sẽ chuyển tiếp nó đến tất cả các client đã đăng ký kênh. Một bản tin MQTT thông thường bao gồm các thành phần được mô tả như Hình 1.5 dưới đây:



Hình 1.5 Cấu trúc bản tin MQTT

Trong đó:

Phần Fixed header gồm 2 bytes chứa các thông tin điều khiển, các thông tin xác định kiểu bản tin, các Flag tương ứng cho các tính năng khác nhau. Nội dung bản tin được chứa trong Payload.

1.4.2 Một số công nghệ truyền thông không dây

Bảng 1.4 tổng hợp một số công nghệ kết nối không dây sử dụng cho IoT.

Bảng 1.4 Các công nghệ truyền dẫn trong IoT [17]

Protocol	Optimized for Extended Battery Life	Nominal Range Limit	Typical Data Rate	Spectrum
Bluetooth	✓	Personal (<10m)	2Mbps	ISM 2.4GHz unlicensed
NFC	✓	Contact (<4cm)	100kbps	ISM 13.56MHz unlicensed
Wi Fi	✗	Local (<100m)	>100Mbps	ISM 2.4GHz/5GHz unlicensed
LoRaWAN	✓	Metro (>10km)	<50kpbs	ISM 900MHz unlicensed
NB-IoT	✓	Metro (>10km)	200kbps	Licensed cellular
2G | 3G	✗	Metro (>30km)	<2Mbps	Licensed cellular
4G LTE	✗	Metro (>30km)	>100Mbps	Licensed cellular

Các yếu tố thường được xem xét khi lựa chọn một loại công nghệ là mức tiêu tốn năng lượng, vùng phủ, tốc độ truyền dữ liệu và băng tần hoạt động. Với các ứng dụng cần khoảng cách truyền dẫn xa, thực hiện trên các thiết bị năng lượng hạn chế, chúng ta thường chọn các công nghệ như LoRa, NB-IoT. Các ứng dụng yêu cầu tốc độ truyền cao sẽ phải trả giá về mặt năng lượng hoặc vùng phủ, chẳng hạn Bluetooth có tốc độ tối đa lên đến 2Mbps nhưng chỉ sử dụng ở cấp độ cá nhân với phạm vi không

quá 10m, các công nghệ mạng di động như (2G, 3G, 4G LTE) và Wi-Fi tiêu tốn mức năng lượng lớn.

1.4.2.1 Wi-Fi

Wi-Fi [1] viết tắt từ Wireless Fidelity là phương thức kết nối không dây sử dụng sóng vô tuyến. Wi-Fi hiện tại đang sử dụng theo tiêu chuẩn IEEE 802.11, có thể hoạt động ở các giải tần 2.4GHz, 5GHz và 60GHz. Phạm vi truyền dẫn của Wi-Fi với các anten cơ bản đạt khoảng 50m. Tốc độ kết nối của Wi-Fi phụ thuộc vào các phiên bản. Chuẩn 802.11n hoạt động ở tần số 2.4GHz có thể đạt tốc độ tố đa 450MB/s, chuẩn 802.11ac hoạt động ở tần số 5GHz có tốc độc lên đến 1.3GB/s. Ngày nay, Wi-Fi được sử dụng rất phổ biến, hầu hết các thiế bị điện tử đều có tích hợp công nghệ này.

1.4.2.2 Zigbee

ZigBee [1] là tên thương mại của tiêu chuẩn IEEE 802.15.4 sử dụng cho mạng không dây cá nhân tốc độ thấp - Wireless Personal Area Network standard (WPANs). Nó hoạt động trên dải băng tần ISM, sử dụng băng tần 868 MHz ở châu Âu, 915MHz tại Mỹ và 2.4GHz ở hầu hết quốc gia khác. Tốc độ truyền dẫn phụ thuộc vào tần số sử dụng, tuy nhiên tốc độ tối đa có thể đạt đến là 250Kbps. Mặc dù Zigbee có tốc độ chậm hơn các công nghệ không dây phổ biến như Wi-Fi nhưng nó tiêu tốn năng lượng thấp và giá thành rẻ. Zigbee thường được sử dụng trong các ứng dụng khoảng cách ngắn (khoảng 10-100m) và dữ liệu truyền tin ít nhưng thường xuyên, được đánh giá phù hợp với các ứng dụng smarthome hoặc trong một khu vực đô thị/khu chung cư.

1.4.2.3 Bluetooth

Bluetooth [18] hoạt động ở dải tần 2.4GHz chung với ZigBee và Wi-Fi, phạm vi truyền trong khoảng 50-150m. Trong mạng Bluetooth, có thể phân các thiết bị ra làm hai loại: master và slave. Một master có thể kết nối với nhiều nhất 7 slave khác nhau,

trong khi mỗi slave chỉ kết nối với duy nhất một master. Do đó, master có thể gửi dữ liệu tới bất kì slave nào, trong khi slave chỉ được phép truyền và nhận với master của chúng. Tại thời điểm nhất định, một thiết bị Bluetooth chỉ có thể là master hoặc slave, tuy nhiên, vai trò này có thể thay đổi.

1.5 Kết luận chương

IoT thường được tiếp cận theo ba hướng nhìn liên quan đến các thiết bị cảm biến thu thập thông tin, giao thức truyền thông và cách xử lý thông tin lấy được. Bức tranh tổng quan về IoT đã được vẽ ra thông qua kiến trúc năm tầng và các nhân tố chính. Trong ba giao thức tầng ứng dụng phổ biến, CoAP được xem như phiên bản rút gọn của HTTP, trong khi, MQTT cung một kiến trúc Pub/Sub khác biệt. Công nghệ Wi-Fi đem lại độ tin cậy, tốc độ truyền tốt nhưng hạn chế về mức tiêu thụ năng lượng, trong khi, Zigbee và Bluetooth được xem là những giải pháp tốt để kết nối các thiết bị trong mạng cá nhân.

Chúng ta có thể thấy thị trường IoT hiện tại rất đa dạng tuy nhiên bị đánh đổi bởi sự phân mảnh. Mỗi nhà cung cấp dịch vụ sẽ chọn cho mình giải pháp từ phần cứng đến phần mềm riêng biệt sao cho phù hợp nhất với yêu cầu cần triển khai. Điều này dẫn đến nhu cầu tạo ra một nền tảng dùng chung, có thể tích hợp nhiều công nghệ, giao thức trong một kiến trúc duy nhất. oneM2M chính là một trong số giải pháp rất đáng được xem xét, luận văn sẽ đề cập chi tiết trong phần sau.

CHƯƠNG 2. TỔNG QUAN VỀ ONEM2M

Chương hai giới thiệu về oneM2M, một tiêu chuẩn toàn cầu cho kết nốt M2M và IoT. Nội dung của chương chủ yếu được chắt lọc từ tài liệu kỹ thuật [19] và [20], tóm tắt và trình bày chi tiết những phần thiết yếu nhất để có thể triển khai một nền tảng, hệ thống theo tiêu chuẩn oneM2M. Kiến trúc hệ thống, giao thức và phương pháp tích hợp đa công nghệ là những nội dung quan trọng được làm rõ.

2.1 Lịch sử phát triển

2.1.1 Sự ra đời của oneM2M

Ngày nay, hầu hết các giải pháp IoT/M2M trong công nghiệp sử dụng các hệ thống độc quyền, bao gồm tất cả các tầng từ vật lý đến ứng dụng để cung cấp các dịch vụ chuyên dụng cho từng đối tượng khách hàng nhất định. Các hệ thống độc quyền này tạo ra khó khăn trong việc mở rộng hệ thống, hỗ trợ dịch vụ mới, tích hợp dữ liệu mới và tương tác với các hệ thống M2M khác. Hơn nữa, việc thiết kế, triển khai và bảo trì những giải pháp độc quyền có thể gây tăng chi phí.

Do đó, rất cần phát triển nền tảng chung cho nhiều ngành dọc, nhằm tăng hiệu quả phát triển của giải pháp M2M. Một nền tảng dịch vụ chung (Common service platform) cũng cần thiết để đơn giản các ứng dụng đa ngành như hệ thống điện, thành phố thông minh, và cho phép triển khai M2M liền mạch giữa các hệ thống không đồng nhất. Vấn đề này thúc đẩy rất nhiều các tổ chức thành lập một dự án hợp tác chung, "oneM2M Global Initiative", để chuẩn hóa một lớp nền tảng dịch vụ chung cho M2M có thể sử dụng rộng rãi và độc lập với từng dịch vụ.

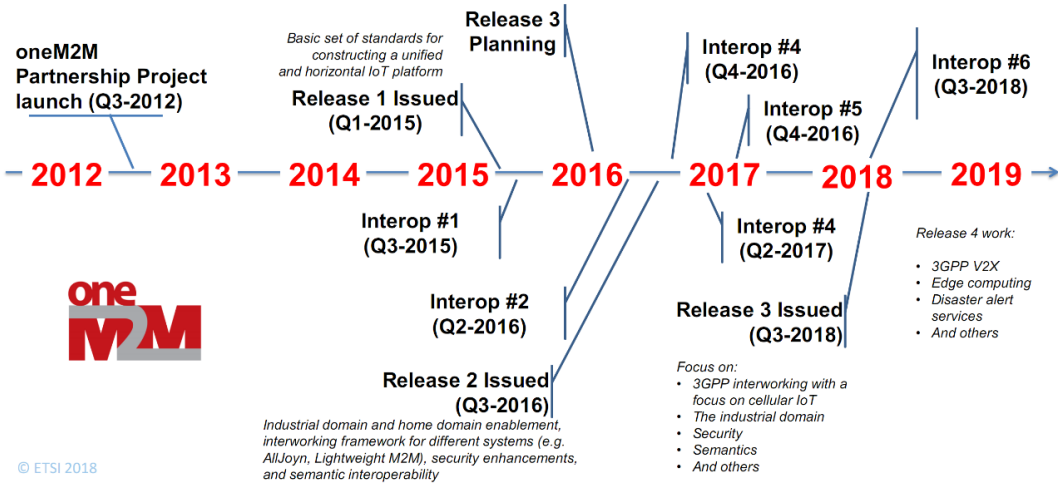
oneM2M Global Initiative là một dự án hợp tác quốc tế, thành lập với nhiệm vụ chung tay tạo ra những thiết bị toàn cầu, không phụ thuộc vào các dịch vụ, giải pháp M2M cụ thể. Lớp dịch vụ dùng chung có thể nhúng trong rất nhiều phần cứng

cũng như phần mềm để đảm bảo chúng có thể chạy được trên phạm vi toàn cầu. oneM2M được đề xướng bởi bảy tổ chức tiêu chuẩn viễn thông: Hiệp hội thương mại và công nghiệp vô tuyến, Association of Radio Industries and Businesses (ARIB), Ủy ban công nghệ viễn thông Nhật Bản, Telecommunication Technology Committee (TTC), Liên minh các giải pháp công nghiệp viễn thông, the Alliance for Telecommunications Industry Solutions (ATIS), Hiệp hội Công nghiệp Viễn thông được Viện Tiêu chuẩn Quốc gia Hoa Kỳ Telecommunications Industry Association (TIA), Hiệp hội tiêu chuẩn kết nối Trung Quốc, the China Communications Standards Association (CCSA), Viện tiêu chuẩn Viễn thông châu Âu, the European Telecommunications Standards Institute (ETSI), Hiệp hội công nghệ viễn thông Hàn Quốc, the Telecommunications Technology Association (TTA). Số lượng các công ty thành viên lên đến con số 270, họ cùng đóng góp vào oneM2M.

Mục tiêu chính của oneM2M là tối giản sự phân mảnh của các lớp dịch vụ tiêu chuẩn M2M trên thị trường bằng cách hợp nhất chúng và cùng nhau phát triển tiêu chuẩn toàn cầu. Để đạt được mục tiêu đó, 7 tổ chức phát triển tiêu chuẩn đã tập hợp các tài liệu M2M có sẵn của họ, dừng các công việc chồng chéo nhau, vươn tới một tổ chức và cộng đồng chung. oneM2M được tái sử dụng những tiêu chuẩn đang hiện có, ví dụ như BBF, OMA. Đặc biệt, oneM2M có kế hoạch cung cấp những lợi ích sau đây đến hệ sinh thái M2M:

- Tiết kiệm chi phí mở rộng mạng lưới và rút ngắn thời gian thương mại hóa cho các giải pháp IoT/M2M
- Đơn giản hóa quy trình phát triển ứng dụng nhờ vào các gói API ở cả phía các đơn vị phát triển và nhà mạng.
- Thúc đẩy các mạng toàn cầu tăng cường các dịch vụ tiềm năng và mở rộng cơ hội thương mại hóa dựa trên việc kết nối giữa các tiêu chuẩn hiện có.

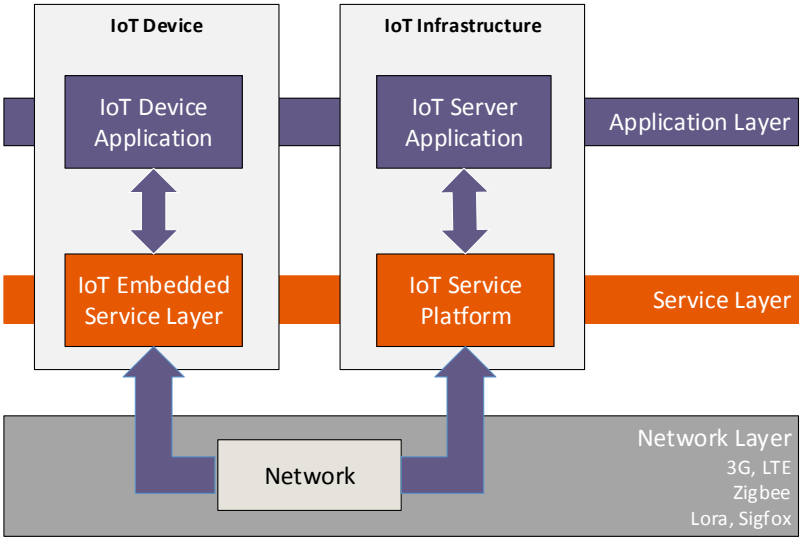
2.1.2 Sự phát triển của oneM2M



Hình 2.1 Sự phát triển của oneM2M

Sự phát triển của oneM2M từ khi ra đời đến nay (năm 2019), được thể hiện trong Hình 2.1. OneM2M bắt đầu được triển khai hợp tác nghiên cứu từ năm 2012. Đến tháng 1 năm 2015, bản Release 1 chính thức được tung ra, tuy nhiên nội dung chủ yếu dựa trên Release 2 của tiêu chuẩn ETSI đã được phát triển trước đó trong giao đoạn 2009-2012. Ở phiên bản đầu tiên này, tiêu chuẩn oneM2M đã đưa ra được các thành phần cơ bản cốt lõi cho hệ thống và có thể hoạt động ổn định. Cùng trong năm 2015, dịch vụ thương mại đầu tiên tuân theo tiêu chuẩn oneM2M được đưa vào hoạt động tại Hàn Quốc. Release 2 được phát hành vào năm 2016 cho phép liên kết hệ thống oneM2M với các hệ thống khác như AllJoyn, Lightweight M2M, đồng thời tăng cường khả năng bảo mật và tương tác về mặt ngữ nghĩa (semantic). Phiên bản mới nhất, Release 3, tập trung vào việc liên kết oneM2M với các mạng di động của tổ chức 3GPP. Dự kiến trong năm 2019, oneM2M phát triển lên release 4 với hướng tích hợp điện toán biên (edge computing).

2.2 Kiến trúc hệ thống



Hình 2.2 Kiến trúc ba tầng hệ thống IoT

Hình 2.2 mô tả một IoT Device (có thể là cảm biến) và IoT Infrastructure (có thể là một máy chủ), cả hai đều có thể nhìn dưới hai ứng dụng và tầng dịch vụ. Chúng kết nối với nhau thông qua tầng mạng. Sau đây là mô tả chức năng của ba tầng:

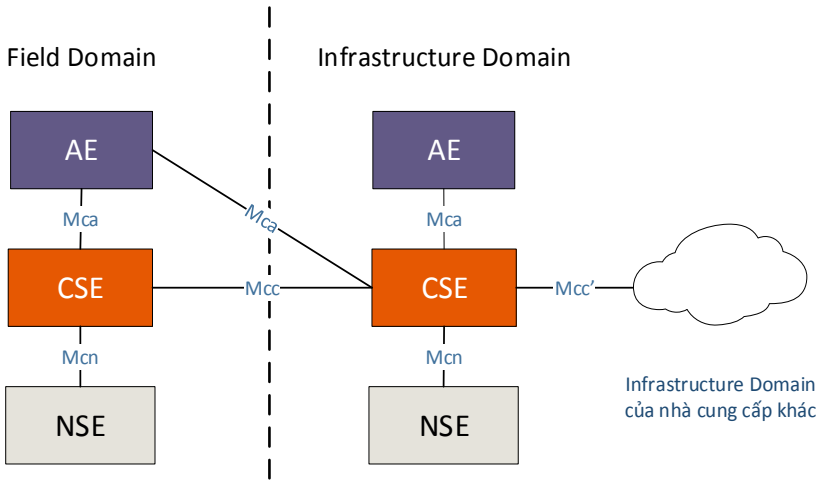
- Tầng mạng (Network Layer) đảm bảo việc truyền tin giữa các thiết bị (có thể là end-device, gateway, sever) với nhau. Các công nghệ được sử dụng trong lớp mạng này rất đa dạng: mạng di động 3G, 4G, LTE hay các mạng LPWAN như Lora, Sigfox,…
- Tầng ứng dụng (Application Layer) gồm những ứng dụng, giao diện web… tương tác trực tiếp với người sử dụng.
- Tầng dịch vụ (Service Layer) thực hiện quản lý, chuyển tiếp, giám sát luồng dữ liệu từ lớp mạng đi lên cũng như ứng dụng đi xuống.

Trong khi tầng mạng chứa rất nhiều công nghệ truyền dẫn khác nhau, tầng ứng dụng có thể dễ dàng lập trình thì thách thức hợp nhất cấu trúc nằm ở lớp dịch vụ. Hệ thống kiến trúc oneM2M được thiết kế tập trung vào những chức năng cơ bản (ví dụ như đăng ký, xử lý bản tin) và nhiều tính năng nâng cao khác (ví dụ như tích hợp với hệ

thống khác). Để làm được điều đó, oneM2M định nghĩa lớp dịch vụ dùng chung hoàn toàn độc lập với lớp mạng.

2.2.1 Kiến trúc mức chức năng

Dựa vào phân tầng nêu trên, oneM2M đưa ra kiến trúc mạng mức chức năng như Hình 2.3 sau đây.



Hình 2.3 Kiến trúc chức năng oneM2M

Miền Infrastructre Domain đại diện cho một nhà cung cấp dịch vụ oneM2M (thường là các đơn vị nhà mạng), miền Field Domain đại diện cho phía người dùng (bao gồm các thiết bị cuối như sensor, thiết bị tập trung như gateway…). Ở mỗi miền đều chứa các thực thể cơ bản AE, CSE, NSE được liên kết với nhau qua các giao diện Mca, Mcn, Mcc và Mcc'.

Trong oneM2M, các thực thể bao gồm:

- **AE (Application Entity)** là một thức thể trong tầng ứng dụng, có triển khai dịch vụ logic của oneM2M và được định danh bằng AE-ID. Chúng ta có thể coi AE đại diện cho *một ứng dụng* giao tiếp với người dùng. Ví dụ như ứng dụng quản lý nhà thông minh, quản lý phương tiện giao thông, đo lượng đường trong máu,…

- **NSE (Network Services Entity)** cung cấp các dịch vụ mạng cho CSE, chẳng hạn như device triggering, hỗ trợ quả lý thiết bị, vị trí. Những dịch vụ này liên quan đến khả năng của lớp mạng.
- **CSE (Common Service Entity)** đại diện cho các dịch vụ ngang hàng. Đây cũng là thành phần quan trọng nhất trong đặc điểm kỹ thuật của oneM2M.

CSE thường gồm một số chức năng gọi là **CSFs (Common Services Funtions)**. Một vài CSFs cung cấp chức năng quản trị cho lớp dịch vụ, ví dụ như:

- Registration (REG), chức năng đăng ký cho phép một AE hoặc CSE đăng ký với một CSE và sử dụng dịch vụ được cung cấp bởi CSE đó.
- Security (SEC), chức năng bảo mật cho phép thiết lập bảo mật của kết nối giữa các dịch vụ và dữ liệu riêng tư.
- AE and a service layer management (ASM), chức năng quản lý AE và các tính năng tùy chỉnh khác như tìm lỗi và nâng cấp.
- Device management (DMG), chức năng quản lý thiết bị thực hiện quản lý một số chắc năng của các thiết bị, ví dụ như cập nhật firmware.
- Communication management and delivery handling (CMDH), chức năng quản lý liên kết và điều khiển phân phối chịu trách nhiệm phân phối bản tin lớp dịch vụ.
- Network service exposure (NSE), chức năng này giúp kiên kết giữa lớp dịch vụ với các dịch vụ được cung cấp trong lớp mạng cơ bản.

Ngoài ra, một số chức năng được thêm vào để đăng ký giữa các AE và CSE:

- Data management and repository (DMR), chức năng quản lý và lưu trữ dữ liệu chịu trách nhiệm lưu trữ dữ liệu người dùng và xử lý chúng. Người dùng có thể đăng ký và được thông báo khi có sự thay đổi dữ liệu.

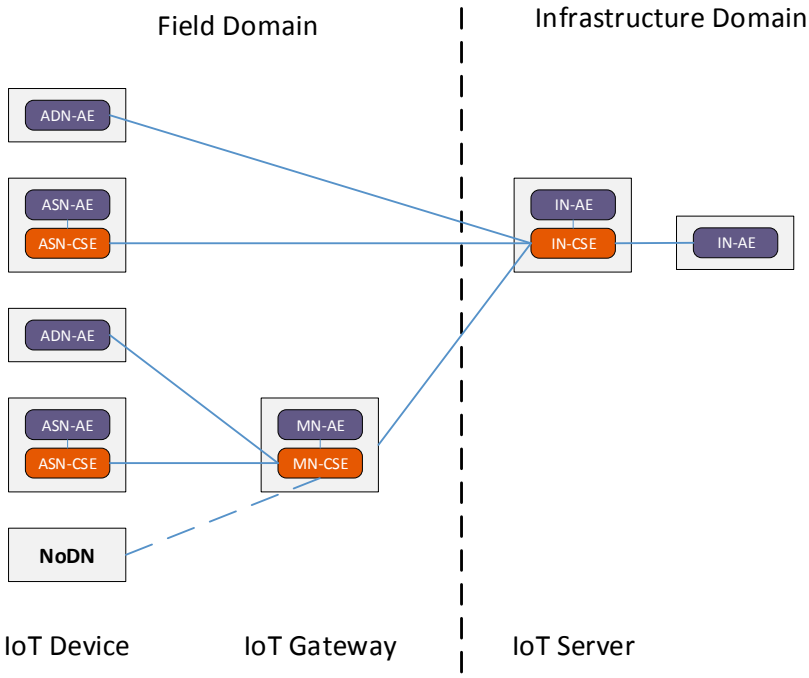
- Discovery (DIS), chức năng khám phá tạo ra các dữ liệu và dịch vụ được phép truy cập bởi CSE và AE khác.
- Subscription and notification (SUB), chức năng đăng ký và thông báo quản lý những đăng ký thay đổi trong nền tảng oneM2M.
- Service session management (SSM), chức năng quản lý phiên dịch vụ hỗ trợ các phiên dịch vụ kết nối điểm-điểm.
- Service charging and accounting (SCA), chức năng này cung cấp cơ chế hỗ trợ trả phí dựa trên dịch vụ.
- Group management (GMG), chức năng quản lý nhóm hỗ trợ triển khai và quản lý các nhóm thành viên.
- Location (LOC), chức năng định vị cho phép M2M AE lấy thông tin về vị trí địa lý.

Các thực thể giao tiếp với nhau thông qua các giao diện sau:

- **Mca** kết nối AE với CSE, hai thực thể này có thể nằm trong cùng một thiết bị vật lý. Mca giúp AE sử dụng các dịch vụ được cung cấp trong CSE.
- **Mcn** kết nối NSE với CSE giúp CSE có thể dùng các dịch vụ đang có sẵn trong NSE. Chẳng hạn khi NSE là mạng di động, CSE có thể dùng chung các dịch vụ được cung cấp bởi mạng 3GPP.
- **Mcc** kết nối giữa các CSE trong cùng một Infrastructre Domain, cho phép chúng trao đổi, dùng chung các dịch vụ của nhau.
- **Mcc'** tương tự Mcc, nhưng các CSE khác miền Infrastructre Domain.

2.2.2 Kiến trúc mức vật lý

Kiến trúc mức vật lý thể hiện các liên kết rõ ràng hơn khi gắn các thực thể logic vào thiết bị vật lý. Hình 2.4 mô tả các node mạng trong oneM2M và giao diện kết nối của chúng với nhau.



Hình 2.4 Kiến trúc mức vật lý

Trong mỗi mạng oneM2M bắt buộc phải có một IN.

IN (Infrastructure Node): Trong mỗi Infrastructre Domain bắt buộc có duy nhất một IN, có thể xem nó như server của nhà cung cấp dịch vụ. IN-CSE có thể kết nối với tất cả các CSE khác (ASN-CSE, MN-CSE, IN-CSE khác).

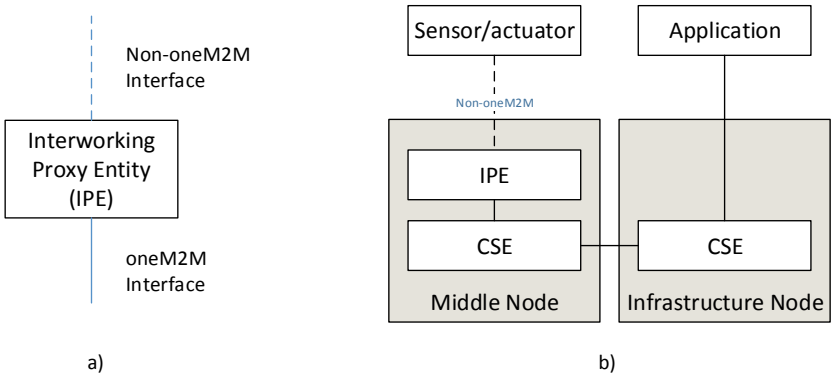
Các phần tử liệt kê dưới đây, có thể tồn tại hoặc không:

MN (Middle Node): Là thiết bị tập trung, ví dụ như gateway.

ADN (Application Dedicated Node): Đại diện cho những ứng dụng, bắt buộc phải có ít nhất một AE và không bao gồm CSE, ADN là đại diện cho một nút mạng bên ngoài có thể giao tiếp với hệ thống oneM2M qua các giao diện Mca trên nền các giao thức tầng ứng dụng được hỗ trợ.

ASN (Application Service Node): Gồm một CSE và ít nhất một AE. ASN-CSE, ứng với ASN là một node mạng trên trong hệ thống oneM2M.

Non-oneM2M Node (NoDN): là những thiết bị xây không sử dụng tiêu chuẩn oneM2M. Để kết nối chúng vào mạng, chúng ta cần sử dụng một AE trung gian, thông qua AE kết nối vào hệ thống thông qua giao diện Mca.



Hình 2.5 Interworking Proxy

AE trung gian được mô tả trong Hình 2.5a) là Interworking Proxy Entity (IPE) có nhiệm vụ ánh xạ giao diện dạng non-oneM2M sang dạng oneM2M. Hình 2.5b) minh họa một cấu hình cụ thể giúp kết nối các thiết bị non-oneM2M vào MN. IPE là nơi tiếp nhận các bản tin từ thiết bị sử dụng giao thức non-oneM2M, chuyển đổi nó sang dạng oneM2M rồi đưa tới các phần tử khác trong mạng. Tùy thuộc vào mục đích của người thiết kế, IPE có thể chuyển đổi mô hình dữ liệu đầu vào sao cho phù hợp với mô hình tài nguyên trong oneM2M hoặc thậm chí liên kết với các nền tảng khác tương tự oneM2M như IoTivity của Open Connectivity Foundation (OCF) and AllJoyn của AllSeen Alliance [9].

2.3 Giao thức oneM2M

oneM2M sử dụng kiến trúc REST có đặc điểm hướng tài nguyên (resource-based). Tất cả mọi thực thể trong hệ thống đều được biểu diễn dưới dạng tài nguyên, các tài nguyên này có thể được truy cập theo bộ 4 phương thức CRUD dùng để tạo mới, truy

xuất, cập nhật, xóa. Mỗi phương thức được thực hiện thông qua một cặp bản tin Request/Respone, các bản tin này có thể được biểu diễn theo nhiều kiểu (XML, JSON) và truyền đi trên nhiều giao thức truyền thông như HTTP, CoAP, MQTT.

2.3.1 Kiến trúc REST

Representational State Transfer (**REST**) là kiến trúc phần mềm xác định một tập hợp các quy ước, ràng buộc sử dụng cho việc tạo dịch vụ web. REST không phải là một giao thức, nó bao gồm tập hợp các nguyên tắc cần tuân thủ để thiết kế, xây dựng phần mềm. Mọi dữ liệu (văn bản, ảnh, video,…) đều được coi là tài nguyên được định danh bởi mã định danh chuẩn URI (Uniform Resource Identifier) và có thể biểu diễn theo nhiều cách khác nhau (XML, JSON). Mỗi tài nguyên đều có tính trạng thái (stateful) và chứa đường liên kết đến các tài nguyên khác. REST có thể dễ dàng triển khai với nhiều giao thức như HTTP, CoAP,…

REST server cung cấp quyền truy cập các tài nguyên, REST client được cho phép truy cập và thay đổi các tài nguyên đó. Các phương thức truy cập tài nguyên (operation) được định nghĩa trước (ví dụ với HTTP có GET, POST, PUT, DELETE). Các phương thức này truy cập tới tài nguyên thông qua những giao diện chuẩn.

RESTful được hiểu là một hệ thống đáp ứng đầy đủ 5 ràng buộc của REST. Những ràng buộc này hạn chế cách mà server có thể xử lý và hồi đáp bản tin từ client.

1. **Client-server**: Tách biệt rõ các thực thể client và server. Trong đó, server chứa tập hợp các dịch vụ nhằm xử lý yêu cầu của client.
2. **Stateless (phi trạng thái):** Server và client sẽ không lưu trạng thái của nhau, những trạng thái này có thể chuyển thành trạng thái của tài nguyên. Mỗi cặp request/response chứa đầy đủ thông tin để client/server có thể hiểu nhau. Đặc tính này giúp hệ thống dễ phát triển và mở rộng cũng như bảo trì.

3. **Cacheability (khả năng lưu trữ)**: Các response có thể tái sử dụng từ bộ đệm của server giúp giảm thiểu thời gian xử lý. Khi đó, các request phải đảm bảo tính duy nhất để response không bị nhầm lẫn với các request khác.
4. **Layered system (phân lớp hệ thống)**: Hệ thống được chia làm nhiều phân lớp, việc giao tiếp của một lớp chỉ được tiến hành với lớp phía ngay trên và dưới của nó. Điều có giúp hệ thống khả năng tích hợp thêm các chức năng như cân bằng tải (load balancing), đồng thời bộ dệm dữ liệu trong hệ thống cũng sẽ được cải thiện.
5. **Uniform interface (chuẩn hoá giao diện):** Để đảm bảo ràng buộc này, hệ thống tập trung vào việc xử lý các tài nguyên. Mỗi một tài nguyên sẽ được xác định bằng một URI riêng biệt và có thể tương tác với các tài nguyên khác. Các tài nguyên được biểu diễn dưới dạng bản tin và có môt tả dữ liệu (metadata) của chính nó. Những bản tin này có thể trao đổi giữa các thành phần qua API.

2.3.2 Giao diện lập trình ứng dụng oneM2M

Giao diện lập trình ứng dụng oneM2M API (Application Program Interface) API là các phương thức kết nối với thư viện, cung cấp khả năng truy xuất đến một tập các hàm hay dùng, giúp AE và CSE có thể trao đổi bản tin thông qua một trong các giao diện của oneM2M (Mca//Mcc/Mcc'). Các API được thiết kế theo một số nguyên tắc sau:

- APIs tuân theo những nguyên tắc của hệ thống RESTful
- APIs chỉ ra những đặc tính được hỗ trợ và không được hỗ trợ tại mỗi điểm tham chiếu tuân theo [19]
- APIs định nghĩa cách đánh địa chỉ cho tài nguyên, cách tương tác với tài nguyên và định danh cho tài nguyên đó bằng URI.

- APIs cung cấp định dạng và cú pháp cho các phương thức tác động đến tài nguyên. Trong trường hợp có sử dụng kết hợp các giao thức, mỗi phương thức có khả năng ánh xạ trong các giao thức khác nhau, được miêu tả chi tiết trong [21]–[23].
- Mỗi tài nguyên đều có một cách biểu diễn và được tương tác thông qua các phương thức bao gồm: CREATE. RETRIEVE, UPDATE, DELETE.
- Các bản tin cũng như cách chúng được trao đổi tùy thuộc vào loại tài nguyên được miêu tả trong [19] và có thể ánh xạ với các giao thức tầng ứng dụng.

2.3.3 Phương thức oneM2M

oneM2M sử dụng 6 phương thức CRUD+N. Bên gửi (Orginator) sẽ gửi các bản tin CRUD+N request đến bên nhận (Receiver). Tùy thuộc vào từng phương thức mà bên gửi và bên nhận có thể là CSE hoặc AE. Mỗi bản tin request gửi đi được xử lý và trả về một bản tin response. Mỗi phương thức sẽ tác động lên các tài nguyên, mỗi tài nguyên bao gồm các thuộc tính (attribute) và tài nguyên con (child resource).

CREATE

Thông thường, để tạo mới một tài nguyên, phương thức CREATE được sử dụng. Bên gửi gửi bản tin request đến bên nhận, sau khi bản tin được xử lý, tài nguyên trong cơ sở dữ liệu được tạo mới, đồng thời bản tin response được gửi trả về bên gửi. Bản tin response thông báo kết quả cùng với nội dung của tài nguyên mới được tạo.

RETRIEVE

Phương thức RETRIEVE được sử dụng khi bên gửi muốn lấy/khám phá thông tin về một tài nguyên nhất định. Bên nhận sau khi nhận được request sẽ gửi trả về bản tin response chứa tất cả các thuộc tính của tài nguyên được yêu cầu, tuy nhiên, nó không bao gồm các thuộc tính nằm trong tài nguyên con của tài nguyên được yêu cầu.

UPDATE

Bản tin UPDATE request tác động đến các thuộc tính của một tài nguyên đích xác định. Phương thức này làm thay đổi giá trị của thuộc tính, tạo mới thuộc tính nếu nó chưa có trong tài nguyên đích, và xóa thuộc tính bằng cách cập nhật giá trị null. Bản tin response chứa nội dung các thuộc tính sau khi được cập nhật.

DELETE

Phương thức DELETE trái ngược với CREATE, mục đích của nó là xóa bỏ tài nguyên trong cơ sở dữ liệu. Bản tin response của phương thức thông báo kết quả đã thành công hay thất bại.

NOTIFY

Khi CSE muốn gửi dữ liệu đến AE hoặc CSE khác, nó thực hiện thông qua bản tin NOTIFY Request. Phương thức này thường được sử dụng khi một thực thể muốn theo dõi để biết được các thay đổi trong một tài nguyên xác định.

2.3.4 Tài nguyên oneM2M

Mỗi tài nguyên là một cấu trúc dữ liệu, có địa chỉ duy nhất được đánh bằng một chuỗi ký tự định danh tài nguyên URI. Các tài nguyên này được liên kết với các CSFs để hỗ trợ hoạt động của CSE.

Các loại tài nguyên

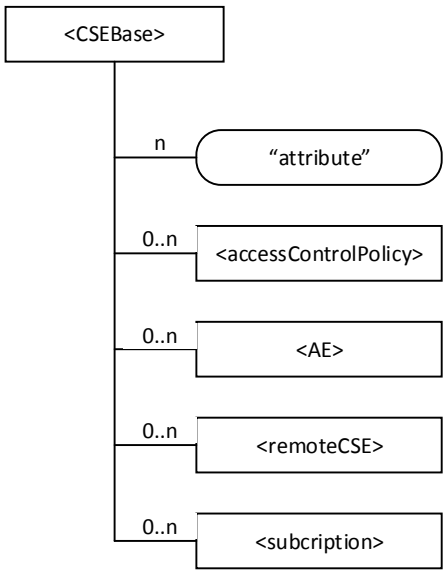
Tất cả các thực thể trong hệ thống oneM2M, như AE hay CSE, dữ liệu của cảm biến, lệnh điều khiển,… đều được biểu diễn dưới dạng các tài nguyên. Các tài nguyên này được phân loại chi tiết, có thể bao gồm các thuộc tính cũng như tài nguyên con. Mỗi tài nguyên được xác định thông qua *resource ID* và có thể truy cập đến dựa vào địa chỉ *URI*. Tùy thuộc vào loại tài nguyên mà chúng có thể có hoặc không các thuộc tính và tài nguyên con. Dưới đây, Bảng 2.1 liệt kê một số ResourceType điển hình.

Bảng 2.1 Một số loại tài nguyên

Resource Type	Short Description
accessControlPolicy	Kiểm soát "ai" được phép "làm gì" và nội dung nó có thể truy cập trong một tài nguyên xác định.
AE	Lưu trữ thông tin về thực thể ứng dụng. Nó được tạo ra sau khi ứng dụng đăng ký thành công với CSE.
container	Dùng để chia sẻ dữ liệu giữa các thực thể, hoạt động như một bộ đệm dữ liệu cho phép trao đổi linh hoạt giữa AE và CSE.
contentInstance	Biểu diễn giá trị của dữ liệu được lưu trong tài nguyên <container>
CSEBase	Tài nguyên gốc đại diện cho một CSE, nó chứa thông tin về CSE dưới dạng các thuộc tính và là cha của tất cả các tài nguyên khác.
remoteCSE	Lưu trữ thông tin liên quan đến M2M CSEs nằm trong các thiết bị khác sau khi đăng ký thành công. Tài nguyên này hỗ trợ việc điều hướng bản tin trong mạng.
subscription	Lưu trữ thông tin liên quan đến đăng ký cho một số tài nguyên. Nó cho phép người đăng ký nhận thông báo không đồng bộ khi có sự kiện xảy ra, chẳng hạn dữ liệu mới từ cảm biến, cập nhật hoặc xóa tài nguyên.

Cấu trúc cây tài nguyên

Cấu trúc cây tài nguyên của mỗi thực thể oneM2M đều được bắt đầu từ gốc là tài nguyên <*CSEBase*> có địa chỉ là một địa chỉ tuyệt đối (absoulute address). Dựa vào địa chỉ tuyệt đối này, tất các những tài nguyên con khác đều được đánh địa chỉ dựa trên <*CSEBase*>. Hình 2.6 mô tả một ví dụ về một cây tài nguyên của <*CSEBase*>, trong đó có chứa các thuộc tính và một số tài nguyên con, gồm có: <*accessControlPolicy*>, <*AE*>, <*remoteCSE*>, <*subcription*>.



Hình 2.6 Tài nguyên *<CSEBase>*

Thuộc tính của tài nguyên

Mỗi tài nguyên đều chứa các thuộc tính để lưu thông tin của chính nó. Các thuộc tính được chia ra làm 3 loại:

- Universal Attributes: xuất hiện ở tất cả các tài nguyên.
- Common Attributes: xuất hiện ở hầu hết các tài nguyên, có ý nghĩa giống nhau ở bất kì nơi nào nó xuất hiện.
- Resource-specific Attributes: là những thuộc tính đặc trưng chỉ xuất hiện tương ứng một số tài nguyên nhất định.

Các thuộc tính xuất hiện ở mọi tài nguyên được liệt kê trong bảng sau:

Bảng 2.2 Universal Attribute

Universal Attribute	Mô tả
resourceType	Loại tài nguyên
parentID	resoruceID của tài nguyên cha
creationTime	Thời gian tài nguyên được tạo
lastModifiedTime	Thời gian cuối cùng tài nguyên bị thay đổi
resourceID	ID của tài nguyên
resourceName	Tên tài nguyên

Biểu diễn tài nguyên

oneM2M định nghĩa ra các XML, JSON và CBOR schemas để biểu diễn các thuộc tính cho mỗi tài nguyên.

2.3.5 Bản tin oneM2M Primitive

Bản tin primitive là bản tin của lớp dịch vụ chung dùng để trao đổi thông qua các giao diện Mca, Mcc, Mcc'. Chúng luôn đi theo một cặp gồm bản tin request primitive và response primitive. Các bản tin này độc lập với tầng dưới. Nếu bên gửi và bên nhận cùng nằm trong một node, các bản tin được trao đổi trực tiếp qua các giao diện logic. Nếu bên gửi và bên nhận nằm ở hai thiết bị khác nhau, bản tin primitive sẽ được ánh xạ qua các giao thức phù hợp.

Cấu trúc Primitive

Một bản tin primitive gồm hai thành phần: phần điều khiển (control) và phần nội dung (content). Trong đó:

- Phần điều khiển: gồm các tham số cần thiết cho quá trình xử lý chính bản tin đó (ví dụ như phân loại, định dạng, địa chỉ đích, địa chỉ nguồn,…)
- Phần nội dung: có thể có hoặc không tùy thuộc vào loại bản tin. Nội dung của phần này chứa mô tả tài nguyên cùng các thuộc tính của nó.

Bản tin primitive được mã hóa và tuần tự hóa (serialize) tùy thuộc vào giao thức và kiểu biểu diễn dữ liệu được hộ trợ. Trong quá trình truyền, phần điều khiển được mã hóa dựa trên giao thức binding được sử dụng, còn phần nội dung được tuần tự hóa sử dụng XML, JSON hoặc CBOR. So sánh với các bản tin trong giao thức tầng ừng dụng, phần điều khiển có chức năng giống như header, phần nội dung giống với payload/body của bản tin. Ví dụ về phần nội dung của bản tin primitive có thể xem ở phụ lục 2.

2.3.5.1 Bản tin request primitive

Bản tin request primitive gồm các tham số bắt buộc và tùy chọn, chúng phụ thuộc vào các phương thức. Bảng 2.3 thể hiện một số tham số chính của bản tin request primitive, và sự phụ thuộc của chúng vào các phương thức C, R, U, D, N. "M" ý chỉ phần bắt buộc, "O" có thể có hoặc không, "N/A" không được trình bày.

Bảng 2.3 Các tham số của Request Message

Các tham số		Phương thức				
		Create	**Retrieve**	**Update**	**Delete**	**Notify**
Bắt buộc	***Operation***	M	M	M	M	M
	To	M	M	M	M	M
	From	O (*)	M	M	M	M
	Request Identifier	M	M	M	M	M
Tùy chọn	***Content***	M	O	M	N/A	M
	Resource Type	M	N/A	N/A	N/A	N/A
(*) Với bản tin tạo tài nguyên AE trường này có thể không tồn tại.						

Các tham số bắt buộc cần có:

***Operation*:** Phương thức được thực thi. Các phương thức bao gồm: Create (C), Retrieve (R), Update (U), Delete (D), Notify (N). Mỗi phương thức sẽ được nhận dạng và thực thi ở bên nhận.

To: Địa chỉ của tài nguyên hoặc thuộc tính đích. Tham số ***To*** có thể được biết trước nhờ quá trình khám phá (discovery). Tuy nhiên, tài nguyên <*CSEBase*> không được phép discovery nên khi sử dụng tham số này chúng ta giả sử <*CSEBase*> đã được cung cấp trước. Địa chỉ đích phụ thuộc vào tài nguyên và phương thức tác động tới tài nguyên. Ví dụ:

- tham số ***To*** ứng với phương thức CREATE đối với tài nguyên <*example*> là "*/m2m.provider.com/exampleBase*",
- tham số ***To*** ứng với phương thức RETRIEVE cũng đối với tài nguyên đó là "*/m2m.provider.com/exampleBase/example*".

Tương ứng với từng phương thức, tham số ***To*** thể hiện:

- **Create (C):** tham số ***To*** chứa địa chỉ của tài nguyên đích, tài nguyên này sẽ là tài nguyên cha của tài nguyên mới sau khi khởi tạo.
- **Retrieve (R):** tham số ***To*** chứa địa chỉ của tài nguyên có thể truy cập và trả về thông của chính nó cho bên gửi.
- **Update (U):** tham số ***To*** chứa địa chỉ của tài nguyên có thể cập nhật các thuộc tính mới lấy từ tham số ***Content***.
- **Delete (D):** tham số ***To*** chứa địa chỉ của tài nguyên mà tất cả các tài nguyên con của nó sẽ bị xóa.
- **Notify (N):** tham số ***To*** chứa địa chỉ của tài nguyên đích tại bên nhận.

From**:** Định danh bên gửi. Tham số này được dùng để bên nhận kiểm tra xem bên gửi có được phép truy cập tài nguyên hay không. Bên gửi có thể là admin, guest,…

Request Identifier**:** Định danh bản tin request.

Các tham số tùy chọn

Content**:** Tài nguyên được trao đổi. Tham số này phụ thuộc vào từng phương thức.

- **Create (C):** ***Content*** là nội dung của tài nguyên mới, tài nguyên này được phân loại dựa vào tham số ***ResourceType***.
- **Update (U):** ***Content*** chứa nội dung cần được cập nhật trong tài nguyên đã có. Ứng với mỗi thuộc tính của tài nguyên đích, ***Content*** gồm tên và giá trị của thuộc tính. Thuộc tính trong tài nguyên đích sẽ bị xóa nếu giá trị được cập nhật là NULL.
- **Notify (N):** ***Content*** chứa thông tin cần được thông báo.
- **Retrieve (R):** ***Content*** gồm danh sách tên và giá trị các thuộc tính của tài nguyên đích.

Resource Type: loại tài nguyên. Tham số này chỉ cần với phương thức CREATE. Các loại tài nguyên tiêu biểu trong oneM2M có thể kể đến là *<accessControlPolicy>, <cseBase>, <remoteCSE>*,… được miêu tả cụ thể trong [19].

2.3.5.2 Bản tin response primitive

Bản tin response primitive được bên nhận gửi trả về bên gửi tương ứng với mỗi bản tin resquest primitvie. Nó gồm các tham số bắt buộc và tùy chọn, chúng phụ thuộc vào các phương thức. Bảng 2.4 thể hiện một số tham số chính của bản tin Response primitive, và sự phụ thuộc của chúng vào các phương thức C, R, U, D, N. "M" ý chỉ phần bắt buộc, "O" có thể có hoặc không, "N/A" không được trình bày.

Bảng 2.4 Các tham số của Response Message

Các tham số	**Ack**	**Thành công**					**Thất bại**	
		C	**R**	**U**	**D**	**N**	**CRUD**	**N**
Response Status Code	M	M	M	M	M	M	M	M
Request Identifier	M	M	M	M	M	M	M	M
Content	O	O	M	O	O	O	O	O

Các tham số bắt buộc:

***Response Status Code*:** Tham số này chỉ ra kết quả sau khi bên nhận xử lý bản tin request, chúng có thể là: thành công, thất bại, ACK, trạng thái của quá trình xử lý,… Bản tin ACK (acknowledgement) chỉ ra rằng bên nhận đã chấp nhận bản tin nhưng chưa xử lý. Ví dụ, CSE nhận được request trực tiếp từ bên gửi nhưng request đó không được xử lý ngay lập tức. Trạng thái thành công hay thất bại sẽ được thông báo sau. Chi tiết về các code này được mô tả trong [20].

***Request Identifier*:** tham số này khớp với bản tin request.

Các tham số tùy chọn

Content*:** Chứa nội dung của tài nguyên, có thể được biểu diễn qua nhiều kiểu khác nhau. Nội dung này phụ thuộc vào ***Response Status Code.

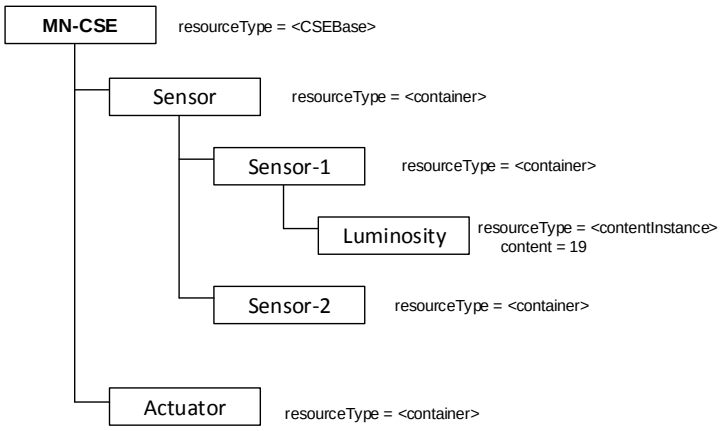
- Nếu ***Response Status Code*** là thành công: ***Content*** được trả về tương ứng với yêu cầu của bản tin Request.
- Nếu ***Response Status Code*** là thất bại: ***Content*** chứa thêm các thông tin mô tả lỗi
- Nếu ***Response Status Code*** là ACK: ***Content*** chứa địa chỉ của tài nguyên <*request*> trong bên gửi.

2.4 Tổ chức và truy cập dữ liệu trong oneM2M

Để lưu trữ dữ liệu, oneM2M dùng tài nguyên <*Container*>, cách truy cập để lấy dữ liệu có thể thông qua cơ chế đăng ký và thông báo (Subcription và Notification) hoặc lấy trực tiếp qua cơ chế khám phá (Discovery).

2.4.1 Container

Container được xem như những kho chứa, lưu trữ dữ liệu thu thập được trong tài nguyên <*container*>. Các dữ liệu này có thể được chia sẻ, theo dõi thông qua các thực thể khác nhau. Tài nguyên <*container*> chứa tài nguyên con và các thuộc tính của chính nó. Dữ liệu cảm biến thu thập được lưu trữ thông qua các tài nguyên con <*contentInstance*>. Để hỗ trợ việc biểu diễn cấu trúc dữ liệu theo thứ bậc, một tài nguyên <*container*> có thể có tài nguyên con chính là một tài nguyên <*container*> khác. Hình 2.7 minh họa một cây tài nguyên <*container*>.

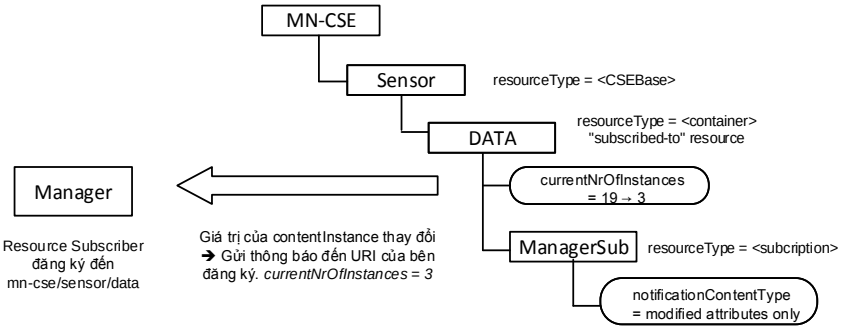


Hình 2.7 Ví dụ về cây tài nguyên

Giả sử Gateway (MN-CSE) được đăng ký bởi 3 thiết bị IoT, trong đó có 2 cảm biến và một thiết bị chấp hành. MN tổ chức dữ liệu thành các tài nguyên <*container*> có resource name tương ứng là Sensor và Actuator. Do trong ví dụ có hai cảm biến nên ta sẽ biểu diễn chúng tương ứng với các tài nguyên con là Sensor-1 và Sensor-2. Dữ liệu thu thập được từ các cảm biến được lưu trong tài nguyên <*contentInstance*>, chẳng hạn như giá trị cường độ ánh sáng được lưu trong thuộc tính content = "19".

2.4.2 Cơ chế đăng ký và thông báo

Các thao tác CRUD khi gửi tới một CSE sẽ làm thay đổi tài nguyên, để nhận được sự thay đổi đó chúng ta có thể dùng tài nguyên <*subcription*>. Tài nguyên <*subcription*> sẽ là tài nguyên con của tài nguyên được đăng ký ("subscribed-to" resource), nó lưu thông tin về đối tượng đi đăng ký (resource subscriber). Đối tượng đăng ký được quyền RETRIEVE và CREATE tài nguyên <*subscription*> trong tài nguyên được đăng ký. Các thuộc tính của tài nguyên <*subscription*> nhằm xác định các thông báo được gửi đi khi nào, tới đâu và bằng cách nào.



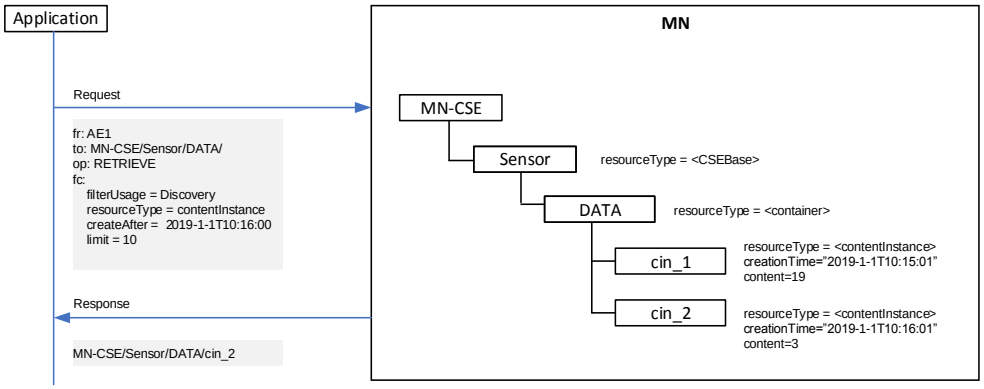
Hình 2.8 Ví dụ về cơ chế đăng ký và thông báo

Hình 2.19 mô tả kịch bản đơn giản của cơ chế đăng ký và thông báo. Mỗi khi có thay đổi về cường độ ánh sáng, Manager cần nhận được thông tin đó để phân tích và đưa ra lệnh điều khiển phù hợp tới các thiết bị chấp hành. Để làm được điều này, Manager sẽ tạo tài nguyên ManagerSub nằm trong tài nguyên đích là *MN-CSE/Sensor/DATA*, thuộc tính *notificationContentType* của tài nguyên này quy định chỉ khi giá trị thay đổi mới gửi thông báo. Giả sử giá trị cường độ sáng lưu trong contentInstance thay

đổi từ 19 xuống 3. CSE thực hiện gửi thông báo đến Manager theo URI được cung cấp trước đó với thông tin chứa giá trị sau khi đã thay đổi.

2.4.3 Cơ chế khám phá

Tài nguyên trong oneM2M còn có thể được lấy/khám phá thông qua phương thức RETRIEVE. Các thông tin khám phá này có thể được hạn chế bằng việc dùng tham số *filterCriteria* lọc Type, Lables, Content Size,…

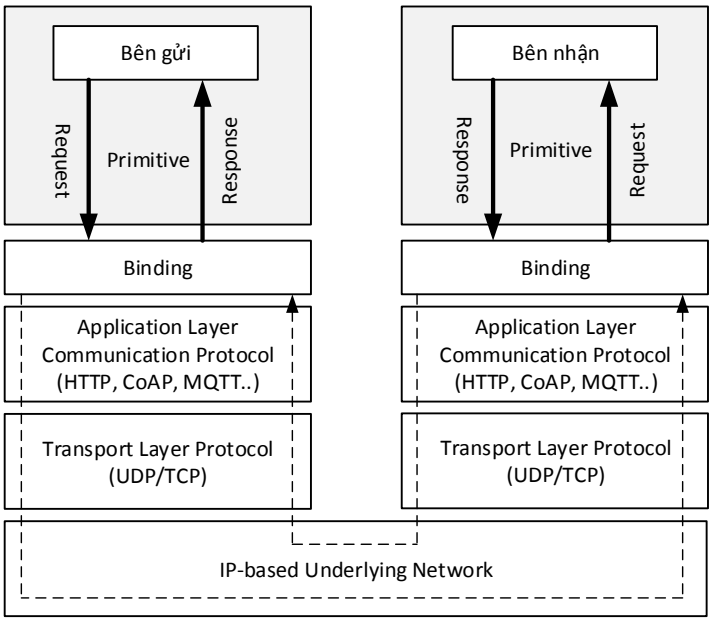


Hình 2.9 Ví dụ về quá trình khám phá

Hình 2.9 minh họa việc một ứng dụng muốn lấy thông tin dữ liệu về giá trị của cảm biến sau 10h:16':00s tại địa chỉ *MN-CSE/Sensor/DATA/,* số kết quả tối đa cần là 10 giá trị. Kết quả được trả về là danh sách địa chỉ của tài nguyên thỏa mãn yêu cầu.

2.5 oneM2M Binding

Khi bên gửi và bên nhận là hai thiết bị độc lập, có thể kết nối qua nền tảng mạng IP, oneM2M Binding thực hiện một chức năng giúp các thực thể oneM2M có thể trao đổi với nhau sử dụng các giao thức tầng ứng dụng phổ thông như HTTP. CoAP và MQTT. Tại đầu ra của Binding, bản tin có thể truyền xuống các tầng dưới theo các giao thức tầng giao vận như TCP, UDP. Mô tả về quá trình bày được thể hiện trong hình dưới đây.



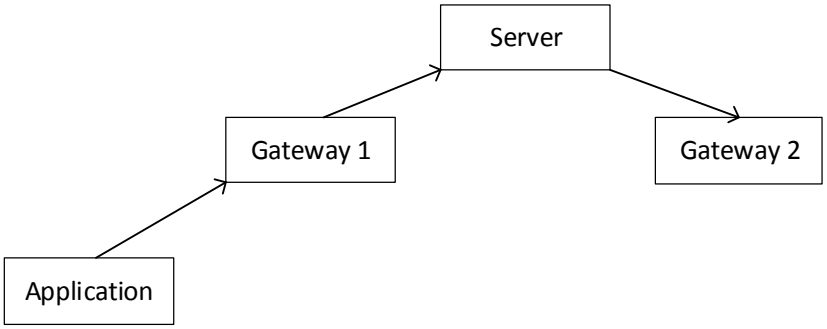
Hình 2.10 Kiến trúc oneM2M Binding

Có hai nhiệm vụ chính phải thực hiện khi sử dụng Binding là: cấu hình các thực thể và chuẩn hóa được cách ánh xạ bản tin oneM2M sao cho phù hợp với từng giao thức. Đối với các giao thức như HTTP, CoAP có thể chạy trên mô hình RESTful hoàn toàn tương thích với oneM2M, vấn đề cần quan tâm là ánh xạ được các tham số trong bản tin primitive vào phần header của từng bản tin của các giao thức. MQTT sử dụng kiểu trao đổi Publish/Subcribe nên khi cấu hình chúng ta cần quan tâm vị trí đặt Broker.

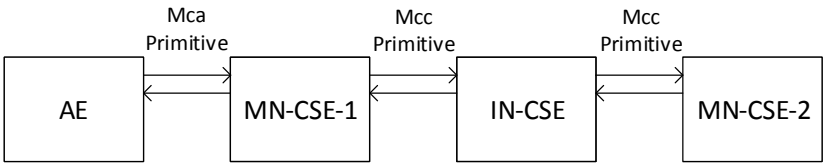
2.5.1 HTTP Binding

Cấu hình Binding

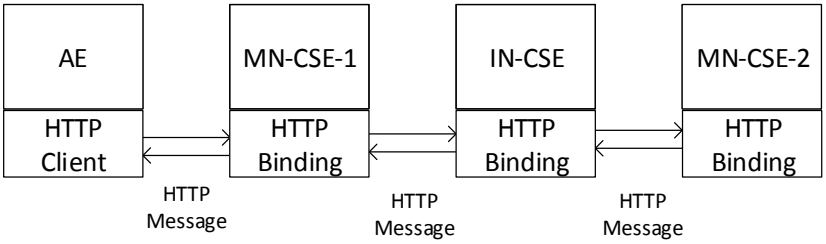
Hình 2.11 dưới đây trình bày một ví dụ về việc thiết lập hệ thống có sử dụng HTTP Binding. Giả sử một ứng dụng thuộc quản lý của Gateway 1 muốn gửi bản tin đến Gateway 2, đường đi của bản tin được mô tả trong Hình 2.11a). Các thực thể ứng dụng, gateway, server tương ứng với các thực thể AE, MN-CSE, IN-CSE. Trong đó, AE đăng ký với với MN-CSE-1, MN-CSE-1 và MN-CSE-2 đăng ký với IN-CSE.



a) Kịch bản cấu hình



b) Cấu hình oneM2M



c) Cấu hình oneM2M sử dụng HTTP Binding

Hình 2.11 Cấu hình HTTP Binding

Hình 2.11b) thể hiện quá trình trao đổi bản tin giữa các thực thể oneM2M. Ban đầu, bản tin được khởi tạo và xuất phát từ AE tới MN-CSE-1, do đây chưa phải đích đến của bản tin nên nó sẽ được chuyển tiếp lên IN-CSE rồi tới MN-CSE-2. Các bản tin được trao đổi qua giao diện Mca, Mcc thuộc loại bản tin request/response primitive. Khi triển khai HTTP binding, các bản tin primive sẽ được ánh xạ với HTTP messager, các thực thể oneM2M sẽ thực hiện vai trò tương đương với HTTP client hoặc HTTP server. Trong kịch bản này, HTTP client tương ứng với AE là nơi bắt đầu gửi bản tin vào mạng, HTTP server đại diện cho các thực thể nhận và chuyển tiếp bản tin bao gồm MN-CSE-1, IN-CSE và MN-CSE-2. Trên thực tế, với nhiều kịch bản khác, khi

AE cần nhận bản tin từ hệ thống oneM2M hoặc các CSE muốn gửi bản tin thông báo, điều khiển, các thực thể sẽ đóng vai trò của cả server và client.

Ánh xạ bản tin

Thông tin trong tham số content của bản tin primitive chứa nội dung của tài nguyên sẽ được ánh xạ với trường HTTP message body. Tất cả những tham số khác (xác định nơi gửi-from, địa chỉ tài nguyên đích-to, phương thức-operation,…) sẽ nằm trong phần HTTP header. Chi tiết về quá trình ánh xạ này được trình bày trong phụ lục 3.

2.5.2 CoAP Binding

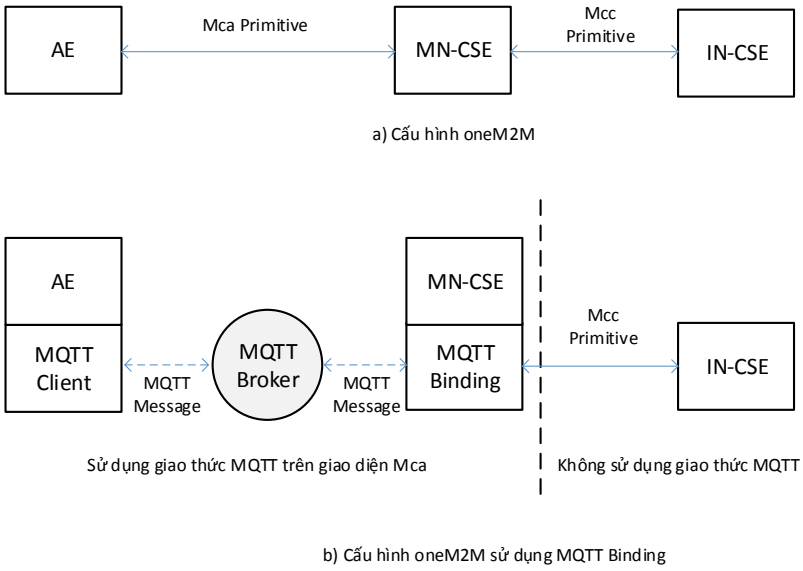
Do mô hình kiến trúc giữa CoAP và HTTP rất giống nhau nên việc ánh xạ các thực thể oneM2M sang CoAP tương tự như HTTP. Mỗi thực thể oneM2M có thể tương ứng với một hoặc cả hai thực thể CoAP client và CoAP server. Ngoài ra, quá trình ánh xạ bản tin được trình bày trong phụ lục 3.

2.5.3 MQTT Binding

Cấu hình Binding

Có rất nhiều cách để triển khai MQTT trong oneM2M, điểm khác biệt cơ bản để phân biệt chúng là MQTT Broker/Server được tích hợp bên trong hay đứng độc lập phía ngoài các thực thể oneM2M. Hình 2.12a) mô tả một mô hình oneM2M cơ bản gồm ba thành phần AE (ứng dụng), MN (gateway), IN (server). Trong đó AE đăng ký với MN-CSE, MN-CSE đăng ký với IN-CSE. Các bản tin primitive được trao đổi qua các giao diện Mca, Mcc.

Cấu hình triển khai oneM2M sử dụng MQTT Binding trong Hình 2.12b đảm bảo kết nối giữa AE với MN-CSE sử dụng giao thức MQTT, kết nối giữa MN-CSE và IN-CSE vẫn sử dụng giao thức oneM2M.



Hình 2.12 Cấu hình MQTT Binding

Trong đó, AE và MN đều đóng vai trò làm các MQTT client, chúng giao tiếp với nhau thông qua MQTT Broker. Broker này được cài đặt độc lập với các thực thể oneM2M, nó đóng vai trò trung gian giữa các MQTT client. Giả sử AE muốn gửi bản tin đến MN-CSE, đầu tiên nó sẽ gửi bản tin MQTT đến một kênh (topic) trong MQTT Broker, sau đó bản tin được điều phối tới MQTT Binding trong MN-CSE.

Do oneM2M sử dụng mô hình request/response để tương tác giữa các thực thể, trong khi đó giao thức MQTT sử dụng mô hình publish/subcribe, chúng ta cần tạo ra các kênh phù hợp để hợp nhất hai mô hình này. Kênh request là nơi các client có thể gửi bản tin vào, sao đó các bản tin response sẽ được đưa đến một kênh response định trước mà client đã đăng ký.

Kênh request có cấu trúc:

/oneM2M/req/<originator>/<target-id>/<serialization-format>

Kênh response có cấu trúc:

/oneM2M/resp/<target-id>/<originator>/<serialization-format>

Trong đó:

- <target-id> là CSE-ID đích
- <originator> tương ứng với trường X-M2M-Origin xác định bên gửi.
- <serialization-format> được xác định bởi định dạng bản tin sử dụng (là xml hoặc json).

Tất cả những thông tin liên quan đến xác thực, phương thức,… đều được lưu trong payload bản tin. Phần header của bản tin MQTT không được ánh xạ với bất kì tham số nào của bản tin primitive. Chi tiết về nội dung này được đề cập trong phục lục 3.

2.6 Các dự án triển khai oneM2M

Một số nền tảng và dự án nguồn mở độc lập đã sử dụng tiêu chuẩn oneM2M trong các ứng dụng và dịch vụ của mình. Các dự án này là nơi hữu ích giúp các nhà phát triển bắt đầu làm quen với oneM2M.

ATIS Open Source Internet of Things (OS-IoT) bao gồm các thư viện nguồn mở đơn giản hóa quy trình phát triển thiết bị IoT, đặc biệt cho người dùng muốn kết nối với hệ sinh thái oneM2M.

IOTDM, một phần của dự án OpenDaylight của Linux Foundation: Phát triển một oneM2M-based IoT Data Broker cho phép các ứng dụng được cấp quyền truy xuất dữ liệu IoT được tải lên từ bất kì thiết bị nào.

OASIS SI, một phần của dự án Open-source Architecture Semantic IoT Service-platform: Phát triển mã nguồn cho hệ thống oneM2M dựa trên nền tảng máy chủ IoT. Nó bảo gồm kết hợp giao thức, trình điều khiển và quản lý tài nguyên cùng các lớp cơ sở dữ liệu linh hoạt.

OCEAN, mã nguồn triển khai cho oneM2M sever/gateway/device và application đều được hỗ trợ. Đồng thời dự án cung cấp các công cụ phát triển bao gồm nền tảng tài nguyên web, công cụ tự kiểm tra. oneM2M được triển khai trên các

phần cứng như Raspberry Pi, Arduino giúp dễ dàng phát triển sản phẩm oneM2M. Dự án còn cung cấp công cụ Mobius như một máy chủ oneM2M.

OM2M, phát triển bởi the Eclipse Foundation và là một trong những dự án của Eclipse's IoT Working Group: Cung cấp một nền tảng oneM2M linh hoạt giúp triển khai M2M theo chiều ngang với server, gateway, và các device. Nó có cấu trúc modular và chạy trên nền OSGi, một nền tảng có khả năng mở rộng tốt thông qua việc phát triển các plug-in.

OpenMTC là sự tích hợp của middleware dựa trên tiêu chuẩn oneM2M, giúp nghiên cứu và phát triển các ứng dụng M2M cũng như IoT. OpenMTC tiếp cận dịch vụ theo chiều ngang giúp dễ dàng tích hợp các thiết bị từ nhiều ngành công nghiệp IoT khác nhau, độc lập với cơ sở hạ tầng phần cứng.

Trong những dự án kể trên, OM2M có thể được xem ở *"mở"* nhất khi chỉ sử mã nguồn mở Java chạy trên Eclipse, không yêu cầu thêm bất kì công cụ hay phần mềm chuyên biệt nào đi kèm. Ngoài ra OM2M có thể chạy trên nhiều hệ điều hành, dễ dàng phát triển tính năng mới thông qua cấu trúc OSGi.

2.7 Kết luận chương

Chương hai hệ thống các kiến thức lý thuyết thiết yếu cần có để triển khai được hệ thống IoT theo tiêu chuẩn oneM2M. Tiêu chuẩn oneM2M có kiến trúc ba tầng cơ bản đại diện cho các kết nối, các dịch vụ và các ứng dụng. Trong đó thực thể dịch vụ dùng chung (CSE) được xem như trái tim của cả hệ thống, nó cung cấp các chức năng đảm bảo kết nối các phần tử trong mạng. Hơn nữa, CSE dễ dàng mở rộng và phát triển các tính năng mới, Binding đảm bảo kết nối qua các giao thức tầng ứng dụng, IPE giúp kết nối đến các thiết bị sử dụng nhiều công nghệ khác nhau.

oneM2M được xây dựng dựa trên kiến trúc REST, khi mọi thành phần đều được biểu diễn dưới dạng các tài nguyên. oneM2M protocol định nghĩa các phương

thức tương tác với tài nguyên thông qua bản tin primitive. Quá trình khai phá, giám sát thông tin cũng được hỗ trợ đầy đủ thông qua các cơ chế đăng ký và thông báo.

Tất cả các tiêu chuẩn của oneM2M chỉ dừng lại trên giấy ở mức lý thuyết, để có thể triển khai một mô hình thật, chúng ta cần đến những dự án liên quan, trong chương tiếp theo, OM2M, một dự án như vậy được nghiên cứu và phát triển.

CHƯƠNG 3. XÂY DỰNG MẠNG ĐA CÔNG NGHỆ TRÊN NỀN TẢNG OM2M THEO TIÊU CHUẨN ONEM2M

Chương ba đi tìm hiểu về OM2M, một dự án được triển khai theo tiêu chuẩn oneM2M. Từ đó, một hệ thống IoT cơ bản có khả năng kết nối đa công nghệ được đề xuất và xây dựng thử nghiệm. Khảo sát hệ thống với các kịch bản đơn giản, chúng ta có thể đánh giá hoạt động và hiểu sâu sắc hơn về oneM2M.

3.1 Dự án OM2M

3.1.1 Sự ra đời

Dự án Eclipse OM2M [24], một chương trình mã nguồn mở dựa trên chuẩn oneM2M và SmartM2M được phát triển bởi LASS-CNRS. Nó cung cấp một nền tảng dịch vụ M2M ngang hàng để phát triển nhiều dịch vụ độc lập dựa trên một lớp mạng cơ bản, hướng đến việc hợp nhất được các ứng dụng theo chiều dọc hệ thống và các thiết bị không đồng nhất. Một số thông tin cơ bản của dự án:

- Nhà phát triển Eclipse Technologies
- Lần đầu ra mắt 20/04/2014
- Phiên bản mới nhất 13/05/2018 (version 1.3.0)
- Được viết trên ngôn ngữ Java
- Phát triển cho hệ điều hành: Windows, Linux, Mac OS X
- Website: http://www.eclipse.org/om2m/

3.1.2 Các đặc điểm và tính năng

Platform tiêu chuẩn

OM2M là một nền tảng mở, tuân theo tiêu chuẩn oneM2M và SmartM2M. Nó cho phép người lập trình tạo ra một thực thể dịch vụ dùng chung (CSE) có thể chạy trong server, gateway, thậm chí ngay trên các thiết bị IoT.

RESTful API

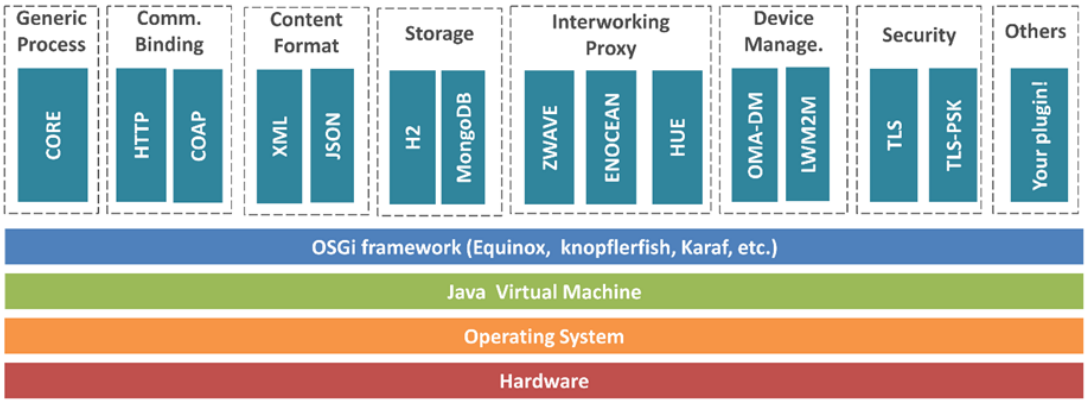
OM2M được xây dựng tuân theo kiến trúc RESTful API. Các API được xây dựng tuân theo nguyên tắc ở mục 2.3.2 và các tài nguyên được xây dựng theo mô tả trong bảng Bảng PL.1.

Modularity & Extensibility

Do được xây dựng trên nền tảng OSGi, OM2M có khả năng mở rộng thông qua việc phát triển các plugin, OSGi container đa dạng như: Equinox, Knopflerfish,…

3.1.3 Kiến trúc OM2M

Ta có thể thấy OM2M là một chương trình Java xây dựng theo kiến trúc OSGi, hướng tới việc phát triển một CSE linh hoạt có thể triển khai bên trong server, gateway, và các thiết bị IoT.

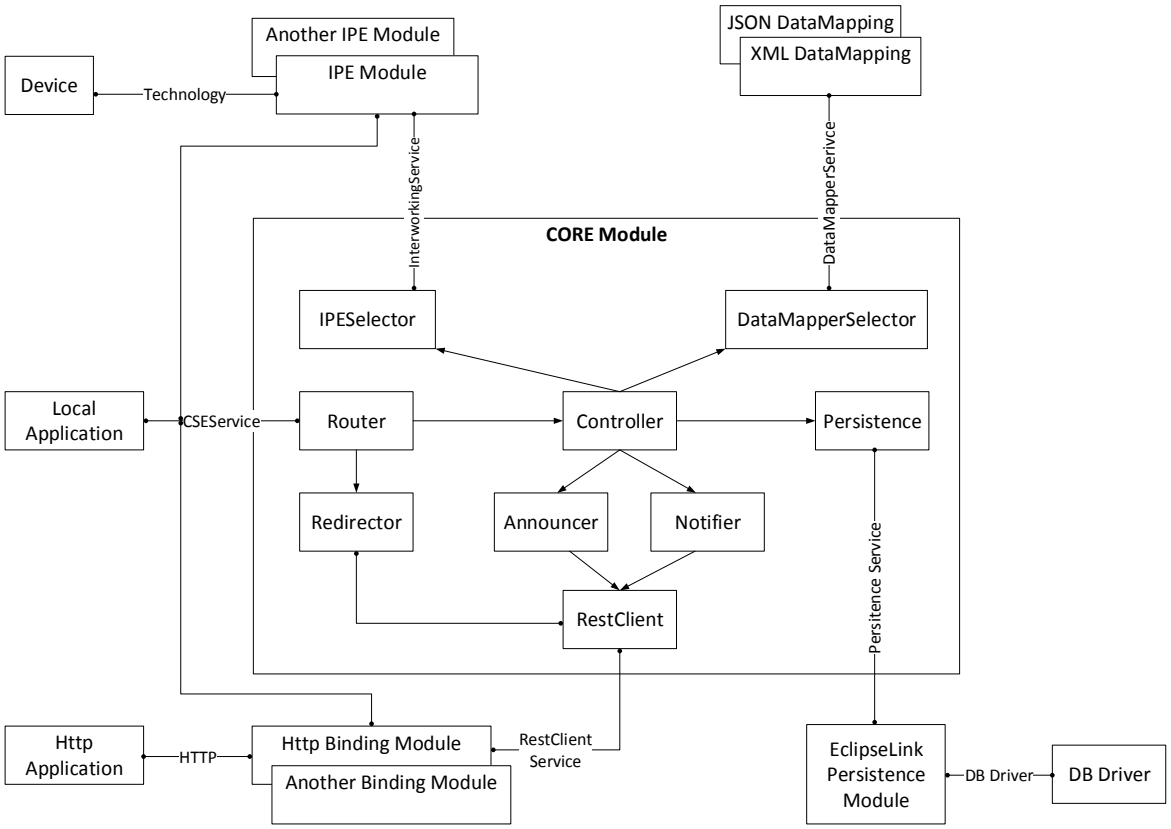


Hình 3.1 Kiến trúc OM2M

Chức năng của CSE được chia nhỏ và được thực hiện thông qua các module (còn được gọi là plugin/bundle). Mỗi module cung cấp một chức năng cụ thể và có thể cài đặt, tắt, bật linh hoạt không cần khởi động lại hệ thống. **CORE** là module chính được triển khai tại mỗi CSE, nó cung cấp những dịch vụ cơ bản giúp các thực thể trong

mạng oneM2M có thể giao tiếp với nhau. Các đặc tính mở rộng của oneM2M đã giới thiệu trong chương hai có thể thực hiện thông qua việc phát triển các module tương ứng. Trong Hình 3.1, ta có thể thấy một số khối module thực hiện các chức năng binding (tích hợp giao thức tầng ứng dụng), định dạng content (theo kiểu XML, JSON,…), kết nối với cơ sở dữ liệu (H2, MongoDB), Interwoking Proxy (giúp các thiết bị IoT sử dụng những công nghệ truyền dẫn khác nhau kết nối liền mạch với hệ thống), Device Manager (thực hiện quả lý thiết bị, hỗ trợ cập nhật firmware),…

3.1.3.1 CORE Module

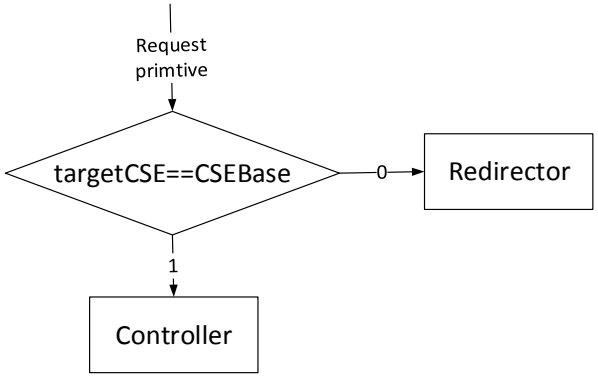


Hình 3.2 Kiến trúc CORE module

Mỗi CSE chứa duy nhất một CORE module, nó cung cấp giao diện *CSEService* để xử lý các bản tin primitive. Các bản tin này có thể được gửi từ những ứng dụng nội bộ (Local Application) hoặc là đầu ra của các Binding Module và IPE Module. CORE nhận bản tin request từ nguồn nào thì trả về bản tin response ứng với nguồn đó thông

qua các giao diện *RestClientService* và *IPEService*. Ngoài ra, để hỗ trợ định dạng bản tin và kết nối với cơ sở dữ liệu, CORE cung cấp thêm các giao diện *DataMapper Service* và *PesistenceService.* CORE module chứa các thành phần:

Router*:* Mọi bản tin khi vào CORE đều được xử lý ban đầu tại khối **Router**. Tại đây, hai tham số trong bản tin request được phân tích là *"To"* và *"ResourceType"*. Khi phát hiện đích đến của bản tin không phải CSE hiện tại, nó sẽ chuyển đến bộ **Redirector**. Trong trường hợp còn lại, **Router** chuyển bản tin đến đúng với khối **Controller** của loại tài nguyên đó. Hình 3.3 mô tả một cách đơn giản hoạt động của khối Router.



Hình 3.3 Lưu đồ hoạt động của khối Router

Redirector đóng vai trò định tuyến trong mạng, khối này phân tích CSE đích, tìm kiếm trong các tài nguyên <*remoteCSE*> của CSE hiện tại rồi sửa tham số *"To"* để chuyển tiếp bản tin đi đúng đường.

Controller thực các hiện phương thức CRUD (bao gồm Create, Retrieve, Update, and Delete) cho mỗi loại tài nguyên. Dựa vào tham số *"ResourceType"* mà **Controller** có thể gọi đúng hàm xử lý loại tài nguyên yêu cầu. Một số thành phần điển hình trong khối **Controller** là AEController, ContainerController, RemoteCSEController.

Các tài nguyên được thể hiện trong code dưới dạng các Object, khối **DatamapperSelector** cung cấp giao diện *DataMapperService* giúp tài nguyên có thể

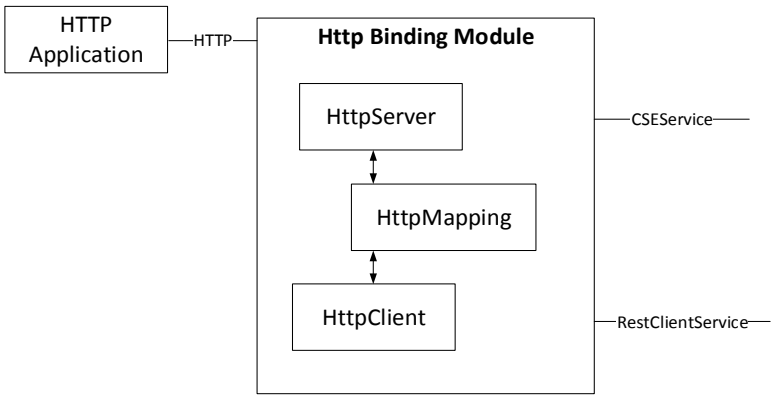
chuyển đổi qua các dạng biểu diễn khác như XML, JSON,… Đồng thời, các tài nguyên này sẽ được lưu trữ trong cơ sở dữ liệu bằng cách sử dụng khối **Persistence**.

Sau khi đã xử lý các phương thức CRUD được yêu cầu từ bản tin request, CORE sẽ thông báo đến các thực thể khác. Khối **Notifier** thực hiện việc gửi thông báo tới tất cả các thực thể đã đăng ký một tài nguyên xác định khi tài nguyên đó có sự thay đổi. Khối **Announcer** thực hiện gửi các tài nguyên tới remoteCSE.

Các bản tin request sau khi được xử lý cần được gửi trả về đúng trên giao diện đầu vào. Các khối **RestClient** và **IPESelector** chịu trách nhiệm đóng gói và chuyển đổi bản tin response primitive thành các dạng phù hợp rồi gửi ra ngoài.

3.1.3.2 Binding Module

Binding Module giúp oneM2M có thể kết nối thông qua nhiều giao thức tầng ứng dụng khác nhau. Mô tả lý thuyết về cơ chế hoạt động và cấu hình của khối đã đề cập trong mục 2.5. Hình 3.4 minh họa chi tiết cấu trúc của Http Binding Moulde được triển khai trong OM2M.



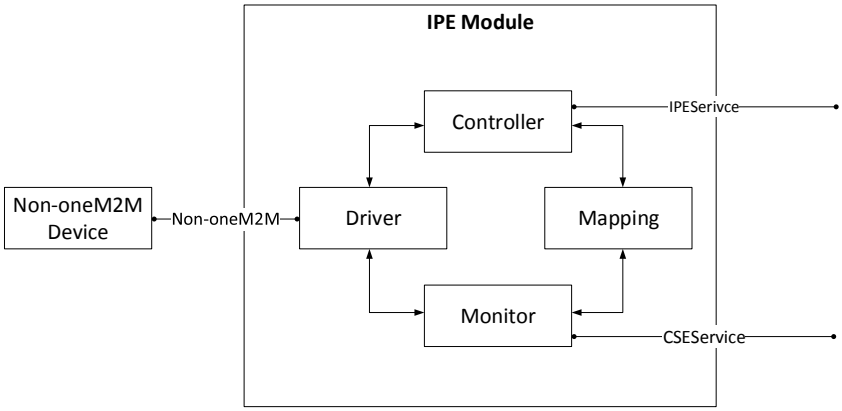
Hình 3.4 Kiến trúc HTTP Binding Module

Các khối HttpClient và HttpServer được xem là Client và Server khi giao tiếp với một HTPPApplication bên ngoài hệ thống. Ví dụ HTTPApplication gửi một HTTP request vào hệ thống, nó sẽ được nhận tại HttpServer, thực hiện chuyển đổi sang request primitive thông qua khối HttpMapping rồi gửi vào CORE bằng giao diện

CSEService. Sau khi CORE xử lý xong gửi về một response primitive thông qua giao diện *RestClientService* về HttpServer, thực hiện chuyển đổi sang HTTP response rồi gửi trả về cho HTTPApplication. Tương tự, khối HTTPClient được sử dụng khi CORE muốn gửi một yêu cầu tới HTTP Application.

3.1.3.3 IPE Module

IPE Module giúp oneM2M có thể kết nối với một thiết bị IoT qua một công nghệ truyền thông xác định. Mô tả lý thuyết về cơ chế hoạt động và cấu hình của khối đã đề cập trong mục 2.2.2, Hình 3.5 minh họa việc triển khai IPE Module trong OM2M.



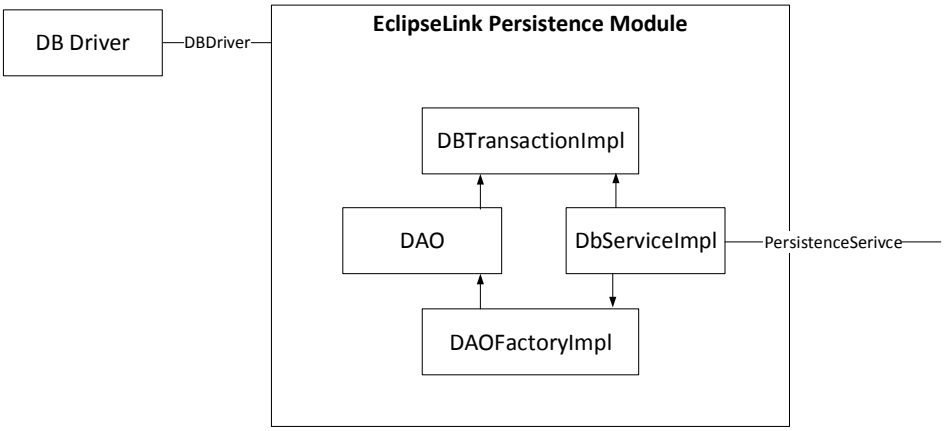
Hình 3.5 Kiến trúc IPE Module

Để có thể giao tiếp với các thiết bị none-oneM2M, trong IPE Module cần có khối Driver thực hiện mọi trao đổi giữa thiết bị với oneM2M. Các bản tin từ thiết bị gửi vào mạng lõi sẽ được chuyển đến khối Monitor, thực hiện chuyển đổi sang request primitve thông qua khối Mapping rồi gửi vào CORE qua giao diện *CSESerivce*. Những bản tin primitive nhận được từ CORE tại khối Controller, thực hiện chuyển đổi sang dạng bản tin phù hợp với thiết bị thông qua khối Mapping, rồi gửi về thiết bị qua giao diện *non-oneM2M*.

Do các bản tin dùng để trao đổi với thiết bị thường rất đơn giản, không chứa các nội dung mô tả tài nguyên, do vậy khối Monitor sẽ thực hiện tất cả các chức năng tương tự một AE để khởi tạo tài nguyên <*AE*>, các kho chứa dữ liệu <*Container*> và

<*ContentInstance*>. Đồng thời, khối Controller thực hiện chuyển đổi các bản tin phức tạp từ CORE về những lệnh điều khiển đơn giản phù hợp với thiết bị.

3.1.3.4 EclipseLink Persistence Module



Hình 3.6 Kiến trúc EclipseLink Persistence Module

DAO cung cấp một giao diện trừu tượng (abstract interface) nhằm đóng gói tất cả các truy cập, giúp tài nguyên lưu trữ được liên tục và không làm lộ chi tiết về cơ sở dữ liệu. Sử dụng DAO ta có thể phân tách logic lưu trữ dữ liệu trong một lớp riêng biệt. Theo cách này, các dịch vụ được che dấu về cách các hoạt động cấp thấp truy cập cơ sở dữ liệu được thực hiện.

DAO là một giao diện định nghĩa các phương thức trừu tượng phụ vụ việc truy cập tài nguyên (gồm create, find, update và delete). Mỗi loại tài nguyên được phân loại và ánh xạ tương ứng với DAO Object dựa vào lớp DAOFactoryImp (ví dụ với tài nguyên <*AE*>, AeDAO được gọi từ hàm getAeDAO()).

Đường kết nối tới oneM2M nhằm mục đích khai phá tài nguyên được xử lý trong lớp DBServiceImpl. Việc liên kết với cơ sở dữ liệu được thực hiện thông qua lớp DBTransactionImpl (các phương thức bao gồm open, commit, close, clear, lock và unlock).

DB Driver tương ứng với từng loại cơ sở dữ liệu khác nhau. Hiện tại OM2M đang sử dụng H2 Database. Ưu điểm của cơ sở dữ liệu này là có thể lưu trực tiếp

trong bộ nhớ, như vậy mỗi CSE có thể lưu trữ dữ liệu của nó tại ngay trên thiết bị phần cứng chứa nó.

3.1.4 Mã nguồn OM2M

Mã nguồn OM2M bao gồm các thành phần chính sau:

org.eclipse.om2m.core: Module xử lý và điều phối chính của chương trình.

org.eclipse.om2m.binding.*: Các plugin giúp tích hợp nhiều giao thức vào trong OM2M, mặc định mã nguồn đã tích hợp HTTP, CoAP và MQTT.

org.eclipse.om2m.binding.site.*: Các thư mục được xây dựng tương ứng với phần tử IN, MN, ASN.

org.eclipse.om2m.client.*: Gồm các mã nguồn mẫu trên nhiều ngôn ngữ lập trình khác nhau để mô tả thực thể AE, hiện tại đã có Java, Js và Arduino.

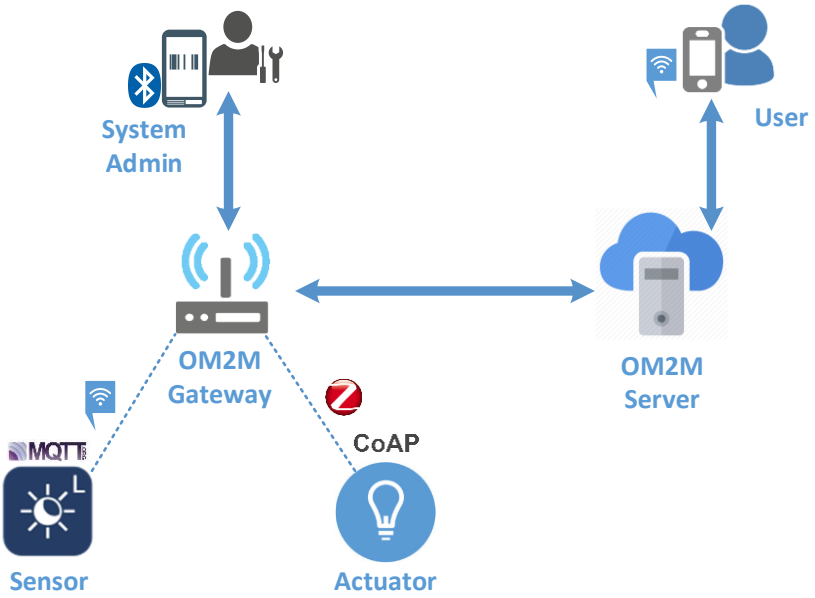
org.eclipse.om2m.comm.*: Xử lý các dữ liệu cơ bản được mô tả trong cấu trúc tài nguyên, xác nhận những yêu cầu REST từ CSEservice.

org.eclipse.om2m.webapp.resourcesbrowser: Xây dựng trình quản lý ứng dụng tham gia vào mạng oneM2M qua trình duyệt.

org.eclipse.om2m.ipe.*: Module đại diện cho các IPE nhằm liên kết nhiều công nghệ truyền dẫn vào trong OM2M.

3.2 Kiến trúc testbed

Tận dụng những thiết bị phần cứng hiện có, kiến trúc hệ thống đề xuất được mô tả như hình dưới đây.



Hình 3.7 Kiến trúc hệ thống triển khai thử nghiệm

Ở miền infrastructure, OM2M Server (IN) được cài đặt trên laptop có kết nối với mạng internet. IN có thể được truy cập từ phía người dùng thông qua một ứng dụng chạy trên smartphone. Ở miền domain, các thiết bị bao gồm:

- OM2M Gateway (MN node) đóng vai trò như một thiết bị tập trung đa công nghệ. Hiện tại, MN được cài đặt trên laptop, sử dụng card mạng có sẵn trong máy cho các kết nối Wi-Fi và Bluetooth. Để kết nối với các thiết bị sử dụng Zigbee, chúng ta sử dụng thêm kit Z1 Zolertia đóng vai trò làm border-router.
- Các thiết bị IoT được sử dụng gồm ESP8266 NodeMCU đóng vai trò như node cảm biến kết nối với MN qua Wi-Fi, Z1 Zolertia làm node chấp hành kết nối với MN qua Zigbee. Ngoài ra, android smartphone có thể kết nối với MN qua bluetooth nhằm mục đích giám sát, bảo trì.

Để xây dựng và tích hợp được các thiết bị vào nền tảng OM2M, chúng ta lần lượt thực hiện việc tích hợp các công nghệ, giao thức:

- Tích hợp thiết bị cảm biến IoT tương thích với OM2M trong mục 3.3.
- Tích hợp các node mạng Zigbee với OM2M trong mục 3.4.
- Thử nghiệm tích hợp thiết bị Bluetooth theo cách sử dụng IPE trong mục 3.5.

3.3 Tích hợp cảm biến IoT

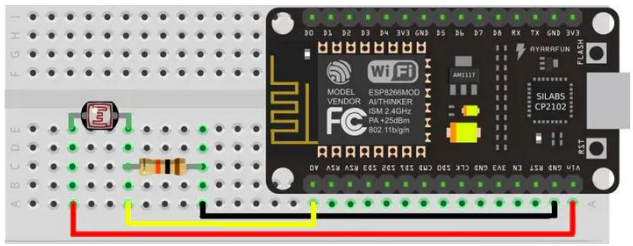
Phần này trình bày cách thiết kế một thiết bị cảm biến Wi-Fi đo cường độ ánh sáng sử dụng ESP8266 NodeMCU, nó có thể gửi dữ liệu trực tiếp vào OM2M Gateway qua giao thức MQTT.

3.3.1 Cảm biến IoT đo cường độ ánh sáng

Linh kiện

Các linh kiện được mô tả trong Hình 3.8 bao gồm:

- ESP8266 nodeMCU, là kit được phát triển dựa trên nền chip ESP8266 với thiết kế dễ sử dụng, có thể sử dụng trực tiếp trình biên dịch của Arduino để lập trình. Ngoài ra, thư viện cho ESP8266 dùng để kết nối WiFi có hỗ trợ TCP, UDP và các ứng dụng HTTP, MQTT, mDNS, SSDP, DNS Servers.
- Quang trở CDS 5mm: Có điện trở thay đổi theo cường độ ánh sáng (giá trị điện trở khoảng 1 – 2 kΩ khi có ánh sáng mạnh; khoảng 1,2 MΩ khi ánh sáng rất yếu).
- Điện trở: Có thể chọn giá trị tùy ý nhưng phải phù hợp để có thể kết hợp với quang trở, chọn 1 kΩ.



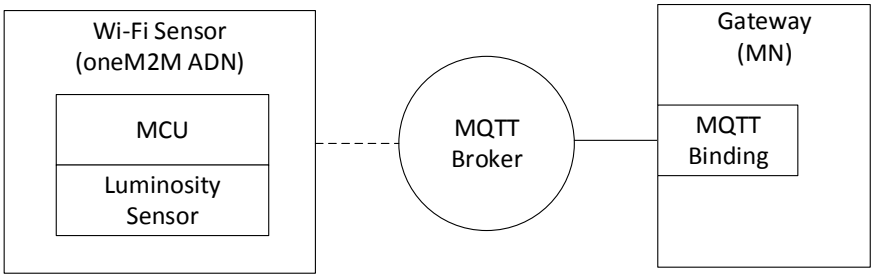
Hình 3.8 Sơ đồ mạch đo cường độ ánh sáng

Nguyên lý

Quang trở được mắc nối tiếp với điện trở. Điện trở được nối với chân A0 của ESP8266, chân còn lại được nối với chân GND. Chân còn lại của quang trở được nối với chân Vin của ESP8266.

Sử dụng hàm analogRead(A0) để đọc giá trị điện áp đo trên điện trở. Hàm này trả về giá trị trong khoảng từ 0 đến 1023 tương ứng với 0V đến 1V trên điện trở (do lấy nguồn từ chân Vin có điện áp 1V). Khi cường độ ánh sáng mạnh, trở trong quang trở sẽ giảm còn khoảng vài kΩ, do quang trở và điện trở mắc nối tiếp nên điện áp sẽ rơi đều trên điện trở (1 kΩ) dẫn đến giá trị trả về của hàm analogRead sẽ cao. Ngược lại, khi cường độ ánh sáng thấp thì trở trên quang trở sẽ cỡ MΩ và điện áp rơi chủ yếu lên quảng trở dẫn đến điện áp trên điện trở gần bằng 0, dẫn đến hàm trả về giá trị xấp xỉ bằng 0.

3.3.2 Cấu hình tích hợp cảm biến IoT sử dụng giao thức MQTT



Hình 3.9 Cấu hình tích hợp Wi-Fi sensor

Để thử nghiệm MQTT binding, chúng ta triển khai cấu hình như mô tả trong phần 2.5.3, bao gồm:

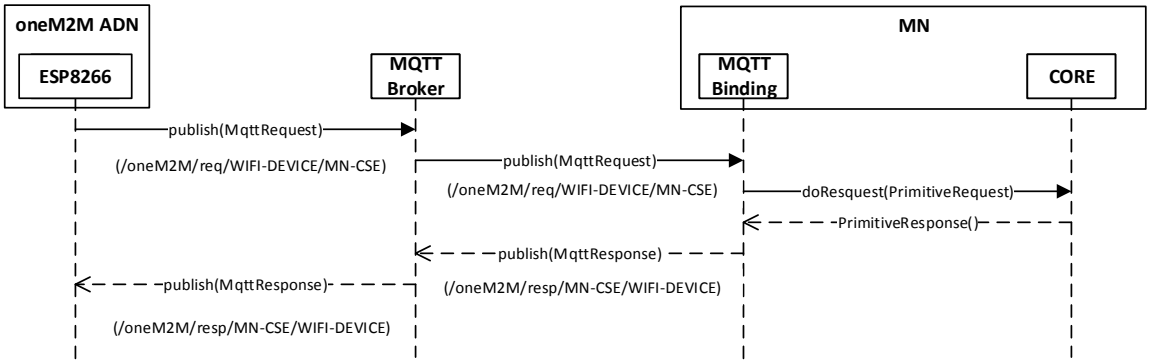
- oneM2M ADN chạy trên NodeMCU đã kết nối với cảm biến ánh sáng.
- MQTT Broker sử dụng Mosquitto broker nằm trong mạng Wi-Fi chung đang kết nối các thiết bị, sử dụng port 1883.
- MN node được cài đặt và bật MQTT binding.

3.3.3 Lập trình oneM2M ADN

oneM2M ADN sẽ gửi những bản tin cần thiết để tạo tài nguyên giúp định danh, mô tả chính nó và lưu trữ dữ liệu cảm biến trong MN-gateway. Muốn kết hợp được cách trao đổi thông tin kiểu request/response và client/server. MQTT binding sau khi được khởi động sẽ tạo ra hai kênh:

- Kênh request: */oneM2M/req/WIFI_DEVICE/mn-cse/json*
- Kênh response: */oneM2M/resp/mn-cse/WIFI_DEVICE/json*

oneM2M ADN sẽ gửi dữ liệu vào kênh request và đăng ký kênh response để nhận các bản tin gửi về từ MN. Tương ứng, MN sẽ đăng ký kênh request để nhận các yêu cầu từ oneM2M ADN và gửi bản tin trả về vào kênh response. Bản tin được trao đổi giữa các khối được thể hiện trong hình:



Hình 3.10 Lưu đồ trao đổi bản tin giữa ESP8266 và MN

Các bản tin MqttRquest được ESP8266 gửi vào kênh request sẽ được MQTT Broker chuyển tiếp đến các subcriber tương ứng (ở đây là MQTT binding). Sau đó, MQTT binding thực hiện ánh xạ bản tin MqttRequest thành PrimitiveRequest và đưa vào lõi. CORE xử lý bản tin và gửi trả về PrimitveResponse và được MQTT binding gửi vào kênh response, MQTT Broker chuyển đến ESP8266.

Như vậy, việc lập trình cho cảm biến Wi-Fi đưa về việc lập trình một MQTT Client có thể kết nối với MQTT Broker. Thư viện *PubSubClient.h* giúp quá trình này trở nên dễ dàng thông qua việc gọi hàm publish và subcribe.

```
WiFiClient espClinet; // declare wifi connection
PubSubClient client(espClinet);  // declare mqtt client
      client.setServer(mqtt_server,1883); // set mqtt broker and its ports
      client.connect(clientID.c_str()); // set client's ID
      client.subscribe(resp_topic); // subcrube response topic
```

```
      client.publish("/oneM2M/req/WIFI_DEVICE/mn-cse/json", req3); //
send message to request topic
```

Trong quá trình triển khai, có thể gặp phải một số lỗi liên quan đến định dạng và kích thước bản tin như sau:

1. Cấu trúc bản tin của MqttRequest chưa được chuẩn hóa và thống nhất giữa tiêu chuẩn oneM2M và dự án OM2M. Sau khi phân tích code trong MQTT Binding, cấu trúc bản tin chỉnh sửa có dạng sau:

```
{"m2m:rqp" : {
      "m2m:fr" : "admin:admin",
      "m2m:to" : "/mn-cse/mn-name",
      "m2m:op" : "1",
      "m2m:ty" : "2",
      "m2m:pc" : {
            "m2m:ae" : {
                  "rn" : "WIFI_SENSOR",
                  "api" : "luminosity-sensor",
                  "rr" : "true"
                     }
               }
            }
      }
```

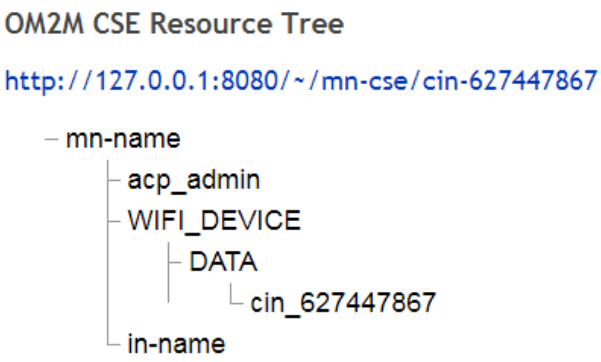
Trong đó, các trường *fr, to, op, ty* tương ứng với tham số *from, to, operation, resource type*, trường *pc* chứa nội dung của bản tin primitive. Trong ví dụ nếu trên bản tin mô tả tài nguyên <*AE*> gồm các thuộc tính *rn* (resource name), *api*, và *rr* (Request Reachability).

2. Bản MqttResponse có dung lượng lớn (600-800bytes) vượt quá dung lượng mặc định mà ESP8266 có thể nhận. Để khắc phục lỗi kể trên, chúng ta cần tăng kích thước tối đa của bản tin MQTT trong thư viện lên 1024.

```
#define MQTT_MAX_PACKET_SIZE 1024
```

3.3.4 Kết quả

Sau khi kết nối thành công, oneM2M ADN được thể hiện trong cây tài nguyên như Hình 3.11 dưới đây.



Hình 3.11 Cây tài nguyên ứng với Wi-Fi sensor

Cảm biến Wi-Fi đã đăng ký thành công với MN dưới tên WIFI_DEVICE và có chứa tài nguyên DATA lưu trữ các giá trị cảm biến. Mỗi giá trị đo được từ cảm biến ánh sáng được lưu trong thuộc tính "con" nằm trong một tài nguyên <*ContentInstance*> được đặt tên dưới dạng dạng cin_*, chúng ta có thể thấy trong hình dưới đây

Attribute	Value
rn	cin_627447867
ty	4
ri	/mn-cse/cin-627447867
pi	/mn-cse/cnt-670203067
ct	20191004T103258
lt	20191004T103258
st	0
cnf	message
cs	2
con	18

Hình 3.12 Tài nguyên <*ContentInstance*> lưu giá trị cảm biến ánh sáng

3.4 Tích hợp mạng Zigbee

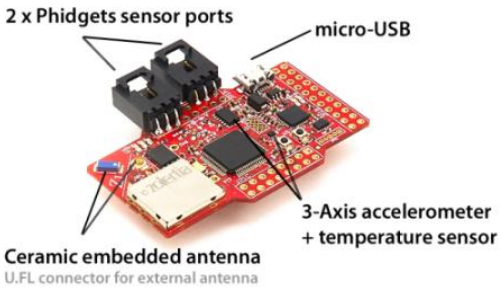
Giả sử đang có một mạng Zigbee hoạt động ổn định, nếu triển khai mỗi node như một oneM2M ADN giống với cách làm cảm biến Wi-Fi, mỗi thiết bị phần cứng đều cần lập trình lại và chuẩn hóa dạng bản tin. Công việc này liên quan nhiều đến thiết kế phần cứng cho từng thiết bị riêng biệt, không phản ánh rõ ràng cách thức kết nối đa công nghệ trong oneM2M. Do đó, nội dung phần này hướng đến phát triển một Zigbee Module có thể giao tiếp với mạng Zigbee nêu trên.

3.4.1 Mạng Zigbee

Mạng Zigbee trong thử nghiệm sử dụng công nghệ Zigbee, giao thức CoAP chạy trên nền IPv6. Các thiết bị cấu thành mạng sử dụng Kit phát triển Z1 Zolertia.

Một số thông số kỹ thuật của Z1 Zolertia:

- Vi điều khiển Ultra-Low Power MCU - MSP430F2617, 16-bit RISC CPU @16MHz clock speed.
- CC2420 Transceiver tương thích với Zigbee hoạt động ở dải tần 2.4GHz.
- 8KB RAM và 92KB Flash memory.
- Tích hợp sẵn 2 cảm biến đo nhiệt độ và gia tốc.
- Sử dụng bất kì nguồn điện nào từ 3 đến 5.25V.



Hình 3.13 Kit Z1 Zolertia

Z1 có thể lập trình và cài đặt Contiki, một hệ điều hành hỗ trợ đầy đủ IPv6, IPv4 và các giao thức 6LowPAN, CoAP, RPL,… Chi tiết về Protocol Stack của Z1 được mô tả trong bảng sau:

Bảng 3.1 Z1 Protocol Stack

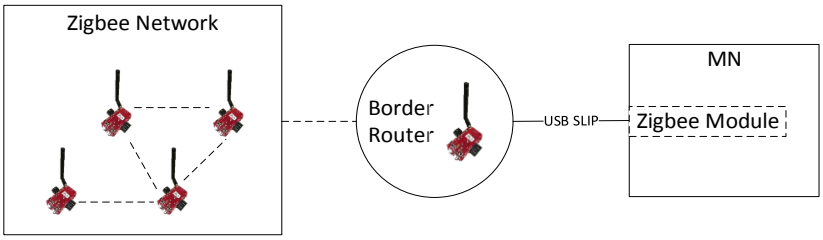
OSI Layer	**IoT Layer**	**IoT Technology**	**Zigbee network**
Application	Application	CoAP, CoAPs	Client - Server
Transport	Transport	UDP	UDP
Network	Net layer ICMP	IPv6 RPL, IPsec	IPv6 RPL
	Adaption layer	6LoWPAN	6LoWPAN
Data link	MAC layer	802.15.4 MAC	Un-beacon mode CSMA

	RDC layer	(none)	ContikiMAC
Physical	PHY layer	802.15.4 PHY	Framer802.15.4 cc2420 driver

Trong mạng Zigbee bao gồm:

- Z1 Border Router đóng vai trò định tuyến các bản tin trao đổi trong mạng Zigbee chạy trên nền IPv6.
- Thiết bị chấp hành Z1 được lập trình như một CoAP-Server. Nó có thể yêu cầu lấy dữ liệu từ hai cảm biến tích hợp sẵn cũng như thực hiện lệnh điều khiển LED.

3.4.2 Cấu hình tích hợp mạng Zigbee



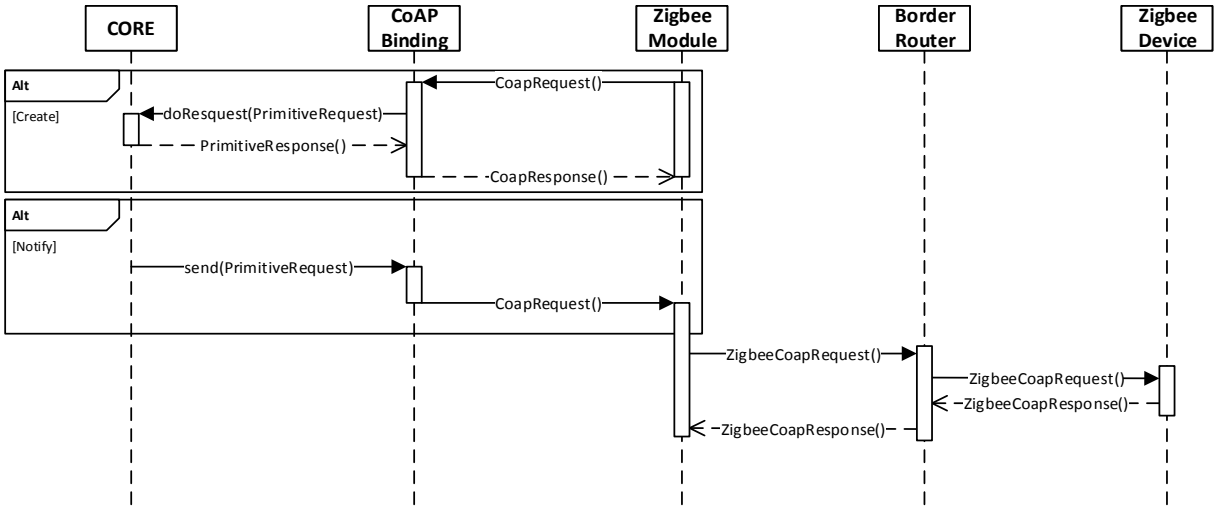
Hình 3.14 Cấu hình kết nối với Zigbee Network

Các thành phần chính trong cấu hình bao gồm:

- Mạng Z1 đang hoạt động gồm các node được cài hệ điều hành Contiki với protocol stack được mô tả trong Bảng 3.1. Trong kiến trúc hệ thống thử nghiệm đã trình bày, chúng ta chỉ thử nghiệm kết nối từ MN tới một Z1 node đóng vài trò như thiết bị chấp hành.
- Border Router được triển khai trên một Z1 node đóng vai trò làm router cho mạng Zigbee. Đồng thời, thiết bị này cũng được kết nối với latop thông qua giao tiếp USB SLIP.
- Zigbee Module được phát triển để có thể giao tiếp với Z1 node bằng giao thức CoAP trên nền IPv6.

3.4.3 Phát triển ZigbeeModule

Dưới đây, Hình 3.15 mô tả quá trình ZigbeeModule tạo tài nguyên và quá trình CORE gửi lệnh điều khiển cho thiết bị chấp hành Z1 (Zigbee Device)



Hình 3.15 Lưu đồ trao đổi bản tin giữa các khối và Zigbee Device

Đầu tiên, Zigbee Module thực hiện đăng ký với MN, khởi tạo các tài nguyên cần thiết mô tả Zigbee Device.

Mỗi khi CORE gửi bản tin điều khiển xuống sẽ thông qua CoAP Binding chuyển thành bản tin CoapRequest đi tới ZigbeeModule. Thay vì việc cố gắng giải quyết các vấn đề liên quan đến sự đồng nhất giữa bản tin CoapRequest (bản tin Coap của oneM2M) với bản tin ZigbeeCoapRequest (bản tin Coap của mạng Zigbee đang hoạt động) trên thiết bị chấp hành, khối Zigbee Module thực hiện ánh xạ bản tin điều khiển từ CORE sang lệnh điều khiển gửi trực tiếp tới Zigbee Device.

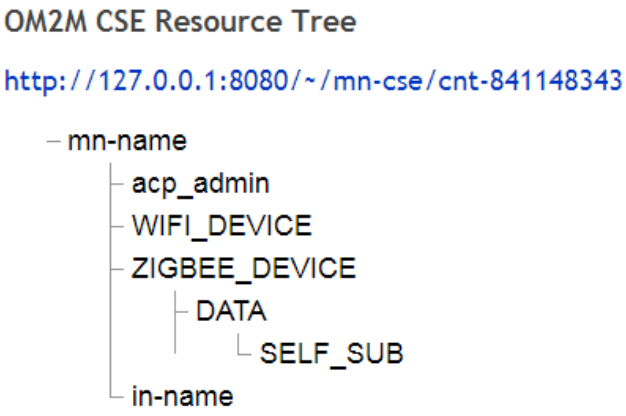
Việc lập trình ZigbeeModule có sử dụng thư viện Californium. Z1 node đã triển khai CoAP-Server dựa trên kiến trúc REST. Do đó, ZigbeeModule có thể dễ dàng lấy dữ liệu hay gửi lệnh điều khiển thông qua các phương thức GET, POST, PUT,… Bản tin GET với URI: coap://[aaaa::c30c:0000:0000:009f]:5683/sensors/battery thực hiện việc lấy giá trị điện áp của nguồn điện đang cung cấp cho kit. Các sensor được hỗ trợ gồm dữ liệu từ hai cảm biến đo nhiệt độ và gia tốc được tích hợp sẵn, các trạng thái của chip CC2420,… Bản tin POST có

URI: coap://[aaaa::c30c:0000:0000:009f]:5683/actuators/leds?color=g

body: mode="on" dùng để điều khiển đèn màu xanh bật.

3.4.4 Kết quả

Sau khi thực hiện kết nối thành công, thiết bị chấp hành Z1 sẽ được lưu trữ dưới tài nguyên ZIGBEE_DEVICE nằm trên MN. Do thiết bị chấp hành cần nhận được thông báo khi có biến động về ánh sáng, nên nó sẽ gửi bản tin đăng ký SELF_SUB đến chính tài nguyên DATA của mình. Chẳng hạn, khi ánh sáng yếu, khối xử lý thực hiện tính toán và gửi bản tin primitive chứa lệnh điều khiển đến tài nguyên DATA của ZIGEE_DEVICE. Tài nguyên này đã được đăng ký, do vậy thông tin này sẽ được thông báo (notify) cho Zigbee Module. Sau khi đọc và chuyển đổi định dạng, Zigbee Module chuyển bản tin về tới ZIGBEE_DEVICE.



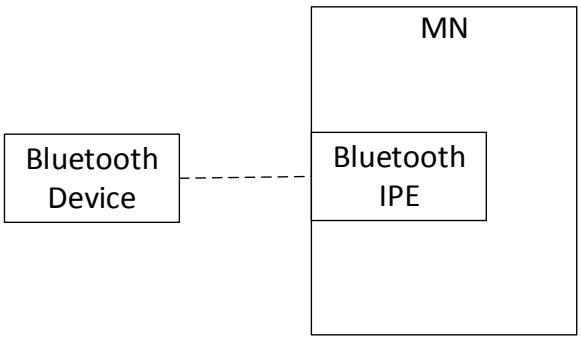
Hình 3.16 Cây tài nguyên của ZIGBEE_DEVICE

3.5 Tích hợp thiết bị Bluetooth

3.5.1 Cấu hình tích hợp thiết bị Bluetooth

Như đã phân tích trong mục 2.5, các công nghệ có thể triển khai một giao thức tầng ứng dụng sẽ được giao tiếp với oneM2M thông qua Binding (ví dụ sử dụng CoAP với công nghệ Zigbee kết nối với oneM2M qua CoAP Binding). Tác giả Nguyễn Thành Long cùng các cộng sự đã thực hiện được việc tích hợp giao thức MQTT và CoAP lên trên nền công nghệ LoRa trong bài báo [6]. Tuy nhiên, không phải bất kì công nghệ nào cũng có thể làm được điều này, ví dụ như Blueooth, các thiết bị ghép

nối với nhau qua giao thức non-IP. Do vậy, việc kết nối với oneM2M dẫn đến cần phát triển một BluetoothIPE module.



Hình 3.17 Cấu hình tích hợp thiết bị Bluetooth

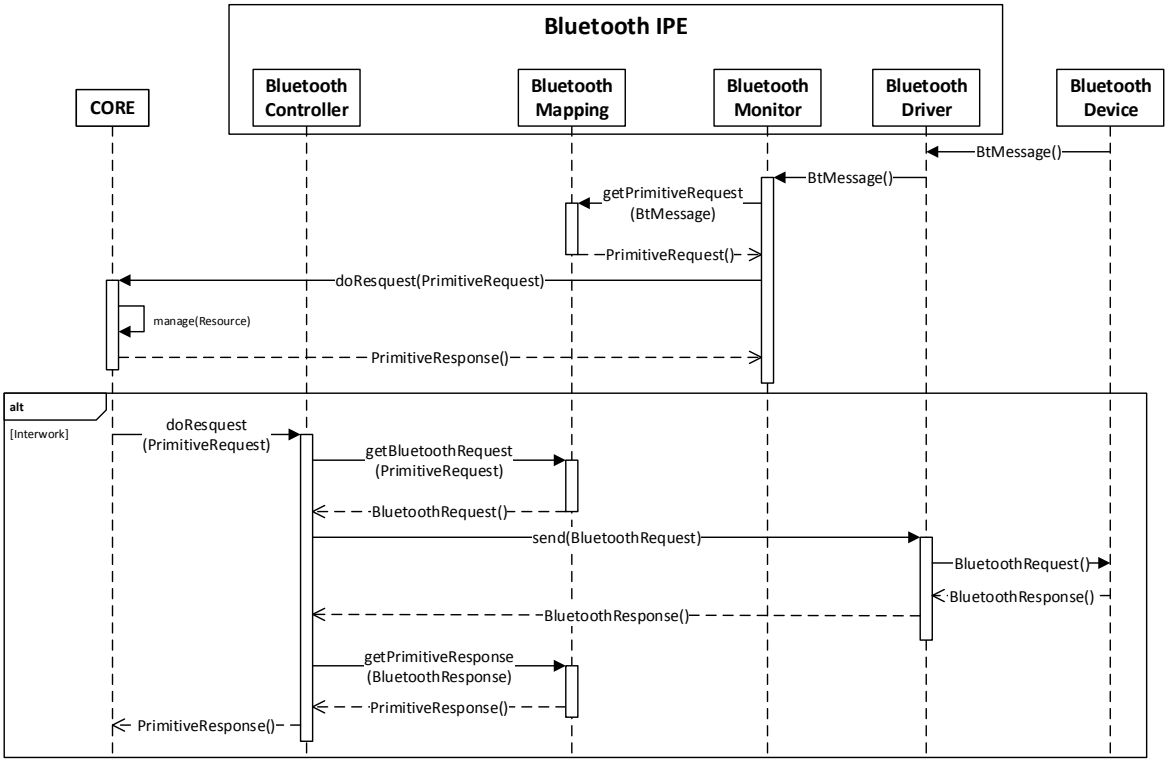
Hình 3.17 mô tả cầu hình kết hợp thiết bị Bluetooh với MN, trong đó:

- Bluetooth IPE được phát triển trong MN trên laptop, dùng chính driver card mạng của máy tính phục vụ giao tiếp.
- Bluetooth Device trong hệ thống hiện tại là một Smartphone có cài ứng dụng được phát triển riêng cho nhu cầu đọc dữ liệu và gửi lệnh điều khiển.

3.5.2 Phát triển Bluetooth IPE

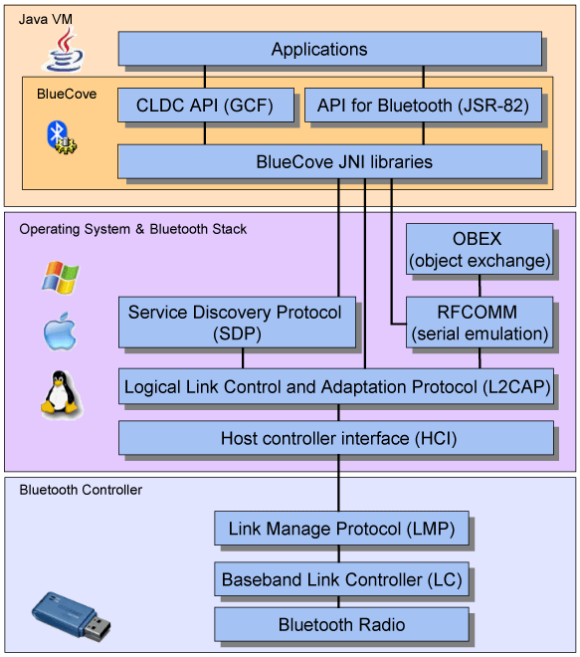
Việc phát triển Bluetooth IPE được phân theo các khối chức năng mô tả trong Hình 3.18, bao gồm:

- Bluetooh Mapping giúp chuyển đổi qua lại giữa các bản tin Primitive (bản tin của oneM2M) và BtMeassage (bản tin của Bluetooh Device)
- Bluetooth Controller phân loại các bản tin Primitve và chuyển về các chức năng/câu lệnh tương ứng để gửi cho Blueooth Device.
- Bluetooh Monitor làm nhiệm vụ ngược lại với Bluetooth Controller. Nó nhận và phân loại các bản tin BtMeassage và chuyển về các bản tin Primitve tương ứng gửi cho CORE.
- Bluetooh Driver cung cấp giao diện kết nối với Bluetooh Device qua sóng Bluetooh.



Hình 3.18 Luồng dữ liệu trao đổi với Bluetooth IPE

Trong bốn khối chức năng được mô tả, ba khối Bluetooth Mapping, Bluetooh Controller và Bluetooth Monitor chỉ thực hiện nhiệm vụ chính là chuyển đổi, điều hướng các bản tin thông qua những luồng dữ liệu trên một số giao diện logic đã được lập trình sẵn. Tuy nhiên, khối Bluetooth Driver được xem là khó triển khai do hoàn toàn không liên quan đến hệ thống OM2M, nó cần được phát triển riêng để phù hợp với các đặc tính của công nghệ Bluetooh. Để có thể thực hiện công việc này, chúng ta sử dụng thư viện BlueCove được mô tả trong Hình 3.19. Thư viện BlueCove cung cấp giao diện lập trình ứng dụng JSR82-API giúp đơn giản quá trình kết nối với hệ thống lõi của hệ điều hành.



Hình 3.19 Kiến trúc thư viện BlueCove

Trong giao tiếp Bluetooth, Bluetooth IPE đóng vai trò như master lắng nghe kết nối đến từ Bluetooth Device. Quá trình thiết lập cho Bluetooth master được mô tả qua đoạn code dưới đây.

```
UUID uuid=new UUID("1101",true);
String url = "btspp://localhost:" + uuid +";name= Bluetooth IPE" ;
StreamConnectionNotifier streamConnNotifier;
streamConnNotifier = (StreamConnectionNotifier)Connector.open( url );

StreamConnection connection=streamConnNotifier.acceptAndOpen();
RemoteDevice dev = RemoteDevice.getRemoteDevice(connection);
```

Việc gửi và nhận dữ liệu được thực hiện thông qua hàm `readData, write`

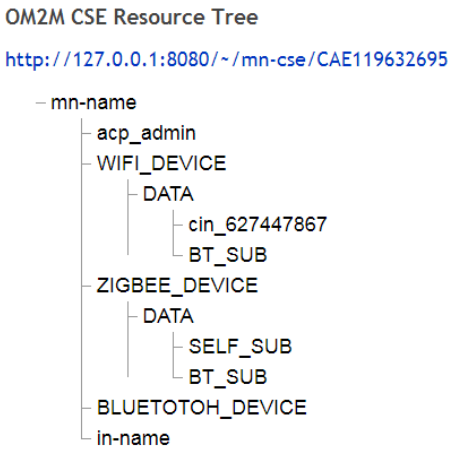
```
mServer=new MyServer(connection);
mServer.readData();
mServer.write(con);
listentoBt();
```

Các khối Controller, Monitor và Mapping được phát triển tương tự trong Zigbee module và các bộ Binding có sẵn.

3.5.3 Kết quả

Bluetooh Device kết nối vào mạng chủ yếu với mục đích giám sát và lấy thông tin nên Bluetooth Monitor sẽ thực hiện gửi bản tin đăng ký đến các tài nguyên của sensor và thiết bị chấp hành. Khi dữ liệu thay đổi, Bluetooth Controller sẽ nhận được và xử

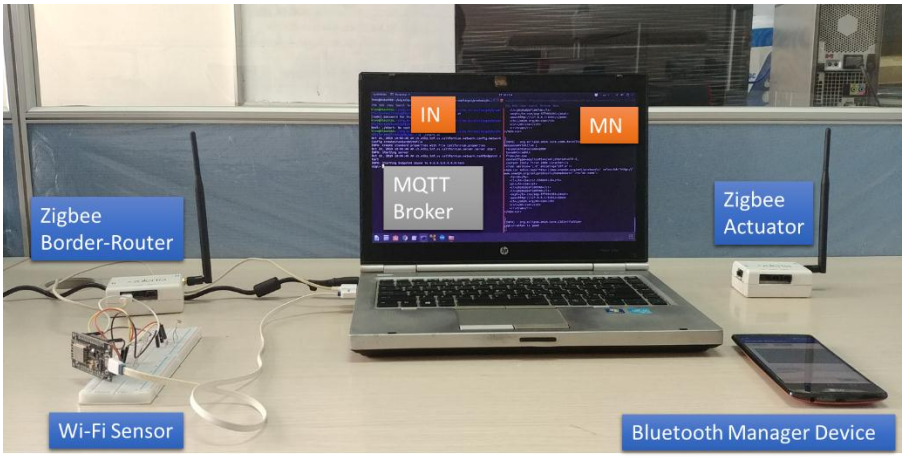
lý. Cây tài nguyên trong Hình 3.20 được tạo ra bởi Blueooth IPE để biểu diễn Bluetooh Device trong hệ thống oneM2M.



Hình 3.20 Cây tài nguyên của Bluetooh Device

3.6 Mô tả và đánh giá hoạt động

Testbed minh họa một ứng dụng IoT cơ bản dùng trong giám sát và quản lý. Các thiết bị IoT gồm có cảm biến ánh sáng và thiết bị chấp hành được kết nối với MN/Gateway nhằm mục đích thu thập dữ liệu và điều khiển thiết bị. Gateway cũng được hỗ trợ truy cập trực tiếp qua bluetooth nhằm mục đích vận hành và bảo trì. Trung tâm xử lý dữ liệu được đặt ở OM2M Server/IN node, tại đây, các ứng dụng hướng người dùng có thể truy cập thông qua Internet. Dưới đây là hình ảnh triển khai test-bed mô phỏng trong thực tế được thực hiện trong Lab.



Hình 3.21 Mô hình testbed trong thực tế

Testbed được triển khai trên các công nghệ Wi-Fi, Zigbee và Bluetooth cùng các giao thức MQTT, CoAP và HTTP. Quá trình tương tác chéo giữa các công nghệ và giao thức được thể hiện một cách trực quan qua hoạt động của các thiết bị có thể chia làm hai pha hoạt động cho toàn hệ thống.

Tại pha đầu tiên, MN-CSE tự động đăng ký với IN-CSE tạo ra một hệ thống OM2M cơ bản. Sau đó các module CoAP binding, MQTT binding và Bluetooth IPE được bật giúp MN sẵn sàng giao tiếp với các thiết bị IoT.

Tại pha thứ hai, khi các thiết bị IoT được bật, chúng sẽ thiết lập kết nối đến MN và tạo cây tài nguyên trong hệ thống. Các tài nguyên lần lượt được tạo là AE (định danh thiết bị trong hệ thống), Container (mô tả thiết bị và chứa dữ liệu), ContentInstance (chứa dữ liệu cần thiết cho các thiết bị). Để thu thập và xử lý dữ liệu, chúng ta dùng Manager ADN, nó sẽ đăng ký đến tài nguyên xác định nhằm lấy thông báo về các sự kiện cần quan tâm. Sau khi xử lý các thông tin thu thập được, chúng ta có thể phát hiện những sự kiện bất thường và cập nhật chúng vào cơ sở dữ liệu, đồng thời gửi lệnh điều khiển tới các thiết bị chấp hành. Manager ADN được lập trình linh hoạt có thể đặt ở MN hoặc IN tùy theo mục đích sử dụng.

Trong kịch bản được thực hiện, thiết bị cảm biến gửi dữ liệu về cường độ ánh sáng qua giao thức MQTT trên nền Wi-Fi tới MN-gateway mỗi phút một lần. Gateway nhận dữ liệu và thông báo đến tới các subscriber (ở đây là Manager ADN). Tại đó, dữ liệu được kiểm tra, nếu nhỏ hơn ngưỡng cho phép sẽ gửi lệnh *"turn LED ON"* đến Zigbee module, và ngay lập tức được chuyển đổi gửi đến thiết bị chấp hành sử dụng giao thức CoAP trên nền Zigbee. Nhận được lệnh điều khiển, thiết bị chấp hành bật đèn LED sáng. Nhờ vào việc Manager ADN được đặt ở MN, dữ liệu có thể xử lý ngay mà không cần đưa lên IN node/OM2M server.

Ngoài ra, để quan sát và giám sát trạng thái của các IN, MN chúng ta có thể dùng các ứng dụng trên điện thoại thông minh. Với kết nối Bluetooth, điện thoại sẽ kết nối với MN và trích xuất trực tiếp dữ liệu từ đó. Với kết nối Wi-Fi, điện thoại kết nối với IN thông qua giao thức HTTP.

3.7 Kết luận chương

OM2M được xây dựng theo đúng tiêu chuẩn oneM2M, các khối mã nguồn được chuẩn hóa và có khả năng thay đổi, mở rộng linh hoạt. Đặc biệt, OM2M "mở" hoàn toàn, giúp các lập trình viên và nhà phát triển có thể tìm hiểu sâu về cấu trúc hệ thống, thực hiện các nâng cấp khi cần thiết.

Dựa trên đó, luận văn xây dựng một mô hình hệ thống IoT cơ bản gồm cảm biến thu thập dữ liệu cường độ ánh sáng gửi về bộ tập trung MN/Gateway, rồi tới máy chủ IN/Server; hệ thống xử lý, tính toán và gửi lệnh điều khiển về thiết bị chấp hành. Để làm được điều đó, chúng ta cần: sửa lỗi Module CoAP binding, chuẩn hóa dạng bản tin sử dụng giao thức MQTT, phát triển Module Bluetooth IPE.

Sau khi khảo sát hệ thống với một số kịch bản đơn giản, để đáp ứng được nhu cầu về mở rộng và phân tán tài nguyên, điện toán biên được xem xét kết hợp vào oneM2M. Nội dung này sẽ được trình bày trong chương kế tiếp.

CHƯƠNG 4. TRIỂN KHAI ONEM2M VỚI ĐỊNH HƯỚNG ĐIỆN TOÁN BIÊN

Chương bốn đi vào tìm hiểu về điện toán biên, điện toán đám mây cũng như hướng triển khai trong oneM2M. Nội dung chương gồm một số hạn chế trong phiên bản oneM2M hiện tại và các giải pháp được xem xét, đề xuất triển khai để khắc phục các tồn tại đã nêu.

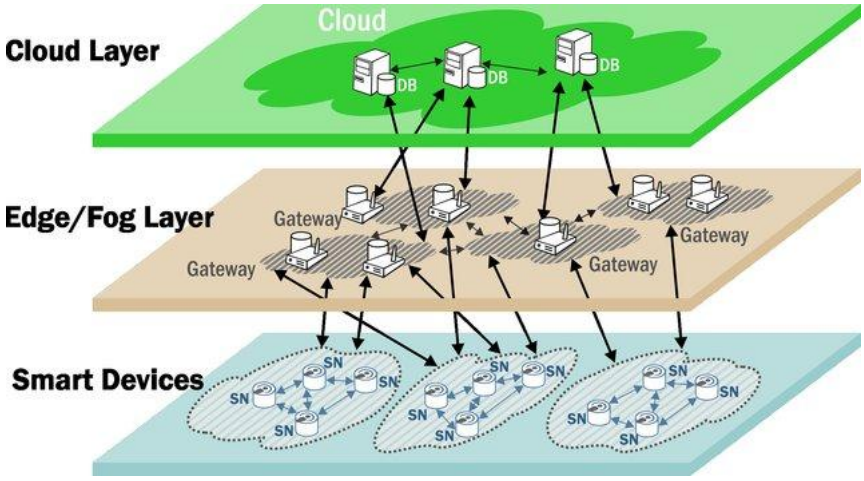
4.1 Điện toán biên

4.1.1 Giới thiệu chung

Số lượng các thiết bị IoT ngày càng gia tăng đồng nghĩa với lượng dữ liệu sinh ra rất lớn. Việc thu thập, gửi, và xử lý lượng dữ liệu lớn này đòi hỏi các giải pháp trở nên thông minh hơn, đưa ra hành động và quyết định nhanh chóng, chính xác hơn. Tuy nhiên, nếu gửi tất cả các dữ liệu đến mạng lưới trung tâm trên đám mây (cloud) sẽ làm gia tăng độ trễ truyền dẫn, làm các nút thắt tới máy chủ quá tải. Do đó, điện toán biên (Edge/Fog Computing) đã và đang được phát triển để giải quyết bài toán này. Theo một nghiên cứu từ IDC, 45% tất cả dữ liệu được tạo ra bởi IoT sẽ được lưu trữ, xử lý, phân tích và hành động tại các biên của mạng vào năm 2020.

Nói một cách đơn giản, điện toán biên chuyển phần lớn việc xử lý dữ liệu tới gần phía các thiết bị IoT. Điện toán biên bao gồm mạng lưới những bộ xử lý và lưu trữ dữ liệu cục bộ trước khi chuyển tiếp đến những trung tâm dữ liệu lớn hơn. Nó giúp tối ưu được hệ thống truyền dẫn, xử lý dữ liệu tại biên làm giảm thời gian gửi và nhận dữ liệu. Trong điện toán biên, mỗi phần tử từ những thiết bị IoT đến bộ tập trung Gateway đều sẵn sàng xử lý dữ liệu ngay tại chỗ khi có thể, thay vì phải đưa hết chúng lên đám mây.

Điện toán đám mây nổi lên từ những năm 2000, dù khái niệm sơ khai về chia sẻ tài nguyên qua mạng có thể truy cập toàn cầu đã có từ năm 1960. Microsoft, AWS và Google đã đi tiên phong trong cuộc đua đó, và đến nay, họ cũng đang đẩy mạnh phát triển điện toán biên. Nếu như điện toán đám mây ra đời cho phép tăng dung lượng lưu trữ, tính linh hoạt và chi phí thấp thì điện toán biên đem tất cả những ưu điểm đó tới nhiều nơi hơn mà không làm giảm tốc độ hay độ tin cậy.



Hình 4.1 Edge/Fog Layer [25]

Phát triển song song cùng điện toán biên, thuật ngữ điện toàn sương mù (fog computing) cũng được đề cập đến như một cách di chuyển "cloud" tản rộng ra các "fog". Điện toán biên hướng đến các chức năng gần với đám mây, điện toán sương mù lại gắn với những thiết bị thông minh. Chúng ta có thể xem nhưng điện toán sương mù là góc nhìn từ đám mây hướng xuống, còn điện toán biên là góc nhìn từ hướng các hệ thống ứng dụng IoT lên.

4.1.2 Lợi ích

Một số lợi ích tiêu biểu có thể kể đến khi sử dụng điện toán biên:

- Dữ liệu nhạy cảm về thời gian có thể được xử lý ngay tại điểm gốc bởi bộ xử lý cục bộ (một thiết bị sở hữu khả năng tính toán riêng).

- Các máy chủ trung gian có thể được sử dụng để xử lý dữ liệu gần nguồn (điều này được giả định là độ trễ trung gian chấp nhận được, mặc dù các quyết định thời gian thực nên được thực hiện càng gần nguồn gốc càng tốt)
- Các máy chủ có thể được sử dụng để xử lý ít dữ liệu thời gian nhạy cảm hơn hoặc để lưu trữ dữ liệu dài hạn.
- Các dịch vụ ứng dụng biên giảm đáng kể lượng dữ liệu phải được di chuyển, lưu lượng truy cập, và khoảng cách dữ liệu được di chuyển. Điều này sẽ làm giảm chi phí truyền tải, giảm thời gian trễ, và nâng cao chất lượng dịch vụ.
- Điện toán biên loại bỏ lượng lớn hiện tượng "nút thắt cổ chai" và giảm khả năng xảy ra các lỗi do không còn quá phụ thuộc vào môi trường tính toán tập trung. Đồng thời an toàn dữ liệu được cải thiện do các tấn công cần đi qua nhiều lớp bảo mật hơn, có thể bị phát biện ngay ở biên hoặc sương mù trước khi tới được đám mây.
- Khả năng mở rộng của điện toán biên tăng lên nhờ hợp lý hoá của các bộ xử lý CPU khi cần thiết, tiết kiệm chi phí khi truyền dữ liệu thời gian thực.

4.2 Điện toán biên trong oneM2M

4.2.1 Kiến trúc mô hình

Trong mô hình đề xuất [26], oneM2M triển khai thêm hai tính năng mới gồm lớp Fog-node hỗ trợ việc phát triển các chức năng cho lớp thiết bị đầu cuối và lớp Cloudlet-node giúp tăng cường kết nối của các lớp dưới tới đám mây. Khi đó các thực thể như ASN, MN đóng vai trò chính trong việc triển khai các tính năng này.

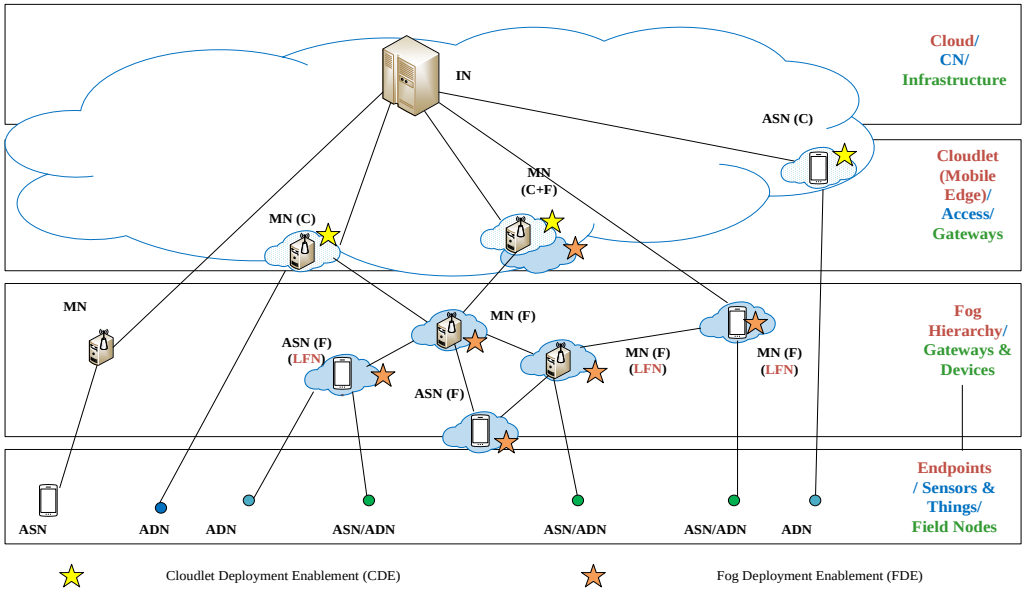
Lớp Fog-node: Bao gồm các nút phân cấp, có thể mở rộng, hỗ trợ các chức năng tự động ở field domain. Các chức tự động này bao gồm khám phá, điều phối và quản lý, bảo mật, vận hành giúp tiết kiệm chi phí mà không phụ thuộc vào một thực thể hoạt động tập trung.

Lớp Fog-node cần có một số đặc tính sau:

- Các Fog node có thể tự mở rộng chức năng thông qua việc nâng cấp phần cứng hoặc phần mềm.
- Vị trí của các node có thể giúp tối ưu hóa hệ thống
- Số lượng node được triển khai tăng hoặc giảm tùy thuộc vào điều kiện triển khai thực tế.
- Việc mở rộng các node có thể tăng khả năng lưu trữ, kết nối.

Lớp Cloudlet-node: Tập hợp các node có thể mở rộng bằng cách sử dụng công nghệ ảo hóa để cung cấp tài nguyên tính toán, lưu trữ và mạng phân tán cho các dịch vụ đám mây/cơ sở hạ tầng.

Để triển khai được các lớp Fog-node và Clouldet-node, chúng ta cần phát triển thêm các dịch vụ mới tích hợp vào bên trong phiên bản oneM2M hiện tại.



Hình 4.2 Kiến trúc oneM2M tích hợp Edge/Fog computing [26]

Endpoint: là các thiết bị đầu cuối, không được triển khai các dịch vụ mới.

Local Fog Node: có khả năng thực hiện các dịch vụ mới và cung cấp dịch vụ đó cho ít nhật một Endpoint.

4.2.2 Những khó khăn tồn tại

Việc chuyển các nhiệm vụ xử lý từ Cloud xuống cho lớp Fog hoàn toàn khả thi và hiện tại có thể thực hiện ngay trong nền tảng OM2M. Trong hệ thống thử nghiệm ở chương 3, chúng ta thấy việc đặt Manager ADN phục vụ xử lý dữ liệu cường độ ánh sáng và đưa ra quyết định điều khiển LED có thể triển khai trên MN mà không cần đưa dữ liệu tới IN. Tuy nhiên, Fog và Cloudlet hướng đến một mạng các MN-Gateway kết nối và trao đổi thông tin với nhau dẫn đến bài toán định tuyến và topo mạng bị thay đổi rất nhiều, khó còn có thể áp dụng cơ chế điều hướng hiện có trong oneM2M.

Để dễ dàng tiếp cận những nội dung sẽ trình bày, một số định nghĩa được giới thiệu và sử dụng trong các phần tiếp theo.

- Các thực thể trong oneM2M có sự phân cấp thứ bậc, khi hai thực thể đăng ký với nhau, thực thể cha được gọi là Registrar CSE, thực thể con được gọi là Registree CSE. Mỗi thực thể được định danh bằng một CSE-ID duy nhất.
- Tài nguyên <*remoteCSe*> của một thực thể lưu trữ các thực thể khác đăng ký với nó, tức là bao gồm CSE-ID của cả Registrar CSE và Registree CSE.
- Một CSE nhận được bản tin được gọi là Hosting CSE

4.2.2.1 Giới hạn kết nối với MN-CSE

Theo như tiêu chuẩn kỹ thuật được viết trong mục 6.4 của tài liệu [19].

Một MN-CSE sẽ chỉ có thể được hỗ trợ đăng ký tới một MN-CSE khác hoặc IN-CSE. Một chuỗi kết nối không được tạo vòng lặp. Ví dụ: Có 3 MN-CSE là A, B, C, nếu A đăng ký với B, B đăng ký với C thì C không đăng ký được với A.

Trong dự án triển khai OM2M, mỗi thực thể MN chỉ được phép đăng ký với duy nhất một thực thể MN/IN khác. Điều này dẫn đến hạn chế rất lớn trong việc mở rộng và phát triển hệ thống mạng.

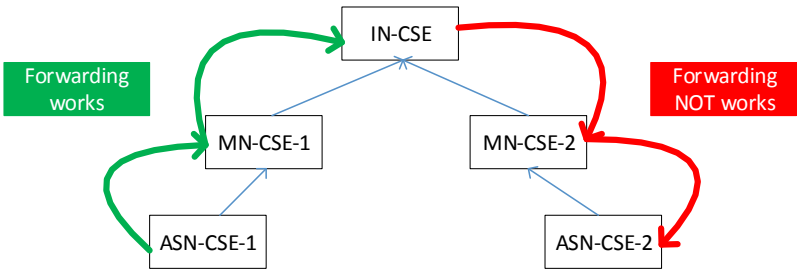
4.2.2.2 Cơ chế retargeting

Giả sử một bản tin cần đi đến một CSE đích xác định (được định danh bằng Destination CSE-ID nằm trong thuộc tính To của bản tin). Khi đi tới một Hosting CSE (được định danh bằng Hosting CSE-ID), có các trường hợp sau xảy ra:

- Hosting CSE-ID trùng khớp với Destination CSE-ID, khi đó bản tin được đưa tới bộ xử lý nằm ngay trên Hosting CSE.
- Hosting CSE-ID khác với Destination CSE-ID, bản tin sẽ được điều hướng tới một CSE khác. Khi đó, Hosting CSE thực hiện tìm kiếm trong tài nguyên <*remoteCSE*>, có ba trường hợp sau xảy ra:
    - Nếu Destination CSE-ID trùng với một CSE-ID nằm trong tài nguyên <*remoteCSE*>, bản tin được điều hướng tới đích.
    - Nếu Destination CSE-ID không trùng với bất kì CSE-ID nằm trong tài nguyên <*remoteCSE*> và Hosting CSE có Registrar CSE. Bản tin được điều hướng tới Registrar CSE-ID.
    - Nếu Destination CSE-ID không trùng với bất kì CSE-ID nằm trong tài nguyên <*remoteCSE*> và Hosting CSE không có Registrar CSE. Bản tin bị hủy và một Hosting CSE gửi trả một thông báo lỗi 4004 NOT_FOUND.

Như vậy, việc bản tin được điều hướng hay không phụ thuộc rất nhiều vào danh sách các CSE-ID nằm trong tài nguyên <*remoteCSE*>. Tuy nhiên, với oneM2M hiện tại, chỉ hai CSE đăng ký trực tiếp với nhau thì thông tin mới được cập nhật trong tài

nguyên <*remoteCSE*>. Để thấy được rõ ràng hạn chế của cơ chế điều hướng này, chúng ta cùng xét một topo mạng cụ thể được mô tả trong hình dưới đây.



Hình 4.3 Topo mạng thể hiện hạn chế của oneM2M

Ở đây ASN-CSE-1 đăng ký với MN-CSE-1 và MN-CSE-1 đăng ký với IN-CSE. Tương tự ASN-CSE-2 đăng ký với MN-CSE-2 và MN-CSE-2 đăng ký với IN-CSE. Xét trường hợp một bản tin gửi xuất phát từ ASN-CSE-1, gửi đến ASN-CSE-2, khi đó Destination CSE-ID là ASN-CSE-2.

Tại MN-CSE-1, <*remoteCSE*> gồm {ASN-CSE-1, IN-CSE}, Registrar CSE là IN-CSE. Bản tin đến MN-CSE-1 có Destination CSE-ID là ASN-CSE-2 không thuộc tập hợp các CSE có trong <*remoteCSE*> nên sẽ được chuyển đến IN-CSE.

Tại IN-CSE, <*remoteCSE*> gồm {MN-CSE-1, MN-CSE-2}, do là nút gốc nên IN-CSE không có Registrar CSE. Bản tin đến IN-CSE có Destination CSE-ID là ASN-CSE-2 sẽ bị gửi trả về thông báo không tìm thấy.

4.2.3 Hướng giải quyết

4.2.3.1 Mở rộng kết nối cho MN-CSE

Theo như hướng dẫn sử dụng OM2M trong [24], mỗi MN-CSE chỉ có thể đăng ký tới duy nhất một CSE khác. Thông tin về Registrar CSE cần được cài đặt trước thông qua file *config.ini*. Bảng 4.1 dưới đây là một ví dụ cho việc cấu hình CSE đăng ký với IN có địa chỉ IP là localhost và sử dụng port 8080.

Bảng 4.1 Các tham số cấu hình remoteCSE

Parameter	Description	Example
org.eclipse.om2m.remoteCseId	Remote IN-CSE Id	in-cse
org.eclipse.om2m.remoteCseName	Remote IN-CSE Name	in-name
org.eclipse.om2m.remoteCseAddress	Remote IN-CSE ip address	127.0.0.1
org.eclipse.om2m.remoteCsePort	Remote IN-CSE listening port	8080
org.eclipse.om2m.remoteCseContext	Remote IN-CSE listening context	/

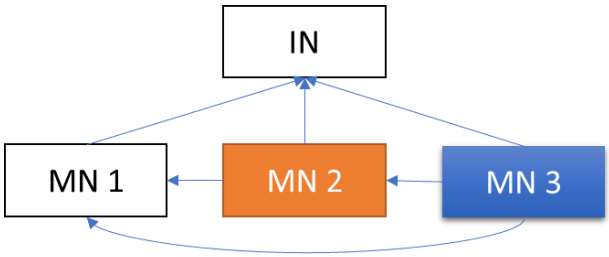
Để có thể mở rộng quy trình đăng ký này, chúng ta sẽ viết lại class `CSEInitializer` quản lý việc khởi tạo và thực hiện đăng ký của CSE. Cụ thể, thông tin về các CSE mà MN-CSE sẽ đăng ký tới được lưu vào các trường ***REMOTE_CSE_ID_String***, ***REMOTE_CSE_NAME_String***, ***REMOTE_CSE_IP_String***, ***REMOTE_CSE_PORT_string***, mỗi thông tin của CSE được ngăn cách nhau bởi một dấu cách " ". Các thông tin này được đọc vào và lưu trữ dưới dạng mảng. Quá trình đăng ký sẽ lần lượt được thực hiện theo thứ tự các CSE được viết từ trái qua phải.

```
private static String[] REMOTE_CSE_ID =
Constants.REMOTE_CSE_ID_String.split(" ");
private static String[] REMOTE_CSE_NAME =
Constants.REMOTE_CSE_NAME_String.split(" ");
private static String[] REMOTE_CSE_IP =
Constants.REMOTE_CSE_IP_String.split(" ");
private static String[] REMOTE_CSE_PORT =
Constants.REMOTE_CSE_PORT_string.split(" ");
```

Ví dụ MN-CSE-2 lần lượt đăng ký với IN-CSE và MN-CSE-1 thì nội dung file config.ini trên MN-CSE cần có các trường:

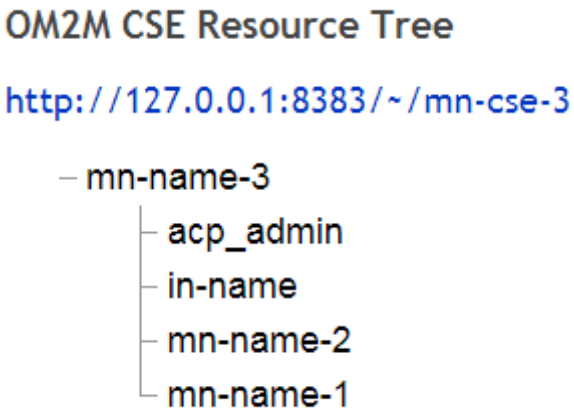
```
org.eclipse.om2m.remoteCseId_string=in-cse mn-cse-1
org.eclipse.om2m.remoteCseName_string=in-name mn-name-1
org.eclipse.om2m.cseBaseAddress_string=127.0.0.1 127.0.0.1
org.eclipse.om2m.remoteCsePort_string=8080 8181
```

Để thử nghiệm kết quả đạt được, ta xét một sơ đồ mạng cơ bản như trong Hình 4.4



Hình 4.4 Topo mạng thử nghiệm

Ở đây, mỗi MN khi bật lên đều đăng ký đến IN. Theo thứ tự tăng dần, các MN sau bật lên sẽ đăng ký với MN trước. Quá trình này được thực hiện nhờ vào việc đọc file cấu hình và thực hiện ở giai đoạn khởi động đầu tiên. Hình 4.5 mô tả cây tài nguyên quan sát được trên MN 3, ở đó đã có IN, MN 1 và MN 2.



Hình 4.5 Cây tài nguyên trên MN 3

4.2.3.2 Tích hợp giao thức định tuyến cho oneM2M

Như đã phân tích ở trên, cơ chế định tuyến/điều hướng bản tin trong oneM2M chưa hoàn thiện. Trong luận văn này, chúng ta sẽ đưa ra và triển khai một cơ chế đơn giản để chuyển tiếp thông tin kết nối giữa các thực thể đã đăng ký với nhau bằng cách thêm một thuộc tính mới là descendantCSEs. Việc thêm một thuộc tính đồng nghĩa với mở rộng được danh sách các CSE-ID trong tài nguyên *<remoteCSE>,* Hosting CSE có thể tìm thấy Destination CSE-ID trong thuộc tính mới và chuyển tiếp bản tin.

Mô tả chi tiết về thuộc tính descendantCSEs được thể hiện trong bảng dưới đây.

Bảng 4.2 Thuộc tính descendantCSEs

Attributes of <*CSEBase*>	**Multiplicity**	**RW/ RO/ WO**	**Mô tả**
descendantCSEs	0..1(L)	RW	Thuộc tính này chứa danh sách các Registree CSE và được lưu trong tài nguyên <*remoteCSE*>. Một descendant CSE là CSE đăng ký với CSE trong <*remoteCSE*> hoặc đăng ký với CSE khác là descendant của CSE trong <*remoteCSE*>. Registree CSE sẽ cập nhật thuộc tính này mỗi khi có CSE đăng ký hoặc hủy đăng ký.

Quy trình khởi tạo, cập nhật và xóa thuộc tính này được trình bày sau đây, trong đó Originator CSE đại diện cho thực thể khởi tạo và gửi bản tin, Receiver CSE đại diện cho thực thể nhận và xử lý bản tin.

Quy trình tạo tài nguyên *<remoteCSE>*

Bước 1: Originator CSE gửi bản tin CREATE Request tới Receiver CSE chứa thông tin về các descendant CSEs của nó.

Bước 2: Receiver CSE thực hiện tạo tài nguyên <*remoteCSE*>. Nếu Receiver CSE đã đăng ký với CSE khác (Registrar CSE), Receiver CSE gửi bản tin cập nhật cho Registrar CSE để thêm thông tin vào thuộc tính descendantCSEs trong tài nguyên <*remoteCSE*> của Receiver CSE trên Registrar CSE. Thông tin bao gồm: CSE-IDs của Originator CSE, những descendantCSEs của Originator CSE.

Bước 3: Receiver CSE gửi trả bản tin Response.

Bước 4: Originator CSE dựa trên thông tin có trong bản tin CREATE response, tạo tài nguyên <*remoteCSE*> là con của tài nguyên <*CSEBase*>. Tài nguyên được tạo này biểu diễn Receiver CSE.

Quá trình cập nhật tài nguyên *<remoteCSE>*

Bước 1: Originator CSE gửi bản tin UPDATE Request tới Receiver CSE bao gồm descendant CSEs của nó.

Bước 2: Receiver CSE thực hiện cập nhật tài nguyên <*remoteCSE*>. Nếu thuộc tính descendantCSEs được cập nhật, và Receiver CSE đã đăng ký với CSE khác, Receiver CSE gửi bản tin cập nhật tới Registrar CSE của nó để thực hiện cập nhật tương ứng cho thuộc tính descendantCSEs trong tài nguyên <*remoteCSE*> của Receiver CSE trên Registrar CSE.

Bước 3: Receiver CSE gửi trả bản tin Response.

Quy trình xoá tài nguyên *<remoteCSE>*

Bước 1: Originator CSE gửi bản tin DELETE Request tới Receiver CSE.

Bước 2: Receiver CSE thực hiện xóa tài nguyên <*remoteCSE*> ứng với Originator CSE. Nếu Receiver CSE đã đăng ký với CSE khác, Receiver CSE gửi bản tin cập nhật tới Registrar CSE của nó để xóa thuộc tính thông tin trong thuộc tính descendantCSEs của tài nguyên <*remoteCSE*> của Receiver CSE trên Registrar CSE. Thông tin bị xóa bao gồm: CSE-IDs của Originator CSE và những descendants của Originator CSE.

Bước 3: Receiver CSE gửi trả bản tin Response.

Bước 4: Originator CSE dựa vào bản tin DELETE response, xóa tài nguyên <*remoteCSE*> ứng với Receiver CSE.

4.3 Kết luận chương

Chương bốn đã đưa ra các định nghĩa cơ bản về Fog/Edge computing và cách tích hợp chúng cùng nền tảng oneM2M. Bước đầu giải quyết được những hạn chế gặp phải trong mở rộng kết nối cho MN và phương thức định tuyến lớp CSE. Đây là cơ

sở để có thể triển khai và đo đạc, đánh giá, so sánh hiệu năng của các thuật toán định tuyến trong oneM2M. Bước kế tiếp trong việc phát triển và nghiên cứu oneM2M theo định hướng điện toán biên cần quan đến việc phân loại bản tin từ nguồn dựa trên các cơ chế dán nhãn QoS phù hợp, từ đó luồng thông tin sẽ được hệ thống cư xử theo các mức độ ưu tiên khác nhau. Những bản tin cần ưu tiên về thời gian trễ ngắn sẽ được tính toán xử lý ngay tại biên; những bản tin có khối lượng lớn, cần tính toán phức tạp sẽ được đưa về Cloud. Cơ chế này sẽ được tích hợp dưới dạng một chức năng (CSF) mới, nhúng vào một số node mạng nhất định có cầu hình tương đối mạnh và có khả năng quản lý một cụm nhỏ các cảm biến cục bộ.

KẾT LUẬN

Luận văn đã trình bày được về tiêu chuẩn oneM2M, và OM2M - dự án triển khai tiêu chuẩn trên ngôn ngữ Java. Thông qua đó, một testbed mô phỏng về hệ thống IoT cơ bản có thể kết nối với nhiều giao thức qua các giao thức khác nhau, thực hiện được một số kịch bản đơn giản đã được triển khai. Ngoài ra, hướng phát triển ban đầu nhằm tích hợp Edge computing vào oneM2M cũng đã được đề xuất. Tuy nhiên, vẫn còn những vấn đề hạn chế trong nghiên cứu của luận văn. **Hạn chế**: Mô hình thực nghiệm sử dụng các Module ảo hóa chạy trong hệ điều hành trên laptop, tuy nhiên, thực tế nó cần được lập trình tích hợp vào ngay trong các thiết bị phần cứng Gateway.

Hướng nghiên cứu tiếp theo của đề tải sẽ là giải quyết được các vấn đề về định tuyến và phát triển Edge computing trong mạng tuân theo oneM2M. Phát triển được công cụ Simulator giúp mô phỏng và đánh giá được một số tham số hiệu năng mạng, song song với đó là nghiên cứu triển khai oneM2M trên các thiết bị phần cứng Gateway có cấu hình giới hạn.

TÀI LIỆU THAM KHẢO

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] J. Manyika *et al.*, "The Internet of Things: Mapping the value beyond the hype," *McKinsey Glob. Inst.*, 2015.

[3] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, and J. Song, "Toward a standardized common M2M service layer platform: Introduction to oneM2M," *IEEE Wirel. Commun.*, vol. 21, no. 3, pp. 20–26, 2014.

[4] "FP7, (9/18/2014). EUInternet of Things Architecture project." [Online]. Available: http://www.iot-a.eu/public.

[5] TS-0026-V3.2.0, "3GPP Interworking." [Online]. Available: http://www.onem2m.org/technical/published-drafts.

[6] T. L. Nguyen, S. Patonico, M. Bezunartea, S. Thielemans, A. Braeken, and K. Steenhaut, "Horizontal Integration of CoAP and MQTT on Internet Protocol - Based LoRaMotes," *IEEE Int. Symp. Pers. Indoor Mob. Radio Commun. PIMRC*, vol. 2018-Septe, pp. 1–7, 2018.

[7] G. Kim, S. Kang, J. Park, and K. Chung, "A MQTT-Based Context-Aware Autonomous System in oneM2M Architecture," *IEEE Internet Things J.*, vol. PP, no. c, pp. 1–1, 2019.

[8] J. Yun, R. C. Teja, N. Chen, N. M. Sung, and J. Kim, "Interworking of oneM2M-based IoT systems and legacy systems for consumer products," in *2016 International Conference on Information and Communication Technology Convergence, ICTC 2016*, 2016.

[9] C. W. Wu, F. J. Lin, C. H. Wang, and N. Chang, "OneM2M-based IoT protocol integration," *2017 IEEE Conf. Stand. Commun. Networking, CSCN 2017*, pp. 252–257, 2017.

[10] J. Kim *et al.*, "Standard-based IoT platforms interworking: Implementation, experiences, and less ons learned," *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 48–54, 2016.

[11] S. C. Choi, I. Y. Ahn, J. H. Park, and J. Kim, “Towards real-time data delivery in oneM2M platform for UAV management system,” *ICEIC 2019 - Int. Conf. Electron. Information, Commun.*, pp. 1–3, 2019.

[12] S. C. Choi, N. M. Sung, J. H. Park, I. Y. Ahn, and J. Kim, “Enabling drone as a service: OneM2M-based UAV/drone management system,” *Int. Conf. Ubiquitous Futur. Networks, ICUFN*, pp. 18–20, 2017.

[13] P. K. Dalela, A. Yadav, and V. Tyagi, “Security enhancement in tower monitoring system of oneM2M network,” in *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems, ANTS 2016*, 2017.

[14] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Comput. Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[15] C. Bormann, Z. Shelby, and K. Hartke, “The Constrained Application Protocol (CoAP),” *J. Chem. Inf. Model.*, vol. 53, no. 9, pp. 1689–1699, 2013.

[16] OASIS, “MQTT Version 3.1.1,” *OASIS Standard*, 2014. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html.

[17] “Abracon | Choose the Right IoT Wireless Connectivity Protocol.” [Online]. Available: https://abracon.com/index.php/news/choose-the-right-iot-wireless-connectivity-protocol. [Accessed: 01-Oct-2019].

[18] P. McDermott-Wells, “What is Bluetooth?,” no. March, pp. 25–27, 2005.

[19] TS-0001-V3.15.0, “Functional Architecture.” [Online]. Available: http://www.onem2m.org/technical/published-drafts.

[20] TS-0004-V3.13.0, “Service Layer Core Protocol.” [Online]. Available: http://www.onem2m.org/technical/published-drafts.

[21] TS-0009-V3.4.0, “HTTP Protocol Binding.” [Online]. Available: http://www.onem2m.org/technical/published-drafts.

[22] TS-0010-V3.0.1, “MQTT Protocol Binding.” [Online]. Available: http://www.onem2m.org/technical/published-drafts.

[23] TS-0008-V3.4.0, “CoAP Protocol Binding.” [Online]. Available: http://www.onem2m.org/technical/published-drafts.

[24] “Eclipse OM2M - Open Source platform for M2M communication,” 2015.

[Online]. Available: https://www.eclipse.org/om2m/.

[25] A. M. Rahmani *et al.*, “Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach,” *Futur. Gener. Comput. Syst.*, 2018.

[26] oneM2M-TR-0052-V0.5.0, “Study on Edge and Fog Computing in oneM2M systems.” [Online]. Available: http://www.onem2m.org/technical/published-drafts.

PHỤ LỤC

Phụ lục 1. Bảng danh sách các loại tài nguyên

Bảng PL.1 Danh sách các tài nguyên oneM2M

Resource Type	Short Description
accessControlPolicy	Kiểm soát "ai" được phép "làm gì" và nội dung nó có thể truy cập trong một tài nguyên xác định.
AE	Lưu trữ thông tin về thực thể ứng dụng. Nó được tạo ra sau khi ứng dụng đăng ký thành công với CSE.
container	Dùng để chia sẻ dữ liệu giữa các thực thể, hoạt động như một bộ đệm dữ liệu cho phép trao đổi linh hoạt giữa AE và CSE.
contentInstance	Biểu diễn giá trị của dữ liệu được lưu trong tài nguyên <container>
CSEBase	Tài nguyên gốc đại diện cho một CSE, nó thông tin về CSE dưới dạng các thuộc tính và là cha của tất cả các tài nguyên khác.
delivery	Chuyển tiếp request từ CSE đến CSE.
eventConfig	Định nghĩa các sự kiện thu thập thông tin với mục đích thống kê.
execInstance	Execution Instance resource gồm tất cả các execution instances của cùng management command mgmtCmd
fanOutPoint	Sử dụng để thực hiện phương thức tới nhiều tài nguyên thuộc cùng một nhóm chung.
group	Tăng cường hoạt động của cây tài nguyên và đơn giản hóa các tương tác trên giao diện API bằng cách thêm tính tăng nhóm. Nó cho phép gửi yêu cầu đến một nhóm người nhận thay vì gửi từng yêu cầu.
locationPolicy	Chứa thông tin về vị trí bản tin gửi/nhận
mgmtCmd	management procedures cmd đại diện cho một phương thức giải quyết các yêu cầu liên quan đến giao thức quả lý hiện tại.
mgmtObj	Đại diện cho ác chức năng quản lý, chúng cung cấp một lớp trừu tượng ánh xạ tới các công nghệ quản lý mở rộng.
node	Chứa thông tin của một Node
pollingChannel	Đại diện cho một kênh, sử dụng khi quản tin request không hướng tới một thực thể xác định.
remoteCSE	Lưu trữ thông tin liên quan đến M2M CSEs nằm trong các thiết bị khác sau khi đăng ký thành công. Tài nguyên này hỗ trợ việc điều hướng bản tin trong mạng
schedule	Chứa thông tinh lập lịch gửi bản tin

statsCollect	Định nghĩ các triggers cho phép IN-CSE thu thập số liệu thống kê về các ứng dụng.
statsConfig	Lưu trữ cấu hình thống kê của các ứng dụng.
subscription	Lưu trữ thông tin liên quan đến đăng ký cho một số tài nguyên. Nó cho phép người đăng ký nhận thông báo không đồng bộ khi có sự kiện xảy ra, chẳng hạn dữ liệu mới từ cảm biến, cập nhật hoặc xóa tài nguyên.

Phụ lục 2. Bản tin primitive

Phần nội dung đại diện cho tài nguyên <*container*> được biểu diễn dưới dạng JSON.


```
{
    "m2m:cnt": {
        "cbs": 0,
                "cni": 0,
                 "ct": "20180406T085712",
        "et": "99991231T235959",
        "lt": "20180406T085712",
        "mbs": 60000000,
        "mia": 1600,
        "mni": 10000,
        "pi": "CAE0120180406T084680_cse01",
        "ri": "cnt20180406T08571214_cse01",
        "rn": "cont_temp",
        "st": 0,
        "ty": 3
              }
}
```

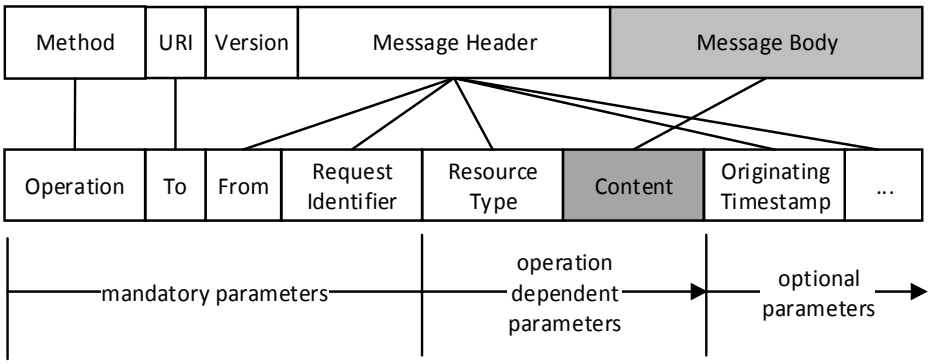

cnt : container
cbs : currentByteSize
cni : currentNrOfInstances
ct : creationTime
et : expirationTime
lt : lastModifiedTime
mbs : maxByteSize
mia : maxInstanceAge
mni : maxNrOfInstance
pi : parentID
ri : resourceID
rn :resourceName
st : stateTag
ty : resourceType

Phần nội dung đại diện cho tài nguyên <*container*> được biểu diễn dưới dạng XML.


```
<?xml version="1.0" encoding="UTF-8"?>
<m2m:cnt xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="cont_temp">
   <ty>3</ty>
   <ri>server/cnt-2951972863155866584</ri>
   <pi>server</pi>
   <ct>20181114T145000</ct>
   <lt>20181114T145000</lt>
   <et>20181114T145000</et>
   <st>0</st>
   <mni>10000</mni>
   <mbs>0</mbs>
   <mia>0</mia>
   <cni>0</cni>
   <cbs>0</cbs>
</m2m:cnt>
```


Phụ lục 3. Ánh xạ bản tin

Ánh xạ kiến trúc bản tin HTTP



Hình PL.1 Ánh xạ bản tin HTTP request và Primitive request

Hình PL.1 mô tả việc ánh xạ từ một bản tin request từ HTTP sang Primitive. Trường Method được ánh xạ với Operation, cụ thể được trình bày trong bảng sau:

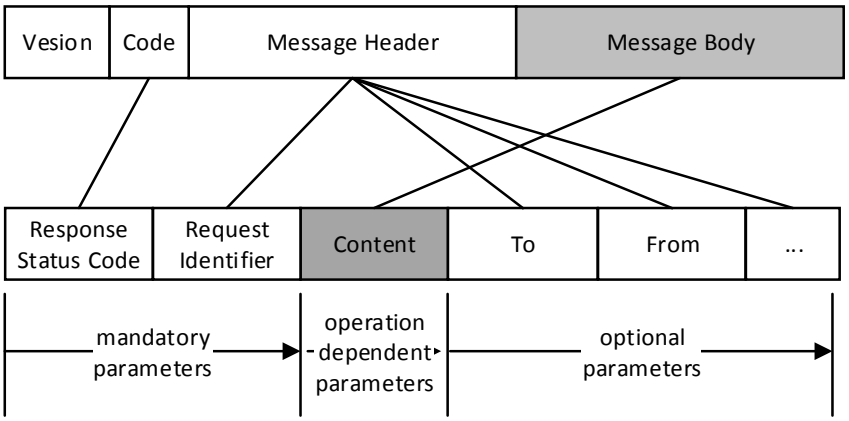
Bảng PL.2 Ánh xạ trường HTTP Method

oneM2M Operation	HTTP Method
Create	POST
Retrieve	GET
Update	PUT
Delete	DELETE
Notify	POST

Trường URI chứa địa chỉ tài nguyên đích tương ứng với trường To. Ngoài ra, trong Header của bản tin HTTP sẽ lưu tất cả những thông tin bổ sung liên quan như ResourceType (ty), From (originator),…

Messager Body sẽ tương ứng với Content của bản tin Primitive, phần này đại diện cho thông tin chứa trong một tài nguyên.

Tương tự, sự ánh xạ giữa bản tin hình PL.2 miêu tả ánh xạ bản tin response từ HTTP sang Primitive. Do bản tin Primitive hoàn toàn độc lập với các giao thức nó được ánh xạ nên trường Version ở đây không có tương ứng nào, trường này được luôn được đặt là “HTTP/1.1”. Sự ánh xạ của Response Status code với Code được liệt kê chi tiết trong tài liệu [21].



Hình PL.2 Ánh xạ bản tin HTTP response và Primitive response

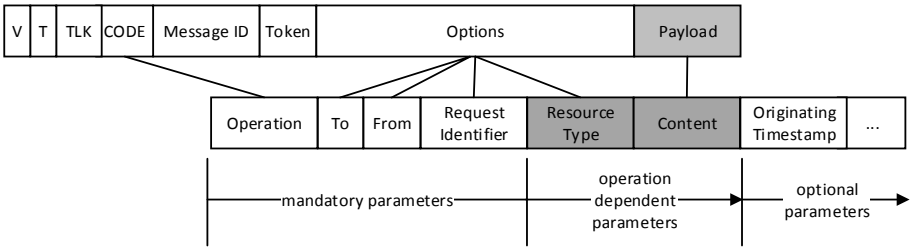
Ví dụ

Dưới đây là ví dụ về cặp bản tin HTTP request/response và giải thích tương ứng với các thuộc tính trong Primitive.

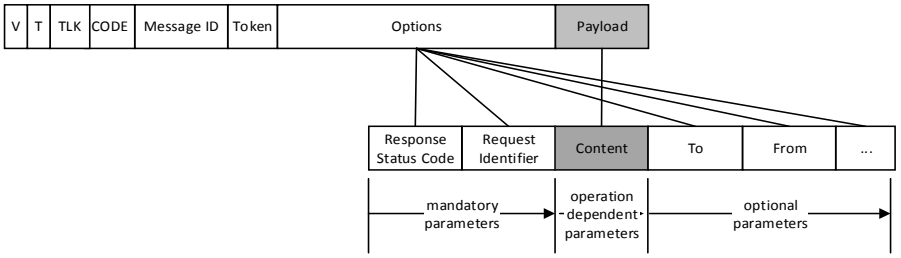
oneM2M Request Primitive:	**oneM2M short names**
HTTP/1.1	
Method: POST	➔ op : Operation
URI: m2msp1.com/CSE01Base	➔ to : To
URI Query String: ?rcn=1	➔ rcn : Result content
From: ae01.com	➔ fr : From
X-M2M-RI:0001	➔ rqi : Request identifier
X-M2M-RVI: 2a	➔ rvi : Release Version Indicator
Content: <AE> representation	➔ pc: primitive content
oneM2M Response Primitive:	
Status: Created	➔ rsc :Response Status Code
Location: http//m2msp1.com/CSE01Base/ae01	➔ uri : URI
X-M2M-RI:0001	➔ rqi : Request identifier
Content: <AE> representation created	➔ pc: primitive content

Ánh xạ kiến trúc bản tin CoAP

Việc ánh xạ từ bản CoAP sang Primitive tương tự HTTP. Hình PL.3 và PL.4 mô tả chi tiết sự tương ứng các trường thông tin. Các trường Options, Payload trong CoAP gần giống với Header. Message Body trong HTTP.



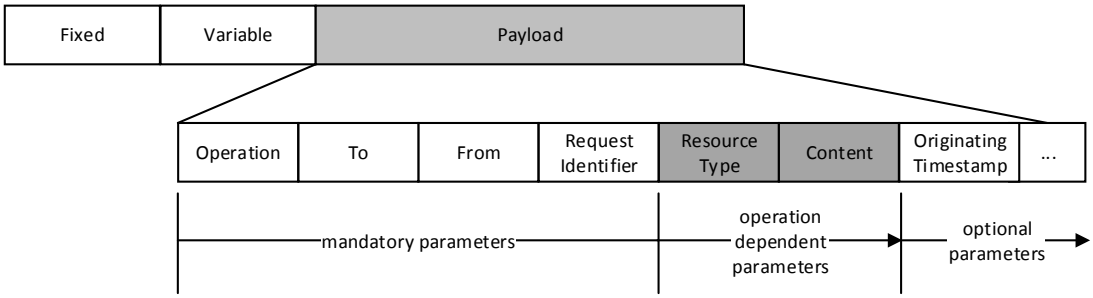
Hình PL.3 Ánh xạ bản tin CoAP request và Primitive request



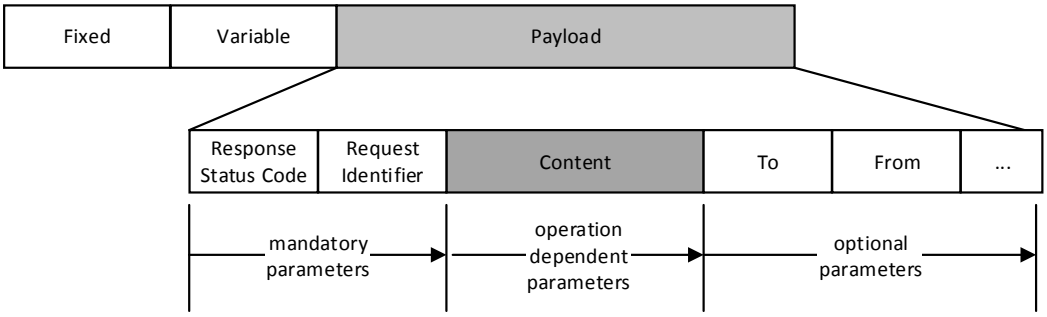
Hình PL.4 Ánh xạ bản tin CoAP response và Primitive response

Ánh xạ kiến trúc bản tin MQTT

Đối với MQTT, tất cả bản tin Primitive được ánh xạ trọn vẹn vào trong Payload như mô tả trong Hình PL.5 và Hình PL.6



Hình PL.5 Ánh xạ bản tin MQTT và Primitive request



Hình PL.6 Ánh xạ bản tin MQTT và Primitive response

Ví dụ về payload của một cặp bản tin Request/Response dưới dạng JSON mô tả trong hình dưới đây:

MQTT payload of request

```
{"m2m:rqp" : {
        "m2m:from" : "admin:admin",
        "m2m:to" : "/in-cse/in-name",
        "m2m:op" : "1",
        "m2m:ty" : "2",
        "m2m:pc" : {
        "m2m:ae" : {
                "rn" : "AE_SENSOR",
                "api" : "test-sensor",
                "rr" : "true"
        }
        }
}
}
```

MQTT payload of response

```
{"m2m:rsp" : {
        "m2m:from" : "admin:admin",
        "m2m:to" : "/in-cse/in-name",
        "m2m:rsc" : "2000",
        "m2m:pc" : {
        "m2m:ae" : {
                "rn" : "AE_SENSOR",
                "ty" : "2",
                ".....",
                "rr" : "true"
        }
        }
}
}
```

Hình PL.7 Ví dụ về MQTT payload