SO SÁNH CÁC PHƯƠNG THỨC TRUYỀN DỮ LIỆU LÊN MÁY CHỦ TRONG IOT GATEWAY

Đào Trần Bằng[1]

[1]Công ty TNHH EoH
Email: bang@eoh.io

Keywords:
iot, cloud, protocol

TÓM TẮT:

Hiện nay ngày càng có nhiều giải pháp IoT được thiết kế và triển khai trong và ngoài Việt Nam: các dự án thành phố thông minh, các thiết bị điện tử thông minh, xe tự lái, vâng vâng...

Khi bắt đầu các dự án IoT, một trong các vấn đề mà người thiết kế giải pháp cần giải quyết chính là việc truyền tải dữ liệu từ gateway lên cloud. Có nhiều sự lựa chọn trong việc kế nối này, mỗi lựa chọn có những ưu điểm khác nhau. Nghiên cứu này nhằm chỉ ra các điểm mạnh và yếu giữa 3 phương thức là HTTP API, Websocket và MQTT.

Có những điểm khác nhau ở tầng mạng máy tính, kiến trúc phần mềm và các nhà cung cấp dịch vụ máy chủ. Tùy vào mục tiêu của giải pháp mà ta có thể lựa chọn được phương thức phù hợp nhất.

1. Mở đầu

Hiện nay ngày càng có nhiều giải pháp IoT được thiết kế và triển khai trong và ngoài Việt Nam: các dự án thành phố thông minh, các thiết bị điện tử thông minh, xe tự lái, vâng vâng...

Khi bắt đầu các dự án IoT, một trong các vấn đề mà người thiết kế giải pháp cần giải quyết chính là việc truyền tải dữ liệu từ gateway lên cloud. Có nhiều sự lựa chọn trong việc kế nối này, mỗi lựa chọn có những ưu điểm khác nhau. Trong đó có phương thức kết nối phổ biến và được nhiều người sử dụng nhất là: HTTP API, WebSocket và MQTT.

Mục tiêu chính của IoT là theo dõi, lưu trữ các dữ liệu đo được vào database, đồng thời điều khiển các sensor để thực hiện công việc khi cần thiết

Nghiên cứu này nhằm chỉ ra các điểm mạnh và yếu giữa 3 phương thức là HTTP API, Websocket và MQTT thông qua 10 đặc điểm nổi bật cũng là 10 tiêu chí đánh giá 1 giải pháp có tốt hay không. Từ đó giúp các nhà phát triển giải pháp chọn lựa được phương thức truyền dữ liệu thích hợp cho sản phẩm của mình.

2. Kết quả nghiên cứu

2.1 Khái niệm

a. IoT Gateway

IoT Gateway là một máy vi tính nhỏ có CPU, RAM, DISK… Chịu trách nhiệm chính trong việc kết nối tới các sensor, thu thập dữ liệu, và truyền tải các dữ liệu này nên máy chủ. Đồng thời IoT Gateway cũng đóng vai trò nhận các lệnh điều khiển từ máy chủ sau đó điều khiển các sensor.

Các IoT Gateway phổ biến hiện tại như Arduino ESP32, Raspberry Pi, STM32…

b. Firmware

Firmware là phần mềm được lập trình chạy riêng trên các IoT Gateway chịu trách nhiệm xử lý các logic của IoT Gateway

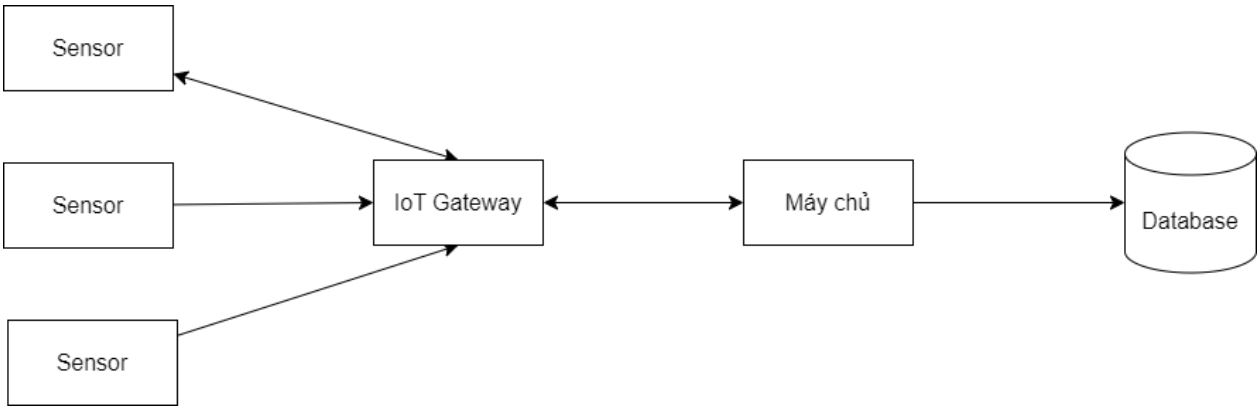
c. Máy chủ

Là phần mềm được đặt trên các máy tính được kết nối mạng internet. Các máy tính này được đặt trong các cụm máy chủ của các nhà cung cấp dịch vụ như Amazon, CMC, Google etc.

d. Sensor

Là các thiết bị phần cứng có tác dụng đo đạc hoặc thực thi: đo CO2 trong không khí, đóng tắt đèn...

2.2 Mô hình hoạt động chung của IoT



IoT Gateway nhận dữ liệu từ các Sensor, sau đó truyền các dữ liệu này lên máy chủ để máy chủ lưu vào database.

2.3 Các phương thức truyền dữ liệu

a. HTTP

IoT gateway gởi thông tin tới máy chủ thông qua các API được cung cấp bởi máy chủ.

b. WebSocket

Là giao thức cho phép máy chủ và gateway trao đổi thông tin 2 chiều với nhau. Đây là giao thức được xây dựng cho các trang web, được hỗ trợ bởi nhiều loại trình duyệt khác nhau. Tuy nhiên giao thức này cũng được hỗ trợ trong các thư viện IoT như Arduino, Raspberry, vâng vâng.

c. MQTT

Là giao thức được xây dựng nhằm tối ưu sự liên lạc giữ các thiết bị phần cứng.

2.4 Các điểm khác biệt

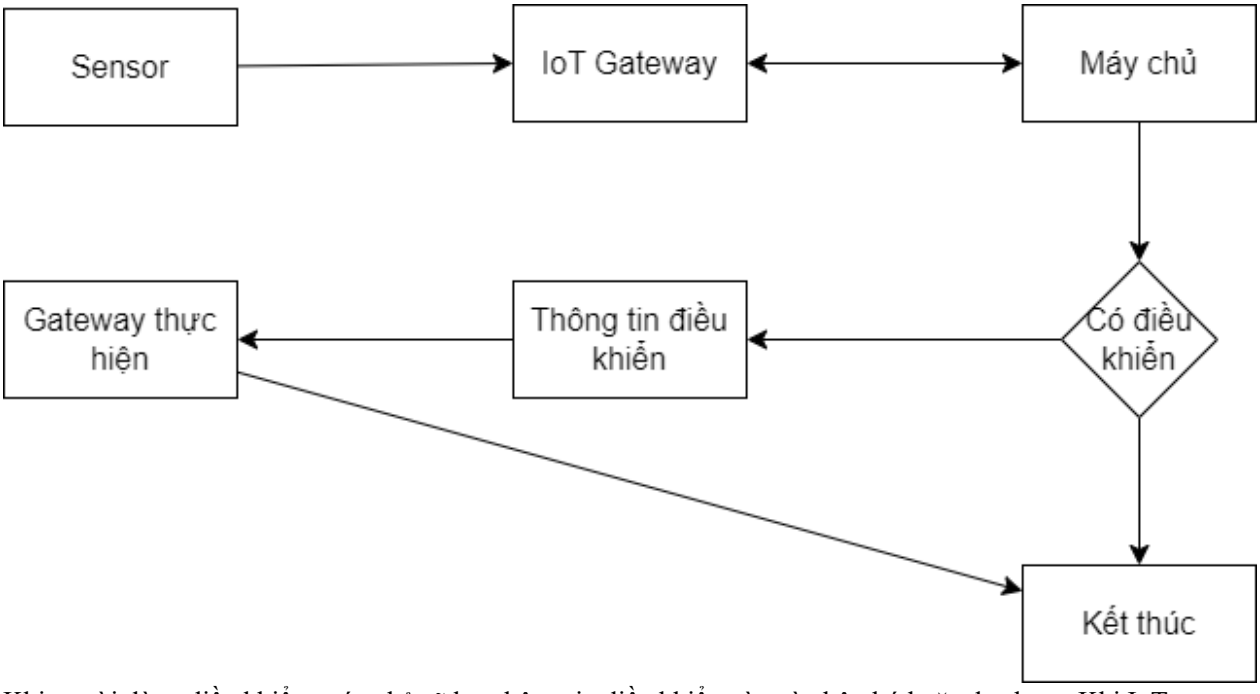
a. Truyền tải 2 chiều

HTTP không hỗ trợ truyền tải 2 chiều. Còn WebSocket và MQTT đều hỗ trợ truyền tải 2 chiều.

HTTP là giao thức chỉ cho phép sự chủ động gởi dữ liệu từ gateway đến máy chủ. Tuy nhiên với giao thức này, máy chủ không thể chủ động gởi thông tin cho IoT gateway. Trong khi đó với WebSocket và MQTT thì máy chủ được chủ động gởi thông tin tới IoT gateway khi cần thiết.

Do đó HTTP phù hợp cho các giải pháp chỉ thu thập dữ liệu, ví dụ đo chất lượng nước, đo chất lượng không khí, vâng vâng, mà không có thao tác điều khiển ngược lại từ máy chủ. Còn WebSocket và MQTT thì phù hợp cho giải pháp có sự điều khiển các thiết bị từ máy chủ.

Với HTTP, để xây dựng giải pháp điều khiển, ta phải làm như sau



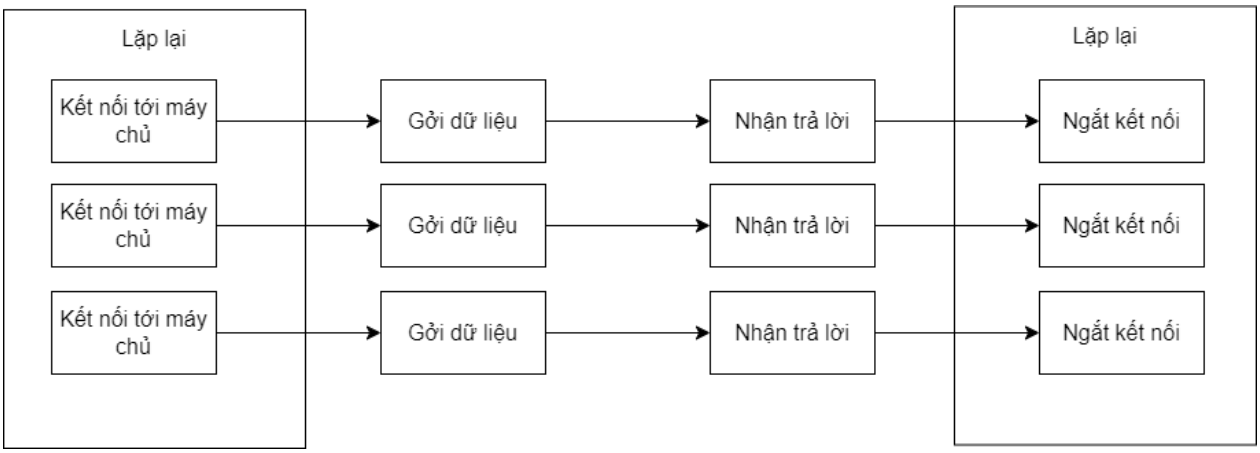
Khi người dùng điều khiển, máy chủ sẽ lưu thông tin điều khiển này vào bộ nhớ hoặc database. Khi IoT gateway gởi dữ liệu lên máy chủ, nếu có thông tin điều khiển, thì máy chủ sẽ trả về thông tin điều khiển, và IoT gateway sẽ thực hiện điều khiển này. Với cách này thì dung phương thức HTTP vẫn có xây dựng giải pháp có kèm điều khiển được, tuy nhiên sẽ bị các vấn đề sau:

- Việc điều khiển ko diễn ra lập tức mà phải chờ gateway gởi dữ liệu lên. Đó tầng suất điều khiển phụ thuộc vào tầng xuất gởi dữ liệu
- Nếu muốn việc gởi dữ liệu diễn ra nhanh thì tầng suất gởi dữ liệu cũng phải nhanh. Nhưng khi tầng suất gởi dữ liệu qua HTTP xảy ra nhanh thì sẽ gặp phải vấn đề khác, được trình bày ở mục tiếp theo.

b. Giữ kết nối

Với HTTP, sau mỗi lần gởi dữ liệu thì IoT gateway và máy chủ sẽ ngắt kết nối. Còn với WebSocket và MQTT thì kết nối sẽ được giữ lại cho việc truyền dữ liệu tiếp sau này.

Thông qua việc giữ kết nối này, WebSocket và MQTT cho phép máy chủ biết được tình trạng kết nối với gateway, đây là bài toán thường thấy với IoT. Còn với HTTP thì để xây dựng tính năng tình trạng kết nối của gateway, nhà phát triển phải xây dựng thêm.



Với việc gởi dữ liệu liên tục, thì việc giữ kết nối cũng giúp thiết kiệm nguồn lực của gateway. Việc kết nối và ngắt kết nối của HTTP sẽ tiêu tốn nguồn lực của IoT gateway, gây lãng phí năng lượng, giảm thời gian hoạt động, vâng vâng. Với cơ chế "Connection: Keep-Alive", IoT gateway vẫn có khả năng giữ kết nối với máy chủ sau khi hoàn thành một lần gởi dữ liệu., tuy nhiên cơ chế này cần phải sửa đổi firmware và thư viện khá phức tạp, đồng thời cũng không đi kèm tính năng theo dõi tình trạng kết nối như của WebSocket và MQTT.

c. Tiết kiệm dữ liệu

Một yếu tố khác cũng cần phải nhắc đến là sự tối ưu dự liệu truyền tải.

Với HTTP, mỗi một lần gởi dữ liệu, thì HTTP yêu cầu các headers (phần khai báo thuộc tính của một lần gởi) rất nhiều. Kể cả việc giữ kết nối thì mỗi một lần gởi dữ liệu cũng yêu cầu các headers phải được gởi lại. Đồng thời, nội dung trả lời từ máy chủ cũng bao gồm các headers thuộc tính.

```
curl 'https://app.e-ra.io/' \

-H 'authority: app.e-ra.io' \

-H 'accept: text/html,application/json' \

-H 'accept-language: en-US,en;q=0.9' \

-H 'cache-control: no-cache' \

-H 'user-agent: Mozilla/5.0 '
```

22:47

Trong khi đó với WebSocket và MQTT hoàn toàn không yêu cầu hoặc yêu cầu rất ít các header này.

Do đó khi cần gởi dữ liệu liên tục, thì sử dụng WebSocket và HTTP sẽ tiết kiệm dữ liệu truyền tải hơn, do đó gởi được nhanh hơn, và chi phí rẻ hơn.

d. Lưu trữ dữ liệu

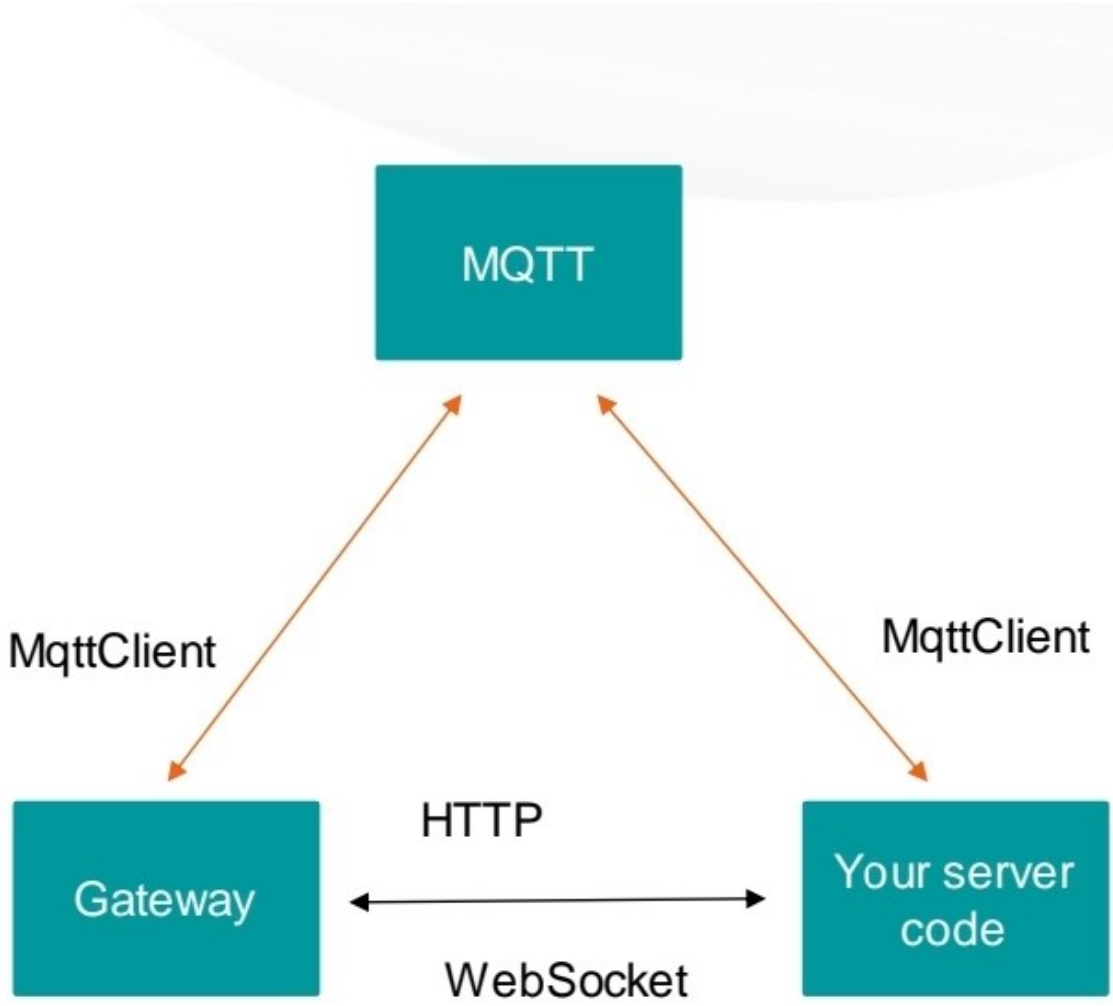
Sau khi dữ liệu được truyền đến máy chủ, thì việc lưu trữ dữ liệu cực kỳ đơn giản đối với HTTP và WebSocket, trong khi đó với MQTT thì phức tạp hơn.

Với HTTP và WebSocket, đây hoàn toàn là chương trình phần mềm do nhà phát triển xây dựng lên, cho nên việc đọc, xử lý và lưu trữ dữ liệu là điều hoàn toàn dễ dàng.

Với MQTT, nếu dùng các plugin kết nối trực tiếp MQTT vào các database, thì việc lưu dữ liệu cũng rất dễ dàng, tuy nhiên đây là các dữ liệu chưa được xử lý cơ bản, và cũng không có sự kiểm soát nào về việc dữ liệu gởi lên có đúng format hay không, có hợp lý hay không. Ngoài ra còn 1 cách khác là xây dựng cơ chế nghe dữ liệu MQTT, sau đó xử lý và lưu trữ, thì việc này khó khan hơn là HTTP và WebSocket, bởi vì 1 chương trình như vậy phải làm các việc sau:

- Kết nối database
- Kết nối MQTT
- Đọc, xử lý, lưu dữ liệu
- Chạy liên tục 24/7
- Quy trình xử lý lỗi

Trong đọc mục chạy liên tục 24/7 đối với HTTP và WebSocket là điều cực kỳ bình thường vì các thư viện đã hỗ trợ việc này từ lâu. Trong khi đó với MQTT để viết ra 1 chương trình tương tự, phải mất nhiều thời gian, công sức và khó hơn.



e. Chia sẻ dữ liệu

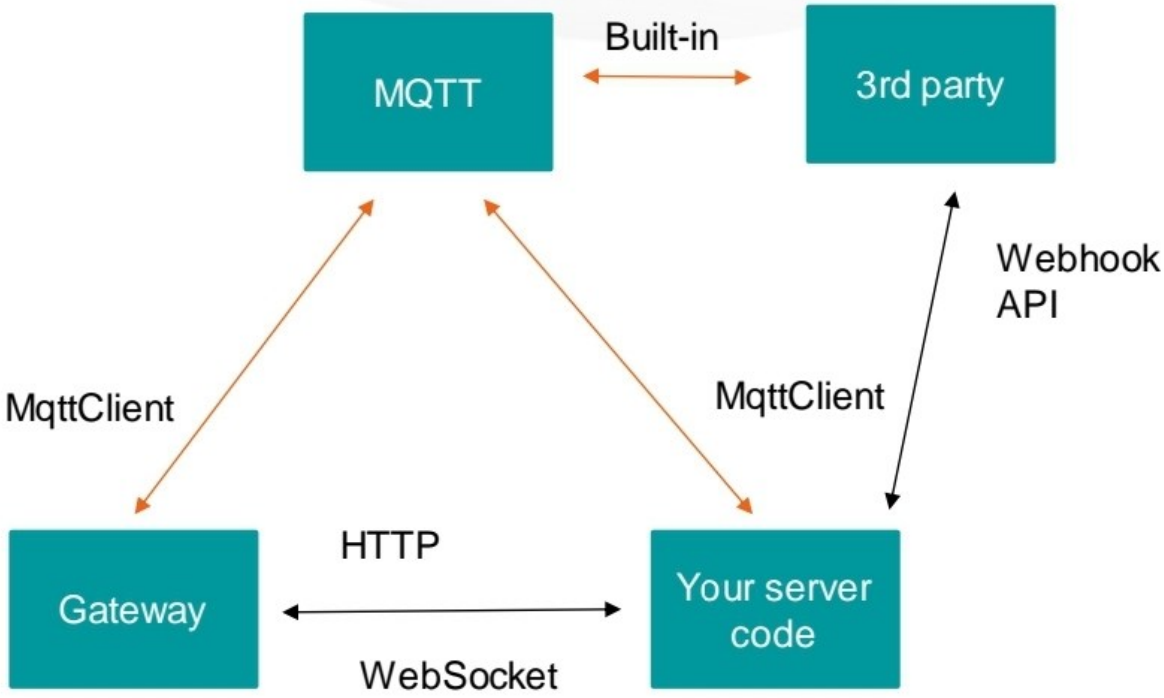
Cơ chế pub/sub trong MQTT cho phép phương thức này có thể chia sẻ dữ liệu một cách cực kỳ linh hoạt

- Chia sẻ cho nhiều nơi cùng lúc
- Chia sẻ các phần dữ liệu riêng lẻ với nhau
- Quản lý bảo mật và quyền riêng tư hiệu quả

Trong khi đó, nếu nhà phát triển muốn chia sẻ dữ liệu với một bên thứ 3 với HTTP và WebSocket, thì phải hoàn toàn xây dựng từ đầu.

Nhu cầu chia sẻ dữ liệu trong IoT là cực kỳ cần thiết và gần như bất kỳ dự án nào cũng yêu cầu tính năng này: chia sẻ thông tin IoT của nhà máy này với tổng công ty, chia sẻ thông tin của nhà máy này với nhà máy khác, chia sẻ thông tin với các ban ngành chính phủ, vâng vâng

Do đó với phương thức MQTT, sẽ tiết kiệm rất nhiều chi phí phát triển giải pháp khi có yêu cầu về chia sẻ dữ liệu.



f. Khả năng tăng tải

Trong khi WebSocket và HTTP có rất nhiều công cụ phục vụ cho việc tăng tải mà bất cứ nhà phát triển phần mềm nào cũng có thể dung tới, việc tang tải cho MQTT là một điều khó khan hơn rất nhiều.

- Kubernetes: nền tảng mã nguồn mở triển khai ứng dụng với khả năng mở rộng
- AWS Elastic BeanStalk: dịch vụ triển khai ứng dụng web của Amazon với khả năng tang tải dễ dàng
- Heroku: nền tảng triển khai ứng dụng web với hỗ trợ tang tải từ chỉ bằng một click chuột

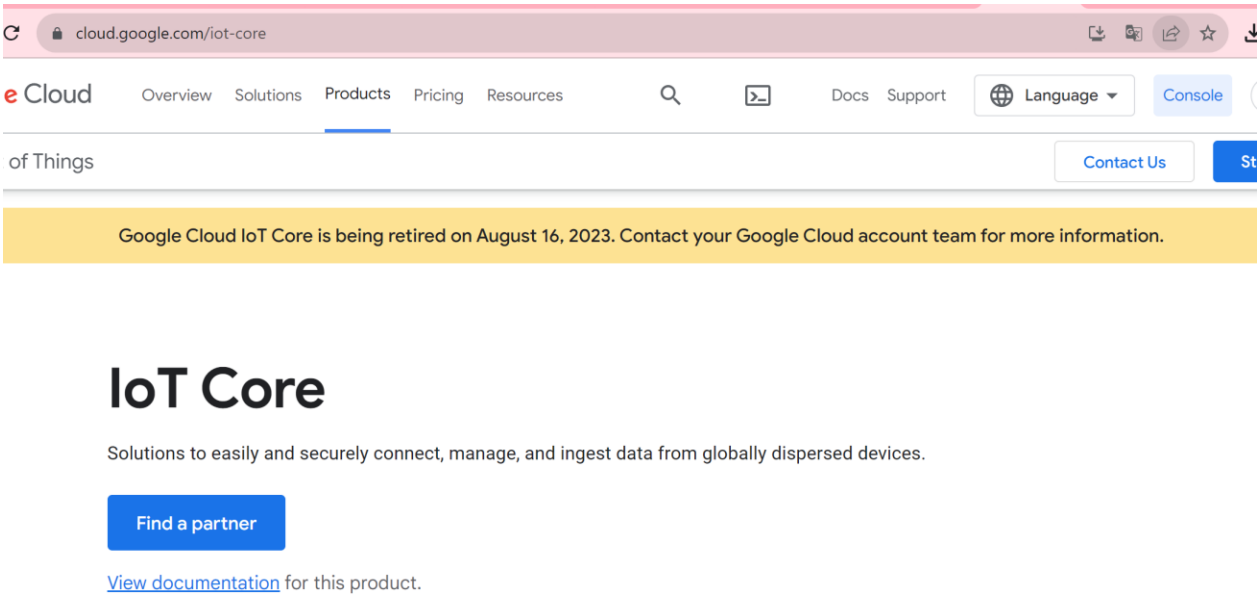
Các hệ thống nâng cấp ngang, nâng cấp dọc với các ứng dụng web đã phổ biến hơn 10 năm, thì với MQTT điều đó chỉ có một số nhà cung cấp dịch vụ MQTT hỗ trợ. Nghĩa là nếu chúng ta quyết định tự triển khai máy chủ MQTT, kể cả trên các nền tảng máy chủ đám mây như Google và Amazon thì cũng không có công cụ nào để có thể tăng tải nhiều được. Để tang tải, bắt buộc chúng ta phải cài đặt lại từ đầu, và trong khoảng thời gian nâng cấp thì dịch vụ sẽ bị đình trệ.

Do đó khi sử dụng MQTT, thông thường các nhà phát triển phải tính toán rất kỹ lưỡng về tải của máy chủ MQTT, vì một khi tính sai, rất khó để tăng giảm theo nhu cầu. Ví dụ chúng ta xây dựng một khu đô thị thông minh với 1000 nhà, thì khi có thêm 1000 nhà khác được xây dựng lên, 90% khả năng là hệ thống không hoạt động được nữa và chúng ta phải tính toán và xây dựng lại.

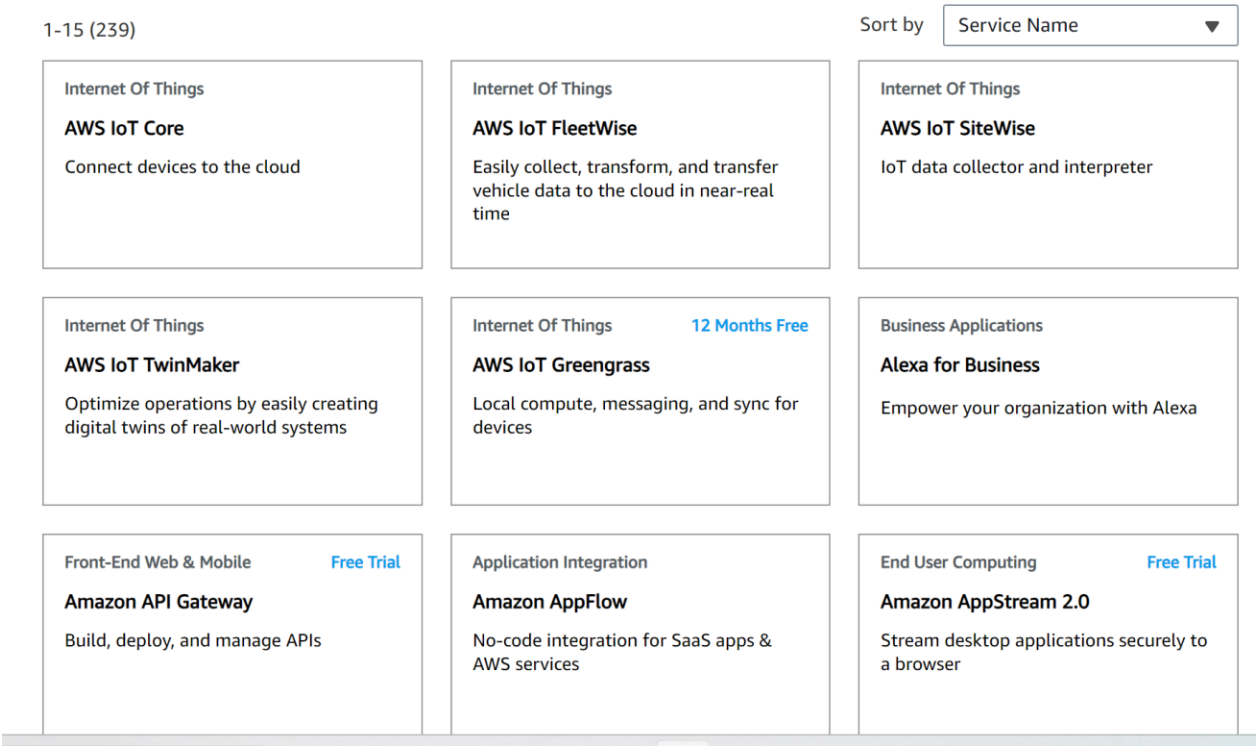
g. Các nhà cung cấp dịch vụ máy chủ đám mây

Các nhà cung cấp dịch vụ máy chủ đám mây hiện tại tập trung lớn vào các ứng dụng web/máy chủ thông thường hơn là máy chủ IoT.

Ví dụ Google đã dừng cung cấp dịch vụ IoT, Amazon thì cung cấp dịch vụ IoT core với chi phí đắt đỏ, còn các nhà cung cấp dịch vụ máy chủ trong nước như CMC thì còn sơ sài.

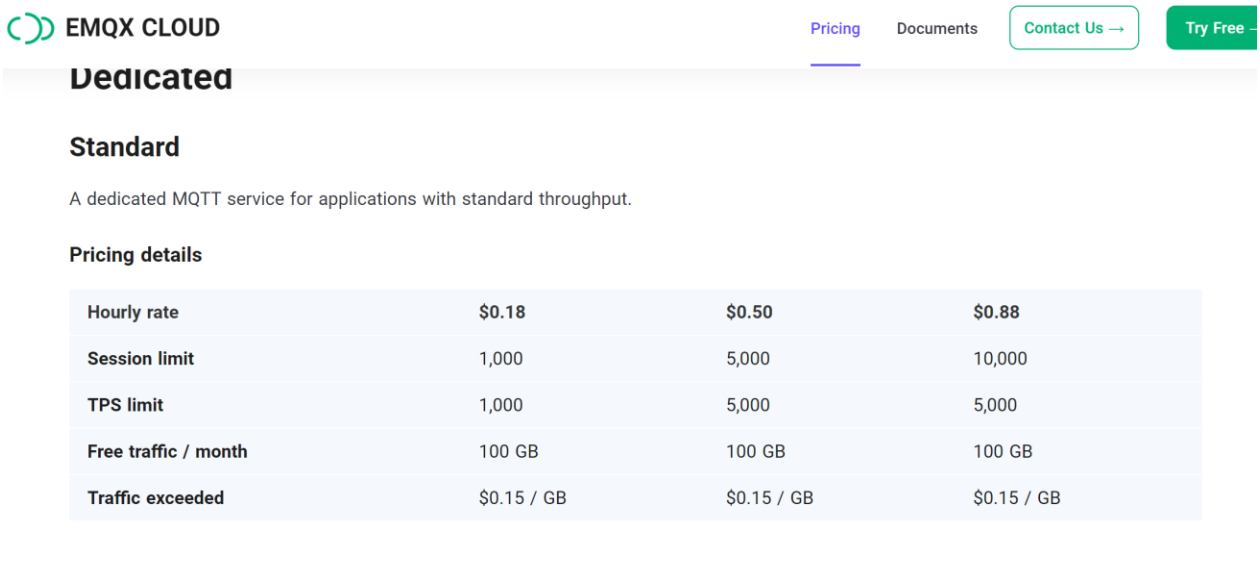


Google IoT Core sẽ dừng dịch vụ vào ngày 16 tháng 8 năm 2023

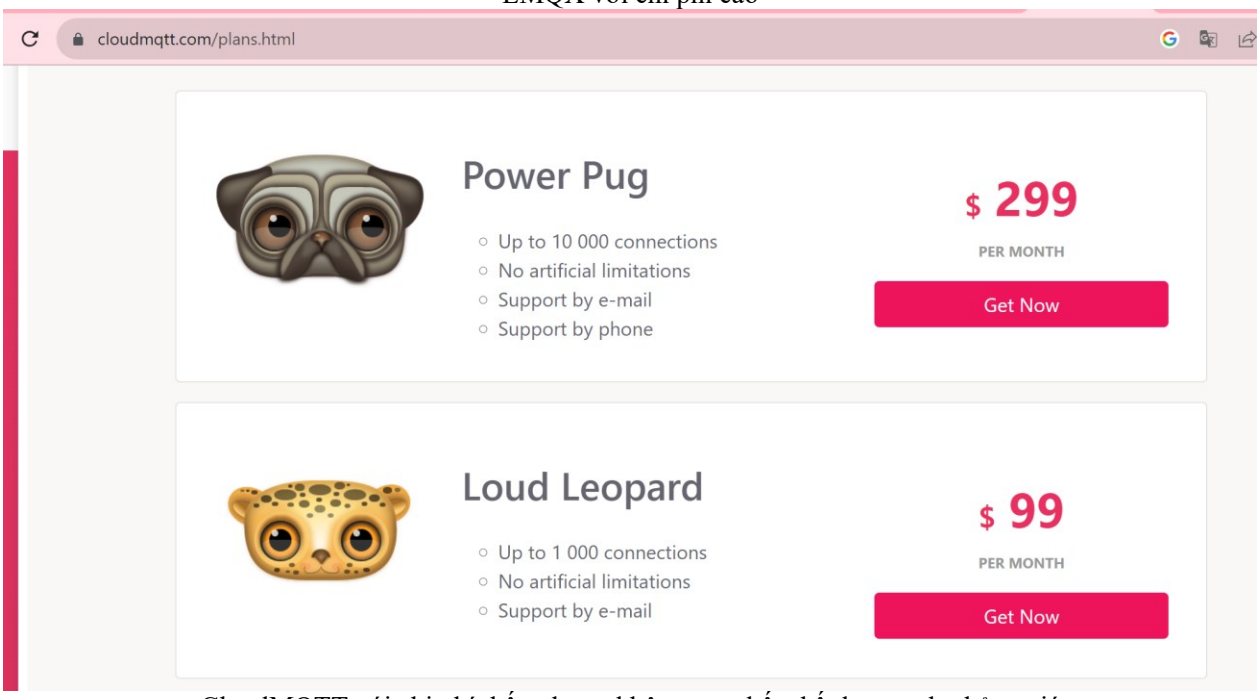


Amazon IoT với chi phí cao

Với dịch vụ máy chủ MQTT, hiện tại chỉ có 1 số ít ỏi các nhà cung cấp, với chi phí cao như Eqmx, HiveMQ hoặc với chi phí rẻ hơn như CloudMQTT, CloudAMQP thì thiếu khả năng tăng tải khi cần thiết.



EMQX với chi phí cao



CloudMQTT với chi phí thấp nhưng không cam kết chất lượng như bảng giá

Sự hỗ trợ từ các nhà cung cấp máy chủ đám mây sẽ giúp nhà phát triển giải pháp tiết kiệm dc chi phí, triển khai dễ dàng hơn và hoạt động hiệu quả hơn.

h. Các tính năng có sẵn

Ngoài việc chia sẻ dữ liệu dễ dàng, thì MQTT còn có nhiều tính năng có sẵn như
- ACL: kiểm soát quyền riêng tư theo người dung, theo từng topic
- QoS: kiểm soát việc thông tin được truyền tải đến đúng nơi nhận
- LWT: khi việc mất kết nối xảy ra thì thông báo với các bên liên quan
- Retained message: ghi nhớ nội dung được gởi cuối cùng
- Vâng vâng …

Các tính năng này cho phép nhà phát triển xây dựng rất nhiều ứng dụng/chức năng cho giải pháp của mình.

Nếu phải xây dựng lại các tính năng này từ đầu, nhà phát triển sẽ phải lãng phí một nguồn lực rất lớn và khó tập trung vào giải pháp chính của mình.

i. Khả năng tùy chỉnh

Đi kèm với các tính năng có sẵn được nhắc ở mục trên, thì khả năng tùy chỉnh MQTT sẽ khó hơn là xây dựng chức năng mới từ đầu trong HTTP và WebSocket.

Hiện tại các thư viện MQTT nổi tiếng như EMQX, HiveMQ đều là mã nguồn mở. Vì thế việc tùy chỉnh các hệ thống này theo ý đồ của nhà phát triển giải pháp là hoàn toàn có thể được. Tùy thuộc vào độ lớn của dự án chúng ta có thể chọn lựa cách tùy chỉnh cho phù hợp.

- Mã nguồn mở HiveMQ: https://www.hivemq.com/developers/community/
- Mã nguồn mở EMQX: https://www.emqx.io/downloads

2.5 Tổng hợp

	HTTP	WebSocket	MQTT
Truyền tải 2 chiều	X	Y	Y
Giữ kết nối	X	Y	Y
Tiết kiệm dữ liệu	X	Y	Y
Lưu trữ dữ liệu	Y	Y	C
Chia sẻ dữ liệu	C	C	Y
Khả năng tang tải	Y	Y	C
Nhà cung cấp dịch vụ	Y	Y	C
Tính năng có sẵn	X	C	Y
Khả năng tùy chỉnh	Y	Y	C

X: không có
Y: có
C: hạn chế

3. Kết luận

Các phương thức truyền dữ liệu từ gateway lên server khác nhau phù hợp cho các giải pháp IoT khác nhau.

Việc chọn lựa phương thức truyền dữ liệu dựa trên nhiều yếu tố từ các yếu tố kỹ thuật liên quan tới hệ thống mạng máy tính, đến các yếu tố kỹ thuật phần mềm và cả sự sẵn sàng của nhà cung cấp dịch vụ

Khi một giải pháp IoT có nhiều tính chất khác nhau thì cũng cần sự kết hợp của nhiều loại phương thức truyền dữ liệu khác nhau.

Việc chọn lựa phương thức kết nối đúng sẽ giảm các rủi ro như hệ thống dừng hoạt động bất chợt, giảm rủi ro phải xây dựng lại giải pháp cho phu hợp, giúp tiết kiệm chi phí và hiệu quả của hệ thống.