TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

----------*----------

Hồ sơ công nghệ giao thức MQTT - Message Queuing Telemetry Transport, CoAP - Constrained Application Protocol, XMPP - Extensible Messaging and Presence Protocol

Giảng viên hướng dẫn : ***PGS. TS. Phạm Văn Hải***

Sinh viên thực hiện : Phạm Minh Tâm

MSSV : 20153298

Lớp : KSTN CNTT K60

Hà Nội, Ngày 2 tháng 10 năm 2019

Danh mục hình ảnh

Danh mục bảng

Thuật ngữ

Ký hiệu viết tắt	Ý nghĩa
IoT	Internet of thing
MQTT	Message Queuing Telemetry Transport
CoAP	Constrained Applications Protocol
XMPP	Extensible Messaging and Presence Protocol
PC	Personal Computer
TCP	Transmission Control Protocol
ACK	Acknowledgment
IP	Internet Protocol
QoS	Quality of Service
UDP	User Datagram Protocol
DTLS	Datagram Transport Layer Security
SSL/TSL	Secure Sockets Layer/Transport Layer Security
XML	eXtensible Markup Language

Mục lục

Mở đầu

Theo một dự báo tới năm 2020, sẽ có khoảng 21 tỉ thiết bị được kết nối với Internet. Như vậy rõ ràng IoT là một mạng lưới khổng lồ nơi mà tất cả mọi "thứ" kết nối với nhau (con người với con người, con người với máy móc, máy với máy). Gần đây, Internet of Things còn bao gồm cả những giao tiếp theo kiểu máy với máy (M2M), hạn chế sự tác động của con người nhưng chủ yếu được áp dụng trong sản xuất năng lượng hay các ngành công nghiệp nặng. Với mạng lưới kết nối khổng lồ này đem đến nhiều lợi ích to lớn cho cả cá nhân và doanh nghiệp. Nhà thông minh ví dụ điển hình và dễ nhận thấy nhất. Với nhà thông minh thì khi không có hoạt động cần tới chiếu sáng thì toàn bộ thiết bị chiếu không cần thiết sẽ tự động tắt, khi bạn thức dậy buổi sáng, rèm cửa tự động mở, café được pha sẵn,…. Các giao thức được sử dụng trong IoT đóng một vai trò vô cùng quan trọng trong mạng lưới kết nối này.

Chuyên đề gồm các nội dung sau:

- Chương 1 : Giới thiệu tổng quan về IoT
- Chương 2 : Giới thiệu giao thức MQTT (Message Queuing Telemetry Transport)
- Chương 3 : Giới thiệu giao thức CoAP (Constrained Applications Protocol)
- Chương 4 : Giới thiệu giao thức XMPP (Extensible Messaging and Presence Protocol)
- Cuối cùng là kết luận và tài liệu tham khảo

Chương 1. Giới thiệu tổng quan về giao thức trong IoT

Internet of Things (IoT) là thuật ngữ dùng để chỉ các đối tượng có thể được nhận biết cũng như sự tồn tại của chúng trong một kiến trúc mang tính kết nối. Đây là một viễn cảnh trong đó mọi vật, mọi con vật hoặc con người được cung cấp các định danh và khả năng tự động truyền tải dữ liệu qua một mạng lưới mà không cần sự tương tác giữa con người-với-con người hoặc con người-với-máy tính. IoT tiến hoá từ sự hội tụ của các công nghệ không dây, hệ thống vi cơ điện tử (MEMS) và Internet. Cụm từ này được đưa ra bởi Kevin Ashton vào năm 1999. Ông là một nhà khoa học đã sáng lập ra Trung tâm Auto-ID ở đại học MIT [2].

Khái niệm Internet of Things trở nên rõ ràng vào năm 2005 khi International Telecommunications Union – ITU công bố bản báo cáo đầu tiên về chủ đề này. Báo cáo nêu: IoT sẽ kết nối các vật thể theo cả 2 cách thông minh và có cảm nhận thông qua sự phát triển kỹ thuật liên kết trong nhận biết thông tin (theo các vật thể), các cảm biến và mạng cảm biến không dây (cảm nhận vật thể), các hệ thống nhúng (suy nghĩ về vật thể) và công nghệ nano (thu nhỏ vật thể). Trong báo cáo ITU cũng xác định các thử thách quan trọng cần giải quyết để khai thác hết tiềm năng của IoT – tiêu chuẩn hóa và sự kết hợp, bảo mật, và các vấn đề đạo đức – xã hội [1].

Ngày nay với khoảng 1,5 tỷ máy tính và trên 1 tỷ điện thoại có kết nối Internet. Sự hiện diện "Internet of PCs" sẽ được chuyển sang IoT trong đó 50-100 tỷ thiết bị kết nối Internet trong năm 2020. Một vài nghiên cứu còn chỉ ra trong cùng năm đó, số lượng máy móc di động sẽ tăng gấp 30 lần so với hiện nay. Nếu không chỉ xem xét các kết nối máy với máy mà là các kết nối giữa tất cả các vật thể thì số lượng kết nối có thể tăng lên tới 100.000 tỷ [1].

Với những hiệu quả thông minh rất thiết thực mà IoT đem đến cho con người, IoT đã và đang được tích hợp trên khắp mọi thứ, mọi nơi xung quanh thế giới mà con người đang sống. Một số ứng dụng phổ biến của IoT như [1]:

- Thành phố thông minh – smart cities [1]
    - Công viên thông minh: giám sát không gian đỗ xe của thành phố.
    - Kiểm tra xây dựng: giám sát các rung động và các điều kiện vật chất trong các tòa nhà, cầu và công trình lịch sử.

  - Bản đồ tiếng ồn thành phố: giám sát âm thanh trong các phạm vi quán bar và các khu trung tâm theo thời gian thực.
  - Tắc nghẽn giao thông: giám sát các phương tiện và mức độ người đi bộ để tối ưu việc lái xe và đi lại.
  - Chiếu sáng thông minh: chiếu sáng thông minh và tương ứng với thời tiết trong hệ thống đèn đường.
  - Quản lý chất thải: phát hiện mức độ rác thải trong các container để tối ưu đường đi thu gom rác.
  - Hệ thống vận tải thông minh: các tuyến đường và cao tốc thông minh với các thông điệp cảnh báo và các điều chỉnh theo điều kiện thời tiết và các sự kiện không mong muốn như tai nạn, tắc đường.
- Môi trường thông minh – Smart Environment [1].
  - Phát hiện cháy rừng: giám sát khí gas đốt cháy và các điều kiện cảnh báo cháy rừng để đưa ra vùng cảnh báo.
  - Ô nhiễm không khí: điều khiển khí CO2 thải ra từ các nhà máy, các khí gây ô nhiễm từ phương tiện và khí độc trong các nông trại.
  - Phòng ngừa lũ quét và lở đất: giám sát độ ẩm đất, các rung chấn và mật độ đất để phát hiện các mối nguy hiểm theo điều kiện đất.
  - Phát hiện sớm động đất: phân bố giám sát ở các vùng đặc biệt có rung chấn.
- Nước thông minh – Smart Water [1].
  - Chất lượng nước: nghiên cứu về sự thích hợp của nước trên các sông, vùng biển đối với hệ động vật và tiêu chuẩn nước để sử dụng.
  - Rò rỉ nước: phát hiện chất lỏng bên ngoài các két và các biến động áp suất bên trong các đường ống.
  - Lũ trên các con sông: giám sát biến động mực nước trên các sông, đập và nguồn nước.
- Đo lường thông minh – Smart Metering [1].
  - Mạng lưới thông minh: giám sát và quản lý tình hình tiêu thụ năng lượng.
  - Tính hình các bể chứa: giám sát mức nước, dầu, khí ga trong các két và bể chứa.
  - Cài đặt hệ thống quan điện: giám sát và tối ưu hiệu quả của các thiết bị năng lượng mặt trời.
  - Lưu lượng nước: đo lường áp suất nước trong các hệ thống dẫn nước.
  - Tính toán trữ lượng hàng: đo lượng mức độ còn và khối lượng hàng hóa.

- Bảo mật và tình trạng khẩn cấp – Security and Emergencies [1].
    - Kiểm soát vùng hạn chế truy nhập: điều khiển truy nhập tới các vùng hạn chế và phát hiện người không phận sự.
    - Sự có mặt của chất lỏng: phát hiện chất lỏng trong các trung tâm dữ liệu, kho dữ liệu và vị trí xây dựng nhạy cảm để chống sự hỏng hóc và ăn mòn.
    - Mức bức xạ: phân phối đo lường phạm vi bức xạ xung quanh các khu năng lượng hạt nhân để cảnh báo rò rỉ kịp thời.
    - Các khí nguy hiểm và nổ khí: phát hiện mức độ khí gas và rò rỉ trong các môi trường công nghiệp, xung quanh các nhà máy hóa chất và trong các mỏ.
- Hệ Thống bán lẻ – Retail [1]
    - Thanh toán NFC: quá trình chi trả dựa trên vị trí hoặc khoảng thời gian hoạt động để chuyên chở công cộng, thể thao, các công viên chủ đề…
    - Điều khiển băng chuyền: giám sát các điều kiện lưu trữ trong các dây chuyền và theo dõi sản phẩm cho mục đích giám sát.
    - Các ứng dụng mua sắm thông minh: đưa ra lời khuyên tại các thời điểm bán theo thói quen khách hàng, sở thích, các thành phần gây phản cảm với người mua hoặc hạn sử dụng.
    - Quản lý sản phẩm thông minh: điều khiển sự luân phiên các sản phẩm trong giá và kho để tự động các quá trình bổ sung.
- Điều khiển trong công nghiệp – Industrial Control [1].
    - Các ứng dụng M2M: kiểm soát tài sản và các máy tự chẩn đoán.
    - Chất lượng không khí trong nhà: giám sát khí độc và mức độ khí oxi trong các thiết bị hóa học để đảm bảo cho công nhân và hàng hóa an toàn.
    - Giám sát nhiệt độ: kiểm soát nhiều độ trong công nghiệp và tủ y học với hàng hóa nhạy cảm.
    - Sự có mặt của khí Ozone: kiểm soát mức độ khí Ozone trong khi làm khô các sản phẩm thịt trong các nhà máy thực phẩm.
    - Định vị trong nhà: vị trí các tài sản trong nhà bằng cách sử dụng các thẻ tích cự (ZigBee) và bị động (RFID/NFC).
    - Các phương tiện tự chẩn đoán: thu thập thông tin từ CanBus để gửi các cảnh báo thời gian thực để nhanh chóng đưa lời khuyên cho lái xe.
- Nông trại gia súc thông minh – Smart Animal Farming [1].
    - Chăm sóc: kiểm soát các điều kiện phát triển của con non trong trang trại gia súc để đảm bảo sống sót và khỏe mạnh.

  - Theo dõi gia súc: vị trí và định danh của các gia súc trên đồng cỏ và phạm vi chăn thả.
  - Mức độ khí độc: nghiên cứu thông hơi và chất lượng không khí trong các nông trại và phát hiện khí có hại.
- Tự động nhà và domotic [1].
  - Sử dụng năng lượng và nước: hỗ trợ giám sát tiêu dùng nước và năng lượng để cho lời khuyên làm thế nào để tiết kiệm chi phí và tài nguyên.
  - Các thiết bị điều khiển từ xa: bật tắt bằng thiết bị điều khiển từ xa để tránh tai nạn và tiết kiệm năng lượng.
  - Các hệ thống phát hiện xâm nhập: phát hiện cửa sổ và cửa ra vào và các xâm phạm để chống đột nhập.
  - Bảo quản hàng hóa và nghệ thuật: giám sát các điều kiện bên trong bảo tang và nơi lưu trữ nghệ thuật.
- Chăm sóc sức khỏe – eHealth [1].
  - Phát hiện ngã: giúp đỡ người già hoặc người tàn tật sống độc lập.
  - Các tủ thuốc: giám sát các điều kiện bên trong tủ lạnh lưu trữ vaccine, thuốc, và các yếu tố hữu cơ.
  - Chăm sóc vận động viên: giám sát các dấu hiệu trong trung tâm chăm sóc.
  - Giám sát bệnh nhân: giám sát điều kiện của các bệnh nhân trong bệnh viện và tại nhà họ.
  - Bức xạ tử ngoại: đo lường lượng tia mặt trời UV để cảnh bảo mọi người không phơi nắng trong những giờ nhất định.

Để tối ưu hóa được mô hình Internet of Things - IoT, các thiết bị phải giao tiếp với nhau bằng các "giao thức" - protocol mà đáp ứng được 2 tiêu chí [10]:

- Năng lượng tiêu thụ thấp: Đây có thể hiểu là nói về mặt vật lý, cách thức để truyền dữ liệu ví dụ hai thiết bị truyền thông qua chuẩn Zigbee, LoRa, LoRaWAN,....(cụ thể các bạn có thể tham khảo: 13 cách thức truyền dữ liệu trong IoT cho các kỹ sư điện)
- Dung lượng bản tin nhẹ để không làm tiêu tốn quá nhiều tài nguyên của CPU. Hiểu nôm na, sau khi có đường truyền kết nối vật lý ở trên, bạn sẽ cần một phương thức "nói chuyện" giữa các thiết bị để nói ít đi mà vẫn hiểu đúng, hiểu đủ.

Ba giao thức truyền tải dữ liệu phổ biến có thể được sử dụng trong các mô hình Internet of Things:

- MQTT (Message Queuing Telemetry Transport)
- CoAP (Constrained Applications Protocol)
- XMPP (Extensible Messaging và Presence Protocol)

Chương 2. Giao thức MQTT (Message Queuing Telemetry Transport)

2.1 Giới thiệu giao thức MQTT

MQTT là một giao thức mã nguồn mở để truyền các messages giữa nhiều Client (Publisher và Subscriber) thông qua một Broker trung gian, được thiết kế để đơn giản và dễ dàng triển khai. Kiến trúc MQTT dựa trên Broker trung gian và sử dụng kết nối TCP long-lived từ các Client đến Broker.

2.2 Tổng quan giao thức MQTT

Kiến trúc mức cao (high-level) của MQTT gồm 2 phần chính là Broker và Clients [3].

Trong đó, broker được coi như trung tâm, nó là điểm giao của tất cả các kết nối đến từ client. Nhiệm vụ chính của broker là nhận mesage từ publisher, xếp các message theo hàng đợi rồi chuyển chúng tới một địa chỉ cụ thể. Nhiệm vụ phụ của broker là nó có thể đảm nhận thêm một vài tính năng liên quan tới quá trình truyền thông như: bảo mật message, lưu trữ message, logs, …

Client thì được chia thành 2 nhóm là publisher và subscriber. Client là các software components hoạt động tại edge device nên chúng được thiết kế để có thể hoạt động một cách linh hoạt (lightweight). Client chỉ làm ít nhất một trong 2 việc là publish các message lên một topic cụ thể hoặc subscribe một topic nào đó để nhận message từ topic này.

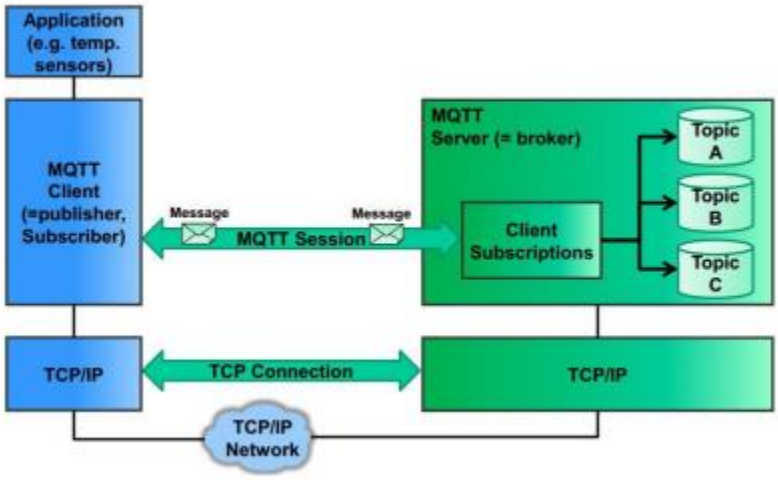
MQTT hỗ trợ tổ chức hệ thống theo các Topics có tính phân cấp, như một hệ thống tập tin (vd: /Home/kitchen/humidity), cung cấp nhiều lựa chọn điều khiển và QoS (Quality of Service).

MQTT Clients tương thích với hầu hết các nền tảng hệ điều hành hiện có: MAC OS, Windows, Linux, Androids, iOS…

MQTT là một giao thức khá nhẹ nên có thể được sử dụng cho truyền thông 2 chiều thông qua các mạng có độ trễ cao và độ tin cậy thấp, nó cũng tương thích với các thiết bị tiêu thụ điện năng thấp.

2.3 Chi tiết giao thức MQTT

2.3.1 Mô hình cơ bản của MQTT



Hình 1: Mô hình cơ bản của giao thức MQTT

Các thành phần chính của MQTT là clients, servers (=brokers), sessions, subscriptions và topics.

MQTT client (publisher, subscriber): Client thực hiện subscribe đến topics để publish và receive các gói tin [3].

MQTT server (broker): Servers thực hiện run các topic, đồng thời nhận subscriptions từ clients yêu cầu các topics, nhận các messages từ clients và forward chúng [3].

Topic: Về mặt kỹ thuật, topics là các hàng đợi chứa message. Về logic, topics cho phép clients trao đổi thông tin và dữ liệu [3].

Session: Một session được định nghĩa là kết nối từ client đến server. Tất cả các giao tiếp giữa client và server đều là 1 phần của session [3].

Subscription: Không giống như sessions, subscription về mặt logic là kết nối từ client đến topic. Khi thực hiện subscribed đến topic, client có thể trao đổi messages với topic. Subscriptions có thể ở trạng thái ‘transient’ hoặc ‘durable’, phụ thuộc vào cờ clean session trong gói Connect [3].

Message: Messages là các đơn vị dữ liệu được trao đổi giữa các topic clients [3].

2.3.2 Giới tin trong giao thức MQTT

2.3.2.1 Định dạng của gói tin

Tất cả các message luôn chứa phần cố định theo bảng 1 [3].

Bảng 1: Header cố định [3]

bit	7	6	5	4	3	2	1	0
byte1	Loại Message				Cờ DUP	QoS level		RETAIN
byte2	Độ dài còn lại							

Byte 1: Chứa loại Message và các cờ (DUP, QoS level, and RETAIN).

Byte 2: (Ít nhất 1 byte) quy định độ dài còn lại.

- Loại Message
  - Vị trí: byte 1, bits 7-4.
  - Một số 4-bit không dấu diễn tả các giá trị được miêu tả dưới bảng 2

Bảng 2: Loại message [3]

Từ gợi nhớ	Giá trị thứ tự	Miêu tả
Reserved	0	Chưa dùng
CONNECT	1	Client yêu cầu kết nối đến Server
CONNACK	2	Kết nối được chấp nhận
PUBLISH	3	Xuất bản message
PUBACK	4	Xuất bản message được chấp nhận
PUBREC	5	Xuất bản đã được nhận (đảm bảo nhận được part 1)

PUBREL	6	Xuất bản release (đảm bảo nhận được part 2)
PUBCOMP	7	Xuất bản hoàn thành (đảm bảo nhận được part 3)
SUBSCRIBE	8	Yêu cầu subcribe từ client
SUBACK	9	Yêu cầu subcriber được chấp nhận
UNSUBSCRIBE	10	Yêu cầu unsubcribe
UNSUBACK	11	Yêu cầu unsubcribe được chấp nhận
PINGREQ	12	Request PING
PINGRESP	13	Response PING
DISCONNECT	14	Client đang mất kết nối
Reserved	15	Reserved

- Các cờ : Bit còn lại của byte đầu chứa các trường DUP, QoS và RETAIN. Vị trí các bit và ý nghĩa được miêu tả trong bảng 3

Bảng 3: Bảng các cờ [3]

Vị trí bit	Tên viết gọn	Miêu tả
3	DUP	Nhận lặp lại
2-1	QoS	Quality of Service (chất lượng dịch vụ)
0	RETAIN	cờ RETAIN

- DUP :Vị trí byte 1, bit 3. Cờ này được bật khi client hoặc server đang cố chuyển lại một gói PUBLISH, PUBREL, SUBSCRIBE hoặc UNSUBSCRIBE. Giá trị này được sử dụng trong các mesage mà có QoSS lớn hơn 0 và yêu cầu ACK. Khi bit DUP được set, phần header thay đổi sẽ chứa Message ID. Nhìn vào giá trị này sẽ biết được gói tin đã nhận được trước đó hay không. Nó không nên sử dụng để tin ngay rằng có duplicates hay không.
- QoS : Vị trí byte 1, bits 2-1. Cờ này sẽ cho biết độ đảm bảo việc nhận message PUBLISH. Giá trị của QoS được miêu tả trong bảng 4.

Bảng 4: Giá trị QoS [3]

Giá trị QoS	bit 2	bit 1	Miêu tả		
0	0	0	Cùng lắm là 1 lần	Gửi rồi quên ngay	<=1
1	0	1	Ít nhất 1 lần	Xác nhận bằng ACK	>=1
2	1	0	Chính xác 1 lần	Nhận đảm bảo	=1
3	1	1	Chưa dùng		

- RETAIN: Vị trí byte 1, bit 0. Cờ này chỉ được sử dụng ở message PUBLISH. Khi client gửi 1 message PUBLISH đến server, nếu cờ Retain được set (1), thì server phải hiểu rằng nên giữ message này ngay cả sau khi chuyển nó đến các subcribers hiện tại. Khi có 1 subcription mới được thiết lập trên 1 topic, message cuối cùng của topic đó nên được gửi đến subscriber với 1 trường Retain được set trong header. Nếu không có messsage nào còn, thì không cần gửi gì hết.
- Độ dài còn lại Vị trí: byte 2. Miêu tả độ dài bao gồm cả phần header và payload có trong message. Việc encoding với độ dài thay đổi sử dụng 1 byte để miêu tả độ dài, vì thế độ dài tối đa sẽ là 127. Những message dài hơn sẽ được miêu tả theo cách sau. 7 bít được dùng để miêu tả giá trị, bít còn lại dùng để miêu tả phía sau còn byte nào miêu tả trường này hay không. Mỗi byte tiếp sau đó cũng như vậy 7 bit để lưu giá trị, 1 bít gọi là bit tiếp tục. Giá trị được tính bằng cách nhân giá trị được diên tả bởi 7 bit và lũy thừa tăng dần của 128. Ví dụ miêu tả độ Remain Length = 64, ta chỉ cần 1 byte, trong đó 7 bytes để miêu tả

giá trị 64, 1 bit còn lại bằng 0. Một ví dụ nữa, giá trị là 321 chẳng hạn 321 = 65*128^0 + 2* 128^1, ta cần 2 byte để biểu diễn. Byte đầu chứa giá trị 65 trong 7 bit và bit còn lại là 1. Byte thứ 2 chứa giá trị 2 ở 7 bit và 1 bit chứa giá trị bằng 0. Trường này được biểu diễn tối đa trong 4 byte. Tức là cho độ dài cho phép sẽ là đến 268 435 455 (256 MB).

Bảng 5: Độ dài ứng với số byte [3]

Số byte	Độ dài min	Độ dài max
1	0 (0x00)	127 (0x7F)
2	128 (0x80, 0x01)	16 383 (0xFF, 0x7F)
3	16 384 (0x80, 0x80, 0x01)	2 097 151 (0xFF, 0xFF, 0x7F)
4	2 097 152 (0x80, 0x80, 0x80, 0x01)	268 435 455 (0xFF, 0xFF, 0xFF, 0x7F)

2.3.2.2 Gói tin CONNECT - Client yêu cầu connect đến server

Khi một một kết nối TCP/IP được thiết lập từ client đến server, thì một session ở mức protocol cũng được tạo sử dụng luồng CONNECT. Server sẽ gửi message CONNACK để trả lời message CONNECT từ client. Nếu server không nhận được mesage CONNET từ client trong một khoang thời gian nào đó sau khi thiết lập kết nối TCP/IP, thì server nên đóng kết nối đó lại. Nếu client không nhận được một message CONNACK từ server trong một khoảng thời gian nhất định, thì client cũng nên đóng kết nối đó lại, và restart session bằng một socket mới đến server rồi tiếp tục gửi yêu cầu kết nối bằng gói CONNECT. Trong cả 2 trường hợp trên, thời gian chờ để nhận được message CONNECT hoặc CONNACK phụ thuộc vào ứng dụng và điều kiện kết nối. Nếu một client kết nối bằng một Client ID đang được kết nối với Server rồi, thì client trước đó phải được disconnect bắt buộc bởi server trước khi thực hiện luồng CONNECT với client mới. Nếu client gửi một một message CONNECT không hợp lệ, server nên đóng kết nối luôn. Không hợp ở đây bao gồm việc khác nhau về Protocol Name hoặc Protocol Version Numbers. Nếu server đã parse message CONNECT rồi mới phát hiện ra có một trường nào đó không hợp lệ,

nó nên gửi lại message CONNACK chứa nội dung mã có nội dung "Kết nối bị từ chối: phiên bản protocol không được chấp nhận" trước khi hủy kết nối này.

2.3.2.3 Gói tin CONNACK - Acknowledge connection request

Message CONNACK được gửi bởi server như để trả lời một yêu cầu CONNECT từ client. Mỗi giá trị trả về của Connack được chỉ ra dưới bảng 6.

Bảng 6: Ý nghĩa gói CONNACK [3]

Enumeration	HEX	Meaning
0	0x00	Kết nối được chấp nhận
1	0x01	Kết nối bị từ chối: Phiên bản protocol không được chấp nhận.
2	0x02	Kết nối vị từ chối: Định danh không hợp lệ
3	0x03	Kết nối vị từ chối: server không phục vụ được
4	0x04	Kết nối bị từ chối: Sai user name hoặc password
5	0x05	Kết nối bị từ chối: Chưa được xác thực
6-255		Dành cho tương lai

Mã trả về là 2 (định danh bị từ chối) sẽ được gửi nếu nếu định danh duy nhất cho 1 client có độ dài không nằm trong khoảng từ 1 đến 23

2.3.2.4 PUBLISH - Message Publish

Một message PUBLISH được gửi từ một client đến server để phân tán chúng đến các subscriber cần chúng [3]. Mỗi message luôn gắn liền với một topic name (cũng được hiểu là Subject hoặc Channel). Topic là có dạng không gian phân cấp, nó định nghĩa ra một cách phân loại nguồn thông tin để cho các subscribers có thể subscribe nếu muốn.Một message được published đến một topic nào đó sẽ được phân phát đến các subscriber quan tâm đến topic đó .

Nếu 1 client subscribe một hoặc nhiều topic, thì mọi message được published đến những topic đó được gửi bởi server đến client như là một message PUBLISH. Response cho PUBLISH message phụ thuộc vào level của QoS.

Bảng 7: Giá trị mức QoS [3]

QoS Level	Giá trị trả về mong muốn
QoS 0	None
QoS 1	PUBACK
QoS 2	PUBREC

Các message PUBLISH được gửi từ Publisher đến server, hoặc là từ Server đến Subsciber [3]. Việc tiếp thu khi nó nhận được messsage phụ thuộc vào QoS level của message:

- QoS 0: Phải đảm bảo chắc chắn rằng các message đến các bên quan tâm.
- QoS 1: Giữ một bản của messsage trên bộ nhớ ngoài, tất nhiên cũng phải đảm báo nó đến các bên quan tâm, và trả lại một message PUBACK đến bên gửi.
- QoS 2: Giữ lại một bản trên bộ nhớ ngoài, đừng gửi nó đến các bên quan tâm, trả về message PUBREC đến bên gửi.

Ở đây, nếu server nhận được message, các bên quan tâm nghĩa là các subscriber đến topic của message PUBLISH đó. Nếu một subscriber nhận một message, thì các bên quan tâm ám chỉ đến các ứng dụng bên phía client mà đã subcribe mọt hoặc nhiều topics và đang đợi một message đến từ server.

Nếu một server mà không xác nhận được một PUBLISH từ một client, thì nó không có cách nào để thông bao đến client đó. Vì thế nó phải đảm bảo một mức hiểu biết nhất định, tùy thuộc vào các rule QoS, client sẽ không được báo rằng việc xác thực message PUBLISH nó gửi đi.

2.3.2.5 SUBSCRIBE - Subscribe to named topics

Message SUBSCRIBE cho phép client subscribe một hoặc nhiều topics với server [3]. Message được published lên server sẽ được chuyển đến client bằng message PUBLISH [3]. Message SUBSCRIBE cũng chỉ ra QoS level mà subscriber

muốn nhận message. Khi nhận được một message SUBSCRIBE message, server trả lời bằng message SUBACK.

Một server bắt đầu gửi một message PUBLISH cho yêu cầu của về subscription của client thậm chí trước khi client nhận được message SUBACK. Nếu một server không xác nhận một yêu cầu SUBSCRIBE từ client, thì nó không có cách nào thông tin cho client. Vì thế nó tạo ra message SUBACK như là để xác nhận, và client sẽ không được thông báo rằng nó không được xác thực.

Một server có quyền chỉ định một level of QoS thấp hơn client yêu cầu [3]. Điều này có thể xảy ra nếu server không thể cung cấp các levels of QoS cao hơn. Ví dụ, nếu server không cung cấp một cơ chế lưu trữ tin cậy nào đó nó có thể chỉ cấp cho các subscriptions với QoS là 0.

2.3.2.6 PINGREQ - PING request

Message PINGREQ có nghĩa là message "Kết nối vẫn tốt đúng không?" được gửi từ một client đã kết nối đến server [3].

2.3.2.7 PINGRESP - PING response

Một message PINGRESP được gửi từ server cho một message PINGREQ và nó có nghĩa là "OK" [3].

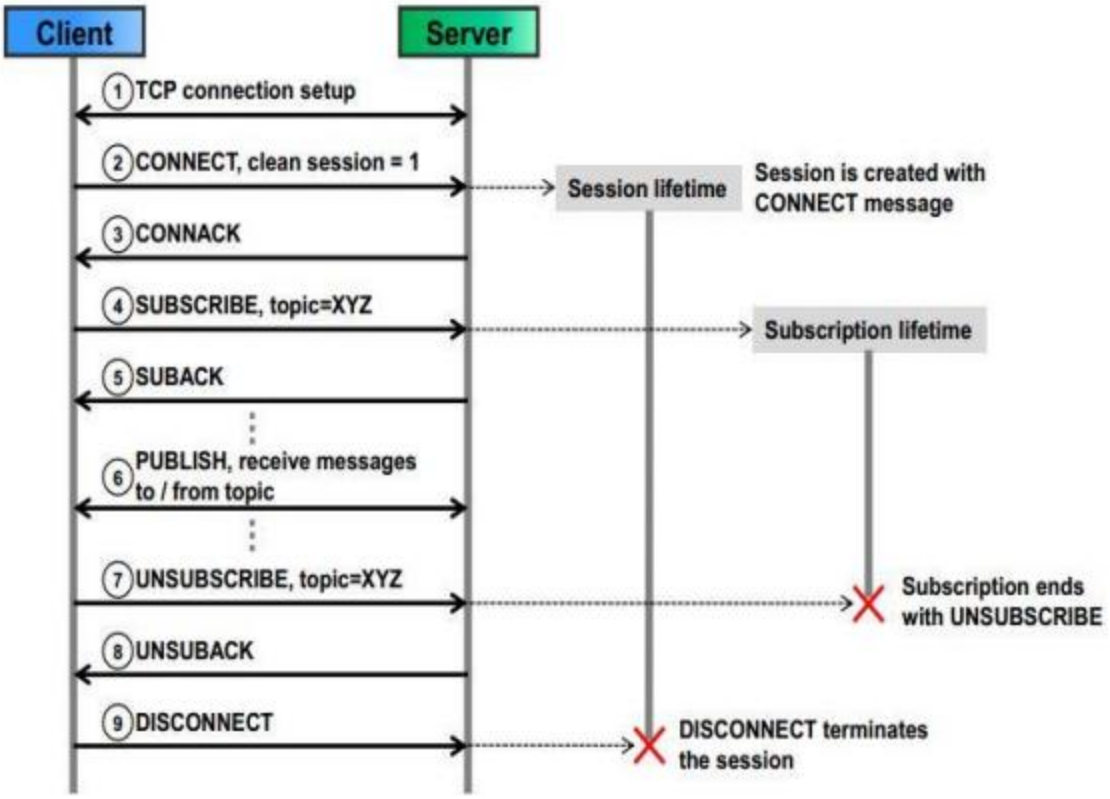
2.3.2.8 DISCONNECT - Disconnect notification

Message DISCONNECT được gửi từ client đến server để báo rằng nó sẽ đóng kết nối TCP/IP đang kết nối. Cái này cho phép một clean disconnection, chứ không chỉ là hủy kết nối. Một server không nên để việc đóng kết nối này cho phía client sau khi nhận message DISCONNECT [3].

2.3.3 Quy trình truyền nhận dữ liệu chính trong MQTT

2.3.3.1 CONNECT and SUBSCRIBE message sequence

Trường hợp 1: Session và subscription được thiết lập với clean session flag = 1 (transient subscription) [3].

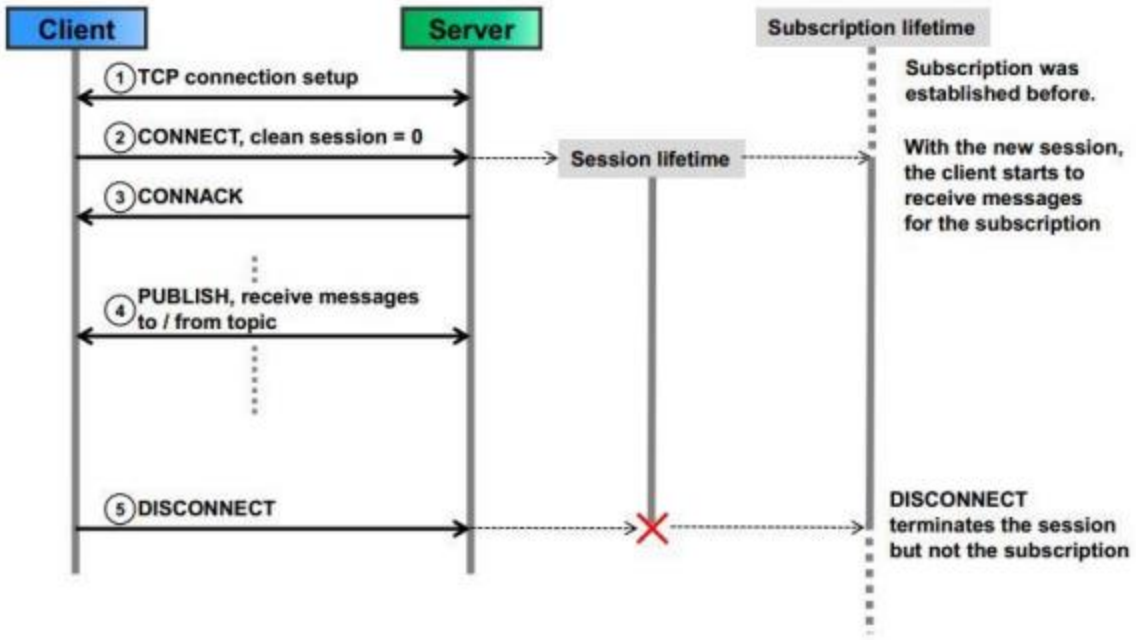


Hình 2: Session và subscription được thiết lập với clean session flag = 1

Quy trình truyền nhận dữ liệu trong MQTT khi session flag = 1:

- Client và Server được kết nối qua giao thức TCP.
- Client gửi gói tin CONNECT yêu cầu kết nối đến Server, clean session = 1. Đây là thời điểm đánh dấu session được thiết lập.
- Server gởi gói CONNACK xác nhận thiết lập kết nối thành công.
- Client thực hiện SUBSCRIBE đến topic XYZ. Đây là thời điểm bắt đầu timeout của một subscription.
- Server gởi gói SUBACK xác nhận quá trình subscription.
- Client PUBLISH để gởi topic - message đến server.
- Sau khi nhận đủ thông tin, client gửi gói UNSUBSCRIBE topic XYZ để kết thúc quá trình Subscribe.
- Server trả về gói UNSUBACK.
- Client gửi gói DISCONNECT để kết thúc session truyền thông.

Trường hợp 2: Session và subscription được thiết lập với clean session flag = 0 (durable subscription) [3].



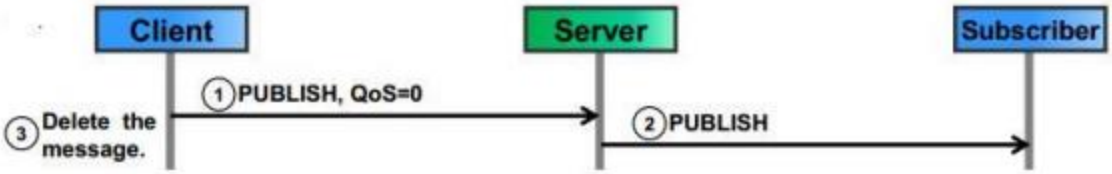
Hình 3: Session và subscription được thiết lập với clean session flag = 0

Quy trình truyền nhận dữ liệu trong MQTT khi session flag = 0:

- Subscription lifetime đã được thiết lập trước.
- Client và Server được kết nối qua giao thức TCP.
- Client gửi gói tin CONNECT yêu cầu kết nối đến Server, clean session = 0. Đây là thời điểm đánh dấu session được thiết lập.
- Server gửi gói CONNACK xác nhận thiết lập kết nối thành công.
- Client PUBLISH để gửi topic - message đến server.
- Client gửi gói DISCONNECT để kết thúc session truyền thông.

2.3.3.2 PUBLISH message flows

- QoS level 0: At most once delivery. Message được phân phối dựa trên best efforts của tầng mạng TCP/IP bên dưới. Một response sẽ không được định nghĩa trong giao thức. Các message đến server hoặc chỉ 1 lần hoặc không bao giờ [3].

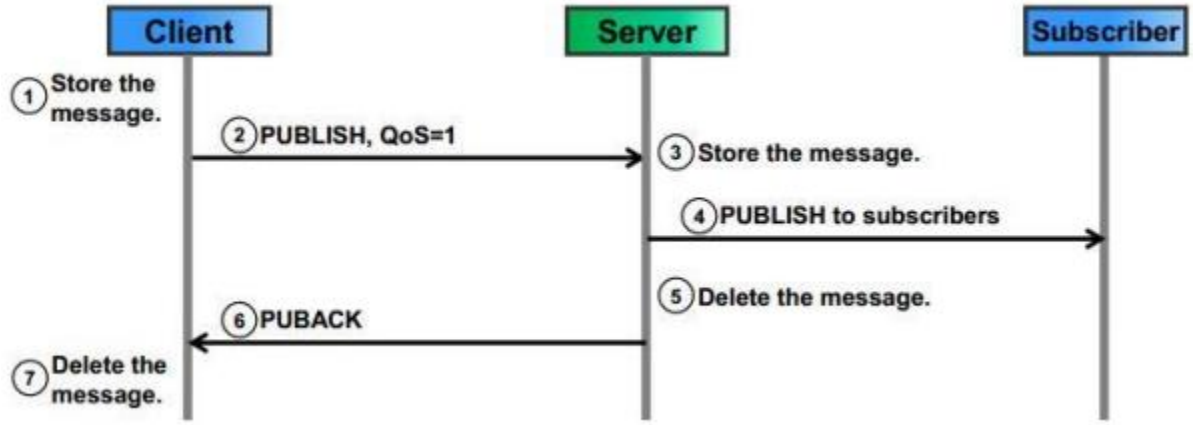


Hình 4: QoS mức 0

- QoS level 1: At least once delivery

Việc nhận được message bên phía server được xác nhận bởi một message PUBACK. Nếu có lỗi do kết nối hoặc gửi đến device, hoặc message xác nhận không nhận được sau một khoảng thời gian nhất định, sender sẽ gửi lại message và set DUP bit trong phần header của message header [3]. Message đến server ít nhất 1 lần. Cả message SUBSCRIBE và message UNSUBSCRIBE đều sử dụng QoS level là 1.

Khi nhận được một message lặp lại từ phía client, server sẽ publish các message đến các subscribers, và gửi một message PUBACK khác.



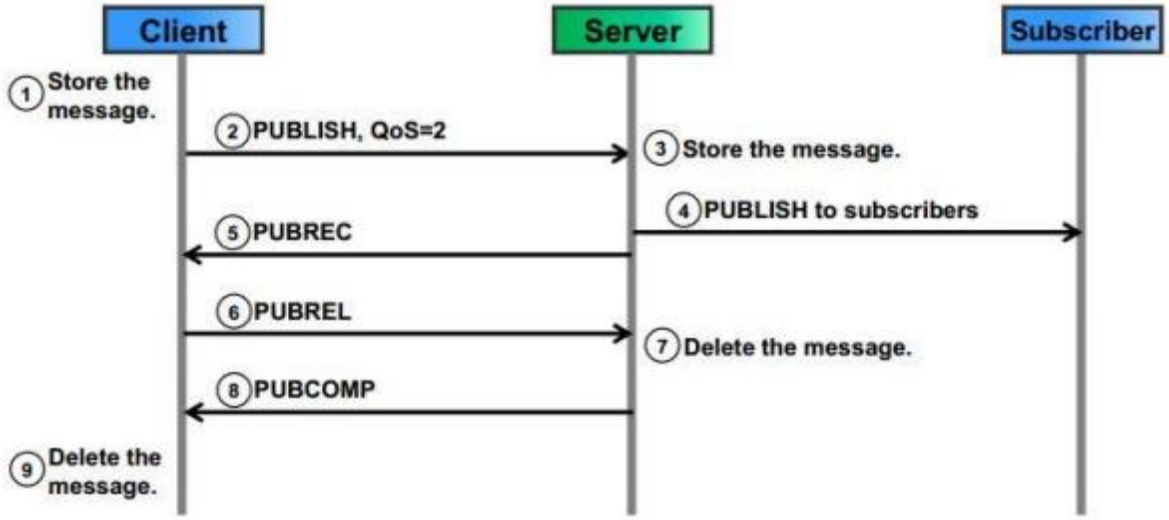
Hình 5: QoS mức 1

- QoS level 2: Exactly once delivery

Một luồng được thêm vào luồng QoS level bằng 1 ở trên để đảm bảo rằng message bị lặp lại không bị chuyển đến ứng dụng [3]. Đây là mức độ cao nhất khi khi phân phối message, không message lặp nào được chấp nhận. Nhờ đó mà lưu lượng mạng sẽ tăng lên.

Nếu phát hiện lỗi, hoặc sau một khoảng thời gian nhất định, luồng protocol sẽ được thực hiện lại từ kết quả của message xác nhận cuối cùng; hoặc là PUBLISH , hoặc

là PUBREL. Luồng protocol đảm bảo rằng message đến các subscriber chỉ đúng 1 lần.



Hình 6: QoS mức 2

Chương 3. Giao thức CoAP (Constrained Applications Protocol)

3.1 Giới thiệu giao thức CoAP

CoAP là một giao thức truyền tải tài liệu theo mô hình client/server dựa trên internet tương tự như giao thức HTTP nhưng được thiết kế cho các thiết bị ràng buộc. Giao thức này hỗ trợ một giao thức one-to-one để chuyển đổi trạng thái thông tin giữa client và server [7].

3.2 Tổng quan giao thức CoAP

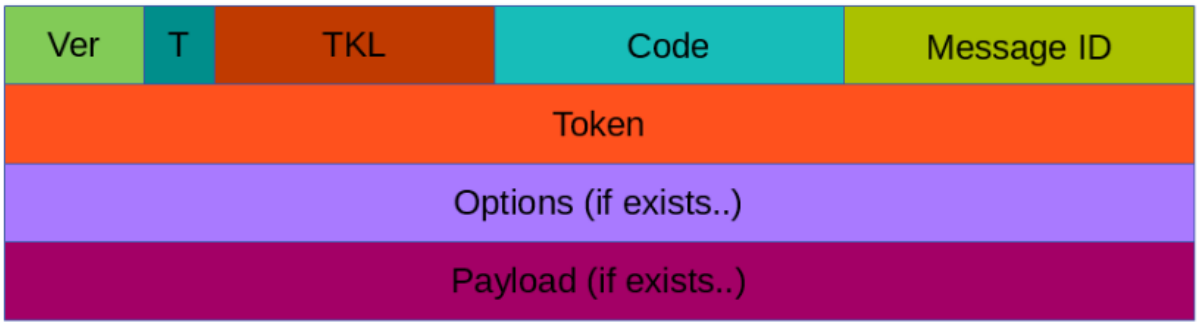
CoAP sử dụng UDP (User Datagram Protocol), không hỗ trợ TCP, ngoài ra còn hỗ trợ địa chỉ broadcast và multicast, truyền thông CoAP thông qua các datagram phi kết nối (connectionless) có thể được sử dụng trên các giao thức truyền thông dựa trên các gói [7].

UDP có thể dễ dàng triển khai trên các vi điều khiển hơn TCP nhưng các công cụ bảo mật như SSL/TSL không có sẵn, tuy nhiên ta có thể sử dụng Datagram Transport Layer Security (DTLS) để thay thế [7].

CoAP theo mô hình client/server. Client gửi yêu cầu đến máy chủ, sau đó máy chủ gửi lại phản hồi. Client có thể GET, PUT, POST và DELETE các tài nguyên. CoAP có tính linh động và hỗ trợ đàm phán nội dung. Điều này cho phép client và máy chủ có thể nâng cấp, thêm mới một cách độc lập mà ko ảnh hưởng gì đến phía còn lại. CoAP đưa ra các yêu cầu quan sát tài nguyên. Cả hai bên đều có thể tác động hoặc xóa các yêu cầu quan sát. Khi cờ quan sát được thiết lập, máy chủ vẫn có thể tiếp tục hồi đáp sau khi các dữ liệu đã truyền đi. Với CoAP, máy chủ cung cấp một hệ thống các tài nguyên cho phép client khám phá tài nguyên và các loại phương tiện truyền thông. Trong CoAP một nút cảm biến thường là một máy chủ. Chúng có khả năng nhận các gói tin gửi đến để hoạt động đúng đằng sau NAT, thiết bị đầu tiên phải gửi yêu cầu đến máy chủ, như được thực hiện trong LWM2M, cho phép các router liên kết chúng lại [7].

3.3 Chi tiết giao thức CoAP

3.3.1 Gói tin trong giao thức CoAP



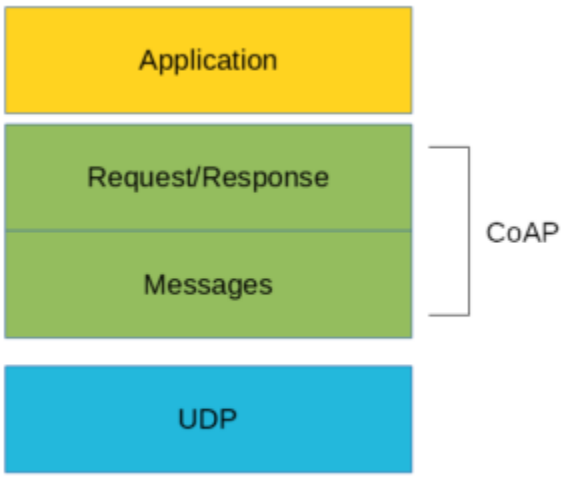
Hình 7: Giới tin trong giao thức CoAP [5]

Trong đó:

- Ver : Đó là số nguyên không dấu 2 bit cho biết phiên bản
- T : đó là số nguyên không dấu 2 bit cho biết loại thông báo: 0 có thể xác nhận, 1 không xác nhận
- TKL : Độ dài mã thông báo là độ dài mã thông báo 4 bit
- Mã : Đó là phản hồi mã (độ dài 8 bit)
- ID tin nhắn : Đó là ID tin nhắn được thể hiện bằng 16 bit

3.3.2 Mô hình truyền nhận dữ liệu

Có hai lớp khác nhau tạo giao thức CoAp: Tin nhắn và Yêu cầu / Phản hồi. Lớp Tin nhắn liên quan đến UDP và với các tin nhắn không đồng bộ. Lớp Yêu cầu / Phản hồi quản lý tương tác yêu cầu / phản hồi dựa trên các thông báo yêu cầu / phản hồi.



Hình 8: Các lớp trong giao thức CoAP [6]

CoAP hỗ trợ bốn loại thông báo khác nhau [6]:

- Xác nhận
- Không thể xác nhận
- Nhìn nhận
- Cài lại

Giao thức CoAp, sử dụng một số thuật ngữ thông dụng sau [6]:

- Điểm cuối : Một thực thể tham gia vào giao thức CoAP. Thông thường, Điểm cuối được xác định với máy chủ lưu trữ
- Người gửi : Thực thể gửi tin nhắn
- Người nhận : Đích đến của tin nhắn
- Khách hàng : Thực thể gửi yêu cầu và đích đến của phản hồi
- Máy chủ : Thực thể nhận được yêu cầu từ máy khách và gửi lại phản hồi cho máy khách

3.3.2.1 Giao thức tin nhắn

Đây là lớp CoAP thấp nhất. Lớp này liên quan đến các thông điệp trao đổi UDP giữa các điểm cuối. Mỗi tin nhắn CoAP có một ID duy nhất; Điều này rất hữu ích để phát hiện các thông điệp trùng lặp. Một thông điệp CoAP được xây dựng bởi các phần này [5]:

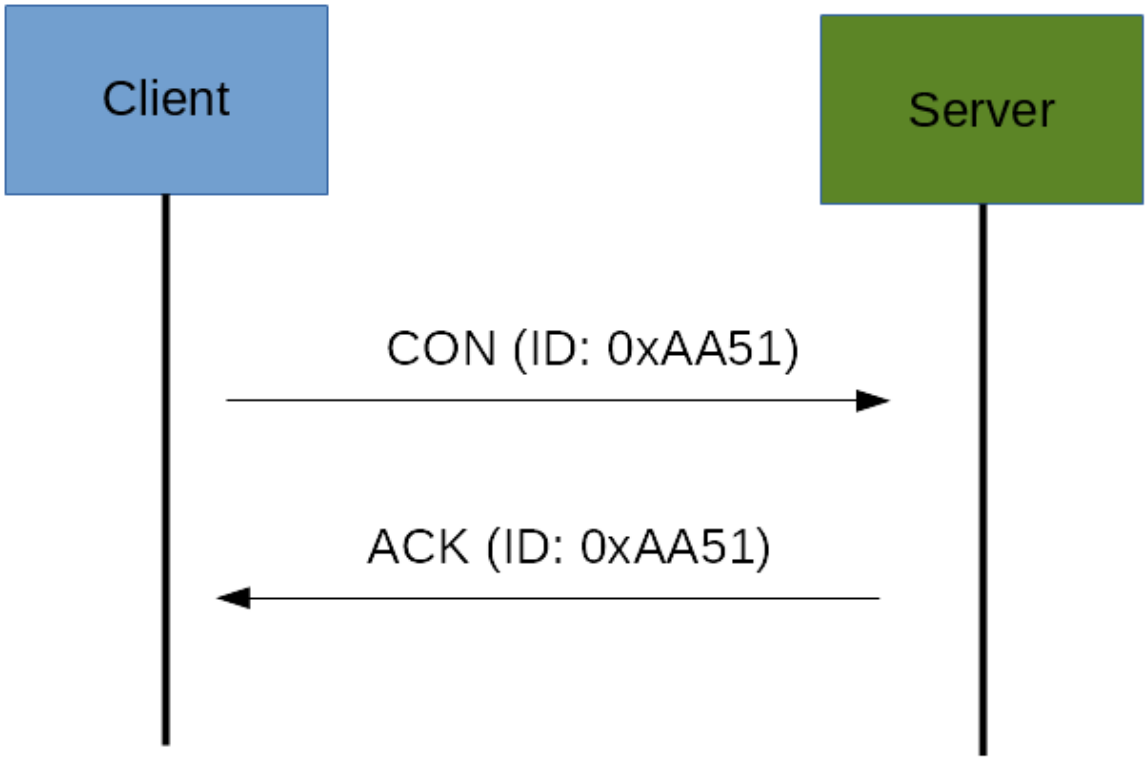
- Một tiêu đề nhị phân
- Một lựa chọn nhỏ gọn
- Khối hàng

Giao thức CoAP sử dụng hai loại thông báo [5]:

- Tin nhắn xác nhận
- Tin nhắn không thể xác nhận

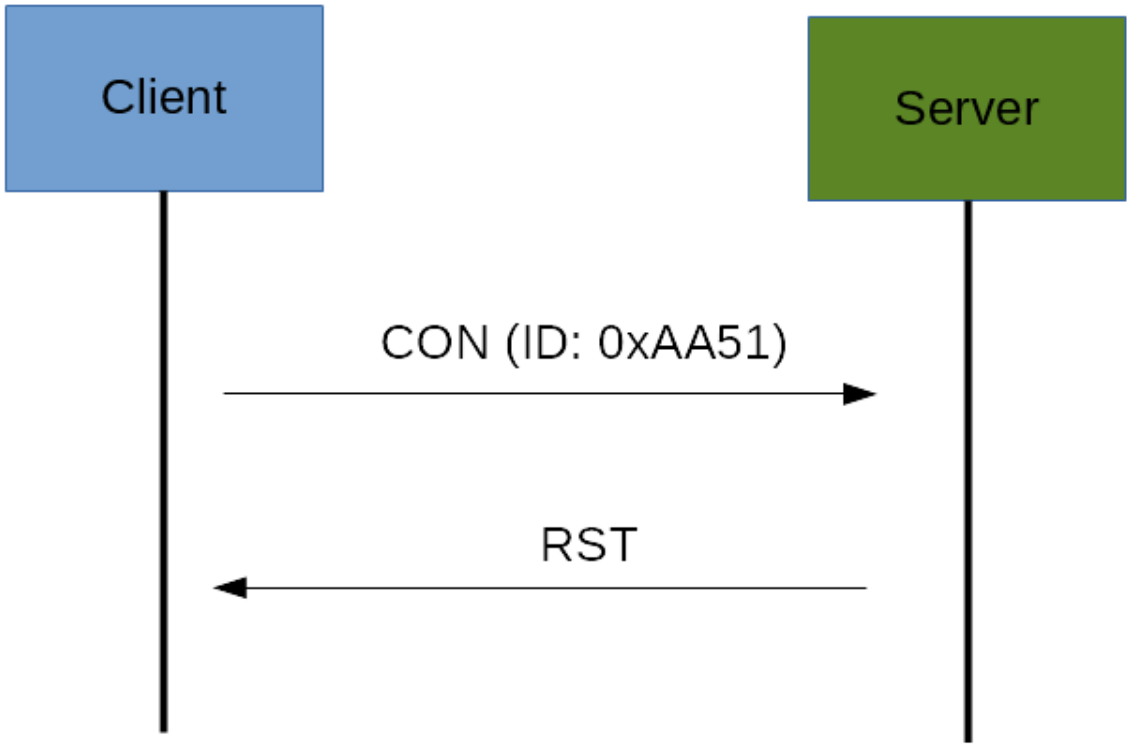
Một tin nhắn xác nhận là một tin nhắn đáng tin cậy. Khi trao đổi tin nhắn giữa hai điểm cuối, những tin nhắn này có thể đáng tin cậy. Trong CoAP, một thông điệp đáng tin cậy có được bằng cách sử dụng tin nhắn có thể xác nhận (CON). Sử dụng loại tin nhắn này, khách hàng có thể chắc chắn rằng tin nhắn sẽ đến máy chủ. Tin nhắn có thể xác nhận được gửi đi gửi lại cho đến khi bên kia gửi tin nhắn xác nhận (ACK). Tin nhắn ACK chứa cùng ID của tin nhắn xác nhận (CON).

Hình dưới đây cho thấy quá trình trao đổi tin nhắn:



Hình 9: Quá trình trao đổi tin nhắn của giao thức CoAP [5]

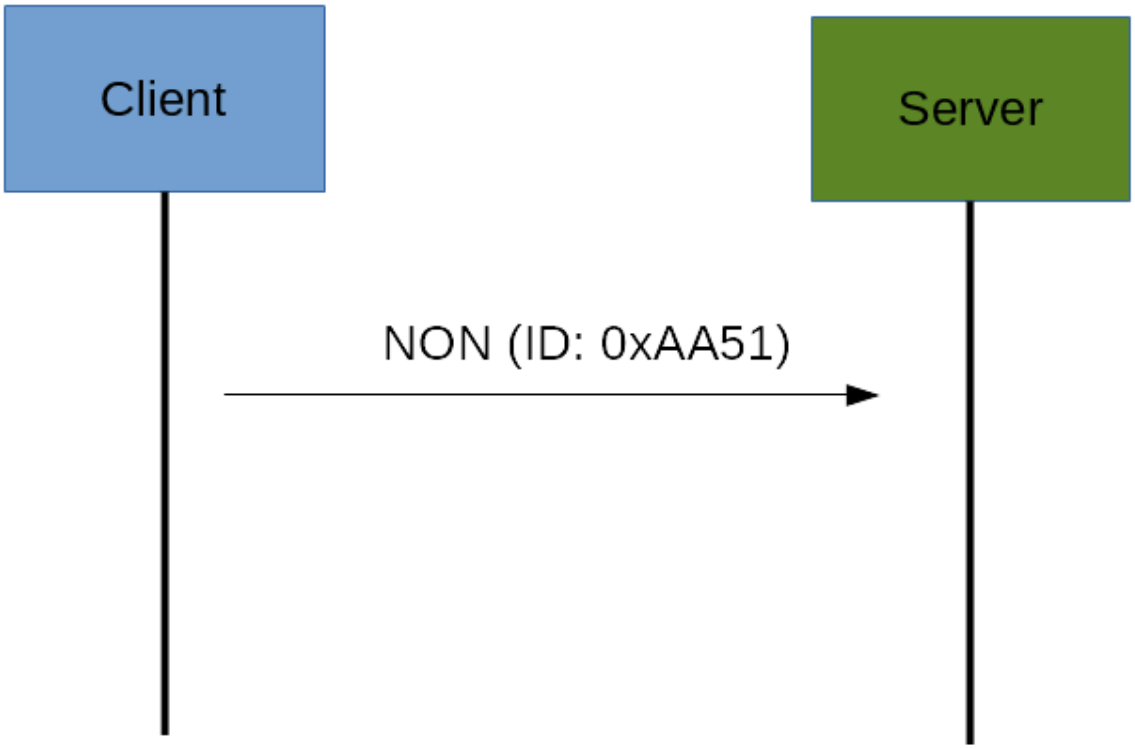
Nếu máy chủ gặp sự cố khi quản lý yêu cầu đến, nó có thể gửi lại tin nhắn Rest (RST) thay vì tin nhắn Acledgeledge (ACK):



Hình 10: Máy chủ gặp sự cố trong giao thức CoAP [5]

Danh mục tin nhắn khác là tin nhắn Không thể xác nhận (NON). Đây là những tin nhắn không yêu cầu Xác nhận của máy chủ. Chúng là những tin nhắn không đáng tin cậy hay nói cách khác là những tin nhắn không chứa thông tin quan trọng phải được gửi đến máy chủ. Đối với thể loại này thuộc về các thông điệp có chứa giá trị đọc từ các cảm biến.

Ngay cả khi những tin nhắn này không đáng tin cậy, chúng vẫn có một ID duy nhất.

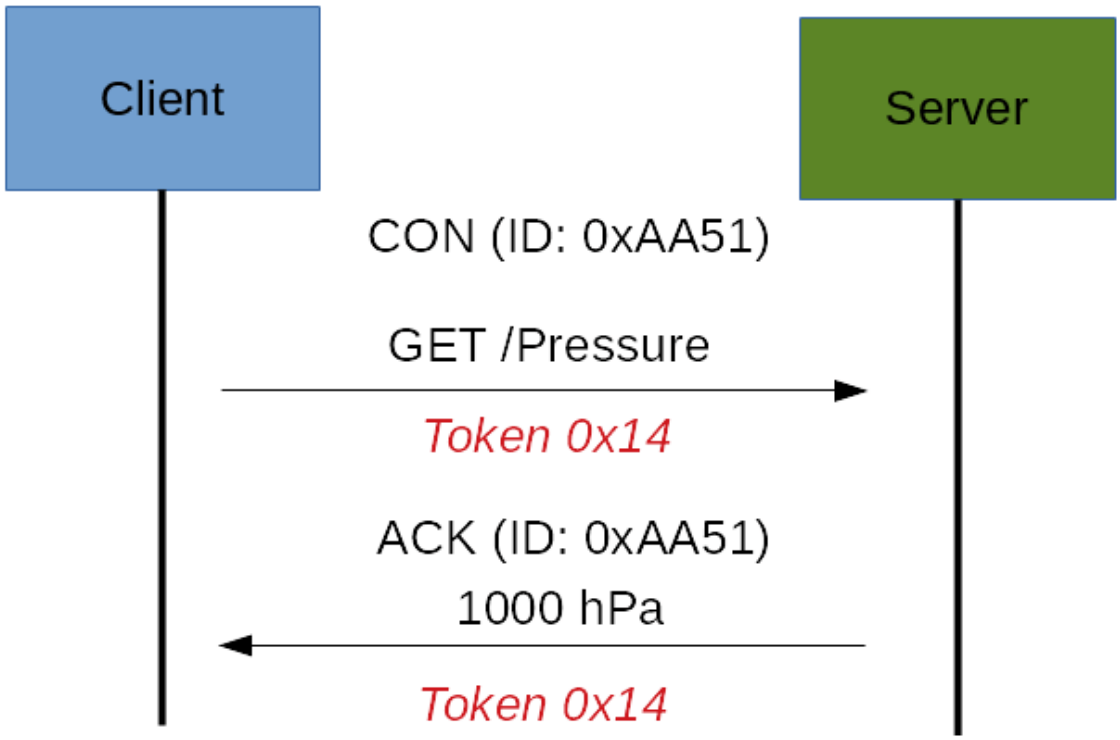


Hình 11: Tin nhắn Không thể xác nhận trong giao thức CoAP [5]

3.3.2.2 Mô hình yêu cầu / phản hồi của CoAp

Yêu cầu / Phản hồi CoAP là lớp thứ hai trong lớp trừu tượng CoAP. Yêu cầu được gửi bằng tin nhắn Có thể xác nhận (CON) hoặc Không thể xác nhận (NON). Có một số kịch bản tùy thuộc vào việc máy chủ có thể trả lời ngay lập tức cho yêu cầu của khách hàng hoặc câu trả lời nếu không có sẵn.

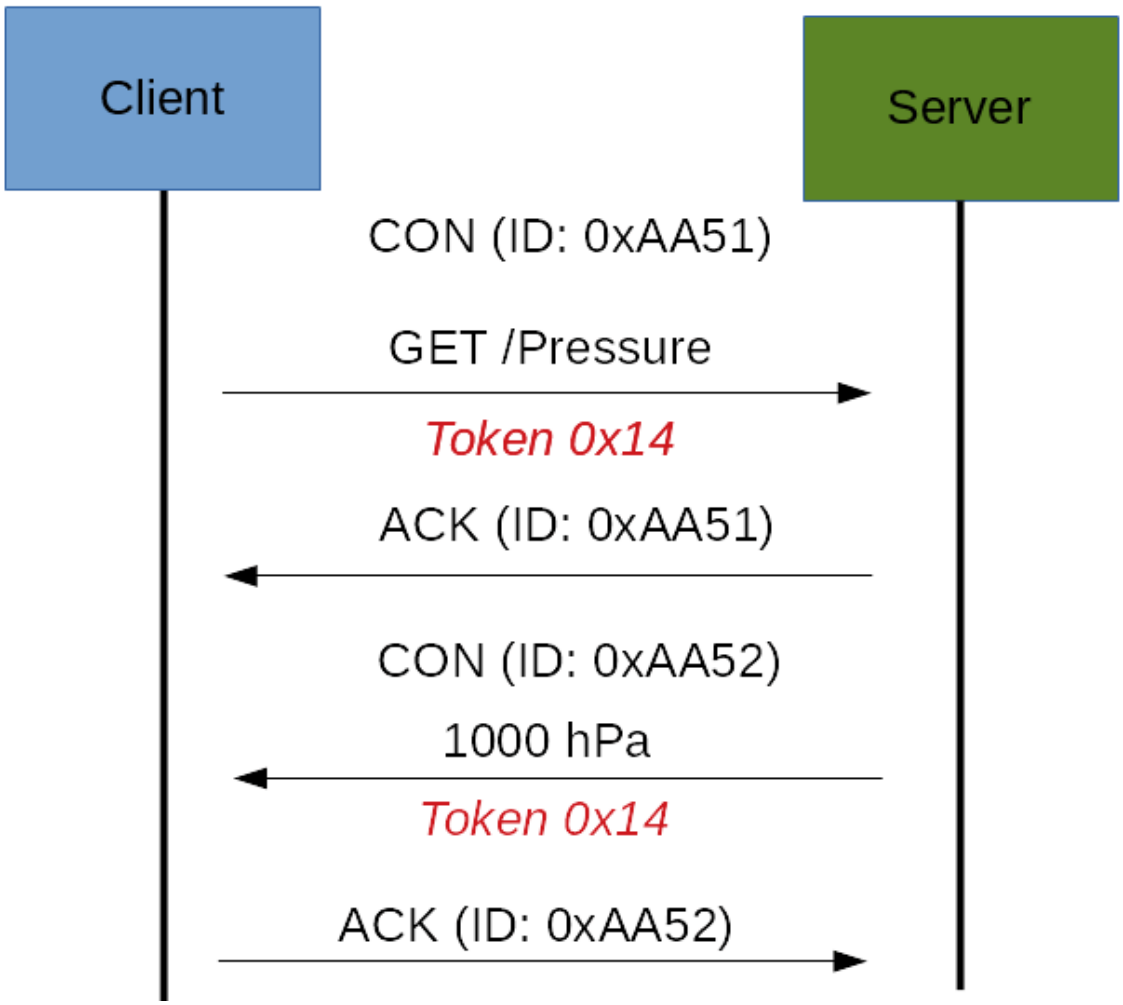
Nếu máy chủ có thể trả lời ngay cho yêu cầu của khách hàng, thì nếu yêu cầu được thực hiện bằng thông báo Xác nhận (CON), máy chủ sẽ gửi lại cho khách hàng một thông báo Xác nhận có chứa phản hồi hoặc mã lỗi:



Hình 12: Giao thức yêu cầu/ phản hồi khi máy chủ có thể xác nhận trong giao thức CoAP [5]

Như bạn có thể nhận thấy trong thông báo CoAP, có một mã thông báo. Mã thông báo khác với ID tin nhắn và nó được sử dụng để khớp với yêu cầu và phản hồi.

Nếu máy chủ không thể trả lời yêu cầu đến từ máy khách ngay lập tức, thì nó sẽ gửi một thông báo Xác nhận với một phản hồi trống. Ngay khi có phản hồi, máy chủ sẽ gửi một thông báo Xác nhận mới cho khách hàng có chứa phản hồi. Tại thời điểm này, khách hàng gửi lại tin nhắn Xác nhận:



Hình 13: Giao thức yêu cầu/ phản hồi khi máy chủ không thể xác nhận trong giao thức CoAP [5]

Nếu yêu cầu đến từ máy khách được thực hiện bằng tin nhắn có thể xác nhận NON, thì máy chủ sẽ trả lời bằng tin nhắn không xác nhận NON.

Chương 4. Giới thiệu giao thức XMPP (Extensible Messaging and Presence Protocol)

4.1 Giới thiệu giao thức XMPP

Extensible Messaging and Presence Protocol (XMPP) là một giao thức mở, trước kia có tên là Jabber [8]. XMPP dựa trên nền tảng là cấu trúc XML (Extensible Markup Language), nói cụ thể hơn XMPP là con đường vận chuyển các gói tin XML giữa các thực thể trên mạng.

4.2 Tổng quan giao thức XMPP

Không giống như hầu hết các giao thức Instant Messaging khác, XMPP là một giao thức mở, chính vì điều này đã làm cho XMPP được sử dụng khá rộng rãi, bất cứ cá nhân nào đều có thể tiến hành triển khai một dịch vụ XMPP của riêng mình và hoàn toàn có thể tương thích với một số dịch vụ của một cá nhân khác. Và thông thường những ứng dụng này đều được cung cấp dưới dạng mã nguồn mở hoặc miễn phí, điều này sẽ là tiền đề để thu hút số một lượng lớn người sử dụng so với các ứng dụng Insant Messaging khác trong tương lai.

Trong thời gian đầu Jabber chỉ tập trung phát triển hệ thống Instant Messaging, tuy nhiên với khả năng mở rộng của XML nên XMPP cũng đã được xây dựng để cung cấp các chức năng khá đa dạng và phong phú. Về cơ bản XMPP cung cấp các dịch vụ hỗ trợ sau [8]:

- Mã hóa đường truyền (Channel encryption): định nghĩa trong RFC-3920, cung cấp khả năng mã hóa kết nối giữa client – server hay giữa các server với nhau.
- Xác thực (Authentication): định nghĩa trong RFC-3920, cung cấp khả năng xác thực một thực thể khi lần đầu tiên kết nối đến server.
- Hiện diện (Presence): định nghĩa trong RFC-3921, cung cấp khả năng thông báo một thực thể có tồn tại trên mạng hay ko, đồng thời có thể kèm theo một số thông tin về thực thể đó.
- Danh sách liên lạc (Contacts lists): định nghĩa trong RFC-3921, cung cấp khả năng lưu trữ danh sách liên lạc của một thực thể trên server.
- One-to-one messaging: được định nghĩa trong RFC-3920, cho phép 2 thực thể gửi tin nhắn qua lại với nhau.

- Multi-party messaging: định nghĩa trong XEP-0045, cung cấp chức năng cho một thực thể tham gia vào một nhóm và trao đổi tin nhắn qua lại giữa các thực thể khác trong nhóm đó.
- Thông báo (Notifications): được định nghĩa trong XEP-0060, cung cấp chức năng tượng tự với "multi-party messaging", tuy nhiên đã được tối ưu hóa theo chiều từ một đến nhiều thực thể với một chủ đề cụ thể. © Service discovery: định nghĩa trong XEP-0030, cung cấp khả năng xác định các chức năng được cung cấp của một thực thể.
- Capabilities advertisement: được định nghĩa trong XEP-0115, là một chức năng mở rộng của dịch vụ "presence", xác định các hành vi xử lý có thể đáp ứng của một ứng dụng khi nhận được một thông điệp "presence" từ một ứng dụng khác.
- Structured data forms: định nghĩa trong XEP-0004, cung cấp khả năng trao đổi thông tin có cấu trúc giữa các thực thể.
- Workflow management: định nghĩa trong XEP-0050, thường được sử dụng kết hợp với data forms để quản lý các quy trình công việc.
- Peer-to-peer media sessions: định nghĩa trong XEP-0166, cung cấp khả năng tạo và quản lý các phiên trong việc trao đổi các dữ liệu đa phương tiện như âm thanh, hình ảnh, tệp dữ liệu

Với số lượng dịch vụ đa dạng trên, XMPP cung cấp khả năng xây dựng các ứng dụng khá phong phú, sau đây là một số ứng dụng tiêu biểu [8]:

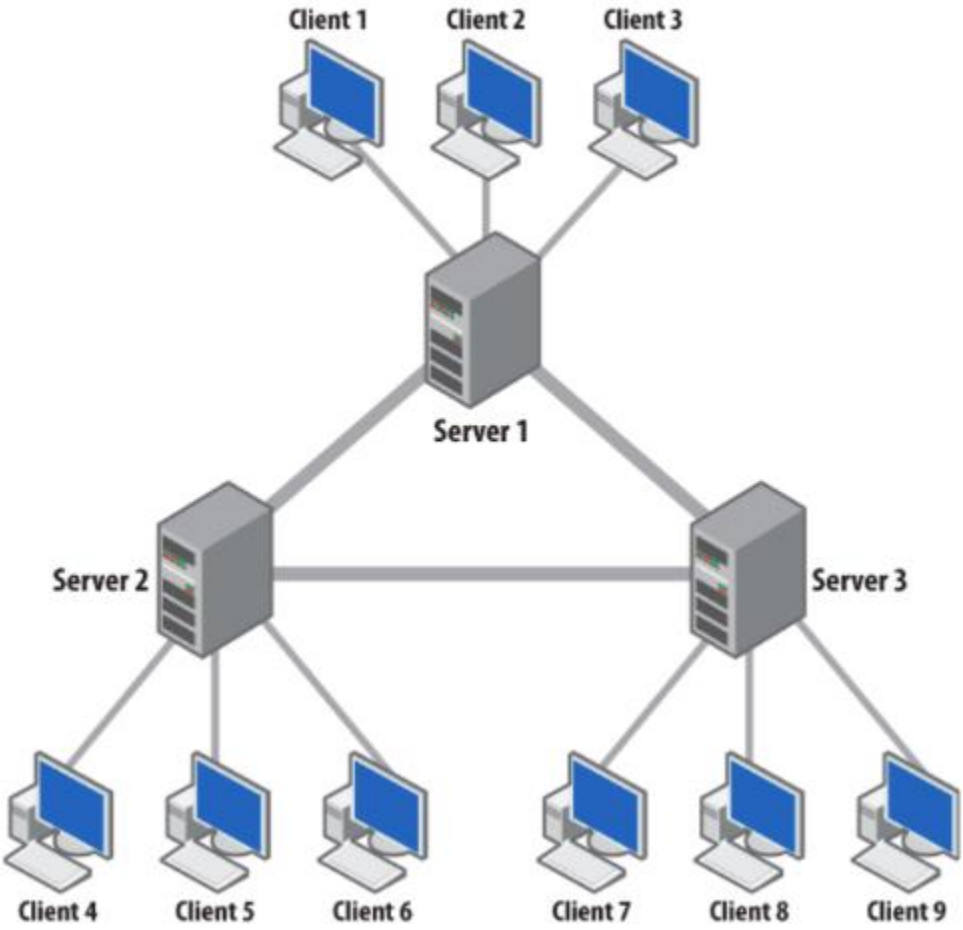
- Instant Messaging.
- Groupchat.
- Gaming.
- System control.
- Geolocation.
- Middleware and cloud computing.
- Data syndication
- Voice over IP
- Identity services.

XMPP là mô hình phân quyền client-server phi tập trung, được sử dụng cho các ứng dụng nhắn tin văn bản. Có thể nói XMPP gần như là thời gian thực và có thể mở rộng đến hàng trăm hàng nghìn nút. Dữ liệu nhị phân phải được mã hóa base64 trước khi nó được truyền đi trong băng tần. XMPP tương tự như MQTT, có thể chạy trên nền tảng TCP.

4.3 Chi tiết giao thức XMPP

4.3.1 Kiến trúc giao thức XMPP

XMPP sử dụng kiến trúc phân cấp client – server. Nghĩa là các client không kết nối và trao đổi thông tin trực tiếp với nhau mà phải thông qua server [8].



Hình 14: Kiến trúc phân cấp client – server XMPP [8]

Điểm hay của việc sử dụng kiến trúc phân cấp client – server là có thể phân chia một cách rõ ràng các mối quan tâm trong việc xây dựng ứng dụng. Đối với việc phát triển client thì tập trung vào giao diện, mức độ tiện lợi cho người sử dụng, trong khi đó đối với phát triển server thì tập trung vào độ bảo mật, tin cậy và khả năng mở rộng của ứng dụng. Nó đơn giản hơn rất nhiều trong việc tổ chức quản lý các ứng dụng sử dụng công nghệ ngang hàng. Và do đó cộng đồng XMPP sẽ xây dựng client đơn giản nhất có thể, trong khi đó sự phức tạp sẽ được đẩy mạnh khi phát triển các server.

- Kiến trúc của XMPP có khá nhiều điểm tương đồng với kiến trúc World Wide Web và Email tuy nhiên cũng có một số điểm khác biệt:
- XMPP giống Email trong khả năng kết nối liên tên miền, còn World Wide Web thì không hỗ trợ.
- Trong khi Email hỗ trợ "multiple hop" nghĩa là khả năng kết nối giữa nhiều server với nhau, thì XMPP chỉ hỗ trợ kết nối "single hop" là chỉ kết nối giữa 2 server với nhau, còn World Wide Web thì không hỗ trợ chức năng này.

4.3.2 Địa chỉ

Mỗi thực thể XMPP trên mạng được xác định bởi một địa chỉ gọi là JabberID (JID). Cơ bản XMPP dựa trên hệ thông tên miền DNS để xây dựng cấu trúc của một địa chỉ thay vì sử dụng địa chỉ IP. JID khá giống với một địa chỉ email có dạng user@domain.tld [8].

- Domain Là giá trị full qualified domain name (FQDN) của server nơi cài đặt một dịch vụ XMPP, ví dụ test.xmpp
- Users Khi một tài khoản được tạo tại một dịch vụ XMPP, hệ thống sẽ tự động gán giá trị tài khoản được tạo vào một JabberID có dạng tương tự như một địa chỉ email (user01@test.xmpp), giá trị này thường được gọi là "bare JID" (user@domain.tld).
- Resources XMPP có khả năng cung cấp cho một tài khoản có thể đăng nhập nhiều lần vào hệ thống tại cũng một thời điểm, tuy nhiên mỗi lần đăng nhập tài khoản phải xác định một giá trị gọi là tài nguyên (resource) khác nhau. Khi đó giá trị tài nguyên này sẽ được thêm vào cuối giá trị JID của tài khoản đó ví dụ user01@test.xmpp/home. Giá trị tài nguyên này có thể là tên địa điểm, tên máy, tên phần mềm mà tài khoản dùng để đăng nhập. Giá trị có dạng user@domain.tld/resource thường được gọi là "full JID".

4.3.3 XML Streams

Hai khái niệm cơ bản trong việc trao đổi dữ liệu giữa các thực thể XMPP là : XML streams và XML stanzas [9].

- XML streams là một luồng trao đổi thông tin giữa các thực thể thông qua mạng. Để bắt đầu một XML streams, thực thể sẽ mở một "streams header" là

một gói tin XML với nhãn <stream> và một vài thông tin đi kèm. Kết thúc là một gói tin XML với nhãn </stream>.

- XML stanzas là một đơn vị truyền thông tin, tương tự như một gói tin hay bản tin ở các giao thức khác, mỗi stanza đều có một ý nghĩa và tác dụng riêng. Có 3 loại XML stanzas gồm: message, presence, và iq.

Quá trình trao đổi dữ liệu giữa hai máy đều thông qua XML streams.

INITIAL STREAM	RESPONSE STREAM
<stream>	
	<stream>
<presence> <show/> </presence>	
<message to='foo'> <body/> </message>	
<iq to='bar' type='get'> <query/> </iq>	
	<iq from='bar' type='result'> <query/> </iq>
[...]	
	[...]
</stream>	
	</stream>

Hình 15: Trao đổi thông tin giữa 2 streams [9]

Kết luận

Internet of Things (IoTs) hiện đang là một thuật ngữ cực kỳ phổ biến, gần như là một từ "khóa" hot nhất hiện nay. Đây là từ tổng quát để thể hiện một thế giới mà trong đó: các thiết bị có thể "nói chuyện" được với nhau, và có khả năng kết nối internet để thực hiện một công việc nào đó. Để tối ưu hóa được mô hình Internet of Things - IoT, các thiết bị phải giao tiếp với nhau bằng các "giao thức" – protocol đáp ứng được tiêu chí năng lượng thấp và dung lượng bản tin nhẹ. Trong đó có 3 giao thức truyền tải dữ liệu phổ biến là: MQTT, CoAP, XMPP.

MQTT sử dụng cho truyền thông 2 chiều thông qua các mạng có độ trễ cao và độ tin cậy thấp, nó cũng tương thích với các thiết bị tiêu thụ điện năng thấp. CoAP là một giao thức truyền tải tài liệu theo mô hình client/server dựa trên internet tương tự như giao thức HTTP nhưng được thiết kế cho các thiết bị ràng buộc. XMPP là con đường vận chuyển các gói tin XML giữa các thực thể trên mạng.

Mỗi giao thức có ưu và nhược điểm riêng. vì vậy cần chọn công nghệ phù hợp với từng trường hợp.

Tài liệu tham khảo

[1] Tổng quan về Internet of Things (IoTs) https://tek4.vn/tong-quan-ve-internet-of-things-iots/

[2] TỔNG QUAN VỀ INTERNET OF THINGS http://kdientu.duytan.edu.vn/media/50204/tong-quan-ve-iot.pdf

[3] GIAO THỨC MQTT http://kdientu.duytan.edu.vn/media/50205/giao-thuc-mqtt.pdf

[4] Giao thức MQTT trong IoT là gì ? Những ứng dụng của MQTT như thế nào https://smartfactoryvn.com/technology/internet-of-things/giao-thuc-mqtt-la-gi-nhung-ung-dung-cua-mqtt-nhu-the-nao/

[5] Giao thức CoAP: Hướng dẫn từng bước https://helpex.vn/article/giao-thuc-coap-huong-dan-tung-buoc-5c6ba207ae03f61e2464e0ef

[6] CoAP Protocol: Step-by-Step Guide https://dzone.com/articles/coap-protocol-step-by-step-guide

[7] Những điều cần biết về giao thức CoAP, sự khác biệt giữa CoAP và MQTT https://bkaii.com.vn/tin-tuc/230-nhung-dieu-can-biet-ve-giao-thuc-coap-su-khac-biet-giua-coap-va-mqtt

[8] Tài liệu Xây dựng ứng dụng instant message theo giao thức xmpp https://xemtailieu.com/tai-lieu/xay-dung-ung-dung-instant-message-theo-giao-thuc-xmpp-34650.html

[9] Extensible Messaging and Presence Protocol (XMPP): Core https://xmpp.org/rfcs/rfc6120.html

[10] IoT - Internet Of Thing: 5 giao thức dùng để "nói chuyện" mà bạn cần biết https://bkaii.com.vn/tin-tuc/175-iot-internet-of-thing-5-giao-thuc-dung-de-noi-chuyen-ma-ban-can-biet