WHITE PAPER

MQTT—IIOT 시대에 Edge-device 연결 활성화

Chase Shih
Moxa Product Manager

MOXA®

개요

MQTT 프로토콜은 거의 30 년 동안 사용되어 왔지만, 이 프로토콜의 설계는 자동화 엔지니어링의 최신 트렌드인 IIoT(Industrial Internet of Things) 애플리케이션에 적합합니다. 이는 특히 디바이스가 정기적으로 폴링되는 "수동적 알림"과 반대로 디바이스가 필요할 때만 데이터를 제공하는 "능동적 알림"을 강조하는 애플리케이션에 해당됩니다. MQTT 의 브로커/클라이언트 설계를 통해 시스템의 모든 기기가 동시에 온라인 상태가 될 필요가 없도록 하였습니다. 클라이언트 (즉, "디바이스"또는 "사물")는 클라이언트들간에 메시지를 주고 받는 중개 역할을 하는 브로커와 직접 통신을 합니다.

Moxa는 산업용 인터넷 (Industrial Internet of Things)의 연결을 가능하게 하는 엣지 연결성, 산업용 네트워킹 및 네트워크 인프라 솔루션의 선도적 제공 업체입니다. 전 세계적으로 5 천만 대 이상의 장치를 연결했으며 70 개국 이상의 고객에게 판매되는 유통 및 서비스 네트워크를 보유하고 있습니다.

여의시스템은 국내 유일 MOXA 기술지원센터(MARC:MOXA Authorized Repair Center)를 보유한 공식 디스트리뷰터로서 오랫동안 축적된 산업 자동화 기술로 다양한 자동화 네트워크 솔루션과 전문적인 서비스를 지원합니다. MOXA 솔루션을 통해 귀사의 산업 네트워크를 완전히 보호하여 OT와 IT 영역 간의 차이를 줄이도록 설계된 차별화된 솔루션을 제공합니다.

Authorized Distributor MOXA 여의시스템

여의시스템 연락처

Tel: 031-737-8888

Fax: 031-737-8800

E-mail: moxa@yoisys.com

Webpage: www.yoisys.com

MOXA®

서문

사람들의 삶의 질을 향상시키는 것은 항상 새롭고 더 나은 기술 향상을 추구하는 주요 동기 중 하나였으며, 현재 인터넷에 점점 더 많은 디바이스를 연결하려는 추진과 함께 소위 IoT 애플리케이션을 위한 더 나은 제품을 개발하는 것이 오늘날 가장 인기 있는 주제 중 하나입니다. 산업용 IoT(IIoT) 엔지니어의 가장 큰 과제 중 하나는 "엣지 디바이스"라고 불리는 "사물"이 안정적인 유선이나 무선 연결에 대한 접근성을 갖추지 못할 수 있다는 것입니다. 엣지 디바이스는 중앙 시스템에 데이터를 간헐적으로 제공하기 때문에 이러한 디바이스에서 데이터를 수집하는 방법은 큰 문제입니다. MQTT, AMQP 및 CoAP 를 포함한 몇 가지 프로토콜들이 IIoT 연결 요구 사항을 충족시킬 수 있습니다. 그러나 MQTT 는 대부분의 IIoT 응용 프로그램에서 최고의 선택이 되었습니다. 아래 그림 1 을 참조하면 IoT 개발자의 절반 이상이 MQTT 를 통신 프로토콜로 사용함에 따라, MQTT 가 IoT 애플리케이션에 가장 적합한 프로토콜임을 알 수 있습니다.

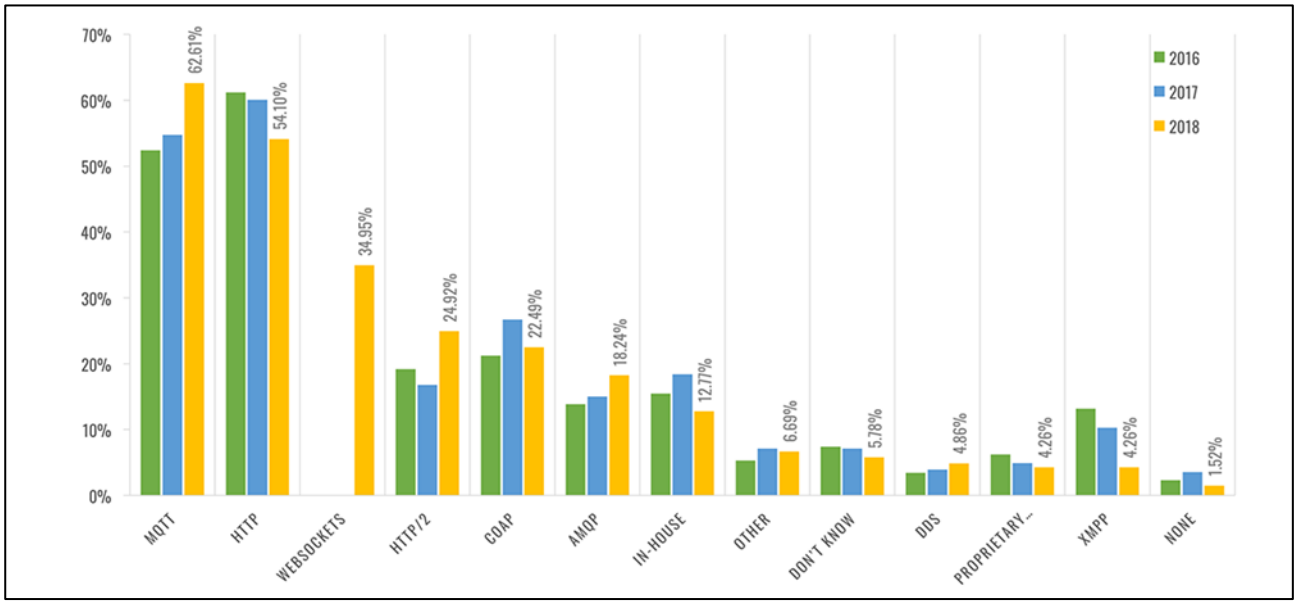

Figure 1. MQTT 는 IoT 애플리케이션을 위한 톱 프로토콜

MQTT 란?

MQTT 메시징 프로토콜은 IBM 과 Cirrus Link 에 의해 1999 년에 처음 개발되었으며, 버전 3.1 을 시작으로 2013 년에 ISO 표준으로 채택되었습니다. MQTT 는 발행-구독 패턴(그림 2 참조)을 사용하여 메시지를 교환합니다. 그림에서 볼 수 있듯이 MQTT 시스템은 하나의 브로커와 여러 개의 클라이언트로 구성되어

있으며, 여기서 클라이언트는 발행자나 구독자가 될 수 있습니다. 발행자는 "토픽"과 "페이로드"로 구성된 MQTT 패킷의 형태로 브로커에게 데이터를 보냅니다. 그런 다음 브로커는 관심을 표명한 주제에 따라 구독자에게 데이터를 분배합니다. MQTT 프로토콜은 애플리케이션이 JSON 프로토콜이나 일반 텍스트를 사용하는 것이 일반적이지만 데이터 전송을 위한 표준 형식을 지정하지 않습니다. MQTT 는 다른 프로토콜에 비해 IoT 애플리케이션과 완벽하게 어울리는 장점이 있습니다.

Figure 2. 발행-구독 패턴

발행-구독 메세징 패턴(Publish-Subscribe Messaging Pattern)

다른 요청-응답 패턴 프로토콜과 비교하여, MQTT 에서 사용하는 발행-구독 패턴은 IoT 개발자가 특정 공통 연결 문제를 해결할 수 있도록 합니다. 예를 들어, 요청-응답 패턴은 데이터가 성공적으로 전송되고 수신되도록 클라이언트와 서버를 동시에 온라인 상태로 유지해야 합니다. 그러나, 특히 IIoT 애플리케이션의 경우, 디바이스가 필요한 데이터를 수신기에 충분한 네트워크 연결을 유지하는 것이 불가능할 수 있으므로 요청-응답 패턴이 해당 애플리케이션에 적합하지 않습니다.

MQTT 의 발행-구독 패턴은 디바이스가 동시에 네트워크에 연결되어 있지 않는 상황에 잘 맞습니다. MQTT 브로커는 이 점에서 중요합니다.

브로커는 "발행자"로 지정된 클라이언트로부터 전송된 데이터를 받아들인 후, "구독자"로 지정된 클라이언트에게 전송하는 정보센터로서의 역할을 합니다.

브로커는 구독자에게 데이터를 보낼 때, 우선 대상 클라이언트가 온라인 상태인지 여부를 확인합니다. 그렇지 않을 경우, 브로커는 구독자가 온라인 상태가 될 때까지 데이터를 보유한 다음 보낼 수 있습니다.

이 전략의 이점 중 하나는 브로커만 항상 온라인에 있어야 한다는 것입니다. 발행자와 구독자 모두 접속이 가능하거나 데이터를 송수신해야 할 때만 온라인으로 접속하면 됩니다.

이벤트 중심(Event-driven)

발행-구독 패턴을 사용할 때 MQTT 클라이언트는 특정 조건이 충족될 때만 데이터를 브로커에게 공개합니다(예: 경고 신호는 특정 디바이스의 온도가 너무 높다는 것을 나타낼 수 있음). 이러한 유형의 동작을 설명하는 또 다른 방법은 클라이언트가 데이터를 요청하는 다른 디바이스를 수동적으로 기다리는 대신 데이터를 능동적으로 업데이트하는 것입니다. IoT 애플리케이션의 경우 전송되는 데이터 패킷 수에 따라 통신 요금이 부과됩니다. MQTT 는 요청-응답 패턴과 비교하여 데이터 전송을 완료하기 위해 단방향 통신만 필요하므로 비용을 절감합니다.

다 대 다 통신(Many-to-many Communication)

MQTT 의 주요 장점 중 하나는 발행-구독 패턴을 사용하여 다 대 다 통신을 쉽게 설정할 수 있다는 것입니다. 다 대 다 통신을 실현 한 M2M (Machine-to-Machine) 개념은 IIoT 에서 가장 인기있는 주제 중 하나입니다. 공장 M2M 애플리케이션에서 각 스테이션의 머신은 다른 스테이션의 머신과 자체 프로세스 상태를 공유합니다. 이러한 방식으로 정보를 공유하면 작업자의 수동 입력 없이도 생산 최적화를 자동화 할 수 있습니다. MQTT 는 M2M 통신을 구현하는 데 사용되므로, 기계는 서로 직접 연결하는 대신 브로커와의 연결만 설정하면 되므로 핸드 쉐이킹에 상당한 시간이 절약됩니다. 하나의 브로커가 모든 머신 간의 통신을 처리하는 데 전념하기 때문에 데이터 전송이 보다 안정적입니다.

통신서비스품질 설계(QoS Design)

MQTT 프로토콜은 세 가지 QoS 레벨을 사용하여 데이터의 우선 순위를 지정합니다.

- **QoS 0: 최대 한번**

  이 경우 클라이언트는 메시지를 브로커에 한 번만 발행합니다. 브로커는 메시지 수신을 승인하지 않거나 구독자와의 통신에 관한 알림을 클라이언트에게 제공하지 않습니다. 유일한 보증은 발행자가 메시지를

보냈다는 것을 알고 있다는 것입니다. 그러나 브로커 나 구독자가 메시지를 받았는지 여부는 알 수 없습니다. QoS 0 은 서비스 품질 정책이 가장 빠르지만 안정성도 가장 낮습니다.

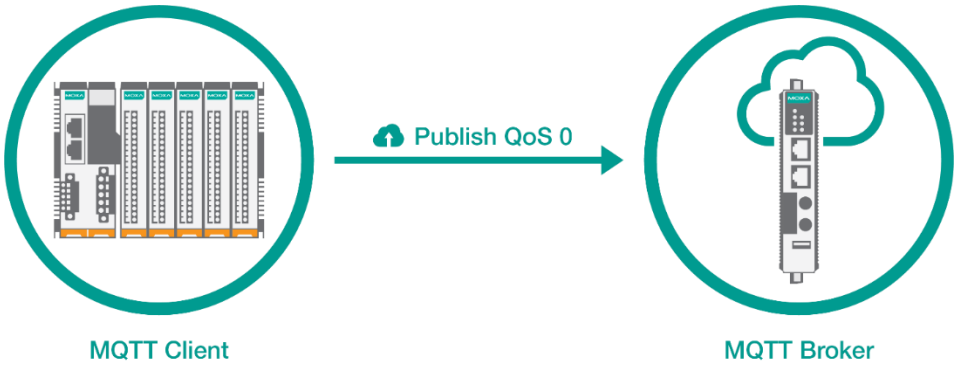

Figure 3. QoS 0: 최대 한번

- **QoS 1: 적어도 한번**

이 경우 클라이언트가 브로커에 메시지를 발행하면 클라이언트는 브로커가 클라이언트가 메시지를 받았는지 여부를 확인해야 합니다. 발행자가 미리 설정된 시간 간격 내에 브로커로부터 승인을받지 못하면 승인을받을 때까지 메시지를 계속 다시 발행합니다. QoS 0 과 비교할 때 QoS 1 은 시간이 지남에 따라 속도가 느릴 것으로 예상 할 수 있지만 더 안정적입니다.

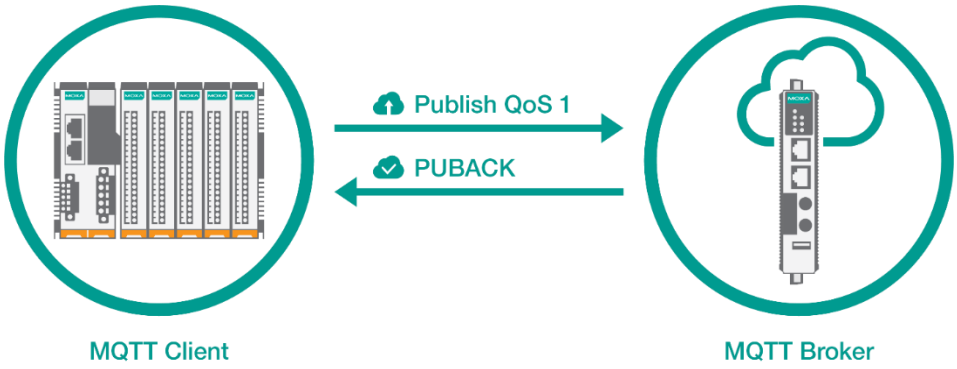

Figure 4. QoS 1: 적어도 한번

- **QoS 2: 정확히 한번**

이 경우 클라이언트와 브로커는 4 개의 메시지를 교환합니다. 클라이언트는 먼저 데이터를 브로커에 공개 한 다음 클라이언트와 브로커가 PUBREC, PUBREL 및 PUBCOMP 의 세 가지 메시지를 교환하여

데이터가 한 번만 전달되도록 합니다. QoS 2 는 가장 느리지 만 MQTT 서비스 품질 정책 중 가장 안정적입니다.

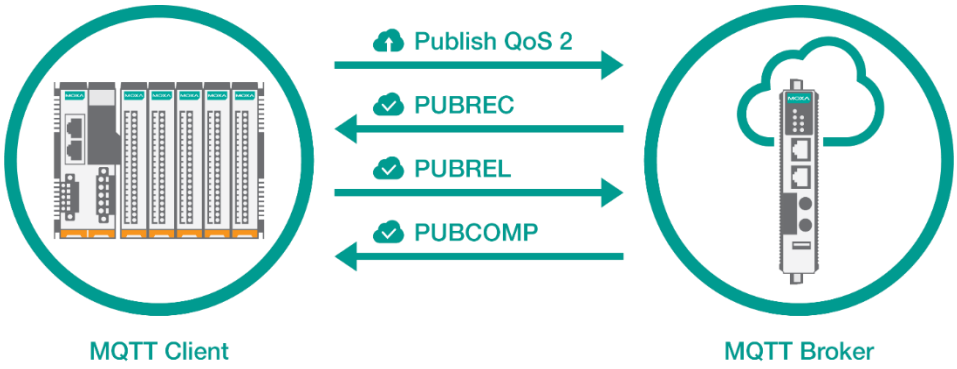

Figure 5. QoS 2: 정확히 한번

보안(Security)

보안은 IIoT 응용 프로그램의 주요 관심사입니다. 인터넷에 점점 더 많은 디바이스가 연결됨에 따라 데이터가 해킹 될 가능성을 최소화하는 방법을 아는 것이 최우선 과제입니다. 그림 6 은 IoT 애플리케이션에서 보안이 가장 중요한 관심사임을 분명히 보여줍니다. MQTT 와 관련하여 브로커는 권한이 없는 클라이언트가 브로커에 연결하여 토픽들을 구독하지 못하도록 계정 이름 및 비밀번호를 지원합니다. MQTT 는 또한 데이터 전송을 위해 TLS 암호화를 지원하여 전송 중에 데이터가 해킹 될 가능성을 크게 최소화합니다.

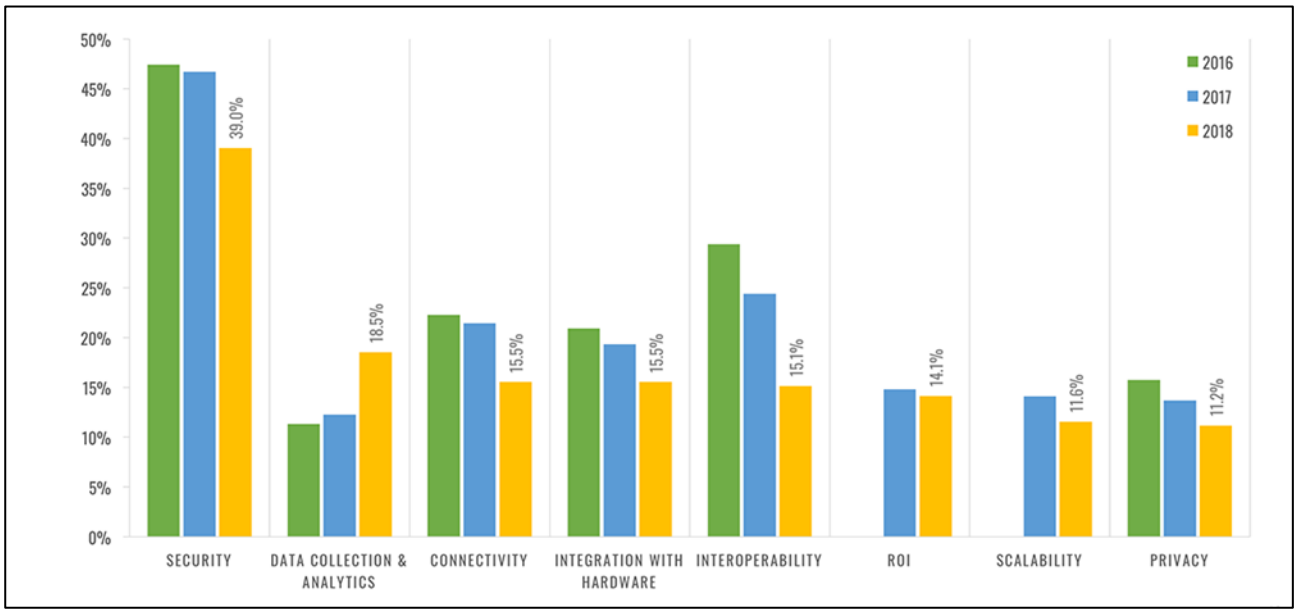

Figure 6. IOT 적용시 보안이 가장 큰 관심사

MQTT Application Architecture

본 글의 시작에서 지적했듯이, 기존의 OT 애플리케이션은 널리 사용되는 MQTT 프로토콜을 사용하는 IIoT 애플리케이션용으로 재장착되고 있습니다. 두 개의 주요 시스템 아키텍처가 사용됩니다.

클라우드에 직접 연결(Connecting Directly to the Cloud)

대부분의 퍼블릭 클라우드 서비스 (AWS, Azure, Google Cloud, Alibaba Cloud 등)는 MQTT 프로토콜을 지원하여 엣지 디바이스가 클라우드에 직접 연결할 수 있도록 합니다. 경쟁력을 유지하고 산업의 미래를 형성하려면 클라우드 서비스는 최소한 다음과 같은 이점을 제공해야 합니다.

- **시간절약(Time savings)**

  클라우드 서비스는 하드웨어 (클라우드 서버, CPU, 메모리 등)를 유지 관리하므로 이러한 보다 전문적인 유지 관리 작업을 클라우드 서비스의 IT 전문가에게 맡김으로써 사용자는 자체 솔루션 개발에 더 많은 시간을 할애 할 수 있습니다.

- **무중단 서비스(Non-stop service)**

  고객은 클라우드 서비스 제공 업체로부터 거의 100 %의 안정성을 기대하며 안정성과 네트워크 안정성을 가장 중요하게 생각합니다. 모든 클라우드 서비스 제공 업체는 SLA (서비스 수준 계약)에 제공하려는 보장 된 서비스 안정성 수준을 명확하게 명시합니다. 예를 들어 Amazon Compute SLA 에서 Amazon 은 월간 가동 시간을 99.99 %로 약속합니다. 즉, 서비스 중단 시간은 한 달에 4.32 분 미만이 될 것으로 예상됩니다. 이 수준의 서비스를 "논스톱 서비스"라고 말할 수 있습니다.

- 풍부한 데이터 마이닝 도구(Rich set of data mining tools)

  클라우드 서비스는 플랫폼의 일부로 데이터 시각화, 데이터 알고리즘, 가상 머신 및 머신 러닝을 포함한 다양한 도구를 제공합니다. 이러한 도구에 액세스하면 다양한 응용 프로그램을 보다 쉽게

구현할 수 있습니다. 예를 들어 엔지니어는 데이터 마이닝에 클라우드 서비스를 사용하여 서버 유지 관리 노력을 줄이고 운영 효율성을 향상시킬 수 있습니다.

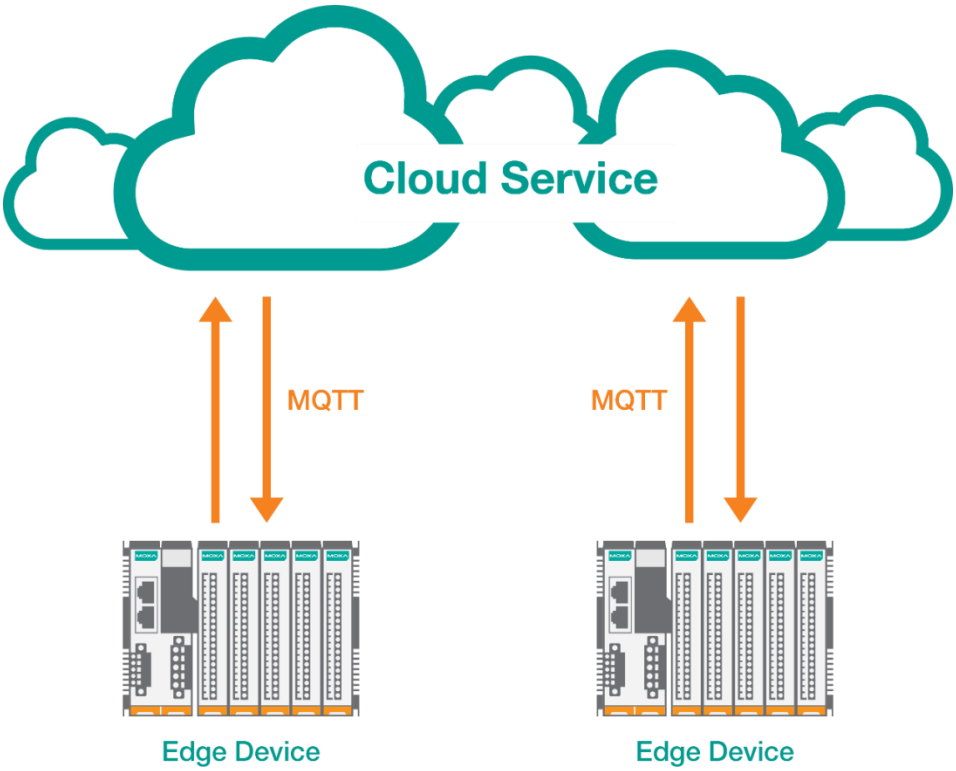

Figure 7. 클라우드에 직접 연결하기위한 아키텍처

로컬 게이트웨이에 연결 (Connecting to a Local Gateway)

엣지 디바이스를 클라우드에 직접 연결하면 이점이 있지만 IIoT 응용 프로그램에 클라우드 서비스를 도입하는 것과 관련된 다양한 문제를 알고 있어야 합니다.

- 첫 번째 관심사는 비용입니다. 클라우드 서비스는 전송되는 데이터 패킷 수에 따라 사용자에게 요금을 청구하므로 엣지 디바이스에서 클라우드 서비스로 데이터를 직접 전송하는 것은 비용 효율적이 아닙니다. 엣지 디바이스가 셀룰러 네트워크를 통해 클라우드에 연결되어 있어도 셀룰러 서비스에 대한 비용을 지불해야합니다.

- 두 번째 관심사는 데이터 보안입니다. 클라우드 서비스는 사용자 데이터를 저장하기 위해 잘 보호 된 환경을 제공하지만 일부 사용자는 여전히 민감한 데이터를 클라우드에 업로드하는 것을 주저합니다.

대부분의 IIoT 응용 프로그램의 경우 현장 사이트에 게이트웨이를 설정하여 엣지 디바이스 데이터를 수집하거나 현장 사이트에서 M2M 통신을 활성화하는 것이 이러한 문제를 피하는 한가지 방법입니다. 게이트웨이는 일반적으로 임베디드 컴퓨터이며 MQTT 브로커 및 MQTT 클라이언트 역할 모두에 대해 반드시 구성 할 필요는 없지만 그렇게 엣지 디바이스구성 할 수 있습니다. MQTT 브로커로서, 게이트웨이는 현장에서 M2M 데이터 전송을 처리 할 수 있습니다. MQTT 클라이언트인 게이트웨이는 현장 사이트 디바이스 데이터를 수집하고 사용 가능한 데이터를 SCADA 시스템, HMI 또는 클라우드 서비스로 보낼 수 있습니다. 게이트웨이 솔루션은 클라우드 대신 필드 사이트에서 M2M 통신을 가능하게 하는 MQTT 를 사용하여 비용을 더욱 최소화합니다.

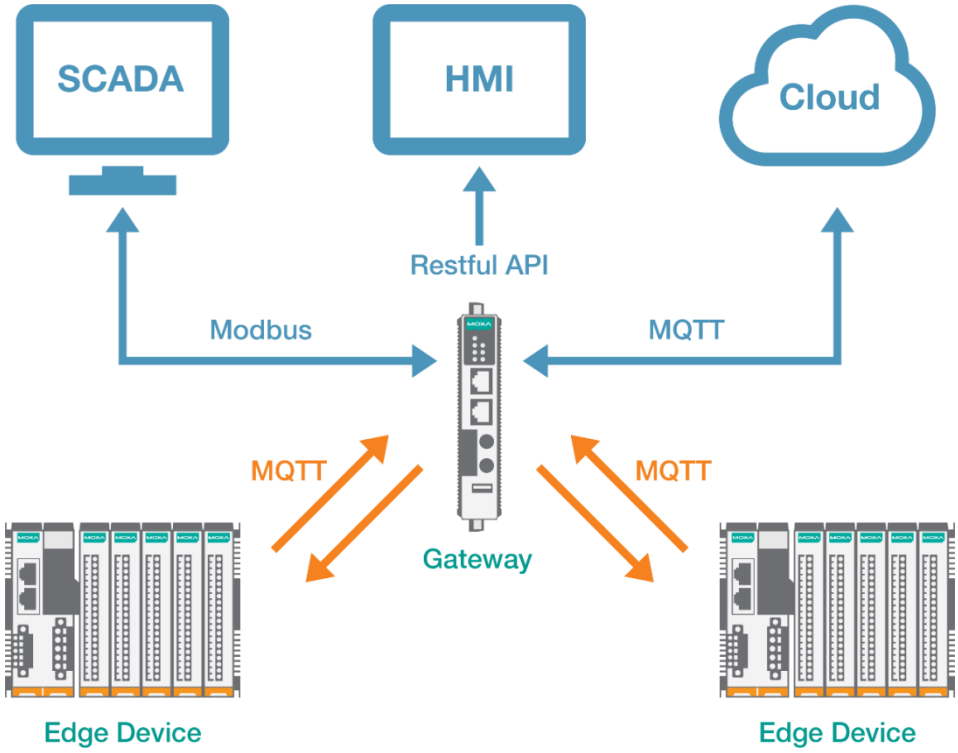

Figure 8. 로컬 게이트웨이에 연결을 위한 아키텍처

IIoT 응용 프로그램으로 변환해야 할 과제들

기존 OT 애플리케이션을 IIoT 애플리케이션으로 변환 할 때 다음과 같은 문제 중 일부 또는 모두가 발생할 수 있습니다.

현재 사용중인 레거시 디바이스들은 MQTT 를 지원하지 않습니다

공장에서 시설 엔지니어는 일반적으로 데이터 액세스 및 환경 모니터링을 위해 원격 I / O 설정을 사용합니다. 또한 프로토콜 게이트웨이는 전력계 데이터를 수집하고 에너지 소비를 모니터링하는 데 사용됩니다. 현재 IIoT 트렌드가 본격화되면서 MQTT 프로토콜을 사용하여 데이터를 클라우드로 전송하려는 경우 시설 엔지니어는 먼저 MQTT 를 지원하는 새로운 원격 I / O 제품 및 게이트웨이를 조사하고 구매해야 합니다. 전 세계 현장에서 여전히 많은 레거시 디바이스가 사용되고 있기 때문에 공장을 IIoT 기반 설정으로 변환하려면 막대한 투자가 필요할 수 있습니다.

기존의 자동화 응용 프로그램과 IT 를 통합하는 것이 더 쉬워졌습니다

IIoT 애플리케이션의 가장 기본적인 측면 중 하나는 OT 데이터를 수집하여 클라우드로 전송 한 후 데이터를 처리 및/또는 분석 할 수 있습니다. IT 와 OT 산업은 서로 다른 전송 프로토콜을 사용한다는 사실에서 문제가 발생합니다. OT 필드에서 가장 널리 사용되는 프로토콜 중 하나인 Modbus 는 작은 헤더와 페이로드가 있는 데이터 패킷을 사용하여 제한된 대역폭 네트워크를 통해 패킷을 전송할 수 있습니다. 반면 IT 엔지니어는 MQTT, RESTful API 및 SNMP 와 같은 IT 프로토콜을 사용하여 데이터를 수집하므로 많은 IT 엔지니어가 Modbus 에 익숙하지 않습니다.

보안은 중요한 요소입니다

IIoT 응용 프로그램의 주요 관심사는 네트워크 보안 유지입니다. 과거의 경험에서 사이버 공격은 공장 외부에서 시작되었으므로 사이버 보안을 개선하기 위한 첫 번째 단계는 보안 라우터를 설치하고 방화벽을 구성하여 해커를 차단하고 일반적으로 외부 공격을 방지하기 위해 네트워크 보안을 업그레이드하는 것입니다. 공장 인트라넷의 엣지 디바이스는 제한된 보안 기능 (있는 경우)을 지원할 뿐만 아니라 암호화되지 않은 프로토콜을 사용하는 경우가 더 많다. 예를 들어 Modbus 는 일반적으로 엣지 디바이스와 데이터를 주고받는 데 사용됩니다. 최근 몇 년 동안 특정 사이버 공격이 산업 네트워크 보안 문제에 집중되었습니다. 예를 들어, 2018 년 8 월 TSMC 는 WannaCry 변종의 사이버 공격으로 인해 약 2 억 달러의 매출 감소를 가져왔습니다.

이 공격은 TSMC 인트라넷의 모든 디바이스가 최신 보안 패치를 설치하지 않았기 때문에 발생했으며, 이는 원칙적으로 공격을 쉽게 방지 할 수 있었음을 의미합니다. 이 사건을 통해 배울 수 있는 중요한 교훈은 네트워크 보안이 어떤 방식으로든 엣지 디바이스 수준에서 구현해야 한다는 것입니다.

Moxa 솔루션

Moxa 가 새롭게 출시한 ioThinx 4510 시리즈 모듈식 원격 I / O 디바이스에는 IIoT 응용 프로그램과 완벽하게 일치하는 주요 기능이 있습니다.

MQTT 클라이언트 지원

ioThinx 4510 시리즈는 MQTT 클라이언트를 지원하므로 ioThinx 4510 에 연결된 디바이스가 클라우드 서비스에 쉽게 연결할 수 있습니다. ioThinx 4510 시리즈는 보급형 원격 I / O 제품으로 권장되었지만 MQTT 를 지원하므로 강력한 자산이 됩니다. 실제로 ioThinx 4510 의 사용자 인터페이스를 사용하여 자신의 MQTT 토픽을 정의한 다음, 이 토픽을 기반으로 어떤 클라이언트가 어떤 데이터를 구독하는지 결정할 수 있습니다. ioThinx 4510 시리즈는 채널 데이터 외에도 구독자에게 채널 모드, 최대 값 또는 최소값 등의 데이터 속성을 제공할 수 있어 네트워크에 연결된 IT 유형의 디바이스가 MQTT 를 통해 ioThinx 4510 의 최신 상태를 얻을 수 있습니다. MQTT 페이로드는 오늘날의 IT 산업에서 널리 사용되는 JSON 형식을 사용합니다. 구독자가 MQTT 패킷을 수신하면 특정 키워드 "값"으로 데이터를 쉽게 검색하여 페이로드에서 찾고 있는 데이터를 찾을 수 있습니다.

내장 Modbus 게이트웨이

ioThinx 4510 시리즈에는 Modbus 게이트웨이를 구현하는 데 사용할 수 있는 3-in-1 시리얼 인터페이스가 내장 되어 있어, 시리얼 Modbus 디바이스에서 데이터를 수집하도록 ioThinx 4510 을 설정하는 데 몇 번의 클릭만으로 충분합니다. I / O 데이터와 마찬가지로 시리얼 Modbus 데이터는 MQTT 에 의해 액세스됩니다. 이 기능 덕분에 단일 ioThinx 4510 시리즈 디바이스를 사용하여 I / O 및 직렬 데이터를 수집 할 수 있으므로 시스템의 복잡성과 비용이 크게 줄어 듭니다. ioThinx 4510 시리즈는 Modbus 데이터 외에도 Modbus / TCP, RESTful API 및 SNMP 를 포함한 다른 프로토콜들을 액세스 할 수 있습니다. 한마디로 ioThinx 4510 시리즈를 사용하면 시리얼 데이터를 클라우드에 쉽고 간단하게 전달 할 수 있습니다.

보안 향상

사용자 데이터를 보호하기 위해 ioThinx 4510 시리즈는 MQTT 전송을 통해 전송 된 데이터를 암호화하기 위해 TLS v1.2 를 지원합니다. 널리 사용되고 인식되는 이 데이터 암호화 기술은 해커로부터 네트워크를 통해 전송되는 데이터를 보호합니다. ioThinx 4510 은 또한 브로커의 계정 이름과 비밀번호를 지원하여 데이터가 인증되지 않은 브로커에 발행되는 것을 방지하고 https 및 SNMPv3 을 통해 RESTful API 를 지원하므로 지원되는 모든 IT 프로토콜이 데이터 암호화로 전송할 수 있습니다. 일반 텍스트로 데이터를 전송하는 Modbus / TCP 프로토콜의 경우, ioThinx 4510 시리즈는 ioThinx 4510 에 액세스 할 수 있는 권한이 부여 된 IP 주소 목록을 사용하여 운영 보안을 강화하는 액세스 제어 기능이 있습니다.

이러한 고급 기능과 다양한 I / O 모듈에 대한 지원을 통해 ioThinx 4510 시리즈는 IT 엔지니어가 현장 데이터를 수집 할뿐만 아니라 OT 엔지니어가 클라우드 서비스에 데이터를 안전하게 전달할 수 있도록 도와줍니다. 실제로 ioThinx 4510 시리즈는 IT 와 OT 세계 사이의 격차를 해소합니다.

References

[1] Amazon Compute Service Level Agreement, Amazon Web Services from https://aws.amazon.com/compute/sla/

[1] Wu, D., Gurman, M. (2018, Aug. 5th). iPhone Chipmaker Races to Recover After Crippling Computer Virus, *Bloomberg* from https://www.bloomberg.com/news/articles/2018-08-05/iphone-chipmaker-races-to-recover-after-crippling-computer-virus