TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

———————— * ————————

ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC

NGÀNH CÔNG NGHỆ THÔNG TIN

ĐỀ XUẤT MỘT MÔ HÌNH GIÁM SÁT DỮ LIỆU TÍCH HỢP CHO HỆ THỐNG INTERNET OF THINGS

Sinh viên thực hiện: **Phan Công Huân**
Lớp CNTT-TT2.03 – K57
Giáo viên hướng dẫn: TS. **Nguyễn Bình Minh**

HÀ NỘI 5-2017

PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

1. Thông tin về sinh viên

Họ và tên sinh viên: Phan Công Huân

Điện thoại liên lạc: 01632305708 Email: nauh94@gmail.com

Lớp: CNTT 2.03 - K57 Hệ đào tạo: Kỹ sư

Đồ án tốt nghiệp được thực hiện tại: Bộ môn hệ thống thông tin, Viện Công nghệ thông tin và truyền thông, Đại học Bách Khoa Hà Nội.

Thời gian làm ĐATN: Từ ngày 16/01/2017 đến 26/05/2017

2. Mục đích nội dung của ĐATN

Đề xuất và xây dựng một mô hình giám sát các thành phần trong một hệ thống Internet of Things gồm nhiều tầng tính toán.

3. Các nhiệm vụ cụ thể của ĐATN

- Đề xuất và xây dựng một mô hình giám sát tài nguyên, thu thập dữ liệu cảm biến trong một hệ thống Internet of Things.
- Cài đặt và thử nghiệm khả năng hoạt động của mô hình đề xuất.

4. Lời cam đoan của sinh viên:

Tôi - *Phan Công Huân* - cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của *T.S Nguyễn Bình Minh.*

Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

Hà Nội, ngày tháng năm

Tác giả ĐATN

Phan Công Huân

5. Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành của ĐATN và cho phép bảo vệ:

Hà Nội, ngày tháng năm

T.S Nguyễn Bình Minh

TÓM TẮT NỘI DUNG ĐỒ ÁN TỐT NGHIỆP

Một hệ thống "vạn vật kết nối Internet" (Internet of Things) không chỉ đơn giản là các thiết bị thông minh kết nối với nhau. Hệ thống đó còn phải bao gồm các thiết bị, các thành phần phần cứng hay thậm chí là các đối tượng số hóa như các chương trình, các phần mềm, các dịch vụ nhằm lưu trữ, tính toán và phân tích lượng dữ liệu sản sinh từ vạn vật được kết nối đó. Điều đó được thể hiện qua một viễn cảnh thực tế về các toà nhà thông minh sẽ bao gồm rất nhiều các thiết bị, đối tượng hoạt động được kết nối với nhau như các cảm biến nhiệt độ, cảm biến báo cháy, bóng đèn, quạt, bộ phát sóng Wifi, network switch, các chương trình điều khiển, theo dõi các thiết bị thông minh chạy tại các máy chủ trong tòa nhà, các trung tâm "điện toán đám mây"... Tất cả các thành phần đó đều có thể sinh ra dữ liệu và thực hiện trao đổi dữ liệu với thành phần khác trong quá trình hoạt động.

Với sự tham gia của nhiều thành phần đa dạng như vậy trong một hệ thống "vạn vật kết nối Internet", nhu cầu cần phải giám sát, quản lý và khai thác các loại dữ liệu đa dạng thu thập được từ các mắt xích một cách tập trung, nhất quán là rất quan trọng. Có thể nói rằng trong xu hướng công nghệ hiện tại và tương lai, dữ liệu chính là một trong nguồn tài nguyên quý giá. Vì vậy, để có thể thu thập và khai thác những nguồn dữ liệu đó phục vụ cho các bài toán phân tích, xử lý, cần xây dựng một hệ thống giám sát và thu thập. Đã tồn tại những mô hình để theo dõi dữ liệu tài nguyên trong môi trường "điện toán đám mây", nhưng chưa có một mô hình cho việc giám sát các tài nguyên và dữ liệu trong một hệ thống "vạn vật kết nối Internet", với đặc trưng là bao gồm rất nhiều mô hình và kiểu loại đa dạng.

Mục tiêu của đồ án là đề xuất một mô hình giám sát và thu thập dữ liệu về tài nguyên tiêu dùng và dữ liệu cảm biến – hai loại dữ liệu chủ đạo trong một hệ thống "vạn vật kết nối Internet", trong bối cảnh đa dạng về đối tượng giám sát và kiểu dữ liệu cần giám sát. Các thử nghiệm được thực hiện đã cho thấy đồ án đã đáp ứng được mục tiêu đề ra.

LỜI CẢM ƠN

Đồ án tốt nghiệp là sản phẩm cuối cùng và tâm huyết nhất của một người sinh viên trong suốt thời gian học tập tại mái trường. Sản phẩm này là thành quả đúc kết của cả một quá trình học tập rèn luyện, và là thước đo đánh giá kiến thúc tiếp thu được của mỗi người sinh viên trong hành trang tiến bước vào sự nghiệp sắp tới. Vì vậy đồ án tốt nghiệp là một sản phẩm rất quan trọng đối với cá nhân mỗi người sinh viên đại học.

Để có thể hoàn thành được đồ án này, ngoài nỗ lực của bản thân, điều đầu tiên, tôi xin trân trọng cảm ơn TS Nguyễn Bình Minh – Viện Công nghệ thông tin và truyền thông, trường đại học Bách khoa Hà Nội, đã trực tiếp hướng dẫn và chỉ bảo tôi tận tình trong suốt quá trình học tập và thực hiện đồ án. Tôi cũng xin trân trọng cảm ơn quý thầy cô giáo ở Viện công nghệ thông tin cùng với tâm huyết của mình, đã truyền đạt kiến thức, chỉ bảo tôi trong suốt 5 học tại trường.

Tôi xin chân thành cảm ơn trung tâm tính toán hiệu năng cao đã hỗ trợ tôi về môi trường và các điều kiện cần thiết để nghiên cứu và thực hiện đề tài này. Xin chân thành cảm ơn anh Hà Quang Dương, anh Trần Hữu Cường, anh Nguyễn Bá Cường, bạn Bùi Ngọc Luân cùng các anh chị nhân viên, các bạn trong trung tâm đã giúp đỡ, định hướng và hỗ trợ về kiến thức cho tôi trong quá trình thực hiện đồ án.

Cuối cùng, con xin gửi lời cảm ơn tới gia đình đã luôn ở bên, động viên và giúp đỡ con hoàn thành đồ án tốt nghiệp này.

MỤC LỤC

BẢNG THUẬT NGỮ VIẾT TẮT

Viết tắt	Diễn giải
IoT	Internet of Things (vạn vật kết nối)
ĐTĐM	Điện toán đám mây
ĐTSM	Điện toán sương mù

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

Chương 1: Mở đầu

Mạng lưới vạn vật kết nối Internet, viết tắt là IoT (Internet of Things), là một viễn cảnh về thế giới, khi mà mỗi đồ vật, con người được cung cấp một định danh của riêng mình, và tất cả có khả năng truyền tải, trao đổi thông tin, dữ liệu qua một mạng kết nối duy nhất. Khái niệm IoT được ra đời từ năm 1999, nhưng trong những năm gần đây, đi kèm với sự phát triển vượt bậc của khoa học công nghệ, xu hướng kết nối các hệ thống ảo, các thực thể, các thiết bị với các hệ thống Internet phát triển rất mạnh mẽ. Xu hướng này có tác động không nhỏ đến nền kinh tế thế giới, cũng như cách mà con người đang làm việc. Thế giới đã phải công nhận và sẵn sàng chuẩn bị cho một *cách mạng công nghiệp lần thứ 4 đang bắt đầu*.

Một xu hướng phát triển cũng rất mạnh mẽ trong những năm qua là mô hình điện toán đám mây. Mặc dù điện toán đám mây chỉ là một cách khác để cung cấp tài nguyên máy tính, không phải là một công nghệ mới, nhưng nó đã châm ngòi cho một cuộc cách mạng trong cách cung cấp thông tinh và dịch vụ của các tổ chức. Sự linh hoạt về tài nguyên sử dụng là một chức năng to lớn đem lại sự phát triển vượt trội của "điện toán đám mây", có thế nói rằng giờ đây tài nguyên máy tính cần thiết cho những tác vụ sử dụng là gần như vô hạn chứ không còn bị bó hẹp trong phạm vi tài nguyên của một cỗ máy vi tính nào đó nữa.

Có một điều không thể phủ nhận, các hệ thống IoT và "điện toán đám mây" (ĐTĐM) ngày nay đã kết hợp gắn bó chặt chẽ với nhau. Với lượng dữ liệu khổng lồ được sản sinh liên tục từ các thiết bị IoT, những yêu cầu mới về khả năng truyền dẫn, tính toán và lưu trữ đã đặt ra các thử thách cho các mô hình "điện toán đám mây" hiện hành. Từ các vấn đề gặp phải khi cần xử lý, lưu trữ lượng dữ liệu khổng lồ đó, một mô hình điện toán mới đã được đề xuất: "điện toán sương mù". Mô hình tính toán mới này, là sự mở rộng của mô hình "điện toán đám mây" xuống các hạ tầng truyền dẫn, với những đặc tính chuyên biệt, đã đáp ứng được những ứng dụng quản lý và khai thác hệ thống IoT, hỗ trợ và giảm tải cho ĐTĐM trong các tác vụ xử lý cần đến thời gian thực.

Với sự tham gia của nhiều tầng tính toán như vậy trong một hệ thống "vạn vật kết nối Internet", nhu cầu cần phải giám sát, quản lý và khai thác các loại dữ liệu đa dạng thu thập được từ các mắt xích một cách tập trung, nhất quán là rất quan trọng. Có thể nói rằng trong xu hướng công nghệ hiện tại và tương lai, dữ liệu chính là một trong nguồn tài nguyên quý giá. Vì vậy, để có thể thu thập và khai thác những nguồn dữ liệu đó phục vụ cho các bài toán phân tích, xử lý, cần xây dựng một hệ thống giám sát và thu thập. Đã tồn tại những mô hình để theo dõi dữ liệu tài nguyên trong môi trường "điện toán đám mây", nhưng chưa có một mô hình cho

việc giám sát các tài nguyên và dữ liệu trong một hệ thống "vạn vật kết nối Internet", với đặc trưng là bao gồm rất nhiều mô hình và kiểu loại đa dạng như máy móc, thiết bị vật lý, các máy ảo, phần mềm, chương trình...

Mục tiêu của đồ án là đề xuất một mô hình giám sát và thu thập dữ liệu về tài nguyên tiêu dùng và dữ liệu cảm biến – hai loại dữ liệu chủ đạo trong một hệ thống "vạn vật kết nối Internet", trong bối cảnh đa dạng về đối tượng giám sát và kiểu dữ liệu cần giám sát. Các thử nghiệm được thực hiện đã cho thấy đồ án đã đáp ứng được mục tiêu đề ra.

Bố cục đồ án gồm 3 phần chính: **Nghiên cứu tổng quan**, **Đề xuất giải pháp** và **Ứng dụng giải pháp đề xuất vào thử nghiệm thực tế**.

Chương 2: Nghiên cứu tổng quan

2.1. Internet of Things (IoT)

2.1.1. Tổng quan

2.1.1.1. Định nghĩa

Internet of Things đại diện cho một sự xu hướng tương tác mới trong thế giới của chúng ta. Giống như cuộc cách mạng World Wide Web (cách mạng Internet) đã kết nối các hệ thống máy tính thành một mạng lưới, và sau đó là kết nối con người đến thế giới Internet, công nghệ Internet of Things là một viễn cảnh tất cả mọi thứ được kết nối với nhau và kết nối vào Internet: các thiết bị, con người, môi trường xung quanh, phần mềm và máy móc.

Mặc dù có rất nhiều định nghĩa không đồng nhất về Internet of Things, nhưng về cơ bản các định nghĩa đều đề cập đến khả năng tích hợp thế giới vật chất vào không gian số hóa của Internet.

- Theo [1], Mạng lưới vạn vật kết nối Internet hoặc là Mạng lưới thiết bị kết nối Internet viết tắt là IoT (Internet of Things) là một viễn cảnh về thế giới, khi mà mỗi đồ vật, con người được cung cấp một định danh của riêng mình, và tất cả có khả năng truyền tải, trao đổi thông tin, dữ liệu qua một mạng duy nhất mà không cần đến sự tương tác trực tiếp giữa người với người, hay người với máy tính. Nói đơn giản, IoT là một tập hợp các thiết bị có khả năng kết nối với nhau, với Internet và với thế giới bên ngoài để thực hiện một công việc nào đó.

- Theo [2], IoT có thể được định nghĩa rộng rãi như là một cơ sở hạ tầng mạng toàn cầu, liên kết các vật thể được định danh duy nhất, các đối tượng ảo, các thiết bị thông qua các đối tượng thông minh, khả năng truyền thông và truyền động (actuation). Nói cách khác, mô hình của IoT được mô tả là "kết nối mọi lúc, mọi nơi và mọi thứ". Theo quan điểm của "Things", rất nhiều thiết bị và đồ vật sẽ được kết nối với Internet. Mỗi cá thể sẽ cung cấp dữ liệu, thông tin hoặc thậm chí cả dịch vụ. Các "Things" có thể là các thiết bị cá nhân mà chúng ta mang theo như điện thoại thông minh, máy tính bảng và máy ảnh kỹ thuật số. Môi trường, nhà riêng, phương tiện giao thông hay các văn phòng kết nối thông qua các thiết bị gateway cũng có thể coi là các "Things". Các thiết bị như cảm biến, vật thông minh và thiết bị truyền động (actuator) cũng có thể coi là các "Things".

Tóm lại, theo quan điểm về kết nối và tương tác, IoT nói đến việc tất các thiết bị có thể kết nối với nhau và kết nối đến mạng Internet, bằng các loại hình kết nối đa dạng như Wi-Fi, mạng viễn thông (3G, 4G), Bluetooth, hồng ngoại..., và trao đổi dữ liệu nhằm thực hiện một công việc nào đó. Vì vậy điểm mấu chốt ở đây là các thiết bị phải có khả năng kết nối trực tiếp hoặc gián tiếp với nhau thông qua một loại hình kết nối nào đó, nếu không nó sẽ không thể được coi là một thiết bị IoT.

Các thiết bị IoT sẽ rất đa dạng, tồn tại xung quanh cuộc sống con người, có thể kể đến như điện thoại thông minh, máy giặt, điều hòa, bóng đèn, đồng hồ, quạt điện, cảm biến nhiệt độ, đồng hồ thông minh ... và còn nhiều thiết bị khác nữa mà có thể kết nối đến Internet.

2.1.1.2. Ứng dụng

Tiềm năng ứng dụng của IoT là vô cùng to lớn. Trong tương lai, công nghệ IoT sẽ tham gia vào mọi lĩnh vực của cuộc sống. Với IoT, cách con người sống, trải nghiệm, tương tác với thế giới sẽ có nhiều thay đổi. Cuộc sống của con người có thể trở nên dễ dàng hơn, thoải mái hơn và tốt đẹp hơn.

Mặc dù vẫn đang ở trong thời kì đầu phát triển của công nghệ IoT, chúng ta đã có thể thấy được sự hiện hữu của công nghệ này trong nhiều mặt cuộc sống như giao thông, chăm sóc sức khỏe, nhà thông minh, nông nghiệp thông minh... Có thể lấy ví dụ như trong lĩnh vực y tế, thiết bị IoT có thể được sử dụng để theo dõi sức khỏe từ xa kết hợp với thông báo khẩn cấp. Các thiết bị theo dõi này có thể là vòng đeo tay có khả năng theo dõi huyết áp, nhịp tim, và cung cấp các thông tin này cho một thiết bị xử lý như điện thoại thông minh nhằm phác thảo những biểu đồ, gợi ý cho sức khỏe. Một ví dụ khác như các thiết bị có khả năng cảm ứng như bóng đèn, cảm biến nhiệt độ được lắp đặt trong một ngôi nhà và sẽ gửi dữ liệu về thiết bị xử lý chính để đưa ra các hành động điều khiển như tắt bật bóng đèn vào buổi tối, bật tắt quạt khi trời nóng… Trong công nghiệp sản xuất, các nhà máy lắp ráp với hệ thống robot được điều khiển bởi trung tâm máy tính sẽ mang tới độ chính xác và năng suất lao động cao gấp nhiều lần so với con người.

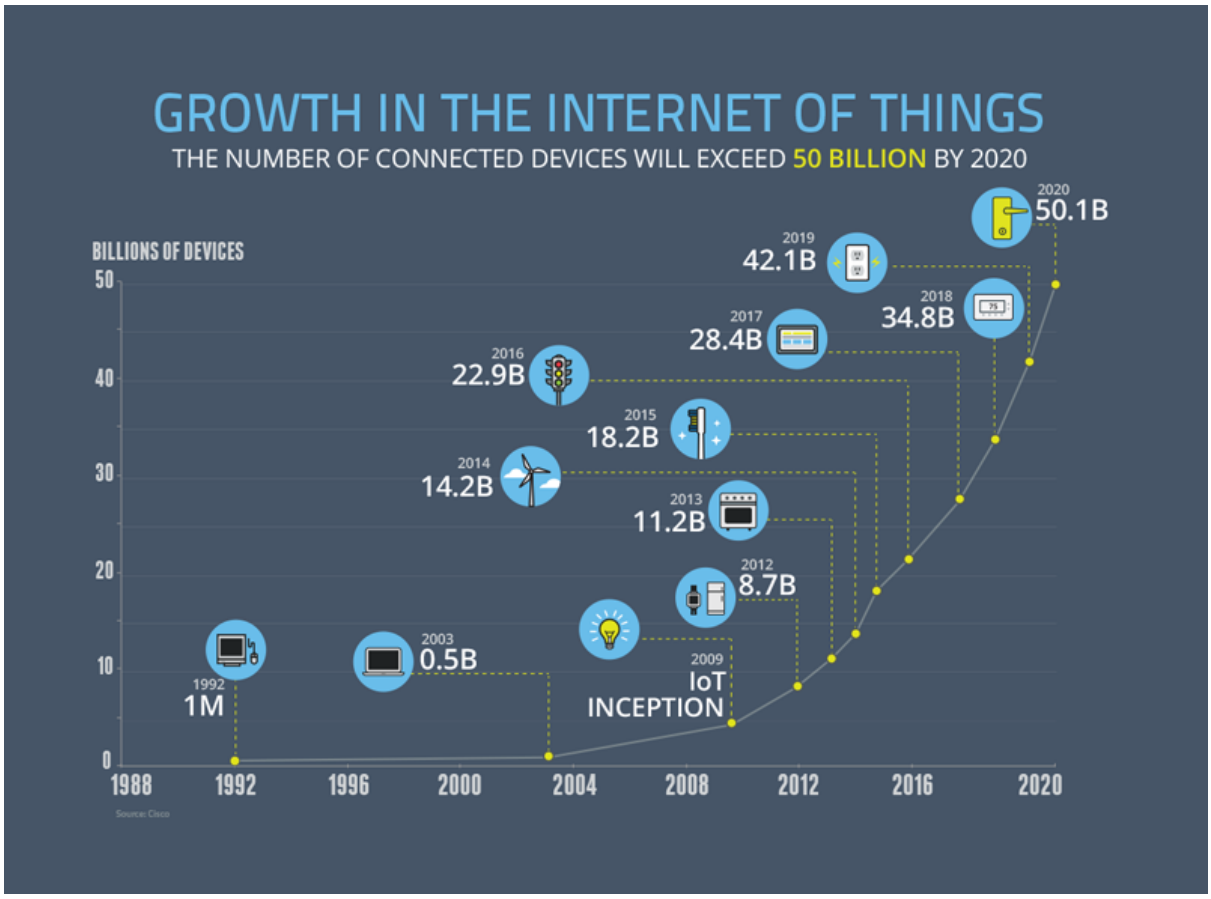
Mặt khác, hệ thống IoT cũng có thể thực hiện các hành động, không chỉ cảm nhận mọi thứ xung quanh. Hệ thống mua sắm thông minh, có thể theo dõi thói quen mua hàng của người dùng ở một cửa hàng bằng cách theo dõi điện thoại di động của họ. Người dùng sau đó có thể được gửi các cập nhật về sản phẩm yêu thích, vị trí các đồ dùng họ cần, hay gợi ý về một sản phẩm có liên quan ngay tại cửa hàng đó. Tất cả đã được tự động chuyển vào điện thoại của người dùng.

2.1.1.3. Internet of Things là xu hướng công nghệ của tương lai

Mặc dù đã đã xuất hiện từ lâu nhưng kỷ nguyên Internet of Things chỉ thực sự được sự được chú ý và bùng nổ trong những năm gần đây, sau sự phát triển mạnh mẽ của các thiết bị di động thông minh, máy tính bảng, các thiết bị cá nhân và sự phát triển của các công nghệ kết nối không dây.

Để có thể thấy được sự phát triển của lĩnh vực này, Cisco đã đưa ra số liệu vào năm 1984, khi mới thành thập, thì chỉ có khoảng 1.000 thiết bị được kết nối mạng toàn cầu, đến năm 2010, con số này đã tăng lên mức 10 tỷ, và theo dự đoán, đến năm 2020, con số có thể lên đến 50 tỷ thiết bị. Theo Gartner, Inc. (một công ty nghiên cứu và tư vấn công nghệ), sẽ có gần 26 tỷ thiết bị trên IoT vào năm 2020.[3].

ABI Research ước tính rằng hơn 30 tỷ thiết bị sẽ được kết nối không dây với "Kết nối mọi thứ" (Internet of Everything) vào năm 2020[4].



Hình 1 Sự phát triển của IoT (nguồn[1])

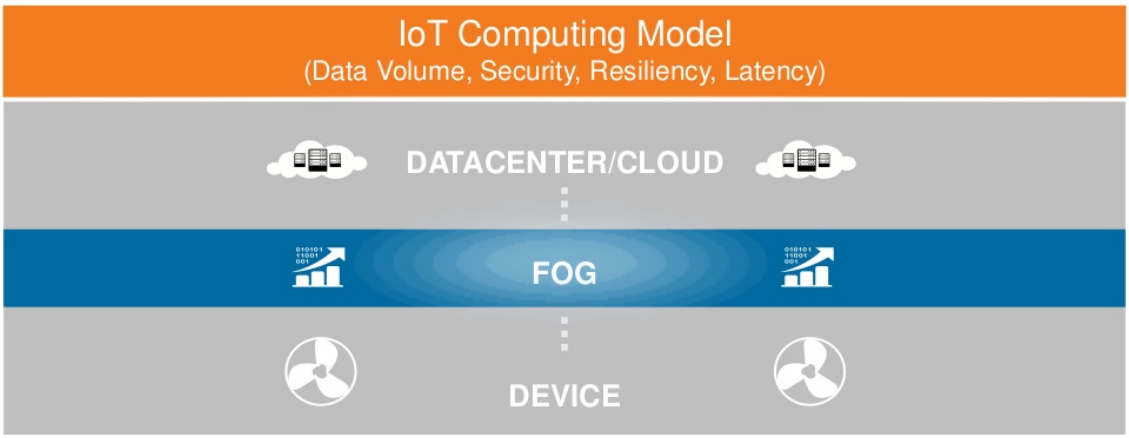
IoT sẽ bao gồm rất lớn các thiết bị kết nối với nhau. Các ông lớn như Intel, Cisco, Qualcomm đang đầu tư rất nhiều tiền để phát triển các công nghệ cho thiết bị IoT. Như vậy có thể nói, các thiết bị có khả năng kết nối Internet chính là xu hướng của tương lai.

2.1.2. Mô hình kiến trúc 3 tầng của hệ thống IoT

Sự phát triển mạnh mẽ của công nghệ IoT đã sản sinh ra một lượng lớn dữ liệu cần lưu trữ và xử lý. Mô hình đẩy dữ liệu lên tầng Cloud để phân tích và xử lý đã tỏ ra không còn phù hợp trong bối cảnh bùng nổ dữ liệu như hiện tại. Thay vì đẩy một khối lượng khổng lồ dữ liệu lên Cloud, điều này có thể gây tiêu tốn chi phí truyền tải, quá tải băng thông mạng, không đảm bảo tính bảo mật dữ liệu, các chuyên gia của Cisco đã đề xuất một mô hình tính toán mới, có tên gọi là Fog Computing (điện toán sương mù), nhằm giải quyết các vấn đề trên [5].

Mô hình của một hệ thống IoT khi đó có thể được chia một cách cơ bản thành 3 tầng như sau [5]:

[1] https://www.ncta.com/platform/broadband-internet/behind-the-numbers-growth-in-the-internet-of-things/



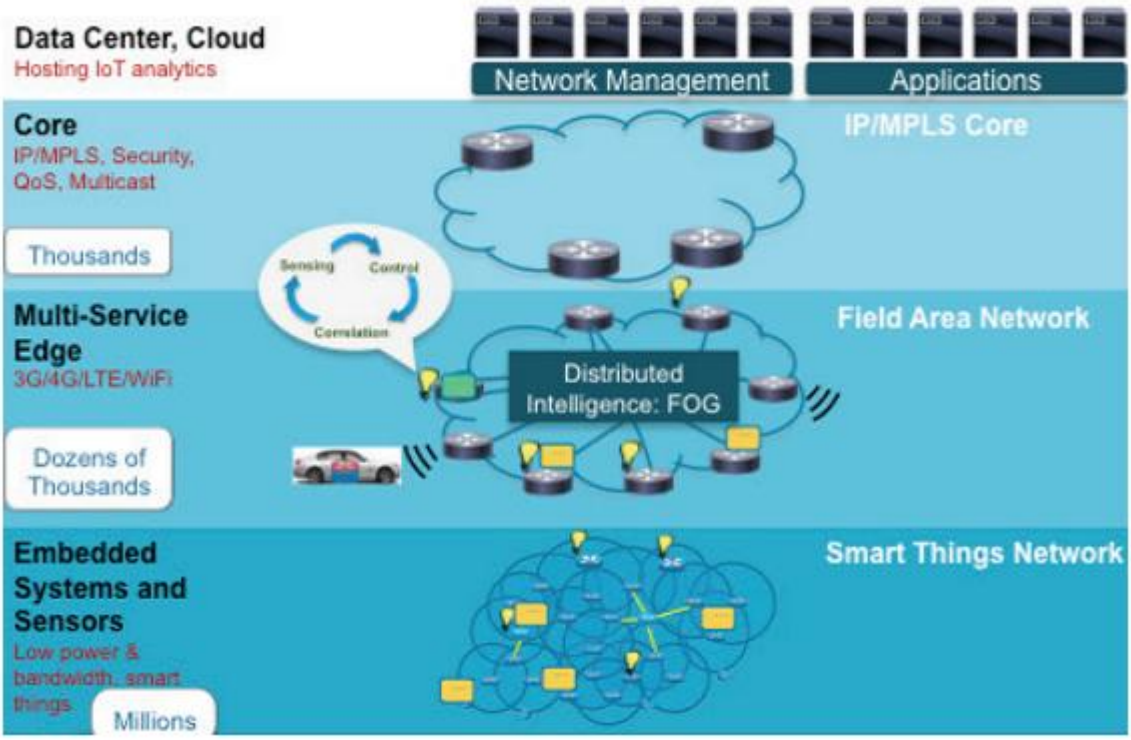
Hình 2 Mô hình kiến trúc 3 tầng của hệ thống IoT (nguồn[2])

- Tầng dưới cùng là tầng "**thiết bị**" (Device), đây là nơi tập hợp hàng trăm nghìn các vật thể như hệ thống nhúng, các thiết bị cảm ứng, hay các mạch tích hợp có khả năng kết nối với nhau, thu thập dữ liệu về môi trường xung quanh và có thể thực hiện một số tương tác với môi trường. Tầng **"**thiết bị**"** sẽ đẩy dữ liệu thu thập được lên tầng "sương mù" (Fog), nơi có chứa các thiết bị, các nút có khả năng tính toán đã được thiết lập để thu nhận luồng dữ liệu đó.
- Tầng **"sương mù"**, đúng theo ý nghĩa khoa học của nó, sẽ là tầng trung gian giữa tầng "đám mây" và tầng "mặt đất" (tầng "thiết bị"). Đây là nơi chứa các cơ sở hạ tầng truyền dẫn, phục vụ trao đổi dữ liệu qua lại giữa tầng 2 tầng "**đám mây**" và tầng "thiết bị". Bên cạnh đó, một số nút có khả năng tính toán nhỏ (máy chủ nhỏ, laptop, smartphone ...) cũng bao gồm trong tầng này. Mặt khác, đúng như tính chất sương mù, gần mặt đất, phân tán, mờ ảo, hỗn độn, tầng này sẽ được triển khai ở vị trí rất gần với tầng "thiết bị", bên cạnh đó là sự đa dạng về chủng loại, cấu trúc, đặc điểm, giao thức kết nối mạng của các thiết bị, đối tượng trong tầng này. Số lượng các thiết bị tầng này sẽ ít hơn tầng "thiết bị" và sẽ có một số tiêu chuẩn ràng buộc chặt chẽ hơn như các giao thức mạng mà các thiết bị sẽ tuân theo.
    - Các thành phần tầng "sương mù" sẽ được triển khai rất gần tầng "thiết bị", nhằm giảm thiểu hóa chi phí truyền dẫn giữa các tầng (độ trễ băng thông), sử dụng tối ưu hóa tài nguyên tính toán của các nút, và giảm thiểu chi phí tài nguyên cho các dịch vụ triển khai trên tầng "đám mây". Tại tầng "sương mù", dữ liệu có thể được tiền xử lý để tiếp tục đẩy lên tầng "đám mây", hoặc trực tiếp ra lệnh điều khiển các thành phần trong tầng "thiết bị".

[2] https://www.slideshare.net/Cisco/enabling-the-internet-of-everything-ciscos-iot-architecture

    - Có thể hình dung một cách rõ ràng rằng số lượng các thiết bị ở tầng "thiết bị" sẽ là rất lớn, có thể lên đến hàng nghìn, thậm chí hàng triệu, và do đó, lượng dữ liệu sản sinh ra là cực kỳ lớn. Với lượng dữ liệu lớn và sản sinh liên tục như vậy, khả năng đáp ứng của băng thông sẽ vượt quá mức giới hạn, chi phí truyền tải sẽ là rất lớn. Vì vậy cần phải phân tán tài nguyên tính toán ra các thiết bị tầng "sương mù", có số lượng lớn, gần hơn với những nguồn sản sinh giữ liệu để tiến hành tiền xử lý, tổng hợp, chắt lọc theo thời gian thực rồi mới gửi lên tầng "đám mây", thường là các trung tâm tính toán tập trung, có thể phân bố ở rất xa. Điều này giúp khai thác tối đa khả năng của từng tầng trong mô hình.
- Tầng **"đám mây"**(cloud) sẽ là các trung tâm dữ liệu lớn, các cụm máy chủ cấu hình mạnh, tập trung theo từng cụm và nằm rải rác khắp nơi trên thế giới. Số lượng các thiết bị, đối tượng ở tầng này sẽ ít hơn nhiều 2 tầng còn lại, bên cạnh đó là mô hình kiến trúc chặt chẽ, tuân theo các tiêu chuẩn điện toán. Có thể coi tài nguyên tính toán ở tầng này là gần như vô hạn. Ở trên tầng đám mây với khả năng tài nguyên dồi dào, các ứng dụng phân tích, xử lý liệu sẽ được triển khai nhằm phân tích một lượng dữ liệu cực lớn thu được tầng "thiết ị" và tầng "sương mù". Đồng thời đây cũng là nơi ra các quyết định điều khiển cho các tầng bên dưới và là nơi triển khai của các kho lưu trữ.

Tùy theo từng hệ thống và bài toán cần giải quyết riêng, tầng "sương mù" trong hệ thống IoT còn có thể được chia thành nhiều tầng hơn, nhằm nhấn mạnh một số đặc điểm cốt lõi của hệ thống[6].



Hình 3 Mô hình ĐTSM mở rộng (nguồn [6])

2.1.2.1. Tầng Cloud: tổng quan về điện toán đám mây

Theo định nghĩa của Viện tiêu chuẩn và Công nghệ quốc gia Mỹ [8], ĐTĐM là mô hình cho phép truy cập mạng để lựa chọn, sử dụng chung nguồn tài nguyên tính toán và có thể tùy chỉnh được (ví dụ như mạng, máy chủ, bộ nhớ lưu trữ, các ứng dụng và các dịch vụ) một cách thuận tiện, theo nhu cầu sử dụng, đồng thời cho phép cung cấp và giải phóng chúng một cách nhanh chóng, giảm thiểu tối đa sự quản lý hoặc thao tác của nhà cung cấp dịch vụ. Với các dịch vụ sẵn có trên Internet, doanh nghiệp cần không phải mua và duy trì hệ thống máy tính cũng như phần mềm, mà họ chỉ cần tập trung vào kinh doanh lĩnh vực riêng của mình bởi đã có các nhà cung cấp dịch vụ lo cơ sở hạ tầng và công nghệ thay họ.

Đa số người dùng Internet hiện nay đã và đang sử dụng những dịch vụ ĐTĐM phổ thông như email, album ảnh và bản đồ số.

Tính linh hoạt của ĐTĐM là cung cấp tài nguyên công nghệ thông tin theo nhu cầu sử dụng. Điều này tạo sự mềm dẻo, thuận lợi cho việc sử dụng tài nguyên tính toán, loại bỏ ràng buộc phải đầu tư cơ sở hạ tầng phần cứng cụ thể cho một nhiệm vụ. Trước khi có ĐTĐM, các trang web hoặc các ứng dụng được chạy trong một máy chủ cụ thể hoạt động trong một hệ thống. Với sự ra đời của ĐTĐM, các tài nguyên được hợp nhất và sử dụng như một kho chung. Cấu hình hợp nhất này cung cấp một môi trường mà ở đó các ứng dụng thực hiện một cách độc lập mà không quan tâm đến cấu hình cụ thể nào [20].

2.1.2.2. Tầng Fog: Tổng quan về điện toán sương mù (Fog computing)

Mô hình dịch vụ ĐTĐM "dùng bao nhiêu trả bấy nhiêu" là một phương án hiệu quả để sử dụng và quản lý các trung tâm dữ liệu cho các ứng dụng dựa trên nền Web và xử lý song song. Mô hình này càng tỏ rõ sự quan trọng trong thời đại bùng nổ của về lượng dữ liệu với các bài toán phân tích dữ liệu lớn.

ĐTĐM đã giải phóng các doanh nghiệp và người dùng cơ bản khỏi những rối rắm và phức tạp khi tự triển khai một hệ thống trung tâm dữ liệu. Tuy nhiên mô hình này gặp phải những vấn đề liên quan đến thời gian truyền tải và thời gian xử lý, đặc biệt với những ứng dụng yêu cầu độ trễ thấp. Bên cạnh đó sự phát triển mạnh mẽ của công nghệ IoT, với một số đặc tính như tính di động cao, phân bố theo địa lý với số lượng lớn, yêu cầu độ trễ thấp và khả năng tính toán phân tán đã đặt ra nhiều vấn đề mới cho mô hình ĐTĐM.

Từ những thách thức cần giải quyết, một mô hình điện toán mới, có tên gọi là "Điện toán sương mù" (Fog computing), được đề xuất nhằm giải quyết các vấn đề đó. "Điện toán sương mù" (ĐTSM) là sự mở rộng của mô hình ĐTĐM lên cơ sở hạ tầng truyền dẫn, nơi gần hơn với các thiết bị Things. ĐTSM một cách tự nhiên, ra đời là để đáp ứng sự phát triển mạnh mẽ của công nghệ IoT, đem đến xu hướng phát triển mới cho các ứng dụng và dịch vụ, khi những tác vụ xử lý, phân tích yêu cầu thời gian thực triển khai ngay gần các nút thu thập dữ liệu, và những tác vụ khai phá, phân tích xu hướng dài hạn (long term) sẽ được thực hiện trên tầng "đám mây" [6].

Đặc trưng của mô hình "Điện toán sương mù"

ĐTSM là một nền tảng ảo hóa mức cao cung cấp khả năng tính toán, khả năng lưu trữ, tài nguyên mạng (networking) giữa các thiết bị đầu cuối và các thiết bị truyền dẫn, còn gọi là các nút Fog, trong một mạng lưới kết nối với các trung tâm ĐTĐM.

Theo [6], các đặc trưng của tầng "sương mù":

- Là các nút mạng, việc truyền tải giữa các nút có độ trễ thấp, các nút có khả năng nhận biết vị trí trong mạng lưới kết nối.
- Các nút Fog phân tán rộng trong không gian địa lý.
- Bao gồm số lượng rất lớn các nút Fog, do tính chất phân tán trong không gian
- Công nghệ kết nối chủ đạo là mạng không dây.
- Có các tác vụ cần sự tương thác theo thời gian thực.
- Tính đa dạng, không đồng nhất, do các nút thiết bị được sản xuất theo nhiều tiêu chuẩn khác nhau, và triển khai trong các hệ sinh thái đa dạng.
- Hỗ trợ phân tích theo thời gian thực và cộng tác với tầng "đám mây", các nút tầng "sương mù" đóng vai trò xử lý dữ liệu gần với nguồn sản sinh dữ liệu.

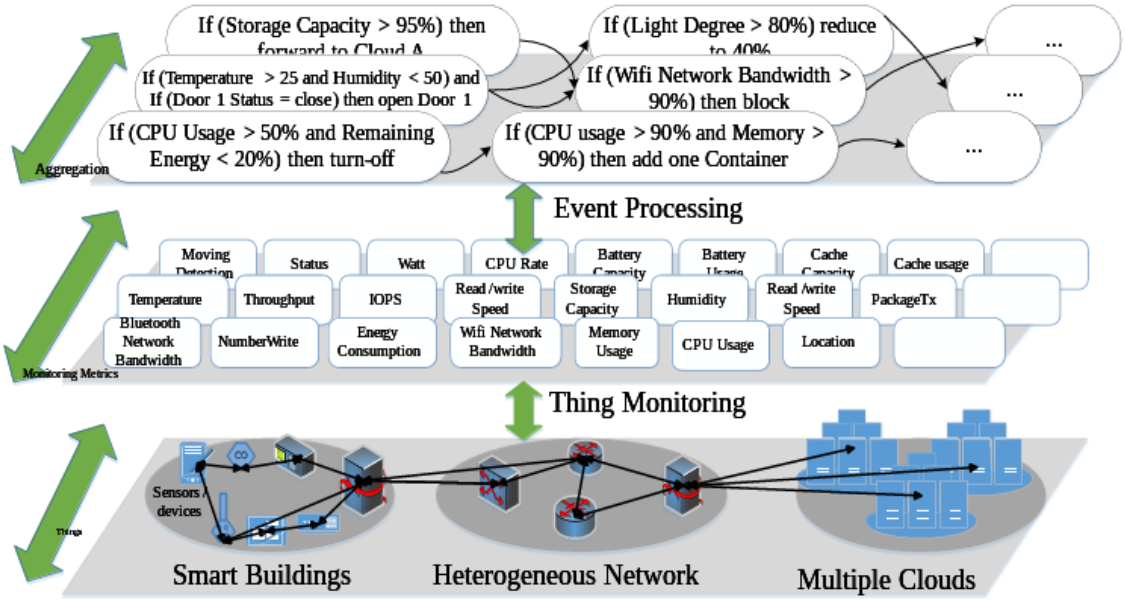
2.2. Bài toán giám sát (monitoring)

2.2.1. Bối cảnh áp dụng

Lấy ví dụ một kịch bản về các tòa nhà thông minh trong thực tế áp dụng kiến trúc 3 tầng IoT. Kịch bản này gồm các đặc điểm:

- Có rất nhiều loại thiết bị như cảm biến môi trường, nhiệt độ, ánh sáng, các thiết bị điện tử, đèn, điều hòa, quạt và còn nhiều nữa, có thể kết nối với nhau. Đa số các thiết bị chỉ có khả năng cảm nhận môi trường xung quanh và gửi/nhận dữ liệu tới/từ các thiết bị khác, mặc dù có một số thiết bị có khả năng tính toán và lưu trữ dữ liệu. Các tòa nhà có thể dùng các nền tảng công nghệ IoT (IoT platform) khác nhau để nhận diện, quản lý, điều khiển và thu nhận dữ liệu từ các thiết bị trên.
- Mạng kết nối là không đồng bộ. Một số thiết bị sử dụng mạng không dây, số khác sử dụng dây dẫn, và một số lượng lớn sử dụng kết nối đơn thuần như Bluetooth (các cảm biến có phạm vi truyền dẫn nhỏ) để gửi/ nhận dữ liệu. Tuy nhiên một đặc điểm chung là hầu hết các thiết bị cần đến một thiết bị trung gian để có thể chia sẻ dữ liệu và kết nối đến Internet (các gateway).
- Như đã trình bày ở phần kiến trúc 3 tầng, các tác vụ xử lý dữ liệu từ Things tiêu tốn nhiều tài nguyên tính toán sẽ được thực hiện trên các "đám mây", và đưa ra những quyết định điều khiển khi cần thiết. Luồng dữ liệu sẽ đi từ các Things, qua hạ tầng truyền dẫn (gateway, router, switch, network hub...) trong tầng "sương mù" để lên đến các dịch vụ trên tầng "đám mây". Tuy nhiên trong một số trường hợp, các tác vụ xử lý nhỏ có thể được thực hiện ngay tại các gateway ngay trong tòa nhà. Một ví dụ như đôi khi hành động tắt bóng đèn khi người dùng ra khỏi phòng sẽ được điều khiển bởi một gateway gần đó (trong cùng phòng với bóng đèn), mà không cần chờ xử lý từ tầng "đám mây".

2.2.2. Hệ thống IoT từ góc nhìn bài toán giám sát



Hình 4 Ví dụ về vấn đề giám sát tài nguyên một hệ thống IoT (nguồn[3])

Hình trên cho ta thấy một góc nhìn của bài toán giám sát một hệ thống IoT gồm 3 tầng. Mọi thiết bị vật lý, đối tượng ảo trong hệ thống IoT 3 tầng khi đó sẽ được gọi chung là Things (tầng dưới cùng trong hình), đều sản sinh dữ liệu và là đối tượng cần theo dõi của bài toán giám sát. Các đối tượng giám sát này có thể là:

- Các cụm máy chủ, máy ảo, các container, các dịch vụ, ứng dụng trên tầng "đám mây" có thể được theo dõi thông qua các thông số đo lượng CPU sử dụng, bộ nhớ sử dụng, dung lượng kho lưu trữ, băng thông mạng...
- Các thiết bị, các ứng dụng trong "tầng sương" mù như gateway, router, switch, máy chủ, các ứng dụng quản lý, bảo mật... có thể được theo dõi với các tham số tài nguyên như giao thức kết nối, trạng thái, dung lượng bộ nhớ, băng thông WIFI, Bluetooth, năng lượng tiêu dùng, CPU sử dụng, bộ nhớ sử dụng, dung lượng kho lưu trữ ...
- Tầng "thiết bị" với các cảm biến, thiết bị thông minh ũng là đối tượng cần theo dõi khi sản sinh ra các thông số đo khác nhau, dựa vào đặc điểm như trạng thái, dung lượng pin, điện năng sử dụng, nhiệt độ môi trường, độ ẩm không khí....

Các đối tượng đa dạng được giám sát như trên cho thấy sự đa dạng về kiểu dữ liệu, mô hình dữ liệu trong quá trình thu thập dữ liệu giám sát (tầng Monitoring Metrics trong hình 4).

2.2.3. Sự cần thiết của bài toán giám sát trong mô hình hệ thống 3 tầng của IoT.

Có 2 điều cho thấy sự quan trọng của bài toán giám sát trong một hệ thống IoT.

[3] Nguyễn, B.M.: Example of IoT systems and monitoring problems (2016)

Đầu tiên, đó là sự cần thiết phải theo dõi sự trao đổi dữ liệu giữa các Things theo thời gian thực, nhằm lựa chọn con đường tối ưu cho truyền dẫn dữ liệu. Một ví dụ thực tế với kịch bản tòa nhà thông minh, khi một cảm biến (ví dụ cảm biến nhiệt độ), gửi giá trị nhiệt độ lên "đám mây", dữ liệu này có thể được trung chuyển qua nhiều thiết bị khác nhau (đèn, qua đường truyền Bluetooth), một gateway cùng phòng, gateway của tầng (wifi router) qua đường truyền mạng không dây, rồi sau đó tập trung về gateway của cả tòa nhà (một máy chủ nhỏ) bằng đường truyền LAN, từ đó chuyển lên tầng "đám mây" qua đường truyền Internet. Sau khi xử lý, lệnh điều khiển từ tầng "đám mây" có thể được truyền trả lại cảm biến nhiệt độ theo một con đường khác. Câu hỏi là làm thế nào để chọn con đường tối ưu cho quá trình gửi và nhận dữ liệu qua lại giữa các Things và tầng "đám mây". Câu trả lời cho vấn đề đó phụ thuộc vào khả năng theo dõi trạng thái của tất cả các Things tham gia vào quá trình truyền nhận dữ liệu trong ví dụ, để từ đó tìm kiếm một giải pháp tối ưu. Bên cạnh đó, do tính chất đa dạng của dữ liệu thu thập được như đã trình bày ở phần 2.2.2, các đối tượng giám sát phải được theo dõi đồng thời nhằm cho một cái nhìn trên cùng một ngữ cảnh, thời điểm.

Điều thứ 2, dựa trên những dữ liệu giám sát thu thập được từ các Thing, những nhà quản trị sẽ mô tả những yêu cầu hoặc điều kiện để đảm bảo sự hoạt động ổn định của một hệ thống IoT. Có thể dẫn ra một số điều kiện như chạy một tác vụ phân tích trên một gateway nào đó nhằm tang cường xử lý khi lượng dữ liệu cảm biến tăng lên, lập lịch cho việc triển khai các tác vụ tính toán trên các nút tầng Fog hạn chế về tài nguyên, đảm bảo sử dụng tài nguyên một cách tiết kiệm và tối ưu… Các diễn giải sau có thể làm rõ hơn cho điều này:

- Tại tầng "sương mù", các nút sẽ chạy các ứng dụng, dịch vụ hoặc đơn giản là một tiến trình nhằm thu thập, tiền xử lý dữ liệu và trực tiếp điều khiển các sensor, actutor. Các nút (máy chủ, laptop, smartphone, RasberryPi, Adruino, mạch tích hợp) thường có tài nguyên tính toán hạn hẹp như bộ xử lý yếu, bộ nhớ trong và lượng điện năng tiêu thụ có giới hạn. Do vậy việc theo dõi lượng tài nguyên tiêu thụ, khả năng tính toán của các thiết bị này là rất cần thiết trong việc triển khai ứng dụng/dịch vụ, tránh tình trạng ứng dụng/dịch vụ yêu cầu lượng tài nguyên vượt quá khả năng đáp ứng của thiết bị, gây ra lỗi hệ thống. Bên cạnh đó, việc theo dõi lượng dữ liệu từ các cảm biến sẽ giúp hệ thống chủ động điều chỉnh việc thu thập, như tăng hoặc giảm tần suất lấy dữ liệu.
- Tại tầng "đám mây", sẽ có các ứng dụng, các dịch vụ được triển khai nhằm thực hiện quá trình xử lý, phân tích, lưu trữ dữ liệu thu được từ các Things. Bài toán giám sát được áp dụng nhằm phục vụ cho việc theo dõi, tối ưu hóa chi phí tài nguyên sử dụng trong khi vẫn đảm bảo tính ổn định, tự co dãn một cách linh hoạt của các ứng dụng/dịch vụ.

2.2.3.1. Một số ứng dụng của bài toán giám sát hệ thống IoT áp dụng vào thực tế

Các ví dụ sau cho thấy khả năng kết hợp giám sát, khai thác dữ liệu từ 3 tầng của hệ thống IoT nhằm giải quyết một bài toán nào đó:

- Tình huống khẩn cấp đám cháy xảy ra trong rạp chiếu phim, có rất nhiều nạn nhân và cần sự phối hợp phản ứng kịp thời của hệ thống phản ứng khẩn cấp: điều xe cấp cứu, xe chữa cháy, chọn khung đường ngắn nhất cho các phương tiện đến hiện trường, đồng thời thông báo cho bệnh viện gần nhất chuẩn bị tiếp đón nạn nhân.
    - Các nạn nhân sẽ được xác định danh tính và tình trạng sức khỏe hiện tại thông qua các thiết bị đeo tay. Hệ thống phản ứng khẩn cấp sẽ đưa ra các phương án thích hợp cho nhân viên cấp cứu trong quá trình đưa nạn nhân đến bệnh viện.
    - Mạng lưới cảm biến giao thông sẽ chọn con đường tối ưu nhất cho xe cứu thương, xe chữa cháy đến hiện trường nhanh nhất có thể.

  Tình huống trên sẽ yêu cầu sự tham gia của rất nhiều thiết bị IoT phối hợp trong khoảng thời gian ngắn. Để có thể thực hiện được các tác vụ tiếp nhận và xử lý dữ liệu trong thời gian ngắn một cách nhanh chóng, hệ thống cảnh báo cần có khả năng theo dõi khả năng của hệ thống để chủ động cung cấp tài nguyên linh hoạt ở tất cả các tầng cho việc thực hiện các tác vụ như: Thiết lập mạng lưới mạng con kết nối các phương tiện cấp cứu, bệnh viện, hệ thống cảnh báo, bác sỹ, thêm các nút xử lý trên các thiết bị gateway trong rạp phim nhằm có khả năng thu thập, xử lý nhiều dữ liệu hơn [9].
- Giao thông thông minh: tương tác và kết nối xe với xe, xe với điểm truy cập (gateway, nút mạng, đèn giao thông chẳng hạn), điểm truy cập với điểm truy cập. Đèn giao thông thông minh tại ngã tư sẽ tương tác và trao đổi dữ liệu cảm ứng với các cảm biến lắp đặt trên các làn đường, các cảm biến này sẽ phát hiện sự hiện diện của người đi bộ, xe máy, và đo đạc khoảng cách cùng với tốc độ của các phương tiện đang đi đến từ làn đường khác. Các đèn có thể tương tác với đèn xung quanh mình để điều chỉnh thời gian đèn chuyển xanh. Dựa trên những thông tin thu thập được, các cột đèn giao thông thông minh có thể gửi tín hiệu cảnh báo tới các phương tiện đang đi đến ngã tư, cùng với điều chỉnh thời gian trạng thái của đèn (xanh, đỏ, vàng) để phòng tránh tai nạn. Dữ liệu thu thập được bởi các cột đèn giao thông thông minh sẽ được phân tích theo thời gian thực nhằm đưa ra các một số quyết định trong ngữ cảnh cụ thể, ví dụ như thay đổi thời gian sáng của đèn báo phụ thuộc vào tình hình giao thông hiện tại. Những lúc vào giờ cao điểm, lượng dữ liệu từ các thiết bị xung quanh đèn giao thông như cảm biến chuyển động, các

phương tiện giao thông tăng mạnh, hệ thống cụm đèn trên đường có khả năng bị quá tải. Nhờ vào hệ thống theo dõi tài nguyên, hệ thống quản lý cụm đèn của thành phố sẽ phát hiện sớm sự quá tải và có thể tiến hành cân bằng tải ra sang các cụm đèn giao thông ở các con đường khác, hoặc chạy thêm các nút xử lý trên tầng Cloud để cân bằng tải [6].

2.2.4. Vấn đề với bài toán giám sát áp dụng vào hệ thống IoT

Bài toán giám sát khi áp dụng vào hệ thống 3 tầng IoT sẽ gặp phải 3 vấn đề quan trọng.

Vấn đề đầu tiên là sự đa dạng của các thông số đo (metric) cần thu thập. Có rất nhiều kiểu mô hình dữ liệu tương ứng với sự đa dạng của Things. Ngay cả một thành phần trong hệ thống cũng đã có rất nhiều đặc trưng cần phải theo dõi. Vì vậy, một giải pháp toàn diện, không chỉ cho phép biểu diễn và mô hình hóa tất cả các thông số đo (metric) dưới dạng một chuẩn thống nhất, mà còn phải có khả năng dễ dàng mở rộng, đáp ứng sự biến đổi nhanh chóng của các công nghệ IoT. Đây là một điều bắt buộc khi giải quyết bài toán giám sát hệ thống IoT.

Các ví dụ sau là các mẫu dữ liệu đo đạc thu được từ các cảm biến và các hệ thống theo dõi tài nguyên:

- Một mẫu dữ liệu từ thiết bị cảm biến được kết nối với ứng dụng OpenHAB IoT Platform


```
[
  {
    "attributes": {
      "assumed_state": true,
      "auto": true,
      "entity_id": [
        "light.garage_outside_light"
      ],
      "friendly_name": "all lights",
      "hidden": true,
      "order": 1
    },
    "entity_id": "group.all_lights",
    "last_changed": "2016-12-13T13:18:34.692337+00:00",
    "last_updated": "2016-12-13T13:18:34.692337+00:00",
    "state": "on"
  },
  {
    "attributes": {
      "assumed_state": true,
      "friendly_name": "Ventilators",
      "icon": "mdi:fan"
    },
    "entity_id": "switch.ventilators",
    "last_changed": "2016-12-13T13:18:33.639397+00:00",
    "last_updated": "2016-12-13T13:18:33.639397+00:00",
    "state": "on"
  },
  {
    "attributes": {
      "assumed_state": true,
      "friendly_name": "Garage outside light",
      "icon": "mdi:spotlight",
      "supported_features": 0
    },
    "entity_id": "light.garage_outside_light",
    "last_changed": "2016-12-13T13:18:34.689111+00:00",
    "last_updated": "2016-12-13T13:18:34.689111+00:00",
    "state": "on"
  },
```


Hình 5 Mẫu dữ liệu từ thiết bị cảm biến được kết nối với ứng dụng OpenHAB IoT Platform

- Một mẫu dữ liệu cảm biến thu được từ ứng dụng OpenIoT IoT Platform

```
"data": {
"DeviceProps": {
  "commandURL": "http://.../services/OpenIoT/assets/..",
  "lastIP": "195.97.103.225", "commands": true },
"asset": {
  "name": "00:3b:B6:BodyTemperature",
  "description": "asset model protocol" },
"model": "SENSOR_TEMP",
"registrationTime": "2015-04-16T15:39:58Z",
"status": "Active",
"sensorMetaData": [
  {"ms": {
      "dataType": "BodyTemperature", "unit": "Celsius",
      "rate": "10" }
}]}
```


Hình 6 Mẫu dữ liệu cảm biến thu được từ ứng dụng

OpenIoT IoT Platform

- Một mẫu dữ liệu về thông tin tiêu thụ tài nguyên CPU của một ứng dụng thu được từ hệ thống theo dõi:


```
{
        "('memory / usage', None)":
        [{
                'time': '2017-04-03T07:00:00Z',
                'host_id': '128.199.91.17',
                'namespace_id': '1a480b89-10a1-11e7-8989-7eb92be1eeb0',
                'container_base_image': 'huanphan/onem2m:1.0',
                'labels': 'app:onem2m-1',
                'pod_name': 'onem2m-1-z8zn2',
                'type': 'pod_container',
                'value': 8368128,
                'nodename': '128.199.91.17',
                'pod_namespace': 'kube-system',
                'container_name': 'onem2m-1',
                'pod_id': 'eff777e6-10a7-11e7-8989-7eb92be1eeb0',
                'hostname': '128.199.91.17',
                'namespace_name': 'kube-system'
        }, {
                'time': '2017-04-03T07:00:00Z',
                'host_id': '128.199.91.17',
                'namespace_id': '1a480b89-10a1-11e7-8989-7eb92be1eeb0',
                'container_base_image': 'huanphan/onem2m:1.0',
                'labels': 'app:onem2m-1',
                'pod_name': 'onem2m-1-dm6mk',
                'type': 'pod_container',
                'value': 235900928,
                'nodename': '128.199.91.17',
                'pod_namespace': 'kube-system',
                'container_name': 'onem2m-1',
                'pod_id': '8cac43ee-1819-11e7-8989-7eb92be1eeb0',
                'hostname': '128.199.91.17',
                'namespace_name': 'kube-system'
        }]
}
```


Hình 7 Mẫu dữ liệu về thông tin tiêu thụ tài nguyên CPU của một ứng dụng

- Một mẫu dữ liệu từ thiết bị cảm biến LBA nhận diện sự hoạt động của con người [10]

```
{
    "date":"2016-03-17T11:03:33",
    "comment":"LBASense Beta system. Infos: 1) Date parameter is currently equal to
the request time; 2) DeviceID is currently equal/fake for all devices.",
    "sapDetailedInformation":[
    {
        "regionID":1,
        "sapDetailedRegionInformation":[
            {
                "visitorID":780,
                "deviceID":"AA:BB:CC:11:22:33",
                "category":0,
                "globalID":1,
                "firstTimeSeen":"2016-03-16T08:58:07",
                "lastTimeSeen":"2016-03-16T11:41:43"
            },
            {
                "visitorID":316028746,
                "deviceID":"AA:BB:CC:11:22:33",
                "category":0,
                "globalID":0,
                "firstTimeSeen":"2016-03-16T11:41:59",
                "lastTimeSeen":"2016-03-16T11:42:07"
            }]
    },
    {
        "regionID":2,
        "sapDetailedRegionInformation":[
        {
            "visitorID":780,
            "deviceID":"AB:CC:DD:11:22:33",
            "category":0,
            "globalID":1,
            "firstTimeSeen":"2016-03-16T08:58:07",
            "lastTimeSeen":"2016-03-16T11:41:43"
        },
        {
            "visitorID":316028746,
            "deviceID":"A1:B1:C1:13:20:31",
            "category":0,
            "globalID":0,
            "firstTimeSeen":"2016-03-16T11:41:59",
            "lastTimeSeen":"2016-03-16T11:42:07"
        }]
    }
}
```


Hình 8 Mẫu dữ liệu từ thiết bị cảm biến LBA

- Từ các ví dụ trên, có thể thấy mô hình dữ liệu giữa các đối tượng là rất đa dạng, điều này gây ra khó khăn cho việc kết hợp xử lý một cách thống nhất các mô hình dữ liệu khác nhau cho mục đích khai thác thông tin ngữ nghĩa, vì vậy cần thiết phải có một phương pháp xử lý nhằm đưa các mô hình dữ liệu về một dạng thống nhất.

Vấn đề gặp phải thứ 2, hiện tại chưa có mô hình kiến trúc nhằm theo dõi các thành phần trong tầng "sương mù" theo một cách thống nhất, áp dụng để giải quyết cho vấn đề thứ 1. Các hệ thống giám sát cho môi trường ĐTĐM đã có rất nhiều, nhưng mô hình giám sát cho tầng "sương mù" hiện vẫn còn thiếu.

Vấn đề cuối cùng, dựa trên những dữ liệu thu thập được và ngữ cảnh sử dụng, những nhà phát triển các ứng dụng IoT sẽ mô tả những yêu cầu hoặc điều kiện để đảm bảo sự vận hành ổn định của hệ thống IoT. Nói một cách khác, họ cần một giải pháp hệ thống để tạo, tổng hợp, khai thác ngữ nghĩa từ các dữ liệu được thu thập cho việc điều khiển hoặc tối ưu hóa quản lý hệ thống IoT. Điều này được thể hiện ở lớp "tập hợp" (Aggregation) trong hình 4.

Từ các vấn đề trên, tôi đề xuất một kiến trúc hệ thống cho phép theo dõi, thu thập dữ liệu cảm biến và dữ liệu về tài nguyên tiêu thụ từ các tầng của hệ thống IoT, và đề xuất một mô hình dữ liệu thống nhất để đặc tả cho các thông số đo đa dạng

thu thập được. Trên cơ sở mô hình dữ liệu thống nhất này, bài toán tập hợp, sử dụng ngữ nghĩa để khai thác, phân tích dữ liệu có thể được áp dụng.

Chương 3: Đề xuất giải pháp

Từ góc nhìn của bài toán giám sát, một số quan sát có thể được đưa từ một hệ thống IoT:

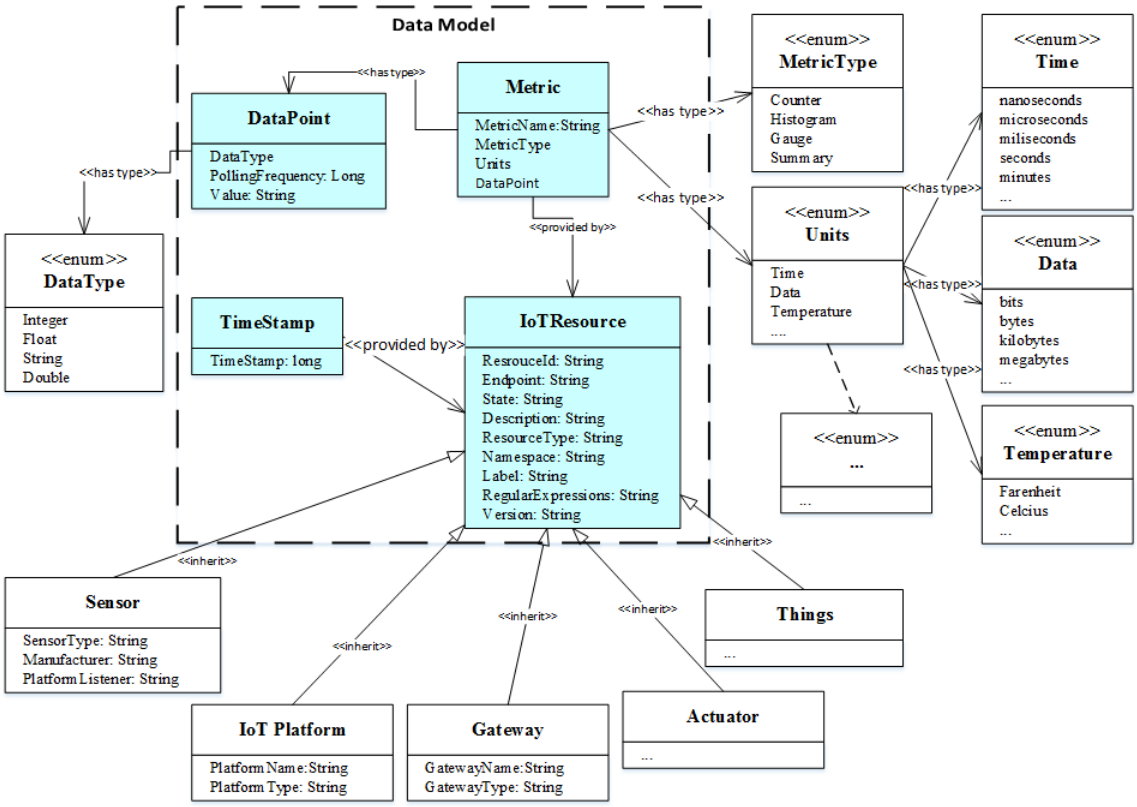
- Dựa trên tính không đồng nhất, dữ liệu cần giám sát có thể được chia làm 3 loại: (i) dữ liệu cảm biến được thu thập từ các thiết bị cảm biến (giá trị nhiệt độ, độ ẩm), (ii) tài nguyên tiêu thụ của các thiết bị (bộ nhớ, CPU, kết nối mạng...) thuộc về 3 tầng IoT, và (iii) log mô tả hoạt động của các thiết bị. Trong khi (i) có thể được thu thập bởi các IoT platform, các dữ liệu (ii) (iii) hiện chưa có một giải pháp toàn diện để giải quyết.
- Theo dõi tài nguyên của hệ thống IoT có thể chia 3 nhóm: khả năng tính toán (compute), bộ nhớ lưu trữ (storage), và kết nối mạng (network). Trong một số trường hợp, ở tầng Things, có thể tồn tại một số cảm biến có khả năng tính toán, lưu trữ dữ liệu. Như trong kịch bản về tòa nhà thông minh, cảm biến nhận diện vị trí trên robot lau nhà có thể xử lý dữ liệu và xác định vị trí trong phòng hoặc trong từng tầng, dựa vào các kết nối trao đổi dữ liệu với các Things xung quanh. Mặc dù hầu hết các loại cảm biến chỉ có chức năng cảm ứng môi trường xung quanh và truyền dữ liệu, tuy nhiên khả năng kết nối mạng luôn luôn cần phải được xem xét đối với các thiết bị loại này. Mặt khác, ở các tầng Fog và Cloud, 3 loại tài nguyên trên hiển nhiên cần phải được xem xét.

→ Dựa vào các quan sát trên, tôi đề xuất ra một mô hình dữ liệu nhằm mô tả các đối tượng dữ liệu: dữ liệu cảm biến từ các thiết bị cảm biến và dữ liệu về tài nguyên tiêu thụ của các thiết bị. Mô hình này phải đảm bảo các tính chất cô đọng, dễ dàng mở rộng, dễ dàng biểu diễn bằng các ngôn ngữ lược đồ phổ biến (tài liệu XML), dễ dàng áp dụng vào các ngôn ngữ lập trình khác nhau.

3.1. Mô hình dữ liệu chuẩn hóa

3.1.1. Mô hình

Dựa trên các tìm hiểu, nghiên cứu các nguồn tài liệu khác nhau [9][11][12][13], tôi đã xây dựng được một mô hình tổng quát nhằm mô tả các đối tượng dữ liệu cần giám sát như sau:



Hình 9 Mô hình cơ bản mô tả các đối tượng dữ liệu

Một mô hình dữ liệu biểu diễn các thông số độ đo (metric) theo thời gian thực sẽ gồm 4 thành phần cốt lõi:

- Thành phần *IoTResource* là thành phần quan trọng nhất, mô tả thông tin cơ bản về các đối tượng được theo dõi. Mỗi đối tượng sẽ kế thừa từ lớp *IoTResource* và có thêm các thông tin mở rộng: Sensor, IoT Platform, Gateway, Things...
- Thành phần *Timestamp* dùng để mô tả nhãn thời gian được khởi tạo của từng mô hình theo *IoTResource*.
- Thành phần *Metric* nhằm mô tả thông tin về độ đo được theo dõi từ *IoTResource*.
- Thành phần *DataPoint* nằm trong *Metric*, nhằm mô tả giá trị thực, kiểu dữ liệu của độ đo thu thập được.

Các thành phần trên ở mức cốt lõi, cô đọng, có thể dễ dàng mở rộng, định nghĩa thêm thành phần mới/ thuộc tính mới khi cần thiết. Tính dễ dàng mở rộng được thể hiện ở các điểm sau:

- *Người sử dụng mô hình có thể dễ dàng định nghĩa thêm các loại IoTResource khác nhau, với các thông tin mô tả chi tiết khác nhau, bằng việc kế thừa thành phần IoTResource (ví dụ như Things, Actuator…).*
- *Thành phần con Units của Metric có thể dễ dàng bổ sung thêm các kiểu đơn vị mô tả độ đo khác nhau như áp suất, số lượng request, thể tích ... nhằm đáp ứng yêu cầu mô tả dữ liệu của từng ngữ cảnh khác nhau.*

- *Mô hình trên có thể dễ dàng được mô tả bằng ngôn ngữ lược đồ phổ biến như XML.*

Bảng sau mô tả chi tiết các thành phần trong mô hình trên

Thành phần	Thuộc tính	Mô tả
IoTResource	ResourceId	Định danh duy nhất của đối tượng được theo dõi trong cơ sở dữ liệu
	Endpoint	Địa chỉ để các bộ thu thập có thể truy cập lấy được dữ liệu theo dõi
	State	Trạng thái của đối tượng
	Description	Mô tả về đối tượng được theo dõi
	ResourceType	Các kiểu đối tượng như Sensor, Actuator, Gateway, Router...
	Namespace	Không gian tên mà đối tượng thuộc về
	Label	Nhãn của đối tượng, được dùng cho các bộ lọc, như lọc theo nhãn nhiệt độ hoặc độ ẩm ...
	RegularExpression	Biểu thức chính quy mô tả cách bộ thu thập đọc giá trị
	Version	Phiên bản của đối tượng
Timestamp	Timestamp	Nhãn thời gian mô hình dữ liệu được khởi tạo
Metric	MetricName	Mô tả tên của metric được thu thập, ví dụ: nhiệt độ môi trường, độ ẩm không khí, bộ nhớ sử dụng...
	MetricType	Mô tả các kiểu metric, có các dạng như sau: - Counter: một dạng số liệu đếm tích lũy theo thời gian, ví dụ như

		đếm số request được thực hiện, số lỗi đã phát hiện trong hệ thống… - Gauge: mô tả kiểu số liệu có giá trị biến đổi theo thời gian thực, ví dụ như nhiệt độ, bộ nhớ sử dụng hiện tại… - Histogram: biểu diễn các số liệu theo dạng phân phối tần suất theo các nhóm, ví dụ chu kì request theo sự tăng dần của số lượng request. - Summary: biểu diễn các số liệu theo một cách thống kê trong một khoảng thời gian, ví dụ số lượng request trung bình trong một phút.
	Units	Kiểu đơn vị của metric, có thể có các đơn vị chuẩn như Time (seconds, miliseconds, minutes..), Data (bits, bytes, kilobytes), Temperature (Celcius, ..) ...
	DataPoint	Kiểu phức hợp mô tả giá trị theo thời gian thực của metric thu thập
DataPoint	DataType	Kiểu giá trị của metric, có một trong các kiểu như: Float, Integer, Double...
	PollingFrequency	Tần suất gửi giá trị của đối tượng (sensor, gateway...)
	Value	Giá trị theo thời gian thực của metric

Bảng 1 Mô tả các thành phần trong mô hình dữ liệu cơ bản

Mô hình dữ liệu trên có thể được biểu diễn bằng tài liệu XML với ngôn ngữ lược đồ XML (XML Schema Definition) sau:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="DataModel">
    <xs:complexType>
```

```
<xs:sequence>
 <xs:element name="Timestamp" type="xs:integer" />
 <xs:element name="Id" type="xs:long" />
 <xs:element name="Metric">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="MetricName" type="xs:string" />
    <xs:element name="MetricType">
     <xs:complexType>
      <xs:enumeration value="Gagge" />
      <xs:enumeration value="Counter" />
      <xs:enumeration value="Histogram" />
      <xs:enumeration value="Summary" />
     </xs:complexType>
    </xs:element>
    <xs:element name="Units">
     <xs:complexType>
      <xs:choice>
       <xs:element name="Time">
        <xs:complexType>
         <xs:enumeration value="nanoseconds" />
         <xs:enumeration value="microseconds" />
         <xs:enumeration value="miliseconds" />
         <xs:enumeration value="seconds" />
         <xs:enumeration value="minutes" />
        </xs:complexType>
       </xs:element>
       <xs:element name="Data">
        <xs:complexType>
         <xs:enumeration value="bits" />
         <xs:enumeration value="bytes" />
         <xs:enumeration value="kilobytes" />
         <xs:enumeration value="megabytes" />
        </xs:complexType>
       </xs:element>
       <xs:element name="Temperature">
        <xs:complexType>
         <xs:enumeration value="Farenheit" />
         <xs:enumeration value="Celcius" />
        </xs:complexType>
       </xs:element>
      </xs:choice>
     </xs:complexType>
    </xs:element>
    <xs:element name="DataPoint">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="DataType">
        <xs:restriction base="xs:string">
         <xs:enumeration value="Float" />
         <xs:enumeration value="integer" />
         <xs:enumeration value="String" />
         <xs:enumeration value="Double" />
        </xs:restriction>
       </xs:element>
       <xs:element name="Value" type="xs:float" />
       <xs:element name="PollingFrequency" type="xs:float" />
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="Resource">
```

```
      <xs:complexType>
       <xs:sequence>
        <xs:element name="ResourceId" type="xs:string" />
        <xs:element name="Endpoint" type="xs:string" />
        <xs:element name="State" type="xs:string" />
        <xs:element name="Description" type="xs:string" />
        <xs:element name="Type" type="xs:string" />
        <xs:element name="Namespace" type="xs:string" />
        <xs:element name="Label" type="xs:string" />
        <xs:element name="RegularExpression" type="xs:string" />
        <xs:element name="Version" type="xs:string" />
        <xs:any minOccurs="0" />
       </xs:sequence>
      </xs:complexType>
     </xs:element>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
</xs:schema>
```

3.1.2. Ứng dụng mô hình vào việc mô tả các đối tượng dữ liệu khác nhau

Mô hình trong phần 3.1.1 có chứa các thành phần cốt lõi của một đối tượng dữ liệu, và khi áp dụng vào triển khai hệ thống, nhà phát triển có thể dễ dàng bổ sung các thành phần mới vào mô hình. *Trong phạm vi đồ án này, tôi không đề xuất một phương pháp chung cho việc chuyển đổi từ các mô hình khác nhau sang mô hình thống nhất, mà chỉ đề cập một phương pháp đối sánh chuỗi, sẽ được trình bày ở chương sau.*

Những ví dụ sau chứng minh cho khả năng áp dụng của mô hình vào việc mô tả các đối tượng dữ liệu đa dạng:

- Ví dụ 1: Dữ liệu một IoT Platform thu thập được từ một cảm biến áp suất (Atmosphere sensor):

```
{
 "Timestamp": "1492074859.14",
 "Resource": {
  "State": "active",
  "Endpoint": "172.17.0.3/demo/sensys_atmosphere_sensor",
  "Regex": "(.*)",
  "PlatformListener": "openhab_pf_1:172.17.0.3"
 },
 "Sensor": {
            "Timestamp": "1492443715.529",
            "Namespace": "IOT_LAB_2",
            "Metric": {
             "DataPoint": {
              "Value": 69,
              "DataFrequency": 10,
              "DataType": "Float"
             },
             "Units": {
              "Pressure": "PSI"
```

```
            },
            "MetricName": "Atmosphere mesurement",
            "MetricType": "Gauge"
           },
           "Description": "SENSYS Atmosphere Sensor Mesurement",
           "SensorType": "IC",
           "Manufacturer": "Panasonic",
           "Label": "task:pressure_warning",
           "ResourceId": "SENSYS_Atmosphere_Sensor:127.0.1.1",
           "ResourceType": "sensor",
           "Version": "v1.3",
          }
}
```


Và dữ liệu trên khi được mô hình hóa qua mô hình đã đề xuất (định dạng XML)


```
<DataModel>
        <Id>143543547</Id>
        <Timestamp>1492074859.14</Timestamp>
        <Metric>
                <MetricName>Atmosphere mesurement</MetricName>
                <Units>
                        <Pressure>PSI</Pressure>
                </Units>
                <MetricType>Gauge</MetricType>
                <DataPoint>
                        <DataType>Float</DataType>
                        <DataFrequency>10</DataFrequency>
                        <Value>69</Value>
                </DataPoint>
        </Metric>
        <Resource>
                <ResourceId>SENSYS_Atmosphere_Sensor:127.0.1.1</ResourceId>
                <Endpoint>172.17.0.3/demo/sensys_atmosphere_sensor</Endpoint>
                <Status>active</Status>
      <ResourceType>sensor</ResourceType>
                <Description>SENSYS Atmosphere Sensor Mesurement</Description>
      <Namespace>IOT_LAB_2</Namespace>
                <Label>"task:pressure_warning"</Label>
                <Version>v1.3</Version>
                <SensorType>IC</SensorType>
                <SensorLocation>Top of building</SensorLocation>
                <Manufacturer>Panasonic</Manufacturer>
                <PlatformListener>openhab_pf_1:172.17.0.3</PlatformListener>
                <Regex>(.*)</Regex>
        </Resource>
</DataModel>
```


- Dữ liệu metric thu thập được về bộ nhớ sử dụng của một nút OpenHAB IoT Platform chạy trên một gateway:


```
{
        "('memory / usage')":
        [{
          "pod_name": "onem2m-4-td7xr",
          "value": 225083392,
```

```
            "pod_namespace": "kube-system",
            "container_base_image": "huanphan/onem2m:semi-final-2",
            "hostname": "139.59.98.138",
            "host_id": "139.59.98.138",
            "container_name": "onem2m-4",
            "type": "pod_container",
            "namespace_name": "kube-system",
            "time": "2017-04-30T18:51:00Z",
            "namespace_id": "c73fb82b-20dc-11e7-a3fd-7eb92be1eeb0",
            "labels": {
              "app": "onem2m-4"
            },
            "nodename": "139.59.98.138",
            "pod_id": "33f9ba58-2dd5-11e7-a3fd-7eb92be1eeb0"
           }]
}
```

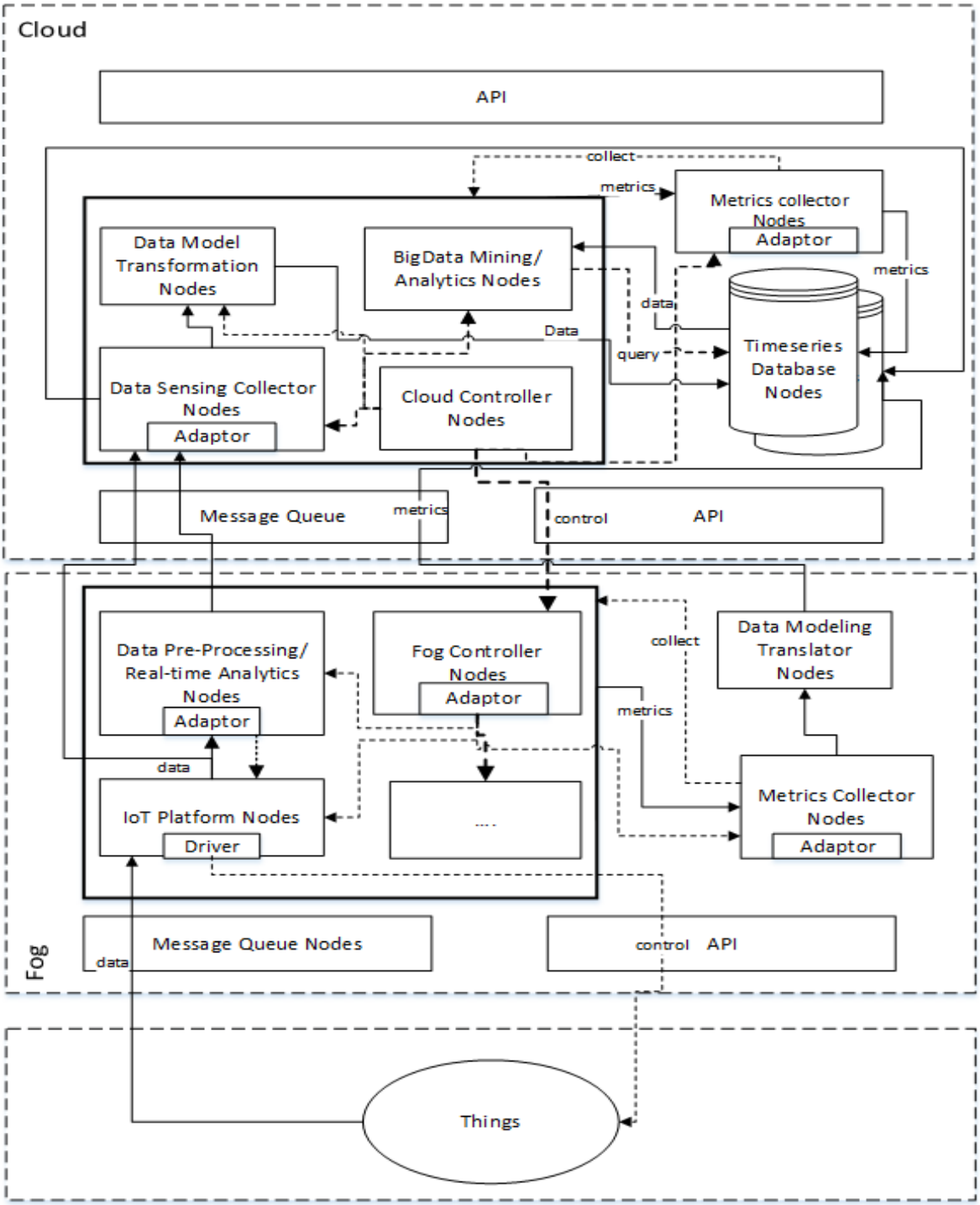

Và dữ liệu trên khi được chuẩn hóa qua mô hình đã đề xuất


```
<DataModel>
        <Id>32534534534</Id>
        <Timestamp>149342332</Timestamp>
        <Metric>
                <MetricName>Memory usage</MetricName>
                <Units>
                        <Data>Bytes</Data>
                </Units>
                <MetricType>Gauge</MetricType>
                <DataPoint>
                        <DataType>Long</DataType>
                        <DataFrequency>10</DataFrequency>
                        <Value>225083392</Value>
                </DataPoint>
        </Metric>
        <Resource>
                <ResourceId>onem2m-4-td7xr</ResourceId>
        <ResourceType>IoT Platform</ResourceType>
                <Endpoint>http://139.59.98.138/onem2m-4-td7xr/memory_usage</Endpoint>
        <Namespace>kube-system</Namespace>
                <State>active</State>
                <Description>IoT Platform memory usage, huanphan/onem2m:semi-final-2, onem2m-
4</Description>
        <Label>app:onem2m-4</Label>
                <PlatformName>OneM2M-4</PlatformName>
                <PlatformType>OneM2M</PlatformType>
                <Version>v1</Version>
        </Resource>
</DataModel>
```


Hai ví dụ trên đã cho thấy mô hình dữ liệu đề xuất hoàn toàn có khả năng mô tả các đối tượng dữ liệu đa dạng một cách linh hoạt, dễ tùy biến.

3.2. Mô hình hệ thống giám sát

Dựa trên góc nhìn giám sát cho cấu trúc 3 tầng của hệ thống IoT đã nói ở chương trước, tôi đề xuất mô hình hệ thống giám sát các đối tượng của hệ thống như hình 9.2.



Hình 9.2 Mô hình hệ thống giám sát

*Mô hình giám sát bao gồm 3 tầng **Things**, **Fog**, **Cloud**, tương ứng với đặc trưng 3 tầng vật lý của hệ thống IoT. Tuy nhiên hai tầng Fog và Cloud không hoàn*

toàn tương đồng về mặt ý nghĩa với 2 tầng tương ứng trong mô hình 3 tầng vật lý của hệ thống IoT. Mô hình giám sát này bao gồm các thành phần cơ bản, cốt lõi mà một hệ thống giám sát cần phải có, các nhà phát triển hoàn toàn có thể mở rộng thêm các thành phần khác trong từng tầng tuỳ thuộc vào từng bài toán cụ thể.

Tầng **Things** sẽ đại diện cho góc nhìn giám sát đối với hệ thống IoT, bao gồm tất cả các thiết bị vật lý/ đối tượng ảo, trực tiếp thu thập dữ liệu cảm ứng môi trường hoặc thực hiện các phản ứng với môi trường xung quanh, tùy theo chức năng.

Tầng **Fog** trong mô hình giám sát này không hoàn toàn tương đồng về mặt ý nghĩa với tầng Fog vật lý trong mô hình hệ thống 3 tầng IoT. Tầng Fog trong mô hình giám sát mang ý nghĩa logic, bao gồm các nút logic (các chương trình, ứng dụng) nhằm thu thập, tiền xử lý dữ liệu và điều khiển các Things theo thời gian thực. Tại tầng này, các nút logic IoT platform sẽ thu thập dữ liệu từ tầng Things. Các nút tại tầng Fog có thể được triển khai trên các thiết bị gateway, router, switch, máy tính nhúng, desktop, điện thoại thông minh... mà có khả năng tính toán, lưu trữ và truyền nhận dữ liệu.

- Tại các nút IoT platform, lớp Driver sẽ cho phép kết nối một cách đồng nhất tới các thiết bị đa dạng tại tầng Things. Dữ liệu thu được từ các thiết bị tầng Things sẽ đi qua nút IoT Platform, chuyển qua nút Data Pre-processing hoặc lên thẳng nút Data Sensing Collector, rồi được chuẩn hóa tại nút "*Data Model Transformation*", trước khi lưu trữ vào Database.
- Một thành phần rất quan trọng trong mô hình này là các nút "Metrics Collector" ở cả 2 tầng Fog và Cloud. Các nút này sẽ có nhiệm vụ thu thập dữ liệu về tài nguyên tiêu thụ như CPU, Memory, Storage từ các thành phần trong cùng tầng, và chuyển qua bộ "Data Model Transformation" trước khi lưu trữ vào cơ sở dữ liệu.
- Các nút Controller sẽ có nhiệm vụ bật tắt, scale in/out các instance của các thành phần.
- Các nút "Data Pre-processing/ Real-time Analytics" có thể được triển khai trên tầng Fog nhằm thực hiện một số tác vụ nhẹ về tính toán, xử lý dữ liệu.

Tầng **Cloud** sẽ triển khai các nút logic (chương trình, ứng dụng) thiên về mục đích xử lý và lưu trữ dữ liệu dài hạn. Một số nút tại tầng này có thể điều khiển hoạt động của các thành phần tại 2 tầng Things và Fog. Tầng này sẽ được triển khai trên môi trường điện toán đám mây.

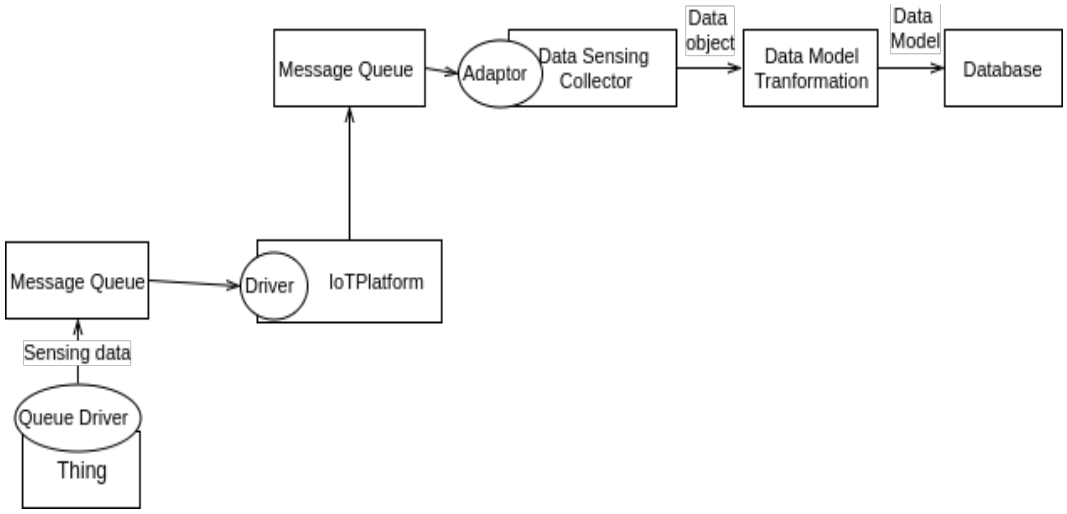
- Nút "*BigData Mining*/*Analytics*" sẽ thực hiện các tác vụ nặng về xử lý dữ liệu, khai thác dữ liệu lớn,
- Các số liệu thu thập được sẽ được lưu trong các cơ sở dữ liệu tối ưu cho kiểu dữ liệu theo dõi theo thời gian thực (timeseries database).

Các luồng hoạt động cơ bản của hệ thống:

(i) Luồng dữ liệu cảm biến

+ Các IoT Platform thu thập dữ liệu từ các thiết bị tầng **Things** sử dụng các driver.

+ Các *Data Sensing Collector* sẽ thu thập dữ liệu từ các IoT Platform thông qua các adaptor, và được chuyển qua nút *Data Model Transformation* để chuẩn hóa, trước khi lưu trữ vào Database.



Hình 10 Luồng dữ liệu cảm biến

(ii) Luồng dữ liệu theo dõi tài nguyên sử dụng của các nút:

+ Các *Metric Collector* thu thập thông số tài nguyên sử dụng theo thời gian thực và đẩy dữ liệu qua nút *Data Modeling Translator* để chuẩn hóa, trước khi lưu trữ vào Database.

(iii) Luồng điều khiển:

+ Các nút *Fog Controller/ Cloud Controller* sẽ điều khiển các nút tại tầng **Fog/ Cloud**

+ Các nút IoT Platform sẽ điều khiển các thiết bị tầng **Things**, các nút *Cloud Controller* sẽ điều khiển các nút *Fog Controller*, các điều khiển liên tầng này sẽ được thực hiện qua giao tiếp API.

(iv) Luồng truy vấn:

+ Các nút *BigData Mining/ Analytics* sẽ truy vấn dữ liệu từ database để thực hiện các bài toán phân tích dữ liệu.

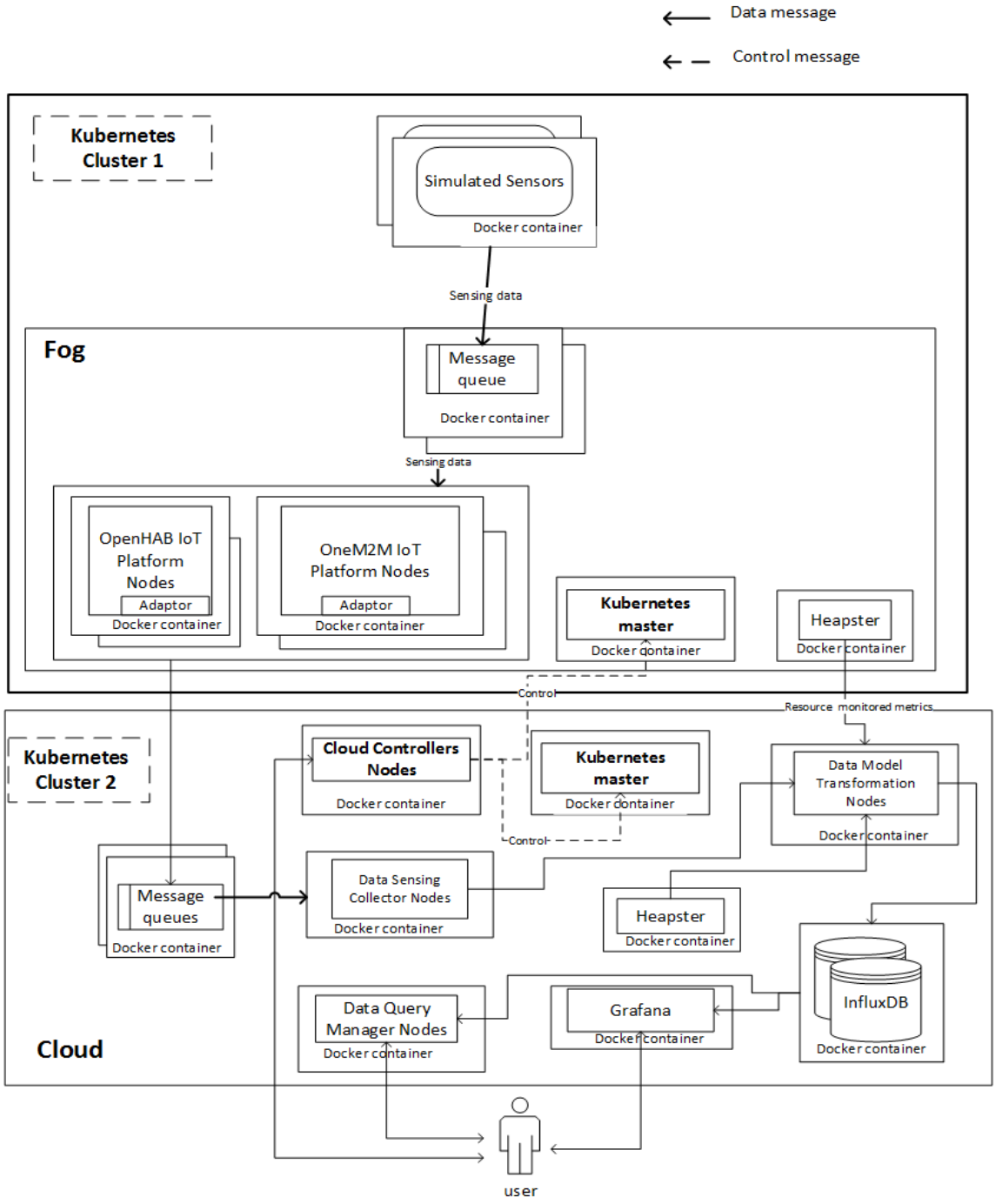
Các luồng i, ii, được biểu thị bằng các mũi tên nét liền, các luồng iii, iv được biểu thị bằng mũi tên nét đứt trên hình vẽ.

Chương 4: Ứng dụng mô hình đề xuất vào thử nghiệm thực tế

4.1. Mô hình hệ thống thực nghiệm

Nhằm kiểm tra khả năng hoạt động thực tế của mô hình đã đề xuất ở chương trên, trong chương này, tôi tiến hành xây dựng một mô hình thử nghiệm, với các ý tưởng từ mô hình đề xuất. Trong mô hình này có sử dụng một số nền tảng mã nguồn mở phổ biến.

4.1.1. Sơ đồ kiến trúc tổng thể



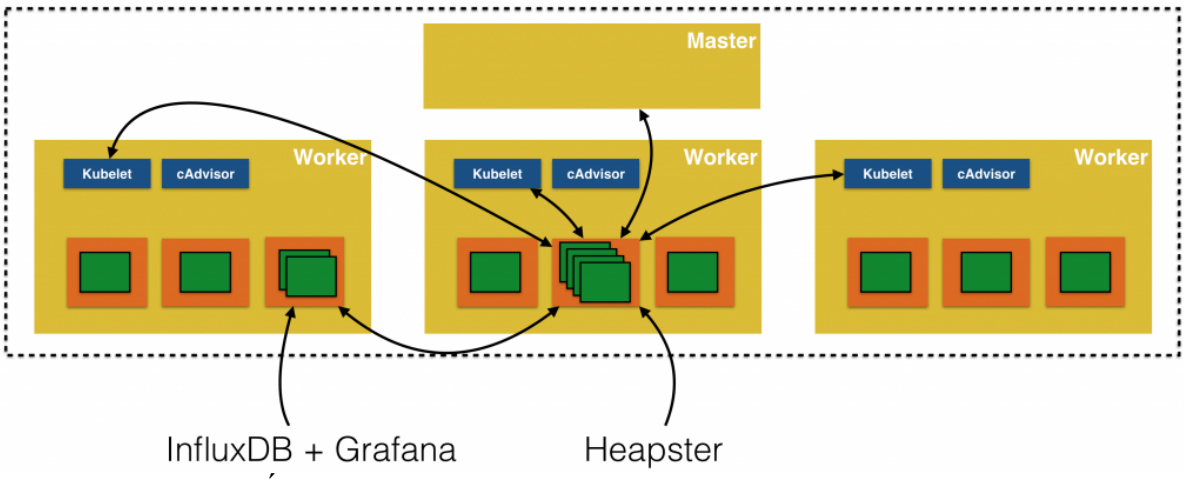
Hình 11 Mô hình hệ thống thực nghiệm

Hệ thống thử nghiệm được triển khai trên 2 cụm máy chủ, cụm thứ nhất sẽ chạy các ứng dụng/ dịch vụ của tầng Things và Fog, cụm thứ hai triển khai các ứng dụng/ dịch vụ tầng Cloud. Hai cụm máy chủ được cài đặt hệ thống mã nguồn mở Kubernetes. Các ứng dụng/ dịch vụ được triển khai trong môi trường ảo hóa Docker Container, và được quản lý bởi hệ thống Kubernetes trên từng cụm, do đó hệ thống sẽ đảm bảo sự hoạt động ổn định, tính sẵn sàng cao của các ứng dụng/ dịch vụ. Một hệ thống Kubernetes sẽ bao gồm một cụm máy chủ kết nối với nhau, trong đó một máy chủ sẽ đóng vai trò quản lý (Kubernetes Master) các máy còn lại (Kubernetes Worker). Các ứng dụng sẽ được Kubernetes Master lập lịch và triển khai tự động lên các máy chủ trong cụm. *Hệ thống Kubernets hỗ trợ các ứng dụng/dịch vụ trong cùng một cụm kết nối với nhau trong một mạng network chung và có thể kết nối ra Internet*. Trong phạm vi đồ án này, tôi sẽ sử dụng chương trình giả lập các sensor thay cho các sensor thực tế, lý do cho việc này sẽ được trình bày ở phần 4.2.1 của đồ án.

Các ứng dụng/ dịch vụ/ thành phần trong hệ thống gồm:

- Các *Simulated Sensors* là ứng dụng giả lập các cảm biến cho môi trường thử nghiệm, có thể điều chỉnh tần suất gửi dữ liệu và số cảm biến hoạt động. Các ứng dụng này được điều khiển bật, tắt bởi *CloudController*.
- Các *Data Model Transformation* sẽ có nhiệm vụ chuyển đổi đối tượng dữ liệu đầu vào sang dạng mô hình dữ liệu đã đề cập ở phần 3.1.
- Các *IoT Platform* được triển khai để thực hiện mô phỏng quá trình thu thập dữ liệu và quản lý các sensor. Trong phạm vi thử nghiệm của đồ án, các *IoT Platform* tập trung vào quá trình thu thập dữ liệu, không tập trung vào việc quản lý, điều khiển các sensor. Các *IoT Platform* sử dụng đều là mã nguồn mở, đã được tôi phát triển thêm các plugin thu thập dữ liệu nhằm đáp ứng cho hoạt động của mô hình.
- Hệ thống sử dụng các *Message Queue* là trung gian trung chuyển dữ liệu qua lại giữa các tầng.
- Ứng dụng *Grafana* được dùng để vẽ các biểu đồ trực quan theo dõi các metric theo thời gian thực.
- *Data Query Manager* cung cấp giao diện cho người sử dụng có thể truy vấn mô hình dữ liệu XML từ cơ sở dữ liệu.
- Các *Cloud Controllers* sẽ thực hiện điều khiển, quản lý các ứng dụng trong các tầng dựa vào các API mà các *Kubernetes Master* cung cấp.
- Nhằm theo dõi, thu thập các thông số về tài nguyên tiêu thụ của các ứng dụng/ dịch vụ và các máy chủ một cách tự động, hệ thống thử nghiệm sử dụng phần mềm mã nguồn mở *Heapster*. *Heapster* sẽ thu thập các các dữ liệu này từ các thành phần *Kubelet* được triển khai tự động trên các máy chủ khi được cài đặt hệ thống *Kubernentes*. Dữ liệu thu được sẽ đẩy vào cơ sở dữ liệu là *InfluxDB*, và ứng dụng *Grafana* sẽ thực hiện các truy vấn

lên cơ sở dữ liệu đó, nhằm cung cấp giao diện biểu đồ trực quan mô tả các dữ liệu thu thập được, từ đó cung cấp cái nhìn toàn cảnh về hoạt động của hệ thống cho người sử dụng (Hình 12).



Hình 12 Hệ thống kết hợp giám sát Heapster và Kubernetes (nguồn[4])

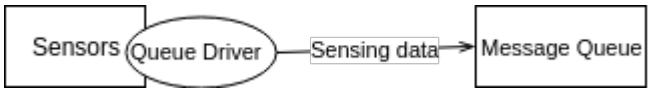
Các công nghệ mã nguồn mở và thiết kế của các thành phần tự phát triển sử dụng trong hệ thống được trình bày ở các phần sau.

4.1.2. Các thành phần tự phát triển

4.1.2.1. Simulated Sensor

Simulated Sensor là phần mềm giả lập sự hoạt động của một cảm biến thông thường. Phần mềm này sẽ sinh các dữ liệu tượng trưng cho giá trị cảm biến đo được với tần suất và khoảng giá trị có thể điều chỉnh được. Những tham số có thể điều chỉnh cho phần mềm này gồm: số lượng sensor, tần suất gửi dữ liệu (số gói tin/ phút), khoảng giá trị của các sensor.

Thiết kế tổng thể của thành phần Simulated Sensor



Hình 13 Thiết kế tổng thể của thành phần Simulated Sensor

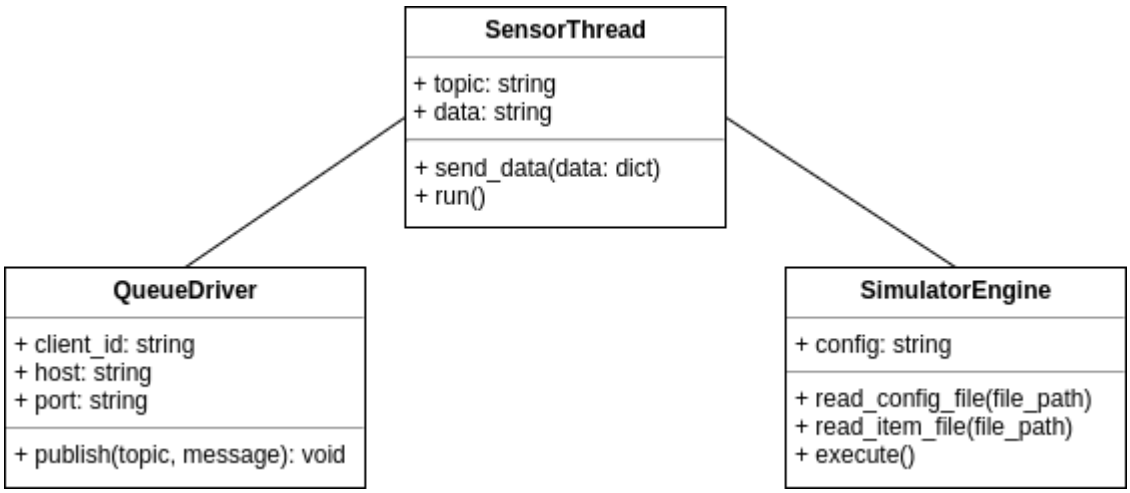
Các sensor sẽ có bộ *Queue Driver* nhằm hỗ trợ kết nối và đẩy dữ liệu vào *Message Queue*

Định dạng dữ liệu gửi đi của một sensor:

[4] https://blog.couchbase.com/kubernetes-monitoring-heapster-influxdb-grafana/

```
{
  "Timestamp": 234234234,
  "DataPoint": {
    "DataType": "Float",
    "Value": "143"
  }
}
```

4.1.2.1.1 Biểu đồ lớp của thành phần Simulated Sensor



Hình 14 Biểu đồ lớp của thành phần Simulated Sensor

Lớp *SimulatorEngine* chứa các phương thức khởi tạo của chương trình. Lớp *SensorThread* là luồng hoạt động chính, chạy nền và thực hiện sinh dữ liệu giả lập. Lớp *QueueDriver* cung cấp các phương thức hỗ trợ kết nối đến Message Queue.

4.1.2.1.2. Mô tả chi tiết các lớp

4.1.2.1.1.1. SensorThread

Kế thừa từ lớp Thead của python.

Thuộc tính

Tên	Kiểu dữ liệu	Mô tả
topic	string	Tên kênh sensor gửi dữ liệu lên message queue
data	string	Dữ liệu mà sensor sẽ gửi đi

Phương thức

Tên	Mô tả
send_data(): void	Thực hiện sinh dữ liệu giả lập theo cấu hình và gửi dữ liệu lên message queue
run(): void	Hàm bắt đầu luồng thực thi của lớp Thread

4.1.2.1.1.2. QueueDriver
Cung cấp các phương thức giao tiếp với Message Queue
Thuộc tính

Tên	Kiểu dữ liệu	Mô tả
client_id	string	Định danh của chương trình kết nối đến message queue
host	string	Địa chỉ kết nối của message queue
port	string	Cổng kết nối của message queue

Phương thức

Tên	Mô tả
publish(topic, message): bool	Gửi dữ liệu lên message queue

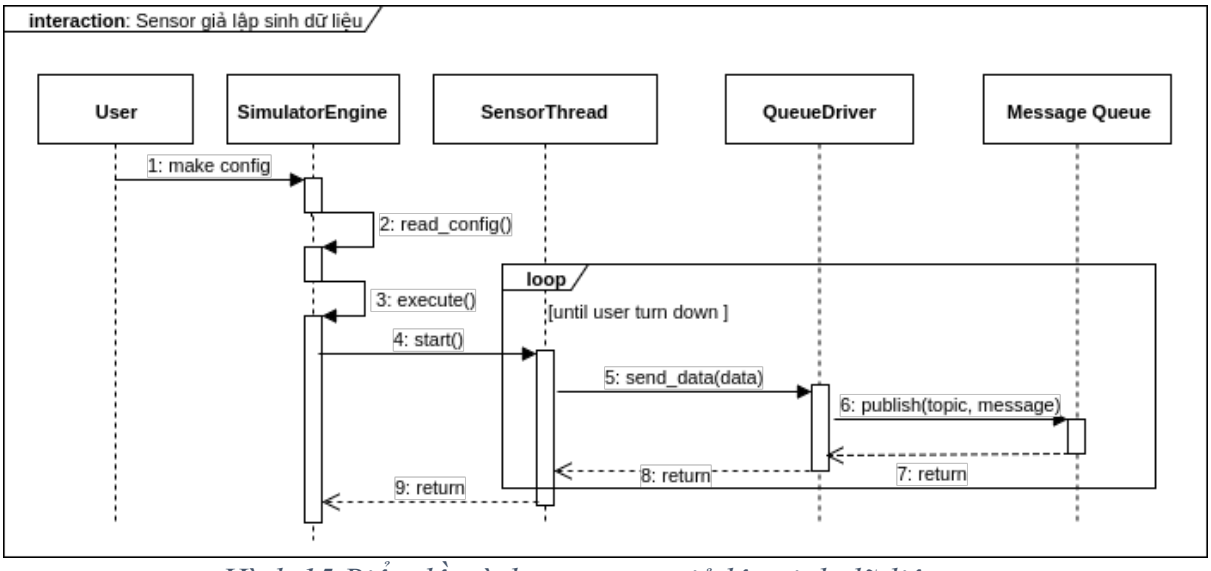
4.1.2.1.1.3. SimulatorEngine
Thuộc tính

Tên	Kiểu dữ liệu	Mô tả
config	string	Lưu các thiết lập của chương trình giả lập

Phương thức

Tên	Mô tả
read_config_file(file_path): string	Tải thiết lập của chương trình từ file
read_item_file(file_name): string	Tải các thiết lập của sensor như tần suất, số lượng, khoảng giá trị từ file
execute(): void	Thực thi chương trình giả lập

4.1.2.1.3. Biểu đồ trình tự



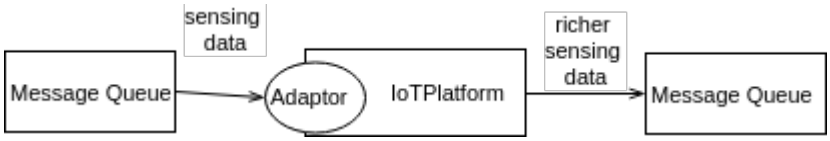
Hình 15 Biểu đồ trình tự sensor giả lập sinh dữ liệu

- Người sử dụng sẽ tạo các thiết lập, *SimulatorEngine* sẽ đọc các thiết lập này và khởi chạy một luồng *SensorThread*.
- Luồng *SensorThread* sẽ chạy ngầm, liên tục sinh dữ liệu theo tần suất và số lượng sensor theo thiết lập, gửi đến *MessageQueue* thông qua *QueueDriver*, cho đến khi người dùng hoặc hệ thống thực hiện tắt chương trình.

4.1.2.2. IoT Platform Adaptor

Thành phần IoT Platform Adaptor là thành phần có thể dễ dàng mở rộng, dễ dàng cài đặt như một plugin vào các IoT Platform có sẵn, thực hiện chức năng kết nối, nhận dữ liệu từ Message Queue, và chuyển đổi kiểu dữ liệu cảm biến đa dạng nhận được sang mô hình dữ liệu chuẩn mà từng loại IoT Platform có thể hiểu được [21].

Hình sau mô tả luồng dữ liệu qua IoT Platform Adaptor:



Hình 16 Mô tả luồng dữ liệu qua IoT Platform Adaptor

Bộ Adaptor sẽ dựa trên các thông tin thiết lập khác nhau của từng loại sensor để thực hiện quá trình chuyển đổi. Một ví dụ về thông tin thiết lập sensor của OpenHAB và OneM2M:

- OneM2M:

```
[
 {
  "metric_name": "Weather Temperature",
  "sensor_type": "Diode",
  "topic": "weather_temperature_sensor",
  "data_frequency": "10",
  "regex": "(.*)",
  "description": "Weather Temperature Sensor",
  "data_type": "Float"
 },
 {
  "metric_name": "Body Temperature",
  "item_type": "IC",
  "topic": "body_temperature_sensor",
  "data_frequency": "10",
  "regex": "(.*)",
  "description": "Body Temperature Sensor"
  "data_type": "Float"
 },
 {
  "metric_name": "Outside Light",
  "item_type": "Semiconductor",
  "topic": "sony_light_sensor_v1",
  "data_frequency": "10",
  "regex": "(.*)",
  "description": "SONY Light Sensor v1"
  "data_type": "Float"
 }
]
```

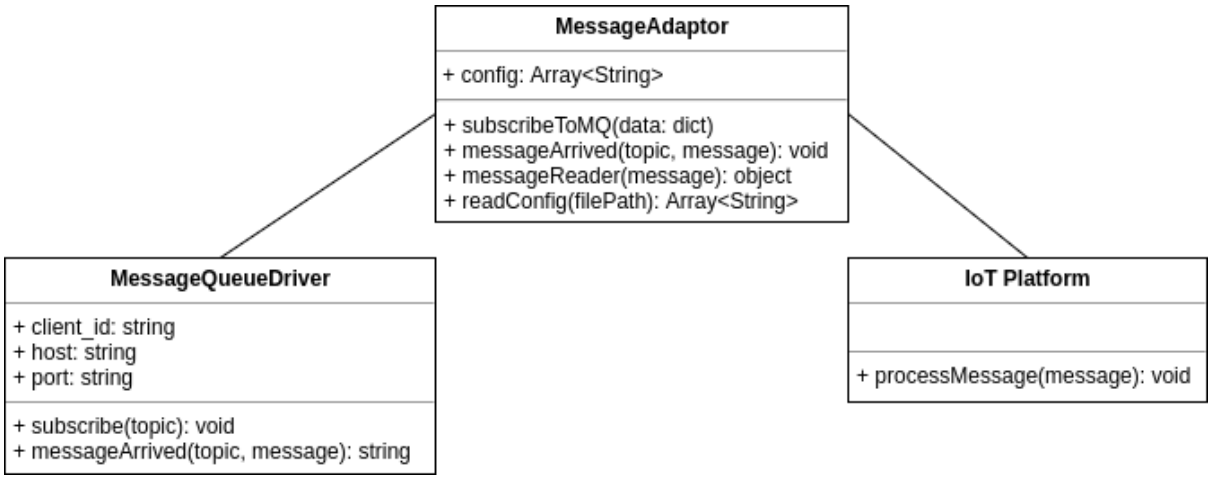
- OpenHAB:

```
Number Weather Temperature "Weather Temperature: [%.3f]]"
{mqtt="<[mqttIn:weather_temperature_sensor:state:default],
>[mqttOut:weather_temperature_sensor:state:*:EXEC(/openhab/configurations/get_timest
amp.sh ${state} ${itemName})]"}

Number Body Temperature "Body Temperature: [%.1f]"
{mqtt="<[mqttIn:body_temperature_sensor:state:REGEX(.*)],
```

```
>[mqttOut:body_temperature_sensor:state:*:EXEC(/openhab/configurations/get_timestamp.sh ${state} ${itemName})]"}

Number Outside Light "Outside Light: [%.1f%%]"
{mqtt="<[mqttIn:sony_light_sensor_v1:state:default],
>[mqttOut:sony_light_sensor_v1:state:*:EXEC(/openhab/configurations/get_timestamp.sh
${state} ${itemName})]"}
```

Dựa vào những thông tin thiết lập trên, bộ *Adaptor* trên mỗi *IoT Platform* có thể đọc được dữ liệu gửi từ các *Sensor*.

4.1.2.2.1. Biểu đồ lớp



Hình 17 Biểu đồ lớp thành phần IoT Platform Adaptor

Lớp IoTPlatform tượng trưng cho các IoT Platform khác nhau, được tích hợp bộ Adaptor như một plugin.

4.1.2.2.2. Mô tả chi tiết các lớp

4.1.2.2.2.1. MessageAdaptor

Lớp *MessageAdaptor* cung cấp các phương thức đọc dữ liệu nhận được từ Message Queue

Thuộc tính

Tên	Kiểu dữ liệu	Mô tả
config	Array<string>	Lưu các thiết lập của các sensor

Phương thức

Tên	Mô tả
subscribeToMQ(data: dict): void	Thực hiện lắng nghe một Message

	Queue
messageArrived(topic, message):void	Phương thức được gọi khi gói tin được QueueDriver nhận
messageReader(message): object	Đọc gói tin chuyển thành đối tượng dữ liệu mà các IoT Platform có thể hiểu được, gửi cho IoT Platform xử lý
readConfig(filePath): Array<string>	Đọc các thiết lập của các sensor

4.1.2.2.2.2. MessageQueueDriver

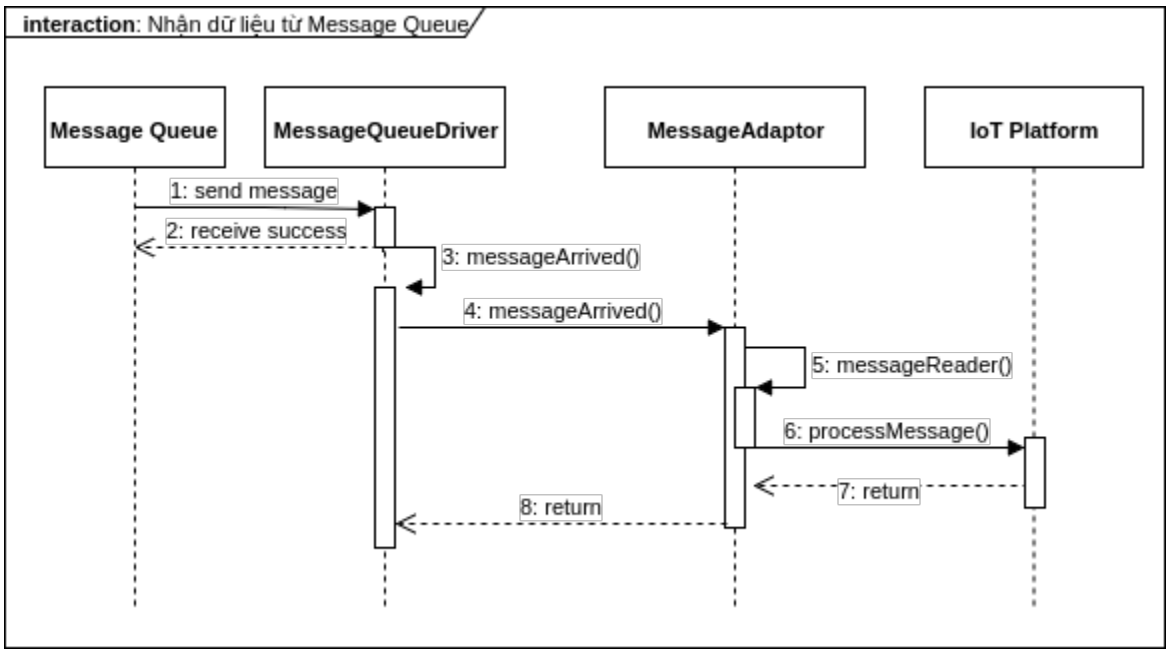
Lớp *MessageQueueDriver* cung cấp các giao thức để kết nối với *Message Queue*

Tên	Kiểu dữ liệu	Mô tả
client_id	string	Định danh của chương trình kết nối đến message queue
host	string	Địa chỉ kết nối của message queue
port	string	Cổng kết nối của message queue

Phương thức

Tên	Mô tả
subscribe(topic): void	Kết nối đến Message Queue với kênh "topic"
messageArrived(topic, message): string	Nhận dữ liệu từ Message Queue và gọi hàm messageArrived() của MessageAdaptor

4.1.2.2.3. Biểu đồ trình tự



Hình 18 Biểu đồ trình tự nhận dữ liệu từ Message Queue

- Thành phần MessageQueueDriver sẽ nhận dữ liệu gửi tới từ Message Queue, đồng thời hồi đáp lại tin thông báo nhận được gói tin, rồi chuyển gói tin đến MessageAdaptor.
- MessageAdaptor nhận gói tin, đọc, chuyển thành đối tượng dữ liệu phù hợp với từng IoT Platform.

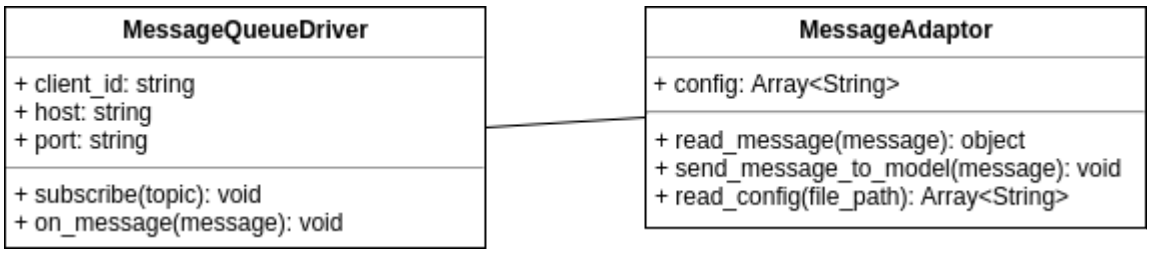
4.1.2.3. Data Sensing Collector

Data Sensing Collector thực hiện chức năng thu thập dữ liệu gửi đi từ các *IoT Platform*, thông qua *Message Queue*, rồi gửi dữ liệu thu được đến thành phần Data Model Transformation. Data Sensing Collector có chứa bộ Adaptor hỗ trợ đọc các mô hình dữ liệu khác nhau thu được từ các loại IoT Platform. Cần phải nạp các file lưu cấu hình kênh kết nối MessageQueue vào Adator để có thể nhận được dữ liệu từ IoT Platform.

Sơ đồ tổng quát hoạt động của Data Sensing Collector

Hình 19 Sơ đồ tổng quát hoạt động của Data Sensing Collector

4.1.2.3.1. Thiết kế lớp



Hình 20 Biểu đồ lớp thành phần Data Sensing Collector

4.1.2.3.2. Mô tả chi tiết các lớp

4.1.2.3.2.1. MessageAdaptor

Lớp *MessageAdaptor* cung cấp các phương thức đọc dữ liệu nhận được từ Message Queue

Thuộc tính

Tên	Kiểu dữ liệu	Mô tả
config	Array<string>	Lưu các thiết lập của các sensor

Phương thức

Tên	Mô tả
read_message(message): object	Đọc gói tin nhận được từ IoT Platform, chuyển thành định dạng mô hình dữ liệu
read_config(file_path): Array<string>	Đọc file cấu hình lưu kênh kết nối Message Queue
send_message_to_model(message): void	Gửi gói tin tới *Data Model Transformation*

4.1.2.3.2.2. MessageQueueDriver

Lớp *MessageQueueDriver* cung cấp các giao thức để kết nối với Message Queue

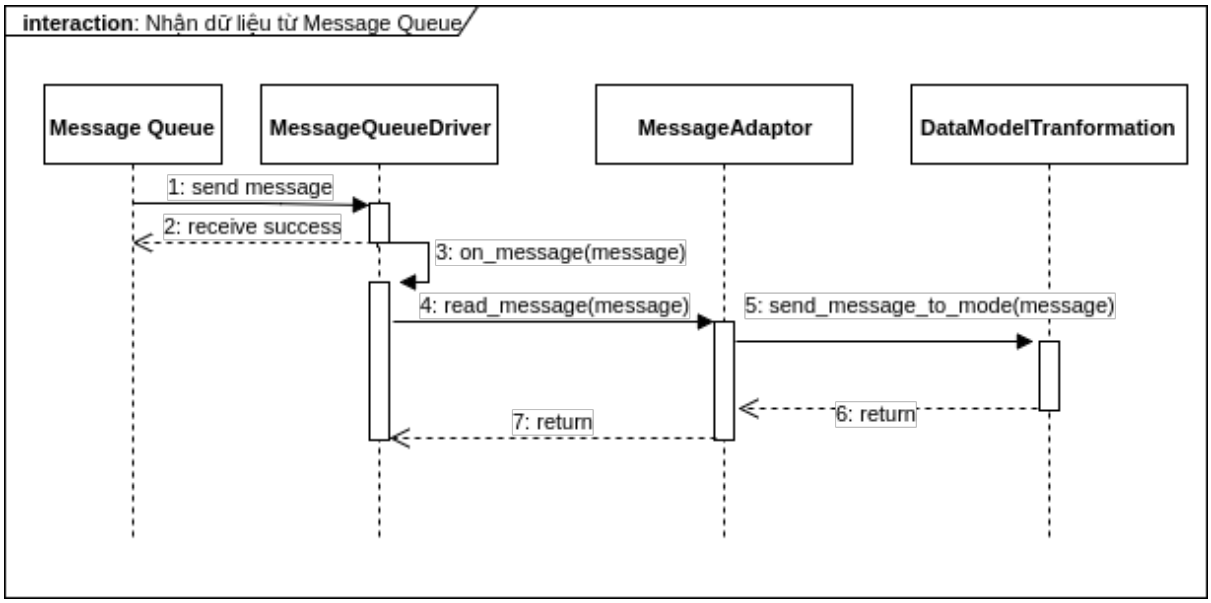
Tên	Kiểu dữ liệu	Mô tả
client_id	string	Định danh của chương trình kết nối đến message

		queue
host	string	Địa chỉ kết nối của message queue
port	string	Cổng kết nối của message queue

Phương thức

Tên	Mô tả
subscribe(topic): void	Kết nối đến Message Queue với kênh "topic"
on_message(message): void	Nhận dữ liệu từ Message Queue và gọi hàm read_message() của MessageAdaptor

4.1.2.3.3. Biểu đồ trình tự



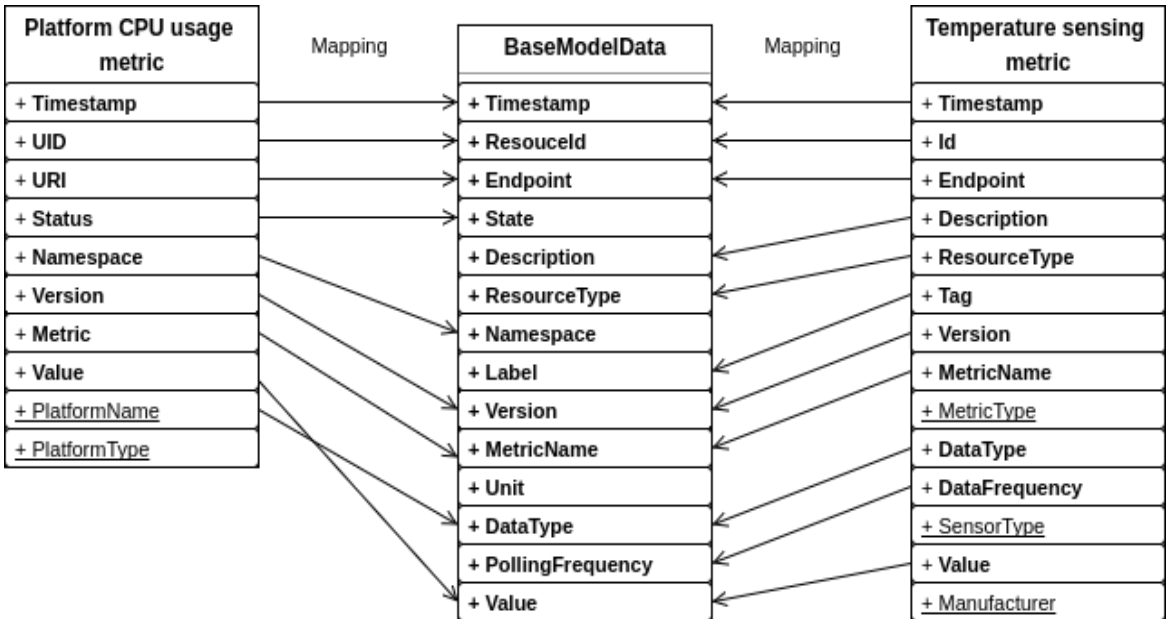
Hình 21 Biểu đồ trình tự nhận dữ liệu từ Message Queue

- Thành phần *MessageQueueDriver* sẽ nhận dữ liệu gửi tới từ *Message Queue*, đồng thời hồi đáp lại tin thông báo nhận được gói tin, rồi chuyển gói tin đến *MessageAdaptor*.

- *MessageAdaptor* nhận gói tin, đọc, chuyển thành đinh dạng json rồi gửi đến *DataModelTransformation*.

4.1.2.4. Data Model Transformation

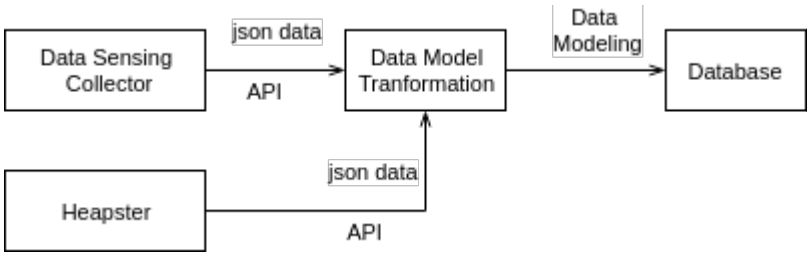
Thành phần *Data Model Transformation* thực hiện chức năng chuyển đổi các mô hình dữ liệu khác nhau sang mô hình dữ liệu chung đã trình bày ở phần 3.1. Dữ liệu sau khi chuyển đổi sẽ được lưu vào cơ sở dữ liệu. Ý tưởng cơ bản tôi sử dụng để chuyển đổi các mô hình khác nhau sang dạng mô hình chung là định nghĩa một mô hình chung với các thuộc tính cơ bản, từ đó bộ chuyển đổi sẽ đọc các mô hình dữ liệu khác nhau và thực hiện đối sánh chuỗi để mapping các trường dữ liệu vào mô hình chung đó. Mô hình chung này hoàn toàn có thể được mở rộng (thêm, bớt trường dữ liệu) một cách dễ dàng, đảm bảo tính linh hoạt, dễ mở rộng của mô hình. Biểu đồ sau mô tả ý tưởng của phương pháp đối sánh mô hình:



Hình 22 Data Model Transformation

Trong biểu đồ trên, mô hình BaseModelData chứa các thuộc tính cơ bản mà một mô hình dữ liệu cần phải có. Hai mô hình Platform CPU usage metric và Temperature sensing metric là các mô hình dẫn xuất. Các kết nối giữa các trường của các mô hình thể hiện khả năng khớp về mặt ý nghĩa của các trường đó (các trường in đậm). Đối với các trường tồn tại trong mô hình cơ sở BaseModelData mà không xuất hiện trong hai mô hình dẫn xuất, các trường sẽ được khởi tạo và gán giá trị mặc định trong lớp dẫn xuất. Ngoài ra ở các mô hình dẫn xuất có thể sẽ có thêm các trường dữ liệu mở rộng (các trường gạch chân). Người lập trình hoàn toàn có thể tự định nghĩa bộ đối sánh chuỗi để khớp các trường theo cách tùy ý.

- Luồng hoạt động cơ bản của Data Model Transformation:

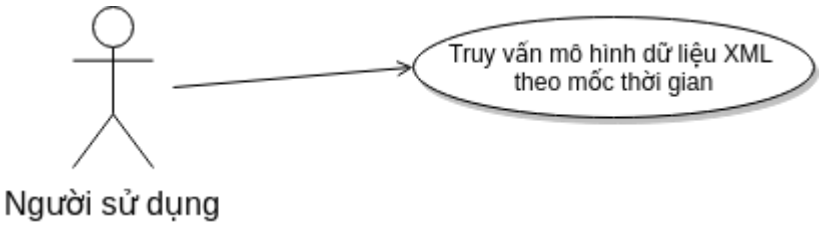


Hình 23 Luồng hoạt động cơ bản của Data Model Transformation

- Data Sensing Collector và Heapster sẽ gửi dữ liệu định dạng json, thông qua giao tiếp API, đến Data Model Transformation.
- Data Model Transformation nhận dữ liệu, chuẩn hóa thành mô hình chung rồi lưu vào cơ sở dữ liệu.

4.1.2.5. Data Query Manager

Thành phần *Data Query Manager* cung cấp giao diện tương tác cho người dùng, giúp người dùng có thể truy vấn tài liệu XML mô hình hóa các số liệu đo từ cơ sở dữ liệu.



Hình 24 Ca sử dụng truy vấn mô hình dữ liệu XML theo thời gian

4.1.2.5.1. Mô tả ca sử dụng

Tên ca sử dụng	Truy vấn mô hình dữ liệu XML theo mốc thời gian
Tác nhân	Người sử dụng
Mô tả tóm tắt	Người sử dụng có thể xem các tài liệu XML mô tả các metric của các ứng dụng theo mốc thời gian.
Điều kiện kích hoạt	Người sử dụng truy cập vào trang xem các metrics, chọn mốc thời gian cần truy vấn.

Thao tác chính	1. Người sử dụng chọn mốc thời gian trên giao diện. 2. Hệ thống thực hiện truy vấn tài liệu XML từ cơ sở theo mốc thời gian tương ứng đối với mỗi ứng dụng đang hoạt động. 3. Hệ thống trả các tài liệu XML cho người sử dụng.
Thao tác khác	Không có
Ngoại lệ	Kết quả trả về rỗng khi mốc thời gian lựa chọn không có ứng dụng nào đang hoạt động.

Hình 25 Ca sử dụng truy vấn XML theo mốc thời gian

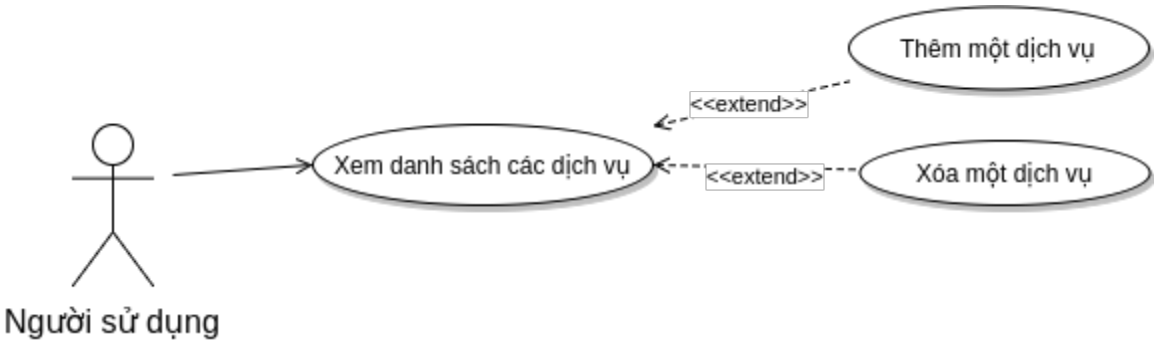
Ví dụ về một tài liệu XML được truy vấn thông qua *Data Query Manager*:

```
<DataModel>
        <Id>143543547</Id>
        <Timestamp>1492074859.14</Timestamp>
        <Metric>
                <MetricName>Atmosphere mesurement</MetricName>
                <Units>
                        <Pressure>PSI</Pressure>
                </Units>
                <MetricType>Gauge</MetricType>
                <DataPoint>
                        <DataType>Float</DataType>
                        <DataFrequency>10</DataFrequency>
                        <Value>69</Value>
                </DataPoint>
        </Metric>
        <Resource>
                <ResourceId>SENSYS_Atmosphere_Sensor:127.0.1.1</ResourceId>
                <Endpoint>172.17.0.3/demo/sensys_atmosphere_sensor</Endpoint>
                <Status>active</Status>
      <ResourceType>sensor</ResourceType>
                <Description>SENSYS Atmosphere Sensor Mesurement</Description>
      <Namespace>IOT_LAB_2</Namespace>
                <Label>"task:pressure_warning"</Label>
                <Version>v1.3</Version>
                <SensorType>IC</SensorType>
                <SensorLocation>Top of building</SensorLocation>
                <Manufacturer>Panasonic</Manufacturer>
                <PlatformListener>openhab_pf_1:172.17.0.3</PlatformListener>
                <Regex>(.*)</Regex>
        </Resource>
</DataModel>
```

4.1.2.6. Cloud Controller

Thành phần *Cloud Controller* cung cấp giao diện tương tác cho người dùng, giúp người dùng có thể điều khiển thủ công quá trình hoạt động của các thành phần trong hệ thống.
Cloud Controller thực hiện điều khiển các thành phần khác thông qua giao diện API của thành phần *Kubernetes master.*

- Các chức năng chính *Cloud Controller* cung cấp cho người dùng:



Hình 26 Tổng quan chức năng của Cloud Controller

4.1.2.6.1. Mô tả các ca sử dụng

Tên ca sử dụng	Xem danh sách các dịch vụ
Tác nhân	Người sử dụng
Mô tả tóm tắt	Sau khi người sử dụng truy cập vào giao diện web, hệ thống sẽ hiển thị danh sách các dịch vụ đang hoạt động, cùng với các thông tin mô tả cơ bản.
Điều kiện kích hoạt	Người sử dụng truy cập vào trang Quản lý dịch vụ
Thao tác chính	1. Người sử dụng chọn thao tác xem danh sách dịch vụ. 2. Hệ thống truy vấn danh sách các dịch vụ đang hoạt động từ cơ sở dữ liệu. 3. Hệ thống trả về và đưa ra màn hình danh sách dịch vụ dưới dạng bảng, sắp xếp theo thứ tự khởi tạo.

Thao tác khác	Không có
Ngoại lệ	Hệ thống sẽ trả về danh sách rỗng nếu hiện tại không có dịch vụ nào đang hoạt động.

Bảng 2 Ca sử dụng xem danh sách các dịch vụ

Tên ca sử dụng	Thêm một dịch vụ
Tác nhân	Người sử dụng
Mô tả tóm tắt	Sau khi người sử dụng truy cập vào giao diện web, hệ thống sẽ hiển thị danh sách các dịch vụ đang hoạt động, người dùng có thể chọn chạy thêm một dịch vụ từ giao diện.
Điều kiện kích hoạt	Người sử dụng truy cập vào trang Quản lý dịch vụ, chọn thêm dịch vụ và nhập vào các thông tin hợp lệ.
Thao tác chính	1. Người sử dụng chọn thao tác thêm dịch vụ trên giao diện quản lý. 2. Hệ thống hiển thị form nhập thông tin mô tả cho dịch vụ. 3. Sau khi người dùng nhập thông tin, hệ thống sẽ kiểm tra dữ liệu vừa nhập. 4. Dữ liệu được kiểm tra sẽ được hệ thống lưu vào cơ sở dữ liệu, đồng thời khởi chạy một dịch vụ tương ứng. 5. Hệ thống trả về kết quả "Thêm dịch vụ thành công".
Thao tác khác	Không có
Ngoại lệ	Quá trình thêm dịch vụ không thành công khi hệ thống đã hết tài nguyên cấp phát cho dịch vụ.

Bảng 3 Ca sử dụng thêm một dịch vụ

Tên ca sử dụng	Xóa dịch vụ
Tác nhân	Người sử dụng
Mô tả tóm tắt	Sau khi người sử dụng truy cập vào giao diện web, hệ thống sẽ hiển thị danh sách các dịch vụ đang hoạt động, người dùng có thể thực hiện xóa các dịch vụ này khỏi hệ thống.
Điều kiện kích hoạt	Người sử dụng truy cập vào trang Quản lý dịch vụ, chọn các dịch vụ cần xóa và nhấn xóa.
Thao tác chính	1. Người sử dụng tích chọn các dịch vụ cần xóa trên giao diện Quản lý dịch vụ. 2. Người sử dụng nhấn nút "Xóa" trên giao diện. 3. Hệ thống sẽ thực hiện xóa thông tin từng dịch vụ mà người dùng chọn khỏi cơ sở dữ liệu. 4. Hệ thống sẽ thực hiện xóa các dịch vụ đang hoạt động trong hệ thống. 5. Hệ thông thông báo kết quả cho người dùng.
Thao tác khác	Không có
Ngoại lệ	Quá trình xóa một dịch vụ không thành công khi thông tin không tồn tại/ tồn tại sai trong cơ sở dữ liệu.

Bảng 4 Ca sử dụng xoá dịch vụ

4.1.3. Các công nghệ mã nguồn mở/ bên thứ ba sử dụng trong đồ án

4.1.3.1. IoT platform: OneM2M, OpenHAB

4.1.3.1.1. Tổng quan

Định nghĩa về IoT Platform

IoT Platform là các ứng dụng, các thư viện được xây dựng nhằm thực hiện các chức năng chính:

- Triển khai các dịch vụ, các trình quản lý, giám sát và kiểm soát các thiết bị đã kết nối.

- Thu thập dữ liệu từ xa từ các thiết bị kết nối.
- Quản lý các thiết bị.

Tính chất

Có 8 tính chất mà IoT Platform cần đáp ứng [19]:

- Kết nối và chuẩn hóa: Đem các giao thức và định dạng dữ liệu khác nhau vào trong một ứng dụng trong khi vẫn đảm bảo chính xác luồng dữ liệu và tương tác tốt với tất cả thiết bị.
- Quản lí thiết bị: Liên tục chạy các bản vá lỗi và cập nhật phần mềm sao cho đảm bảo các thiết bị đã kết nối đều hoạt động tốt.
- Cơ sở dữ liệu: Thu thập và lưu trữ dữ liệu từ các thiết bị đã kết nối.
- Quản lí các hành động: cho phép thiết lập các quy tắc để có thể thực hiện các hành động thông minh dựa trên dữ liệu đã thu thập được.
- Phân tích: Thực hiện một loạt các phân tích phức tạp từ các dữ liệu đã thu thập được để có thể đưa ra những thông tin giá trị.
- Hình ảnh: Cho phép người dùng có thể có thể xem trạng thái thiết bị, các biểu đồ trực quan thông qua các biểu đồ.
- Các công cụ khác: Cho phép các lập trình viên có thể phát triển các plugin, để tạo nên hệ sinh thái phong phú và đa dạng hơn.
- Tính khả mở: Tích hợp được với các hệ thống bên thứ 3, thông qua API hoặc các SDK.

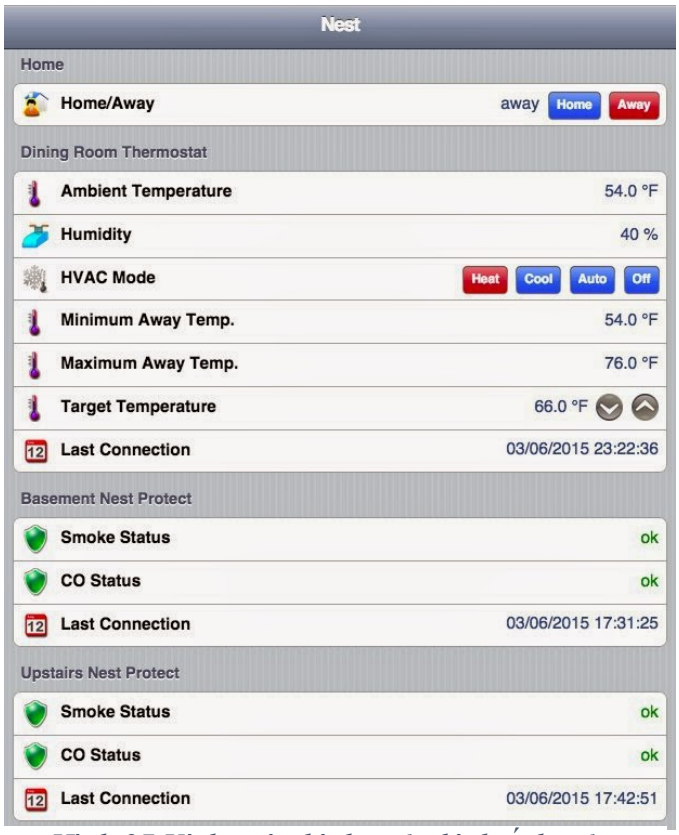
4.1.3.1.2. OpenHAB IoT Platform

OpenHAB là một IoT Platform mã nguồn mở dùng cho hệ thống nhà thông minh cho phép người dùng tích hợp các công nghệ khác nhau thành một hệ thống duy nhất. Openhab cho phép thiết lập các quy tắc tự động hóa trong hệ thống một cách mềm dẻo và cung cấp cho người dùng một giao diện điều khiển thống nhất.

Các tính năng

- Có thể chạy trên bất kì thiết bị nào có môi trường JVM (Linux, Mac, Window).
- Cho phép dễ dàng tích hợp các công nghệ (sensor, device) của các hãng khác nhau vào một sản phẩm hoàn thiện
- Có giao diện điều khiển nền web, và phiên bản mobile dành cho iOS và Android.
- Hoàn toàn là mã nguồn mở.
- Cung cấp REST API để có thể tích hợp với bên thứ 3.

Ví dụ màn hình quản lý thiết bị của OpenHAB (nguồn[5])



Hình 27 Ví dụ màn hình quản lý thiết bị của OpenHAB)

4.1.3.1.3. OneM2M Iot Platform

OneM2M là một phần mềm mã nguồn mở ứng dụng các chuẩn *oneM2M* và SmartM2M. Nó cung cấp nền tảng dịch vụ M2M (Machine to Machine) cho việc phát triển các dịch vụ một cách độc lập với các cấu hình phần cứng, nhằm mục đích tạo sự thuận lợi cho quá trình triển khai các ứng dụng theo chiều dọc và các thiết bị không đồng nhất (nguồn[6]).

Mục đích của chuẩn oneM2M là phát triển các đặc tả kỹ thuật nhằm đáp ứng nhu cầu về một lớp dịch vụ M2M (Machine to Machine: các dịch vụ có sự tương tác giữa các thiết bị) phổ biến, có thể dễ dàng nhúng trong các phần cứng và phần mềm khác nhau, cung cấp các cách kết nối vô số thiết bị với các máy chủ ứng dụng M2M trên toàn thế giới. Mục tiêu của oneM2M là đáp ứng các nghiệp vụ liên quan đến các ứng dụng M2M như: bưu chính viễn thông, giao thông thông minh, chăm sóc sức khỏe, IoT, tự động hóa công nghiệp, nhà thông minh... [14]. Có thể coi OneM2M IoT Platform như một framework, dựa vào đó người phát triển có thể tự

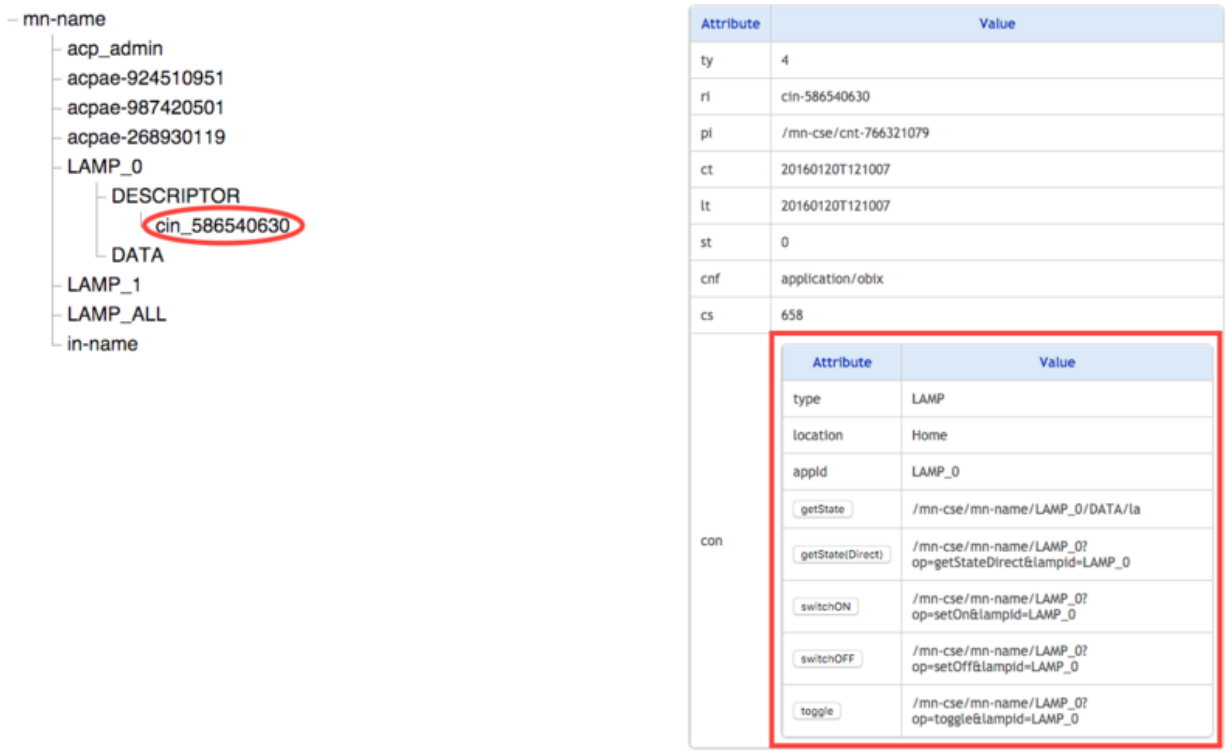
5 https://camo.githubusercontent.com/c7cd82faa47f099d190f81f2580814f6655fa4f5/687474703a2f2f7761746f752e6769746875622e696f2f696d616765732f6e6573742d62696e64696e672d6578616d706c652e6a7067

6 http://www.eclipse.org/om2m/

do, linh hoạt phát triển các ứng dụng quản lý các thiế bị IoT dựa trên các tiêu chuẩn đã được thiết lập.

OneM2M IoT Platform cung cấp giao diện trực quan cho người dùng cuối và REST API cho các nhà phát triển

Một màn hình ví dụ mô tả các chức năng quản lý một bóng đèn:



Hình 28 Giao diện quản lý của OneM2M

4.1.3.2. Message Queue: Mosquitto

Mosquitto là một phần mềm mã nguồn mở được viết bằng Python. Mosquitto đóng vai trò như một message queue, có thể dễ dàng cài đặt và chạy ổn định trên các thiết bị máy tính cỡ nhỏ như Raspberry Pi, Intel Edison, các máy điện thoại thông minh. Mosquitto được ứng dụng rất phổ biến trong các hệ thống IoT.

Message Queue là một mô hình giao tiếp truyền tin bất động bộ. Có nghĩa trao đổi giữa người gửi và người nhận không cần xảy ra đồng thời, tại cùng 1 thời điểm. Người gửi có thể đẩy tin cần gửi vào hàng đợi (queue), và sau đó một số tiến trình độc lập sẽ đẩy tin từ hàng đợi đến người nhận.

4.1.3.3. Timeseries DB: InfluxDB

InfluxDB là một cơ sở dữ liệu mã nguồn mở tối ưu cho việc lưu trữ dữ liệu theo thời gian (open source time series database). InfluxDB có một số các tính năng chính sau:

- Được xây tích hợp sẵn HTTP API.
- Có thể gắn thẻ nhãn cho dữ liệu, cho phép truy cập linh hoạt.

- Sử dụng ngôn ngữ truy vấn SQL.
- Dễ dàng cài đặt và quản lý đồng thời cũng dễ dàng truy cập.
- Được thiết kế để phản hồi các truy vấn thời gian thực, điều đó có nghĩa là mỗi diểm dữ liệu có thể được thêm vào và truy vấn ngược trở lại trong thời gian <100ms.

4.1.3.4. Metric collect agent: Heapster

Heapster là phần mềm mã nguồn mở cho phép theo dõi tài nguyên của các máy chủ và các ứng dụng chạy trong môi trường Container. Heapster hỗ trợ theo dõi, thu thập rất nhiều loại thông số đo khác nhau và có thể đẩy dữ liệu ra nhiều hệ thống hỗ trợ lưu trữ như InfluxDB, Kafka.

4.1.3.5. Docker và Container Orchestration: Kubernetes

Trong phạm vi đồ án, đồ án sử dụng Docker làm môi trường ảo hóa để vận hành các dịch vụ của hệ thống và Kubernetes để quản lý các Docker Container, giúp các dịch vụ có thể chạy ổn định trên các môi trường cài đặt khác nhau.

Docker là một nền tảng để xây dựng, vận chuyển và chạy các ứng dụng phân tán(Build-Ship-Run). Ban đầu viết bằng Python, hiện tại đã chuyển sang Go-lang. Docker đưa ra một giải pháp mới cho vấn đề ảo hóa, thay vì tạo ra các máy ảo chạy độc lập kiểu hypervisors (tạo phần cứng ảo và cài đặt hệ điều hành lên đó), các ứng dụng sẽ được đóng gói lại thành các Container riêng lẻ. Các Container này chạy chung trên nhân hệ điều hành qua LXC (Linux Containers), chia sẻ chung tài nguyên của máy mẹ, do đó, hoạt động nhẹ và nhanh hơn các máy ảo dạng hypervisors.

Kubernetes bao gồm một hệ sinh thái các phần mềm mã nguồn mở được ban đầu được phát triển bởi Google, nhằm triển khai, quản lý, lập lịch, tạo network kết nối các Docker Container trên các cụm máy chủ. Kubernetes đã giúp đơn giản hóa quá trình triển khai và vận hành các ứng dụng chạy trong môi trường Docker Container.

4.2. Kết quả cài đặt và thử nghiệm

4.2.1. Thử nghiệm

Các thử nghiệm được thực hiện để chứng minh cho khả năng thu thập dữ liệu giám sát và cung cấp một điểm nhìn thống nhất lên các dữ liệu đó của kiến trúc hệ thống đã đề xuất.

4.2.1.1. Thiết lập môi trường thử nghiệm

Các thử nghiệm sẽ được thực hiện trên 2 cụm máy chủ, mỗi cụm có triển khai hệ thống Kubernetes riêng.

- Cụm 1 đại diện cho tầng Fog, gồm 4 máy ảo, mỗi máy có cấu hình: 2 cores CPU, 2GB RAM, 40GB storage, trong đó 1 máy ảo đóng vai trò Kubernestes Master, 3 máy còn lại đóng vai trò Kubernetes Worker.
- Cụm 2 đại diện cho tầng Cloud, gồm 2 máy ảo, mỗi máy có cấu hình: 2 cores CPU, 2GB RAM, 40GB storage, trong đó 1 máy ảo đóng vai trò Kubernetes Master, máy còn lại đóng vai trò Kubernetes Worker.
- Mỗi máy ảo đã được cài đặt Docker.
- Các ứng dụng và thành phần của hệ thống sẽ được triển khai trên nền tảng Docker.

Do thử nghiệm chỉ quan tâm khối lượng, tần suất dữ liệu gửi về, không quan tâm dạng dữ liệu nên việc sử dụng các sensor giả lập cho phép dễ dàng thử nghiệm hơn, đồng thời sensor giả lập không cần quan tâm dạng phân phối giá trị của bản thân dữ liệu.

Các kịch bản thử nghiệm và kết quả được trình bày bên dưới:

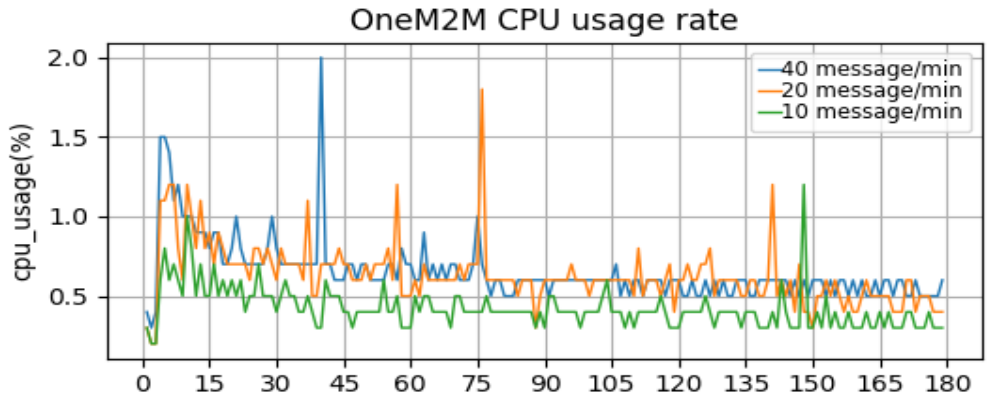
4.2.1.2. Các sensor giả lập có tần suất gửi dữ liệu khác nhau

Thử nghiệm này được tiến hành nhằm chứng minh hệ thống đã xây dựng có thể theo dõi được sự thay đổi về tài nguyên tiêu thụ của các ứng dụng ở tầng Fog, Cloud khi tần suất dữ liệu gửi đi từ các sensor thay đổi.

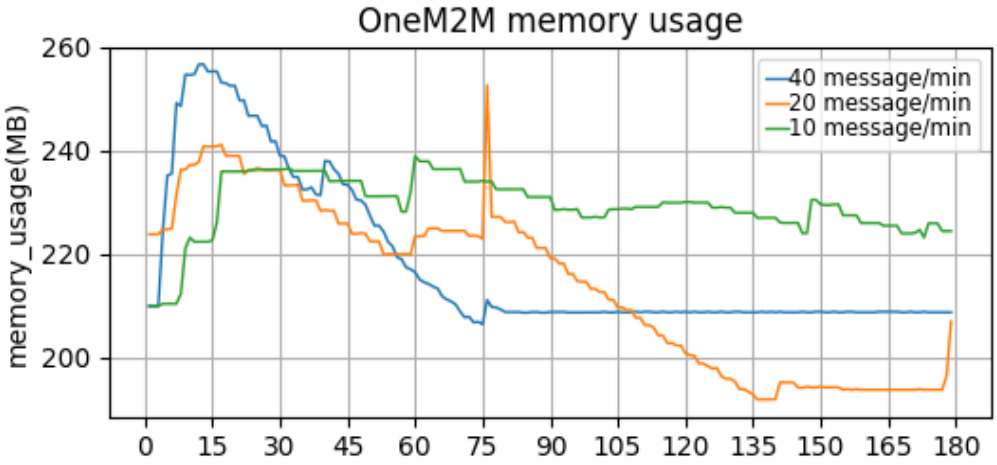
Cách thực hiện: Tiến hành chạy thử nghiệm trong 3 giờ, chạy 2 loại IoT platform OneM2M và OpenHAB, mỗi loại platform sẽ có 3 phiên bản. Mỗi phiên bản lắng nghe và quản lý các sensor có tần suất gửi dữ liệu khác nhau (10, 20 và 40 sensor). Thực hiện giám sát 2 cụm máy chủ. Các biểu đồ được vẽ trên 2 trục toạ độ, tung độ biểu thị giá trị giám sát, và hoành độ thể hiện thời gian giám sát tính theo s.

Kết quả:

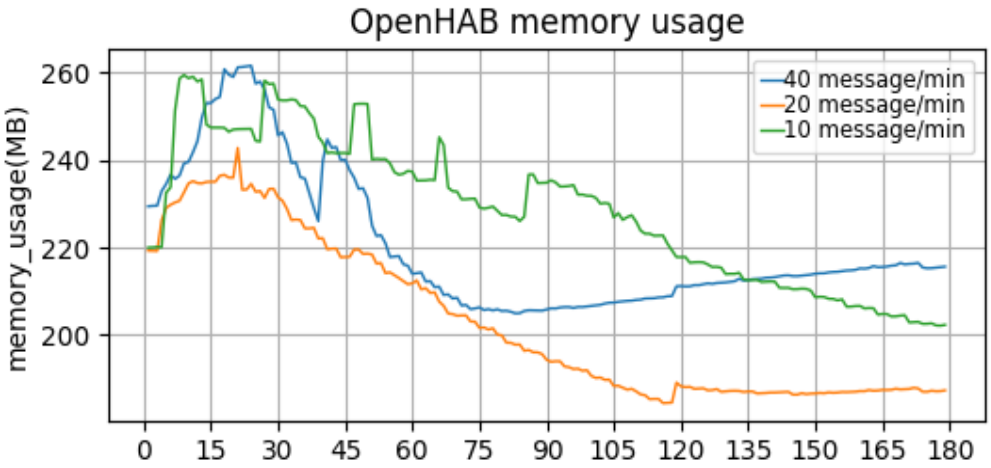
Hình từ 29-32 cho thấy lượng CPU, RAM, output network tiêu thụ có sự khác biệt giữa các phiên bản IoT platform, khi nhận dữ liệu từ sensor có tần suất gửi dữ liệu thay đổi. Có thể thấy rằng khi tần suất gửi dữ liệu từ sensor tăng lên thì lượng tài nguyên sử dụng tương ứng của các IoT platform cũng sẽ tăng do các ứng dụng này sẽ phải thu thập và xử lý một lượng dữ liệu nhiều hơn. Hình 33 thể hiện các giá trị theo dõi được từ các sensor giả lập, và hình 34 minh họa cho tần suất gửi dữ liệu khác nhau của các sensor mà hệ thống thu được.



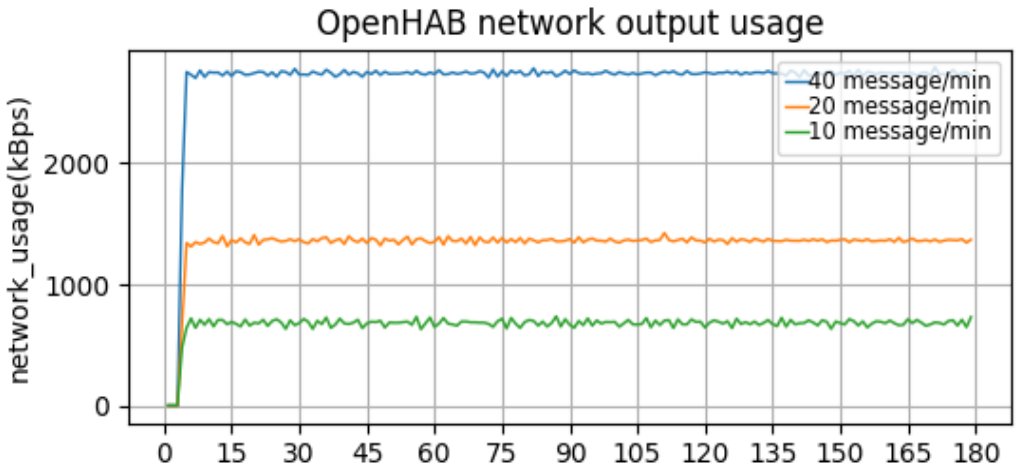
Hình 29 Lượng CPU tiêu thụ của OneM2M



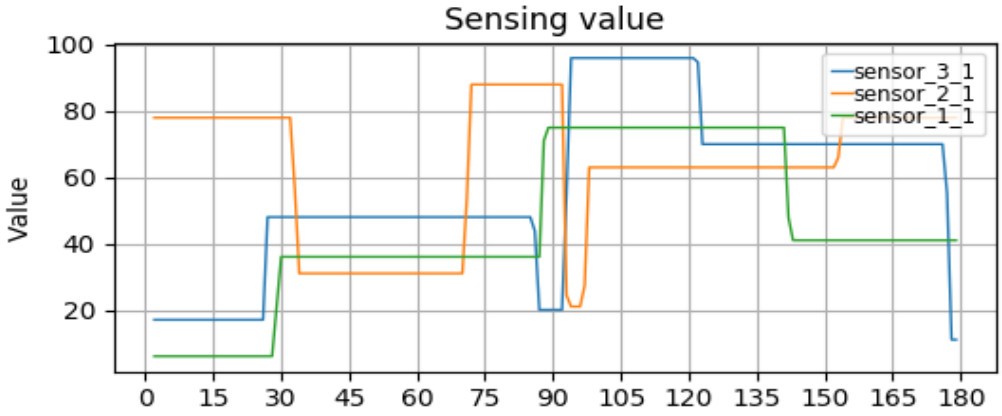
Hình 30 Lượng memory tiêu thụ của OneM2M



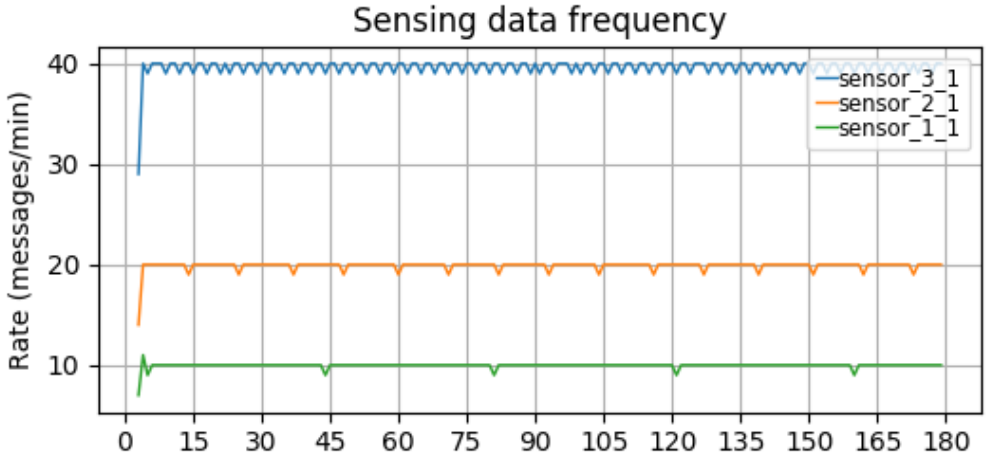
Hình 31 Lượng memory tiêu thụ của OpenHAB



Hình 31 Lượng tiêu thụ tài nguyên mạng của OpenHAB

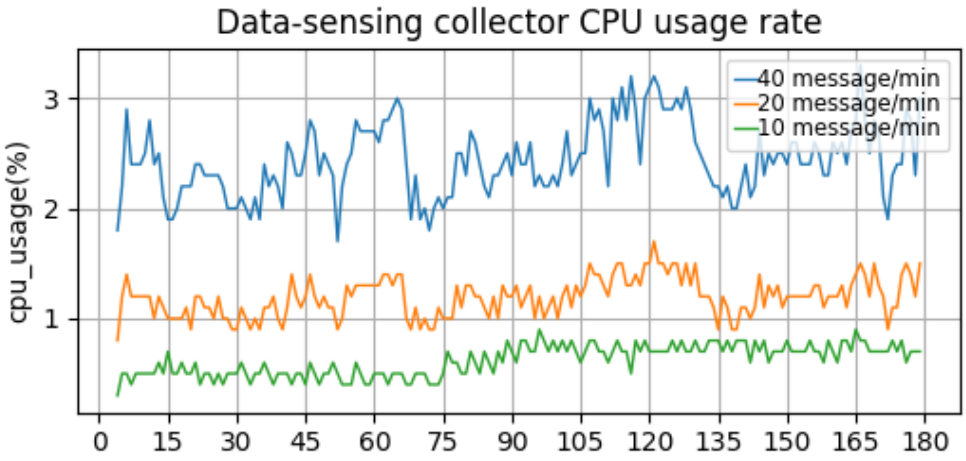


Hình 32 Các giá trị theo dõi được từ các sensor giả lập

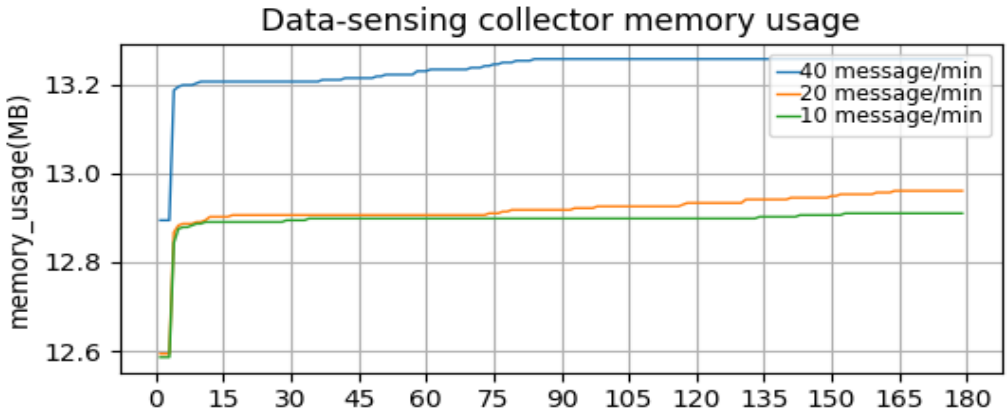


Hình 33 Tần suất gửi dữ liệu của các sensor giả lập

Thành phần nhận dữ liệu từ các IoT platform và tiến hành xử lý là các nút *Data-sensing collector* tại tầng Cloud cũng cho thấy sự khác nhau về tài nguyên tiêu thụ, phù hợp với các tần suất gửi dữ liệu khác nhau từ các sensor qua hình 35-36:

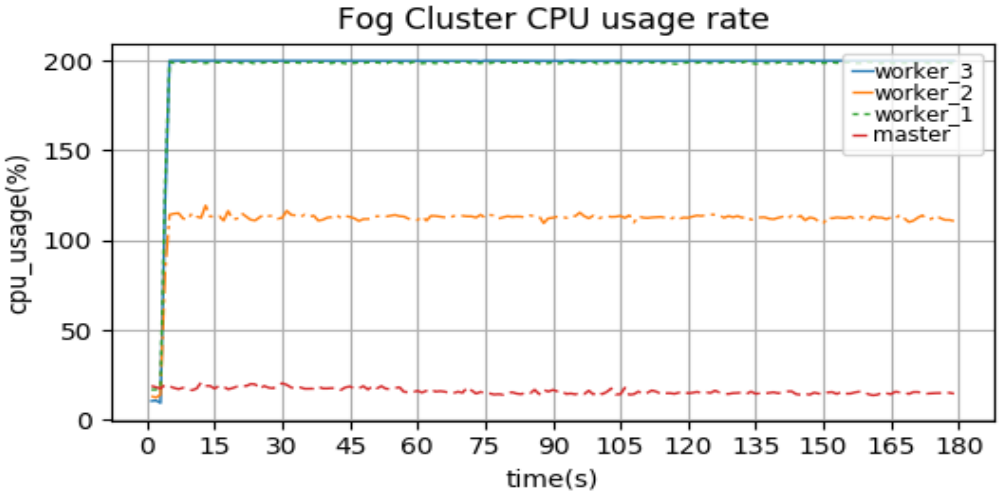


Hình 34 Lượng tiêu thụ CPU của Data sensing collector

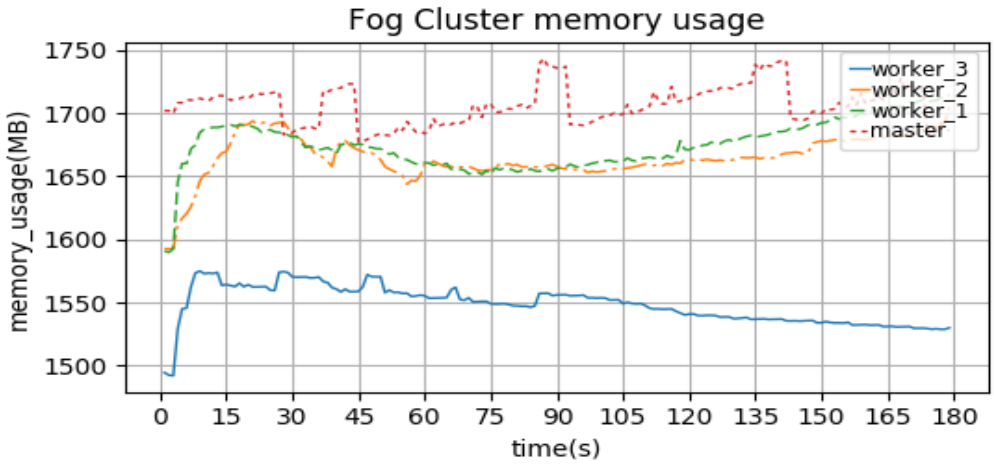


Hình 35 Lượng tiêu thụ memory của Data sensing collector

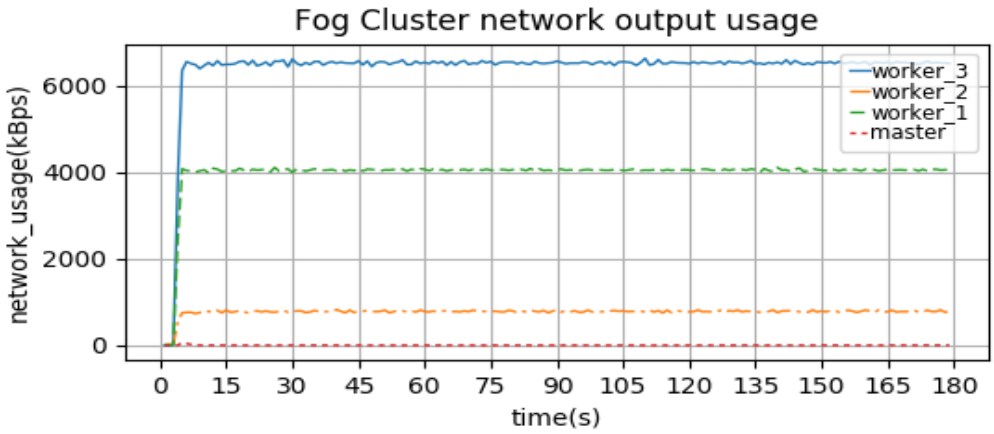
Biểu đồ sau cho thấy hệ thống có thể theo dõi được tài nguyên tiêu thụ của cụm máy ảo Fog và Cloud:



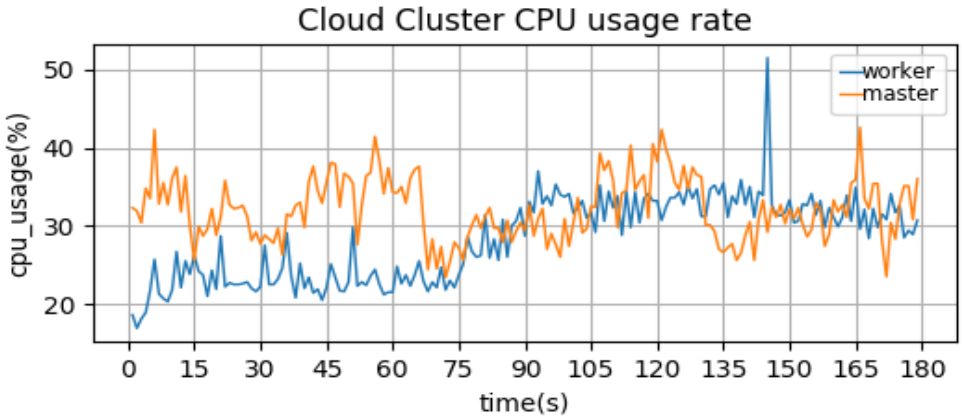
Hình 36 Lượng tiêu thụ CPU của cụm máy tầng Fog



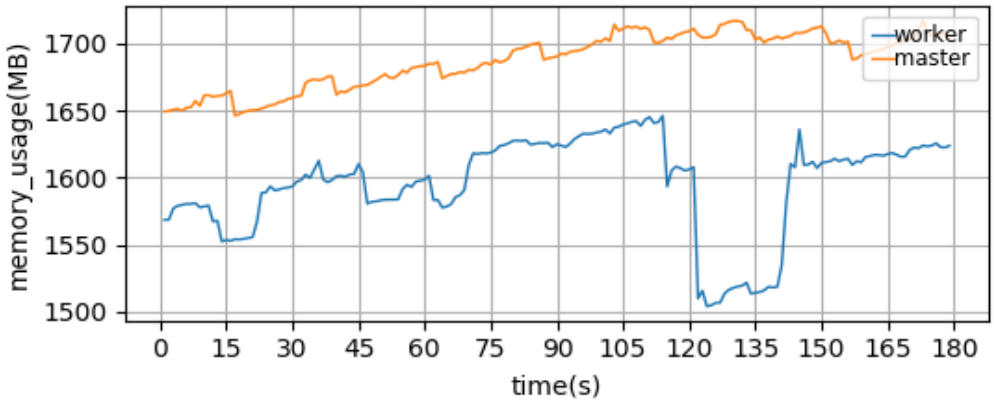
Hình 37 Lượng tiêu thụ bộ nhớ của cụm máy tầng Fog



Hình 38 Lượng tiêu thụ tài nguyên mạng của cụm máy tầng Fog



Hình 39 Lượng CPU tiêu thụ của cụm máy tầng Cloud



Hình 40 Lượng bộ nhớ tiêu thụ của cụm máy tầng Cloud

4.2.1.3. Số lượng sensor mà mỗi IoT platform quản lý là khác nhau

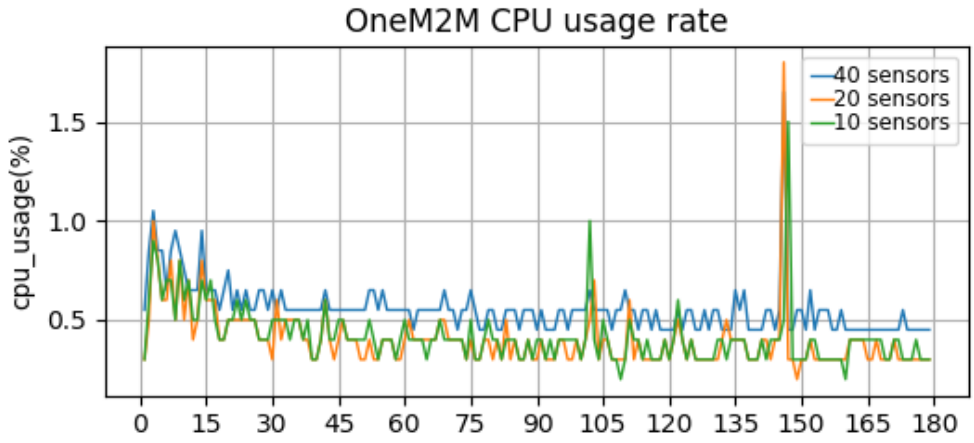
Mục đích: Thử nghiệm này được tiến hành nhằm chứng minh hệ thống đã xây dựng có thể theo dõi được sự thay đổi về tài nguyên tiêu thụ của các ứng dụng ở tầng Fog, Cloud khi thu thập dữ liệu từ một số lượng sensor khác nhau.

Cách thực hiện: Tiến hành chạy thử nghiệm trong 3 giờ, chạy 2 loại IoT platform OneM2M và OpenHAB, mỗi loại platform sẽ có 3 phiên bản. Mỗi phiên bản sẽ lắng nghe và quản lý số lượng sensor giả lập khác nhau (10, 20 và 40 sensor), mỗi sensor giả lập có tần suất gửi dữ liệu là 10 message/phút. Thực hiện monitor 2 cụm máy ảo theo thời gian thực. Các biểu đồ được vẽ trên 2 trục toạ độ, tung độ biểu thị giá trị giám sát, và hoành độ thể hiện thời gian giám sát tính theo s.

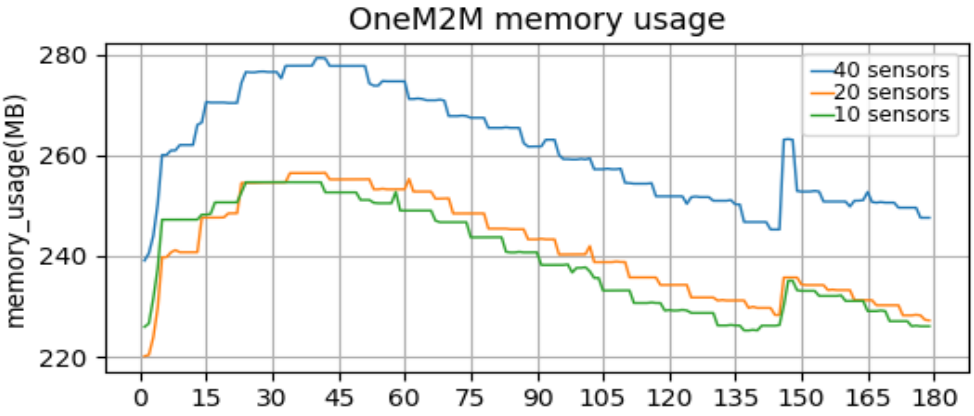
Kết quả:

Cũng cùng một kết luận với kịch bản *Các sensor có tần suất gửi dữ liệu khác nhau*, kịch bản này cho thấy tài nguyên sử dụng của một số ứng dụng trong tầng Fog sẽ khác nhau khi số lượng sensor mà mỗi ứng dụng quản lý là khác nhau.

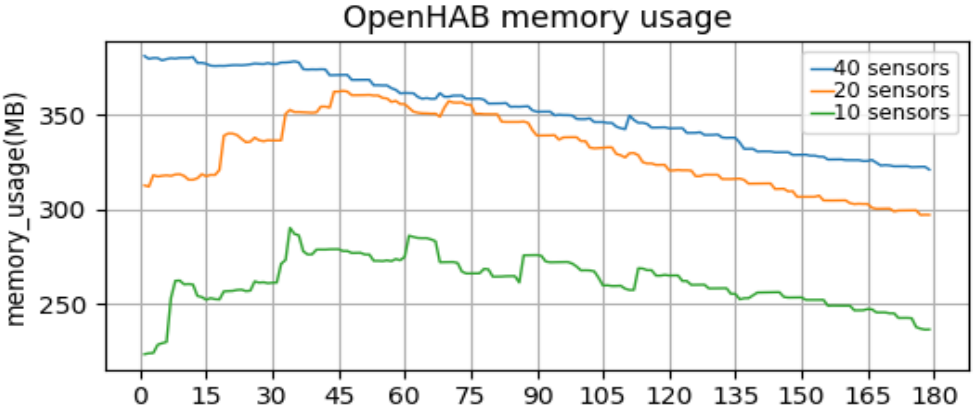
Một cách trực quan, khi quan sát các biểu đồ theo dõi tài nguyên CPU và Memory của các IoT platform từ cả kịch bản 4.2.1.2 và 4.2.1.3, có thể thấy rằng những điểm biến thiên mạnh trên biểu đồ CPU (điểm tăng đột biến) cũng tương ứng với những điểm có sự dao động mạnh trên đồ thị Memory. Nguyên nhân có thể được lý giải là do các IoT platform sử dụng trong thử nghiệm được viết trên nền tảng ngôn ngữ Java, và tại những thời điểm đó, bộ Gabarge Collector được tự động kích hoạt nhằm giải phóng bộ nhớ Heap của chương trình [17], do đó lượng CPU sẽ tăng một cách đột biến, và một lượng bộ nhớ sử dụng sẽ được giải phóng dần dần. Các hình sau cho ta thấy rõ điều này:



Hình 41 Lượng tiêu thụ CPU của OneM2M

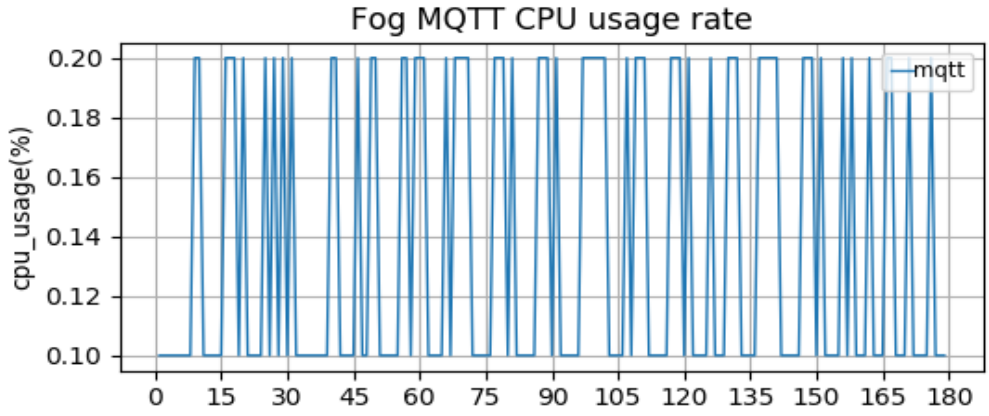


Hình 42 Lượng memory tiêu thụ của OneM2M

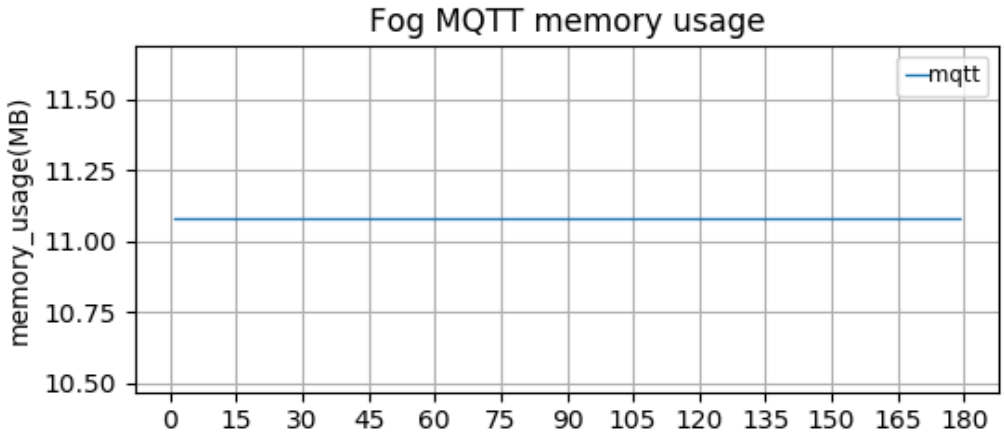


Hình 43 Lượng tiêu thụ memory của OpenHAB

Biểu đồ của Message queue cho thấy sự ổn định về lượng Memory tiêu thụ khi lượng dữ liệu truyền nhận qua là khá ổn định.

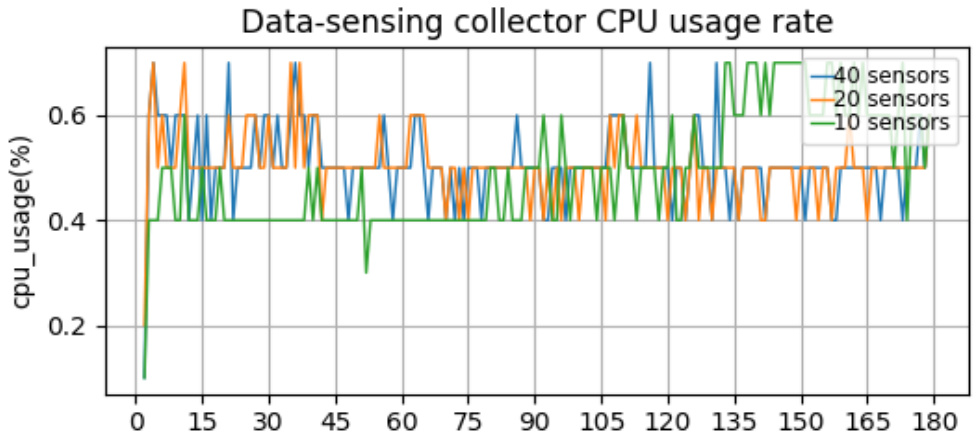


Hình 44 Lượng tiêu thụ CPU của Message Queue

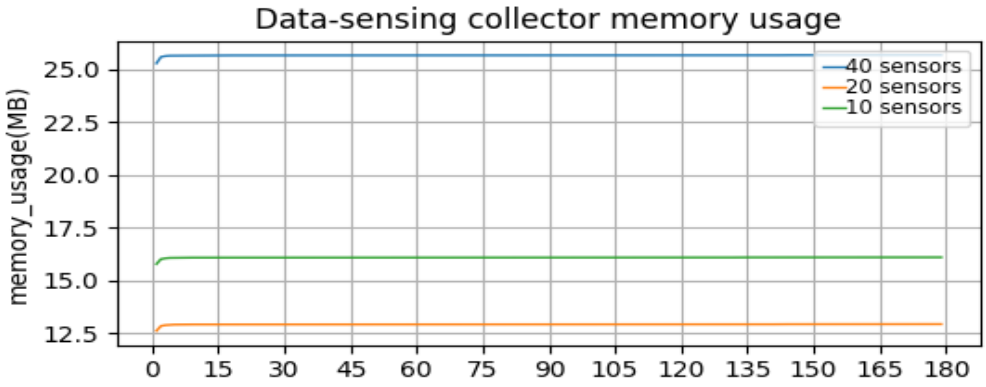


Hình 45 Lượng tiêu thụ memory của Message Queue

Biểu đồ theo dõi các thành phần tại tầng Cloud cũng cho thấy sự khác nhau về lượng tài nguyên tiêu thụ khi số lượng sensor là khác nhau:

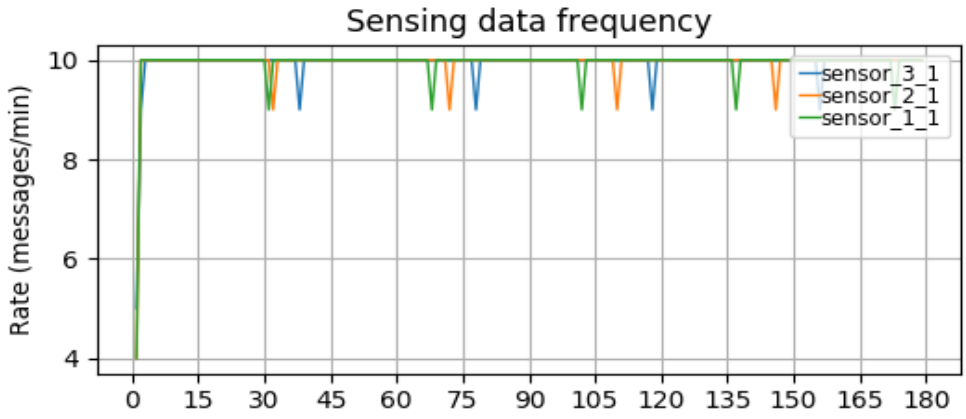


Hình 46 Lượng tài nguyên tiêu thụ của Data sensing collector khi số lượng sensor khác nhau



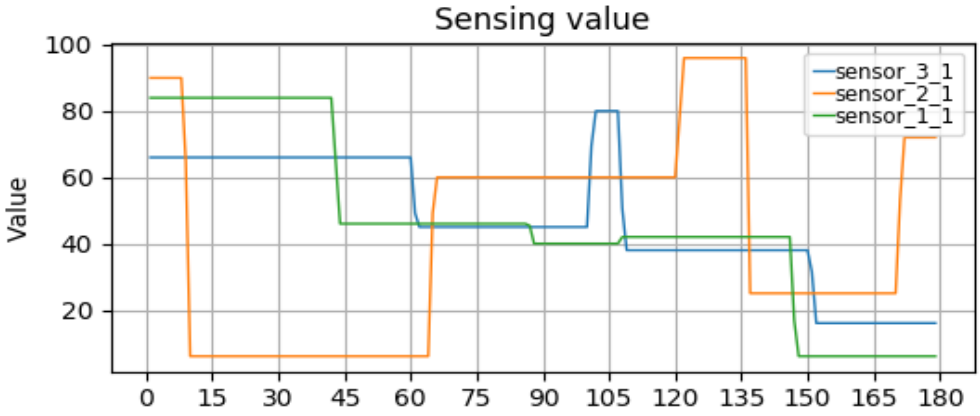
Hình 47 Lượng memory sử dụng của Data sensing collector

Tại tầng Cloud, lượng dữ liệu thu được từ các sensor trong 1 phút ổn định và bằng đúng tần suất gửi dữ liệu của các sensor đã cho thấy không có gói dữ liệu nào bị mất trong quá trình truyền dữ liệu qua các tầng:



Hình 48 Tần suất dữ liệu nhận được tại tầng Cloud

Giá trị cảm ứng đo được từ các sensor:



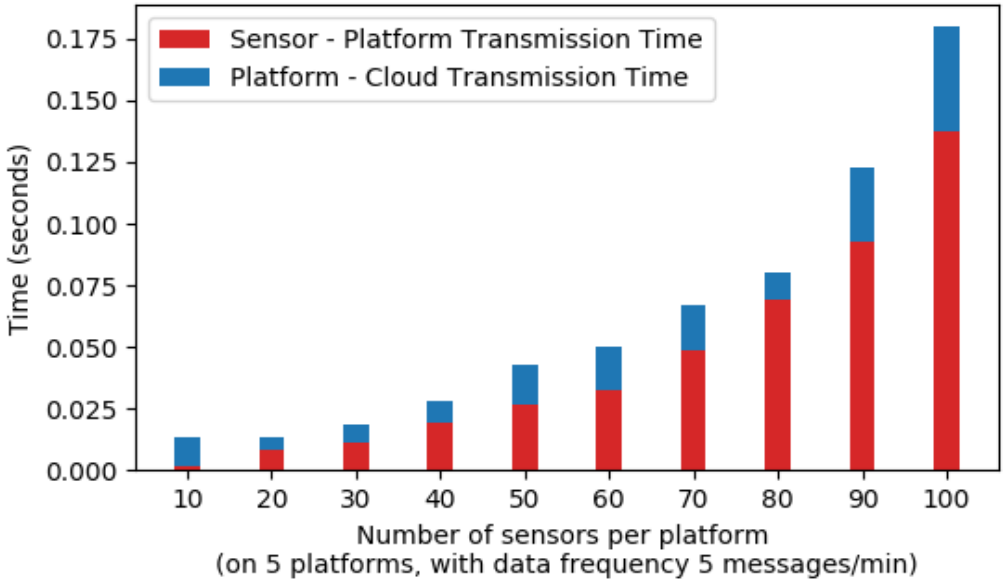
Hình 49 Giá trị cảm ứng thu được từ các sensor

4.2.1.4. Đo độ trễ dữ liệu gửi từ sensor lên tầng Cloud khi số lượng sensor tăng dần

- *Mục đích*: Thử nghiệm này được tiến hành nhằm theo dõi độ trễ về thời gian của dữ liệu được truyền từ các sensor lên tầng Cloud.

- *Cách thực hiện*: Chạy thử nghiệm với 5 platform, mỗi platform sẽ lắng nghe một số lượng sensor tăng dần (từ 10 đến 100), mỗi sensor có tần suất gửi dữ liệu là 5 messge/ 1 phút. Các message queue sẽ đóng vai trò là trung gian nhận/ gửi dữ liệu giữa các thành phần, tầng cloud và tầng Fog sẽ có 1 message queue. Thực hiện đo thời gian truyền tải qua 2 giai đoạn:
    - Giai đoạn một: từ lúc sensor gửi dữ liệu đến khi platform nhận được dữ liệu.
    - Giai đoạn hai: từ lúc platform gửi dữ liệu nhận được từ sensor, đến khi dịch vụ trên tầng Cloud nhận được gói tin đó.
    - Độ trễ truyền tải được đo bằng cách gắn nhãn thời gian (timestamp) khi một gói tin được gửi đi và khi gói tin được nhận.
- *Kết quả*:

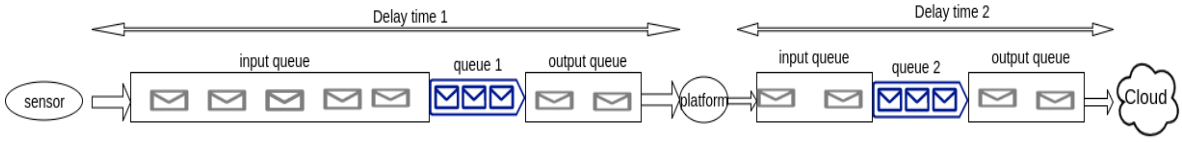
Về lý thuyết, khi số lượng sensor tăng thì lượng dữ liệu truyền tải qua các tầng cũng tăng, làm cho độ trễ truyền tải tăng lên. Số kết nối tại tầng Fog là rất lớn khi có sự tham gia của rất nhiều các thiết bị như truyền dẫn, cảm biến, thiết bị lưu trữ. Số kết nối lên tầng Cloud sẽ ít hơn, khi chỉ có một số nút tại tầng Fog sẽ được kết nối. Vì vậy độ trễ khi gói tin truyền qua các tầng sẽ có sự khác nhau. Biểu đồ sau đã thể hiện điều đó:



Hình 50 Độ trễ truyền tải khi đi qua các tầng

Từ biểu đồ có thể thấy rằng độ trễ truyền tải, từ khi một sensor gửi dữ liệu đi, đến khi một dịch vụ trên tầng cloud nhận được, đã tăng khi số lượng sensor tăng lên. (Trong biểu đồ không thể hiện thời gian xử lý dữ liệu tại các nút vì khoảng thời gian này rất nhỏ so với thời gian truyền tải). Qua hình 50, có thể dễ dàng nhận thấy là độ trễ khi dữ liệu truyền tải tại tầng Fog (từ sensor lên platform - đường màu đỏ) lớn hơn so với độ trễ truyền tải dữ liệu tại tầng Cloud (từ platform lên cloud - đường

màu xanh). Độ trễ truyền tải ở xuất phát từ quá trình các gói tin truyền tải qua các message queue và qua đường truyền Internet. Điều này có thể giải thích là do thông lượng dữ liệu tại một giai đoạn của quá trình truyền tải cao hơn giai đoạn khác. Hình sau sẽ thể hiện rõ hơn điều đó:



Hình 51 Minh hoạ quá trình truyền dữ liệu qua các tầng

Quá trình truyền tải dữ liệu giống như cơ chế đường ống (pipeline), nhiều gói tin xếp hàng trong hàng đợi *input queue* của **queue 1,** khiến cho các gói tin mất thời gian chờ (bị thắt cổ chai) để được **queue 1** đưa ra *output queue* để gửi đi, và ở đầu ra ở **queue 1** số gói tin đi ra cũng thưa hơn (nhỏ giọt), dẫn đến đầu vào *input queue* của **queue 2** cũng ít gói tin hơn, vì vậy thời gian chờ của các gói tin khi truyền qua **queue 2** cũng nhỏ hơn.

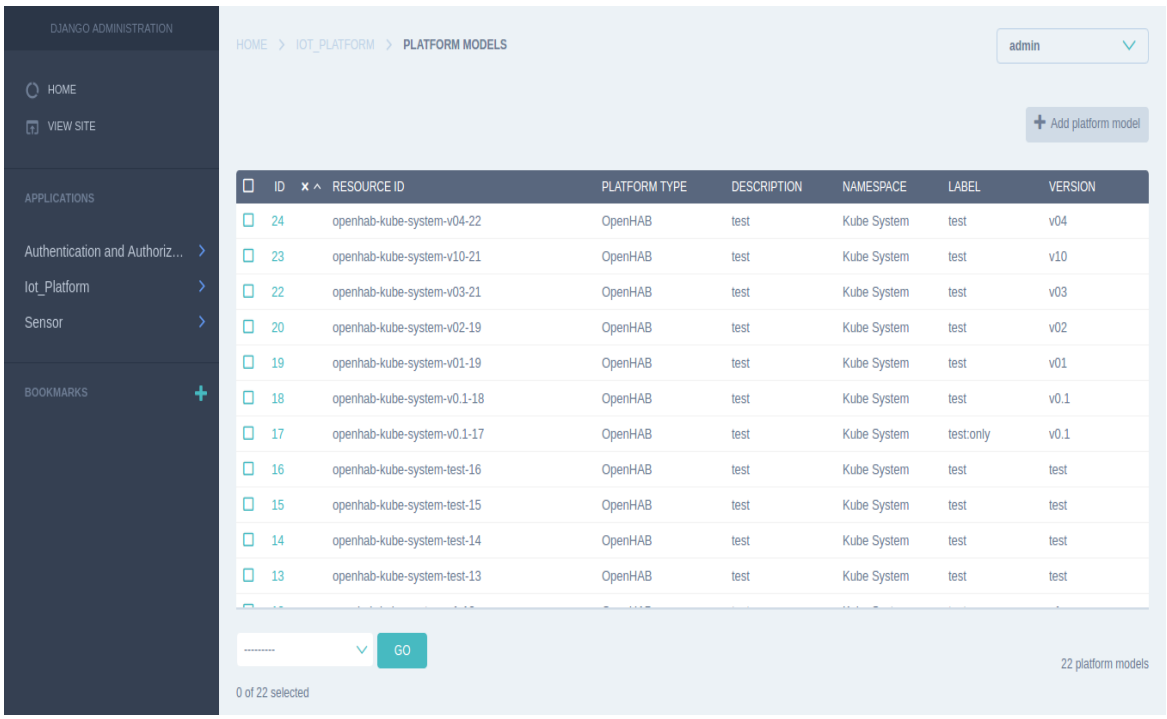
Thử nghiệm này đã cho thấy các thành phần trong tầng Fog đã gánh tải khi cần xử lý một lượng dữ liệu lớn được sinh ra, giảm mức độ chịu tải cho các dịch vụ ở tầng Cloud, tránh gặp phải tình trạng thắt cổ chai khi tất cả dữ liệu đều bị đổ dồn về một điểm duy nhất (đổ dồn về Cloud).

4.2.1.5. Kết luận

Các thử nghiệm 4.2.1.2, 4.2.1.3, 4.2.1.4 đã cho thấy hệ thống xây dựng có khả năng thu thập dữ liệu giám sát các thành phần theo thời gian thực. Hệ thống có thể giám sát được những sự thay đổi về lượng tài nguyên tiêu thụ, lượng dữ liệu trao đổi trong quá trình hoạt động của một hệ thống IoT cơ bản, trong một góc nhìn mô hình dữ liệu thống nhất, theo các tình huống mô phỏng thực tế có thể xảy ra như lượng dữ liệu tăng, lượng thiết bị sinh dữ liệu tăng lên. Các thử nghiệm đã chứng minh rằng mô hình dữ liệu thống nhất và kiến trúc hệ thống đã xây dựng có thể giải quyết được bài toán mà đồ án hướng tới.

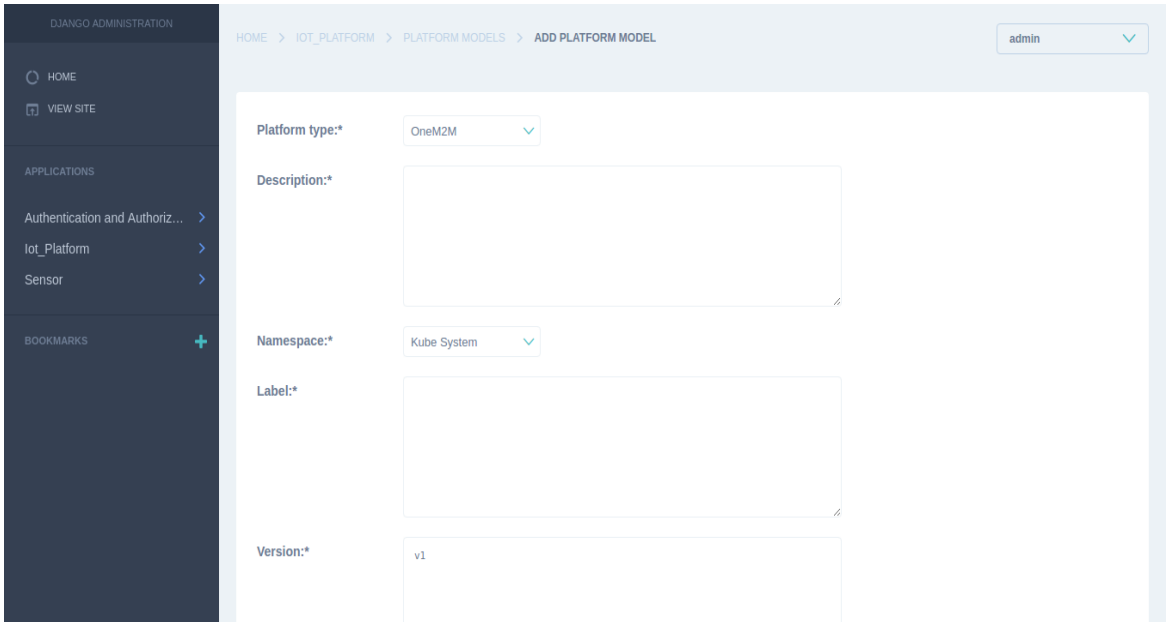
4.2.2. Kết quả cài đặt hệ thống

4.2.2.3. Giao diện quản lý dịch vụ



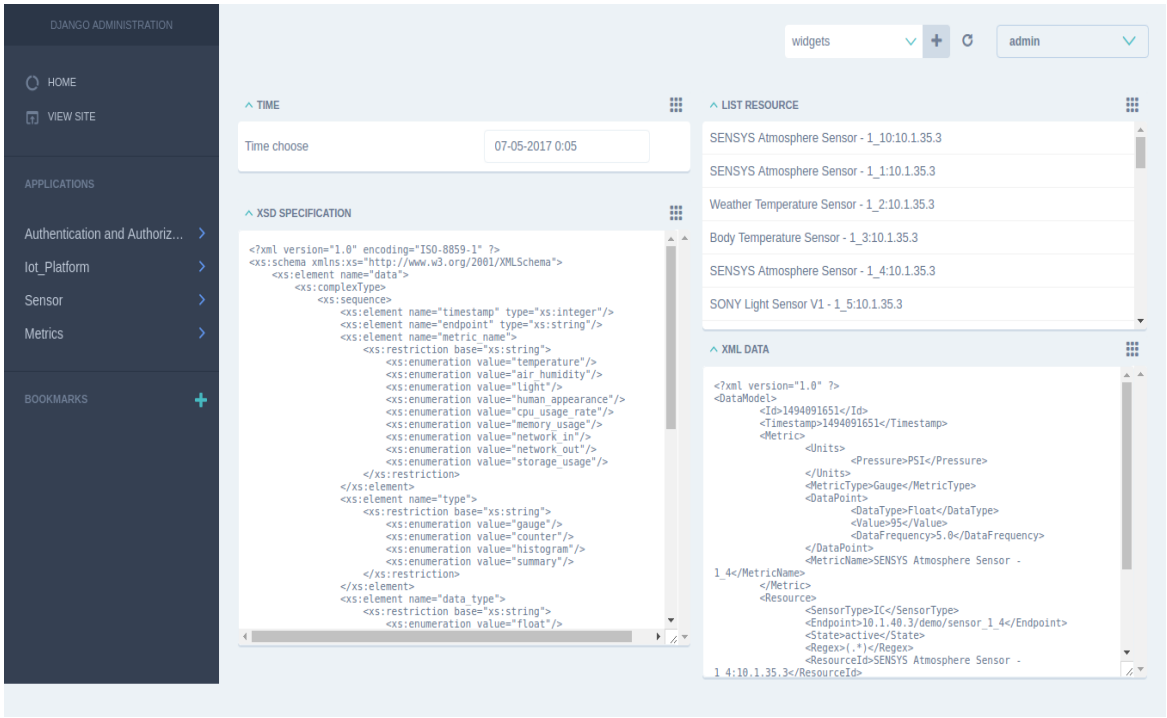
Hình 52 Giao diện quản lý dịch vụ

4.2.2.4. Giao diện thêm dịch vụ



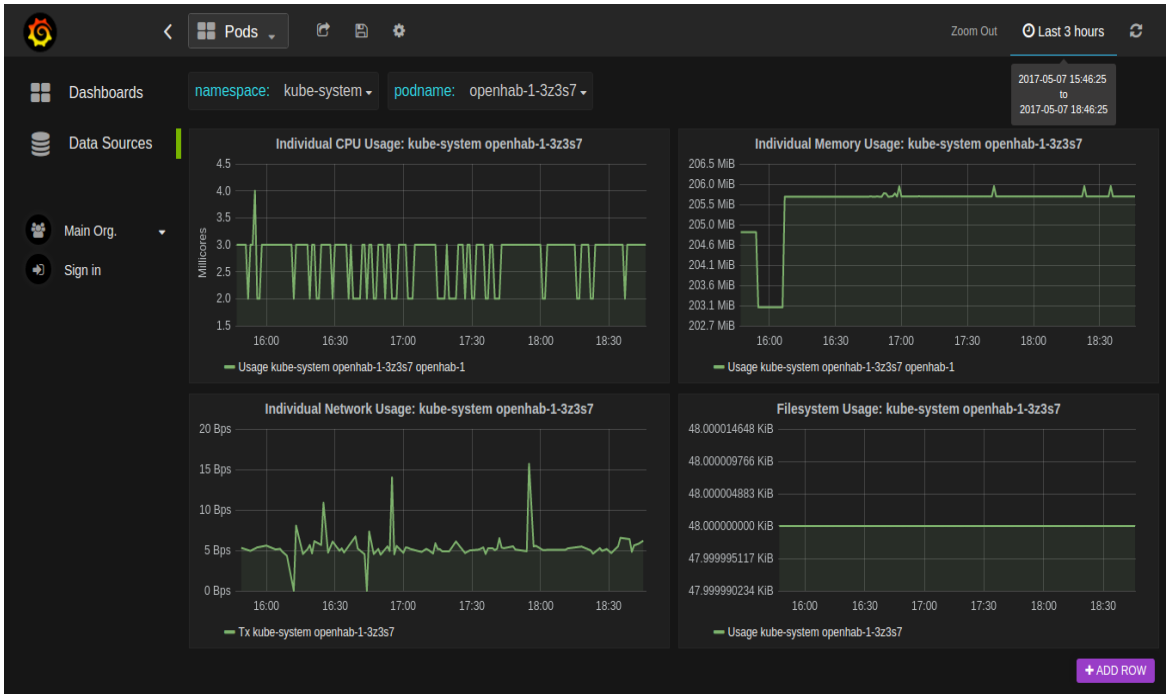
Hình 53 Giao diện thêm dịch vụ

4.2.2.5. Giao diện xem tài liệu XML mô tả các metrics.



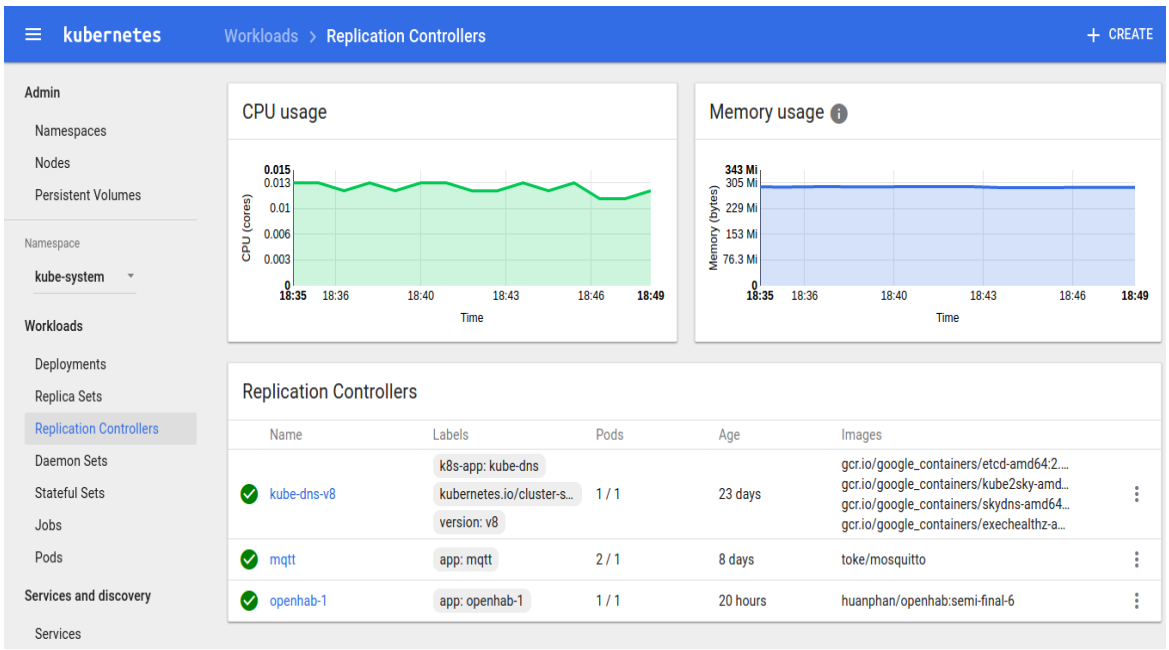
Hình 54 Giao diện xem tài liệu XML

4.2.2.6. Giao diện xem biểu đồ mô tả các metrics theo thời gian thực



Hình 55 Giao diện xem biểu đồ các số liệu theo thời gian thực

4.2.2.7. Giao diện quản lý dịch vụ của hệ thống Kubernetes



Hình 56 Giao diện quản lý dịch vụ của Kubernetes

Chương 5: Kết luận và hướng phát triển tương lai

Đồ án đã đề xuất một mô hình dữ liệu tổng quát và ý tưởng về một kiến trúc hệ thống nhằm giải quyết bài toán giám sát các thành phần trong hệ thống IoT. Qua các thử nghiệm, đồ án đã chứng minh được mô hình dữ liệu đề xuất và kiến trúc đề xuất có khả năng theo dõi được các dữ liệu cảm biến, cũng như các thông tin về tài nguyên tiêu thụ của các thành phần trong một hệ thống IoT, theo một mô hình dữ liệu thống nhất. Khả năng theo dõi được các thành phần đóng một vai trò quan trọng trong việc vận hành và đảm bảo tính hoạt động ổn định của một hệ thống đa dạng thành phần như IoT.

Trong tương lai, tôi có kế hoạch phát triển thêm module logging nhằm theo dõi quá trình vào ra của dữ liệu, tương tác giữa các thành phần, từ đó cung cấp cách quản lý toàn diện hệ thống.

Xây dựng một cách kết nối mạng các thành phần hệ thống với nhau một cách tự động, linh hoạt, tự chủ về mô hình kết nối (Software-defined Network) cũng là một bài toán quan trọng cần thực hiện trong môi trường các thiết bị IoT đa dạng, không đồng nhất và không ổn định. Các thiết bị sẽ có khả năng tự tham gia vào một mạng lưới kết nối nào đó trong phạm vi của mạng lưới. Trong phạm vi đồ án, tôi chưa giải quyết bài toán này, vì vậy trong tương lai, đó sẽ là một hướng phát triển của hệ thống.

Bất cứ một hệ thống IoT nào cũng cần các thành phần làm nhiệm vụ khai phá, xử lý, phân tích lượng dữ liệu lớn thu được. Do đó trong thời gian sắp tới, tôi sẽ phát triển thêm các ứng dụng nhằm phân tích, xử lý và ra quyết định từ dữ liệu mà hệ thống xây dựng đã theo dõi được, qua đó làm hoàn thiện hơn cho hệ thống này.

TÀI LIỆU THAM KHẢO

[1] Wikipedia, Mạng lưới vạn vật kết nối Internet, https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_l%C6%B0%E1%BB%9Bi_v%E1%BA%A1n_v%E1%BA%ADt_k%E1%BA%BFt_n%E1%BB%91i_Internet1, last visited May 2017.

[2] Suwimon, V., Sucha, S.: Internet of things: a review of applications & technologies, pp. 27-49

[3] http://www.gartner.com/newsroom/id/2636073, last visited May 2017.

[4] https://www.abiresearch.com/press/more-than-30-billion-devices-will-wirelessly-conne/, last visited May 2017.

[5] Cisco, Computing overview, last visited May 2017, https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

[6] Flavio, B., Rodolfo, M., Jiang, Z., Sateesh, A.,: Fog Computing and Its Role in the Internet of Things, pp. 13-15 (2012)

[7] https://www.linkedin.com/pulse/20140828132800-5709062-learn-fog-computing-in-5-minutes, last visited May 2017.

[8] NIST, NIST Special Publication, last visited May 2017, http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

[9] Duc, H. L., Nanjangud, N., Hong, L. T,.: HINC – Harmonizing Diverse Resource Information Across IoT, Network Functions and Clouds (2016)

[10] http://www.dfrc.ch/tag/lba-sense/, last visited May 2017.

[11] https://github.com/google/cadvisor/blob/master/docs/application_metrics.md, last visited May 2017.

[12] https://prometheus.io/docs/concepts/metric_types, last visited May 2017.

[13] https://github.com/openhab/openhab1-addons/wiki/Explanation-of-items, last visited May 2017.

[14] http://www.onem2m.org/about-onem2m/why-onem2m, last visited May 2017

[15] Cisco, The Internet of Things How the Next Evolution of the Internet Is Changing Everything, last visited May 2017, http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[16] Luigi, A., Antonio, I., Giacomo, M., The Internet of Things: A Survey (2010).

[17] Dynatrace LLC., How Garbage Collection Works, https://www.dynatrace.com/resources/ebooks/javabook/how-garbage-collection-works/, last visited May 2017.

[18] Bùi Ngọc Luân, Hệ thống tích hợp quản lý các IoT Platform, Đồ án tốt nghiệp, Đại học Bách Khoa Hà Nội, 2016
[19] Trần Đức Nhuận, PDGABPNN - Mô hình dự đoán tiêu dùng tài nguyên trên các ứng dụng trong môi trường phân tán, Đồ tán tốt nghiệp, Đại học Bách Khoa Hà Nội, 2016.
[20] https://github.com/openhab/openhab1-addons/wiki/MQTT-Binding, last visited May 2017.