SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Protokol MQTT a OPC UA Pub/Sub

Učebný text

Oblasť informačné, komunikačné a riadiace systémy

KEGA038STU-4/2018 Ing. Lukáš Beňo

Obsah

Zoznam obrázkov

Zoznam skratiek a značiek

ERP	Plánovanie podnikových zdrojov
IoT	Internet vecí
IIoT	Primyslený internet vecí
MQTT	Riadenie správ telemetrického transportu
SSL	Vrstva bezpečných soket-ov
TLS	Zabezpečenie prenosovej vrstvy
VPN	Virtuálna privátny sieť
QoS	Kvalita služby
M2M	Stroj k stroju

Zoznam tabuliek

Úvod

Termín IoT prvý krát použil v roku 1999 Kevin Ashton, pričom ho využil ako názov pre svoju prezentáciu, ktorá bola určená spoločnosti Procter & Gamble (P&G). Avšak pozornosti termínu IoT sa dostalo o 10 rokov neskôr, keď bol termín použitý ako názov publikácie (That 'Internet of Things' Thing"), ktorú vydal K. Ashton. Termín bol použitý ako popis pre prepojenie fyzických zariadení cez internet. Myšlienka bola veľmi jednoduchá, fyzické zariadenia si mohli vymieňať dáta medzi sebou alebo ovládať sa navzájom. Medzi takéto zariadenia mala patriť napríklad chladnička, televízor, auto alebo budova. V podstate to mohlo byť akékoľvek elektronické zariadenie. Pojem IoT sa dostal aj do oblasti priemyslu, kde k nemu pribudlo ešte slovíčko Industrial a vzniklo IIoT. Aby sa zariadenia mohli radiť do IoT alebo IIoT musia vedieť navzájom komunikovať. Na komunikáciu sa využíva rôzne komunikačné protokoly, ktoré ponúkajú rôzne riešenia komunikácie [1].

Práca sa venuje opisu protokolu MQTT, ktorý sa využíva pri komunikácii založenej na architektúre Pub/Sub. Taktiež sa práca venuje čiastočne aj protokolu AMQP, ktorý si tiež našiel miesto pri Pub/Sub architektúre. Posledná kapitola je zameraná na opis rozšírenia OPC UA PubSub, ktorý v kombinácií s rozšírením TSN pre ethernet vytvára riešenie vhodné pre priemysel, kde je nutné mať zabezpečený deterministický čas.

1 IoT protokoly

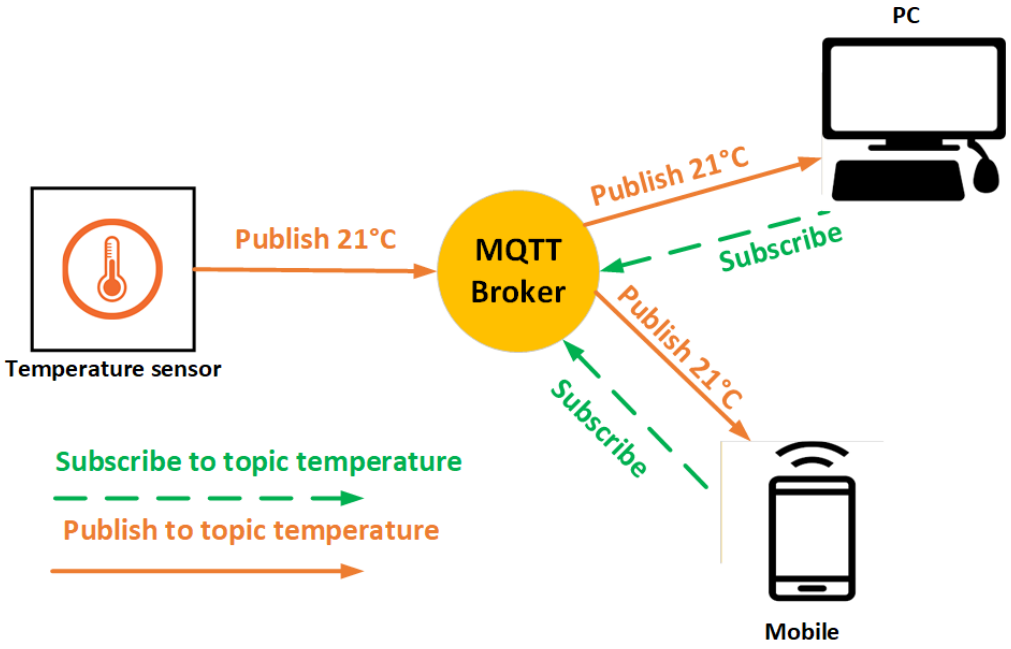
Aby mohli zariadenia navzájom komunikovať je nutné zvoliť protokol, ktorým si budú informácie vymieňať. Nasledujúce kapitoly sa venujú vybraním IoT protokolom, ktoré je možné využiť pre prenos informácií.

1.1 MQTT

Tvorcom protokolu MQTT je Andy Stanford-Clark (IBM) a Arlen Nipper (Eurotech, teraz Cirrus Link). Protokol bol vytvorený v roku 1999 pre účely monitorovania ropovodu, ktorý prechádzal púšťou. Cieľom bolo vyvinúť efektívny protokol pre prenos údajov, ktorý by zároveň dokázali využívať na komunikáciu zariadenia napájané batériou. Batérie v tom období neponúkali veľký výkon a taktiež boli drahé. Zaradenia na monitorovanie ropovodu komunikovali prostredníctvom satelitného pripojenia [2].

1.1.1 MQTT architektúra

Protokol MQTT využíva architektúru publisher/subscriber (Obr. 1), ktorá je opakom HTTP, kde sa využíva princíp request/response. Architektúra publisher/subscriber je založená na udalostiach a umožňuje klientom sa prihlásiť na odber určitých udalostí. Centrálnym komunikačným bodom je MQTT broker, ktorý je zodpovedný za presun správ medzi publisher-mi a subscriber-mi.

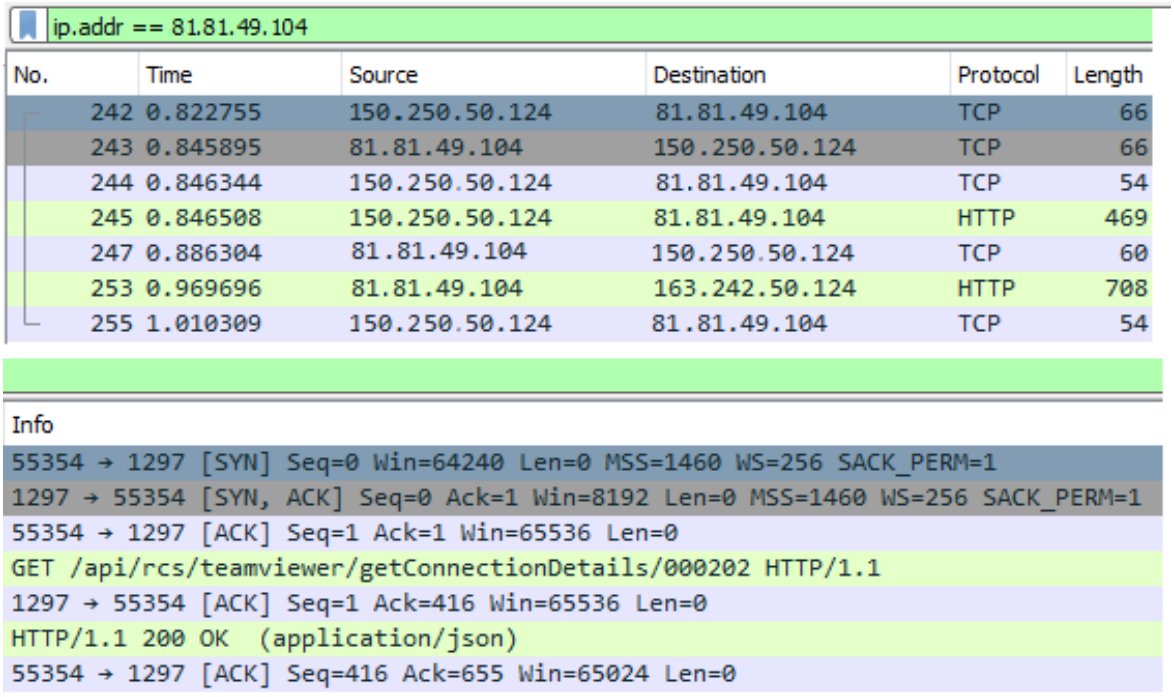


Obr. 1 MQTT architektúra – Subscriber/Publisher

Každý publisher, ktorý chce publikovať správu v MQTT broker-i musí do každej správy pridať aj tému – topic (ďalej len topic), do ktorej chce publikovať. Na základe topic-u vie MQTT broker, ktorým subscriber-om má danú správu poslať, nakoľko subscriber-y sa prihlasujú na odber určitých topic-ov. Preto subscriber-y nemusia nič vedieť o publisher-och a naopak. Vďaka tejto architektúre môžu vznikať vysoko škálovateľné riešenia, nakoľko neexistuje závislosť medzi zariadeniami, ktoré produkujú informácie a zariadeniami, ktoré informácie spracovávajú.

Na obr. 2 je možné vidieť komunikáciu prostredníctvom HTTP. Celá komunikácia prostredníctvom HTTP začína:

1. Vytvorením TCP tunela medzi klientom a serverom, ktorý sa vytvorí pomocou 3-way handshake.
2. Následne je poslaný HTTP request na server.
3. Prichádza odpoveď od servera, že request dorazil.
4. Za potvrdením o dorazený request-u na server, prichádza odpoveď vo forme application/json.
5. Posledným krokom je odpoveď klienta, že dostal response od servera.



Obr. 2 Komunikácia prostredníctvom HTTP

Pri každom HTTP request-e je nutné vždy vykonať procedúru opísanú vyššie. Avšak pri MQTT nie je nutné vždy vytvárať TCP tunel, nakoľko TCP tunel sa vybuduje iba raz pri inicializácií spojenia a následne môže prebehať komunikácia.

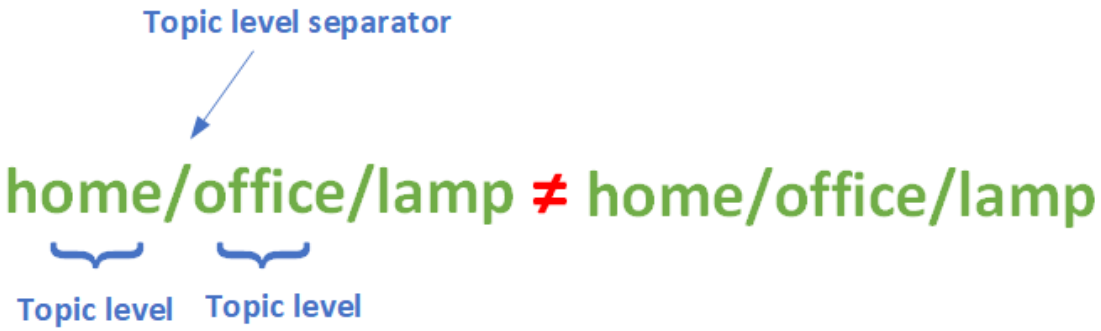
Ako už bolo spomenuté MQTT protokol využíva princíp publisher/subscriber, t. j. klient ma neustále vybudované TCP spojenie k MQTT broker-u., ktorý posúva informácie klientovi v prípade, že je niečo nové. Klientovi posiela iba informácie, na ktoré sa prihlásil. Ak je spojenie prerušené akoukoľvek okolnosťou, broker MQTT si môže správy uchovať a poslať ich klientovi, keď bude opäť online.

Ústredný bod v MQTT na odosielanie správ je topic. Topic je jednoduchý reťazec, ktorý môže mať viac hierarchických úrovní, ktoré sú oddelené lomítkom (Obr. 3). Existujú tri spôsoby ako sa môžeme subscrib-núť [2]:

1. Subscribe na konkrétny topic, v tomto prípade na teplotu z obývačky:
    - **house/living-room/temperature**
2. Subscribe pomocou divokej karty:
    I. **Znamienko „+“ –** dokážeme vytvoriť subscribe pre jednu úroveň hierarchie. Napríklad:
        - **house/+/temperature** – subscribe na teploty zo všetkých miestnosti, t.j. získame hodnoty teplôt z **house/living-room/temperature** a taktiež napríklad aj z **house /kitchen /temperature**

    II. **Znamienko „#“ –** dokážeme vytvoriť subscribe pre viac úrovni hierarchie. Napríklad:
        - **house/# -** subscribe na každý topic začínajúci house

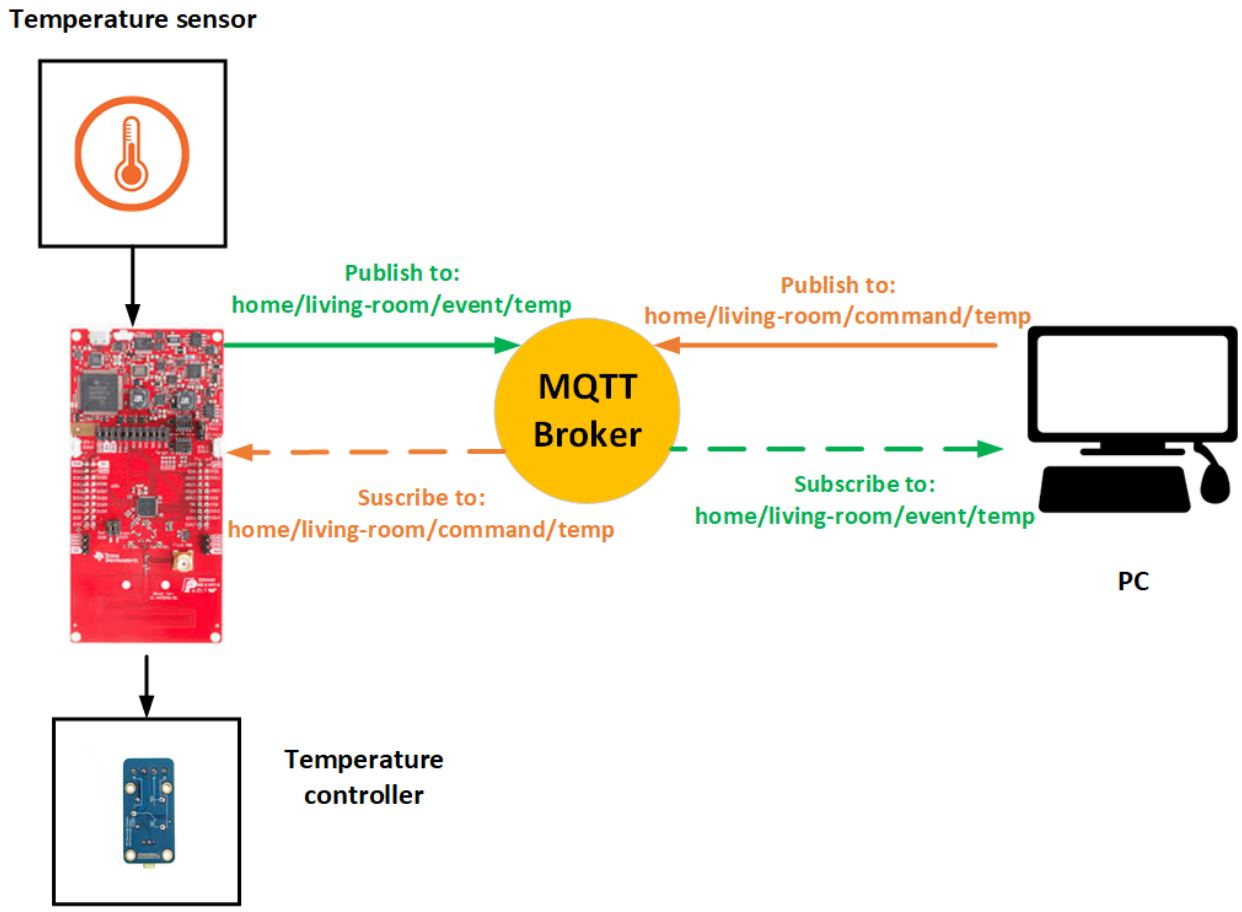
Pri topic-och sa rozlišujú veľké a malé písmena (Obr. 3).



Obr. 3 MQTT – topic

MQTT podporuje obojsmernú komunikáciu, t.j. klient môže byť v úlohe subscriber-a a publisher-a zároveň (Obr. 4). Implementácií MQTT broker-u existuje mnoho, napríklad: Eclipse Mosquitto, CloudMQTT a iné. Protokol MQTT si našiel uplatnenie napríklad aj v mobilnej aplikácií Facebook.

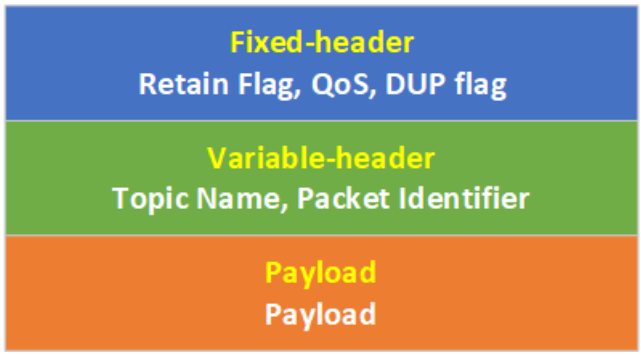
MQTT je vhodný na rýchlu distribúciu údajov. Dokáže uchovať správy v prípade výpadku pripojenia a vie zistiť, či bol údaj doručený alebo nie.



Obr. 4 MQTT – obojsmerná komunikácia

1.1.2 MQTT štruktúra protokolu

Štruktúra MQTT protokolu (Obr. 5) sa skladá z troch hlavných častí [3]:



Obr. 5 MQTT – štruktúra protokolu

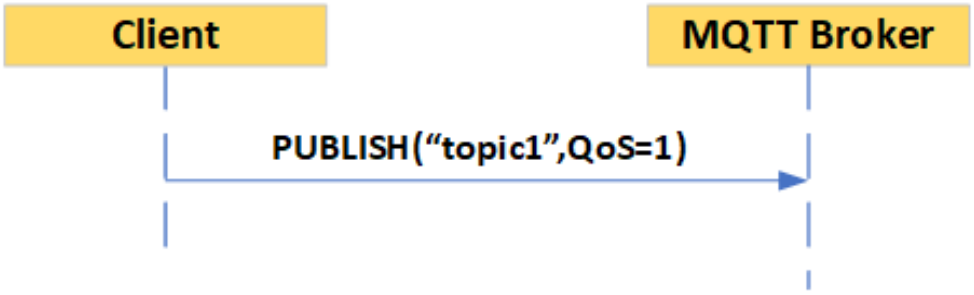
- **Fixed-header**
  - **Retain Flag -** Definuje, či je správa uložená v broker-i, ako posledná známa dobrá hodnota pre daný topic**.** Keď sa nový používateľ prihlási na topic, dostane poslednú správu, ktorá je zachovaná v danom topic-u.
  - **QoS –** Aký druh záruky má správa na dosiahnutie určeného príjemcu. Pre zaručenie QoS sa využíva pakety:

Skratka	Celý názov
PUBLISH	Publish message
PUBACK	Publish acknowledgment
PUBREC	Publish received
PUBREL	Publish release
PUBCOMP	Publish complete

Tab. 1 QoS – celý názov paketov na základe skratiek

QoS má tri hodnoty:

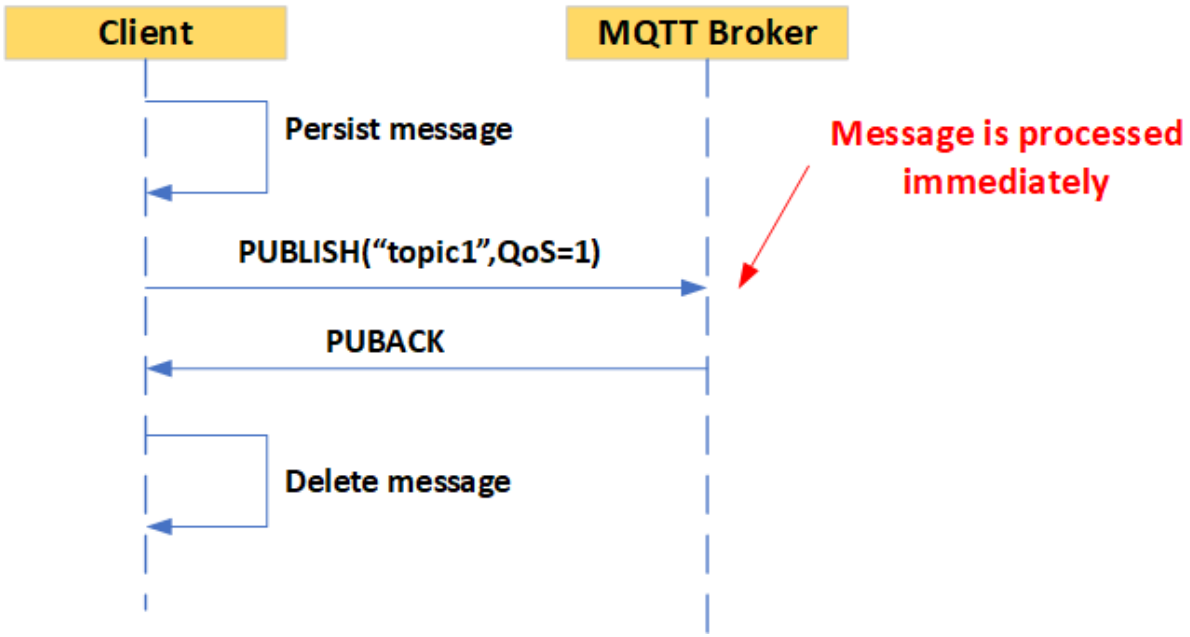
- **QoS 0 –** žiadna garancia doručenia správy (Obr. 6).



Obr. 6 QoS 0 – sekvenčný diagram

- **QoS 1 –** správa bude doručená aspoň raz (Obr. 7). Postup doručenia správy:
    1. Klient si uloží správu, ktorú chce odoslať a publish-ne ju do „topic1“.
    2. MQTT broker príjme správu a okamžite ju spracuje.
    3. MQTT broker posiela PUBACK klientovi o úspešnom doručení správy.
    4. Klient 1 po obdržaní PUBACK vymaže uloženú správu.

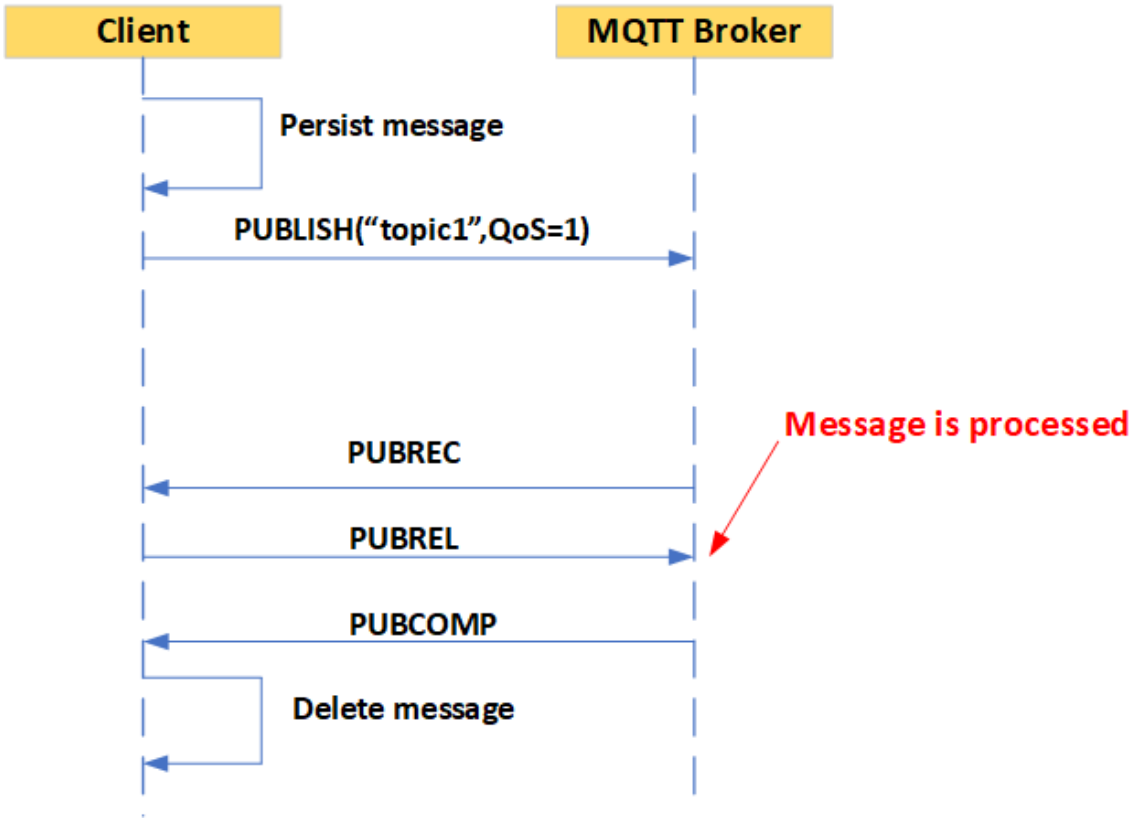
QoS 1 je vhodný v prípadoch, keď nám nevadí, že správa môže byť spracovaná aj viac krát. Pokiaľ správa PUBACK medzi klientom a MQTT broker-om bude stratená, tak dôjde k znovu odoslaniu správy broker-u, ktorý je znova spracuje.



Obr. 7 QoS 1 – sekvenčný diagram

- **QoS 2 –** správa bude doručená len raz. Postup doručenia správy:
    1. Klient si uloží správu, ktorú chce odoslať a publish-ne ju do „topic1“.
    2. MQTT broker získa správu od klienta a odpovie správou PUBREC.
    3. Klient po obdržaní PUBREC odpovie správou PUBREL.

4. MQTT broker môže po obdržaní PUBREL spracovať správu, ktorú mu klient poslal.
5. MQTT broker odpovie klientovi správou PUBCOMP.
6. Klient po obdržaní správy PUBCOMP vymaže správu, ktorú si dočasne uložil.



Obr. 8 QoS 2 – sekvenčný diagram

- **DUP flag -** Označuje, že správa je duplicitná a bola odoslaná z dôvodu, že cielený príjemca (klient alebo broker) nepotvrdil prijatie pôvodnej správy. Toto je relevantné len pre QoS väčšie ako 0.

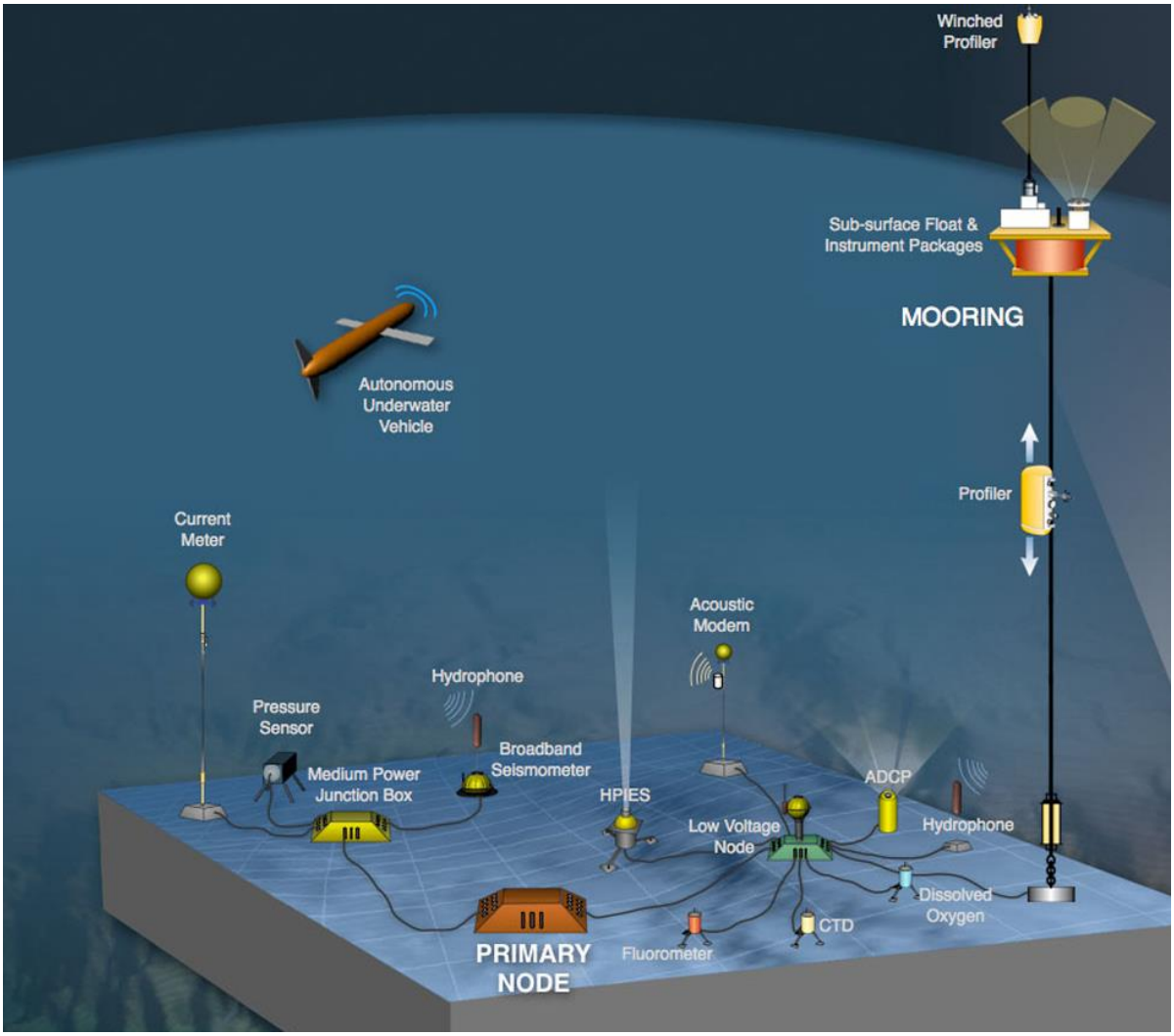
- **Variable-header**
  - **Topic Name** – jednoduchý reťazec, ktorý je štruktúrovaný hierarchicky pomocou lomiek. Napríklad: **„myhome/livingroom/temperature"** alebo **„Germany/Munich/Octoberfest/people"**
  - **Packet Identifier -** jednoznačne identifikuje správu, ktorá preteká medzi klientom a broker-om. Identifikátor paketu je relevantný len pre úrovne QoS väčšie ako nula. Klient alebo broker je zodpovedný za nastavenie tohto interného identifikátora MQTT správy.

- **Payload**
    - **Payload -** skutočný obsah správy. Správa môže obsahovať obrázky, text v akomkoľvek kódovaní alebo zašifrované dáta. Prakticky hocijaké dáta v binárnej forme.

1.2 AQMP

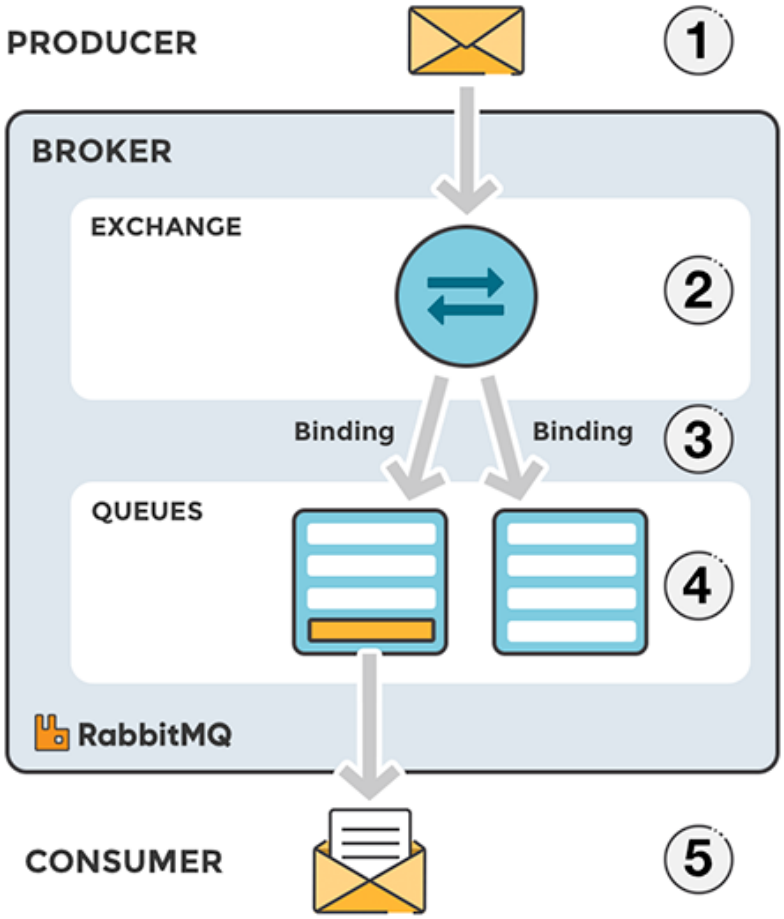
AMQP (Advanced Message Queuing Protocol) je binárny prenosový protokol, ktorý bol vytvorený pre podnikové aplikácie a komunikáciu medzi servermi, ale našiel si uplatnenie aj v IoT. AMQP 1.0 je jedinou štandardizovanou verziou od OASIS (od roku 2012/10) a ISO / IEC (od roku 2014/05) [4]. Protokol AQMP sa využíva napríklad v [5]:

- **NASA** - Nebula Cloud Computing – poskytuje výpočtovú silu pre výskumníkov a vedcov NASA prostredníctvom VPN pripojenia.
- **Google** využíva AMQP na komplexné spracovanie udalostí.
- Využíva sa v jednej z najväčších svetových biometrických databáz v indickom projekte **Aadhar** - domov 1,2 miliárd identít.
- **Ocean Observatories Initiative** – architektúra, ktorá zhromažďuje 8 terabajtov údajov za deň a slúži na monitorovanie oceánov (Obr. 9).



Obr. 9 Ocean Observatories Initiative – prístroje

AMQP protokol funguje principiálne rovnako ako MQTT, t. j. máme producentov/ spotrebiteľov, ktorí komunikujú cez broker. Medzi najznámejšie implementácie AMQP broker-a patrí RabbitMQ alebo Kafka. Napríklad pri implementácií RabbitMQ broker-a (Obr. 10) sa správy najskôr dostanú do komponentu „Exchange“, kde sa na základe kľúču určí, do ktorej „QUEUE“ sa majú správy zaslať.



Obr. 10 RabbitMQ broker

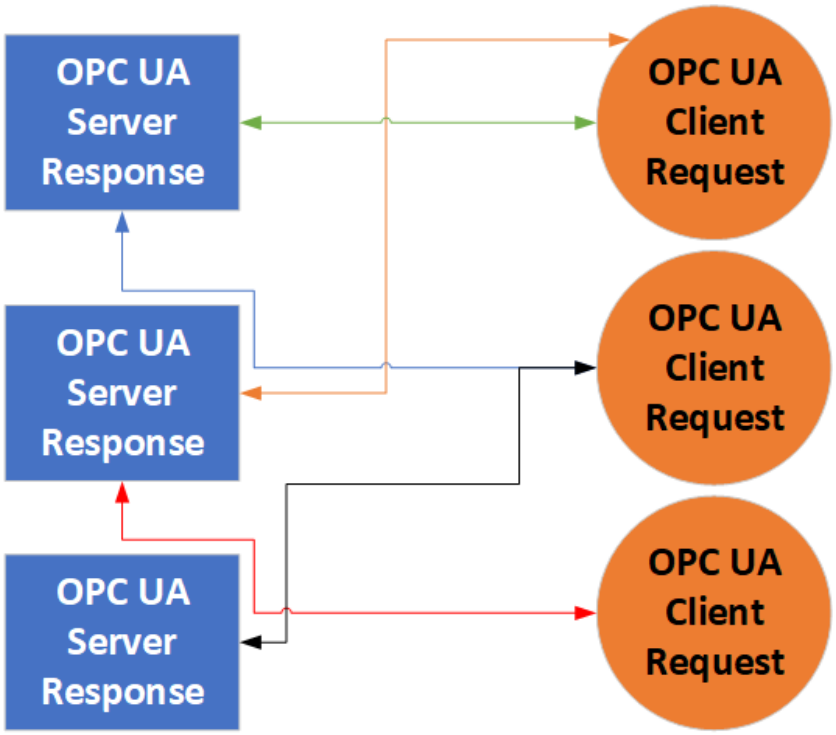
Nasledujúca tabuľka ponúka zhrnutie protokolov MQTT a AMQP.

	MQTT	**AMQP**
Využitie	Integrácia priemyselných zariadení medzi sebou (M2M) a s backend systémami	Komunikácia medzi väčšími infraštruktúrami ale aj medzi zariadeniami (M2M). Vhodný pre prenos väčších dátových celkov
Bandwidth/ prenosová rýchlosť	Vhodný pre zariadenia, ktoré disponujú nižšou prenosovou rýchlosťou – správa je jednoduchšia/menšia	Menej vhodný pre zariadenia, ktoré disponujú nižšou prenosovou rýchlosťou – správa je väčšia/komplexnejšia
Obojsmerná komunikácia	Áno	Áno
Nadviazanie komunikácie smerom klient ⟶ server/broker	Áno	Áno
Nadviazanie komunikácie smerom server/broker ⟶ klient	Nie	Áno
Komunikácia	TCP (port 443)	TCP (port 443)
Wildcard	Áno	Áno
Zabezpečenie	SSL/TLS Meno/Heslo Kryptovaný „Payload“	SSL/TLS Meno/Heslo Kryptovaný „Message body“
Uchovanie správ prípade výpadku	Áno	Áno
Keep-Alive message	Áno	Áno

Tab. 2 Zhrnutie MQTT a AMQP

2 OPC UA – PubSub

OPC UA je komunikačný protokol založený na architektúre klient - server (Obr. 11). Nevýhoda architektúry klient-server spočíva v tom, že pri získanie informácie musí byť vytvorený request na server, ktorý následne odpovie, čiže odošle response klientovi. Tento systém nie je vhodný pre priemyselnú komunikáciu, kde je vyžadovaná komunikácia v reálnom čase.



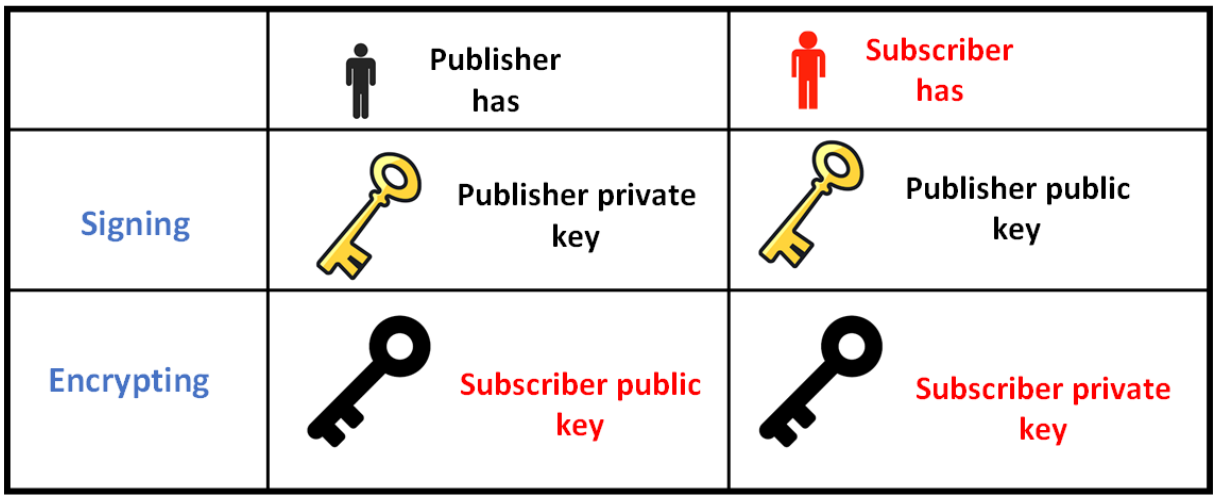
Obr. 11 OPC UA architektúra klient - server

Avšak príchod rozšírenia PubSub pre OPC UA spolu s nástupom rozšírenia TNS pre ethernet vytvára riešenie pre priemysel, ktoré ponúka časovo deterministickú komunikáciu. Architektúra PubSub pre OPC UA ponúka komunikáciu many-to-many a je možné ju kombinovať aj s existujúcou architektúrou klient – server. Výhoda PubSub komunikácie spočíva aj v tom, že nevyžaduje žiadne závislosti na rolách. To znamená, že OPC Klient môže byť publisher a OPC UA Server môže byť subscriber. V skutočnosti, tu nie je ani potreba pre publisher-a alebo subscriber-a byť na OPC Servery alebo Klientovi, aby sa mohli zúčastniť PubSub komunikácie. Pre komunikáciu s broker-om sa využíva protokoly MQTT a AMQP [6].

2.1 OPC UA PubSub - zabezpečenie

OPC UA PubSub architektúra ponúka 3 úrovne zabezpečenia správ [6]:

- Bez zabezpečenia
- Podpísanie, ale žiadne šifrovanie
- Podpísanie a šifrovanie



Obr. 12 Signing a Encrypting

Podpísanie – Subscriber si je istý, že číta správu od korektného/správneho publisher-a, nakoľko má jeho verejný kľuč, ktorý použije na získanie správy, ktorá bola podpísaná privátnym kľúčom publisher-a. Správu môže prečítať každý, kto má verejný kľuč (Obr. 12).

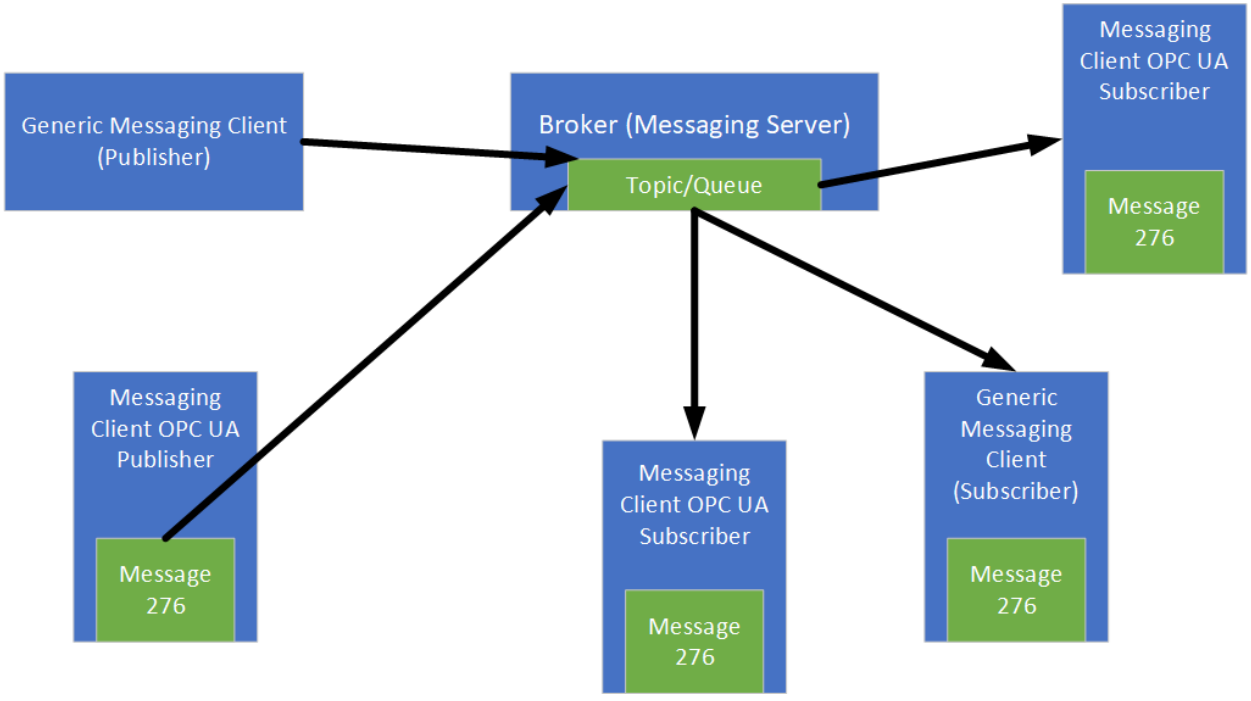
Šifrovanie – Zašifrovanú správu od publisher-a, dokáže prečítať len subscriber, ktorý má privátny kľuč.

Pri kombinácií podpísania a šifrovania dokážeme zabezpečiť, že správu od publisher-a si prečíta subscriber, ktorému je správa určená a zároveň si subscriber môže byť istý, že správa pochádza od publisher-a (Obr. 12).

2.2 Modely komunikácie pri OPC UA PubSub

OPC UA PubSub ponúka dva modely komunikácie [6]:

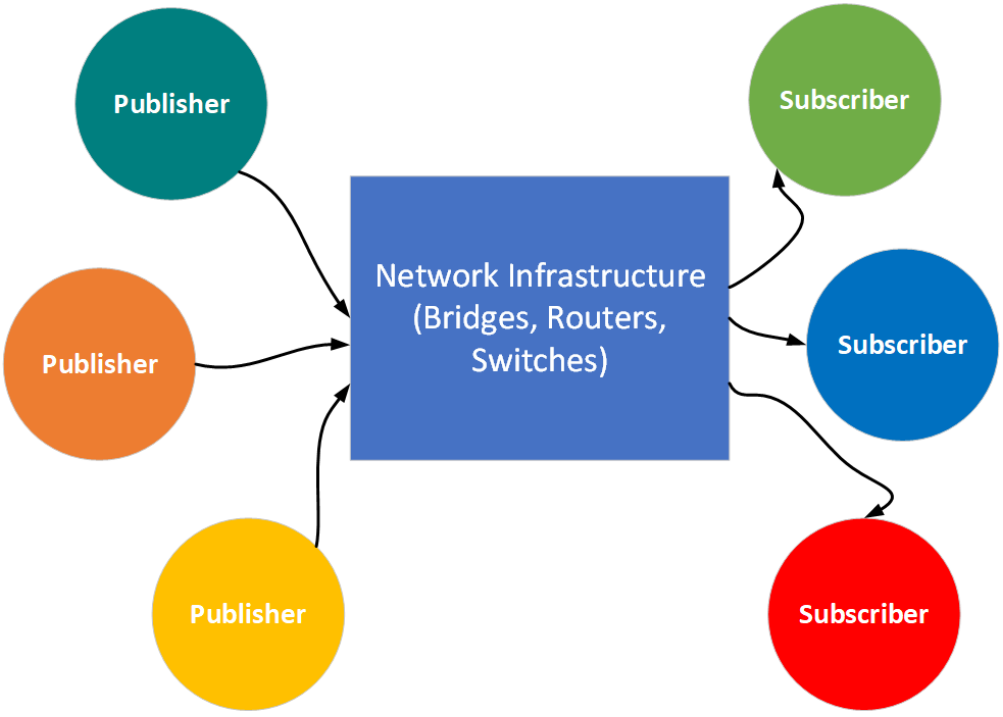
- **S broker-om** - Táto možnosť prekladá, že broker je stredom celej komunikácie (Obr. 13). Žiadna aplikácia nekomunikuje s inou aplikáciou priamo. Broker preposiela správy na základe rôznych biznis kritérií ("queue name", "routing key", "topic" a mnoho iných) a nie na základe IP adries alebo názvov hostiteľov. Výhody tohto modelu sú:
    - Publisher a subscriber nemusia o sebe vedieť a môžu byť umiestnené hocikde.
    - Publisher a subscriber nemusia byť súčasne online. Publisher môže správy poslať do broker-a a ten ich prepošle subscriber-ovi, keď bude znovu online.
    - Publisher a subscriber môžu využívať rôzne protokoly pre komunikáciu s broker-om.
    - Riešenie dokáže obslúžiť veľké množstvo subscriber-ov, nakoľko je ľahko škálovateľné. S narastajúcim počtom subscrib-ov, stačí jednoducho navýšiť počet broker-ov.



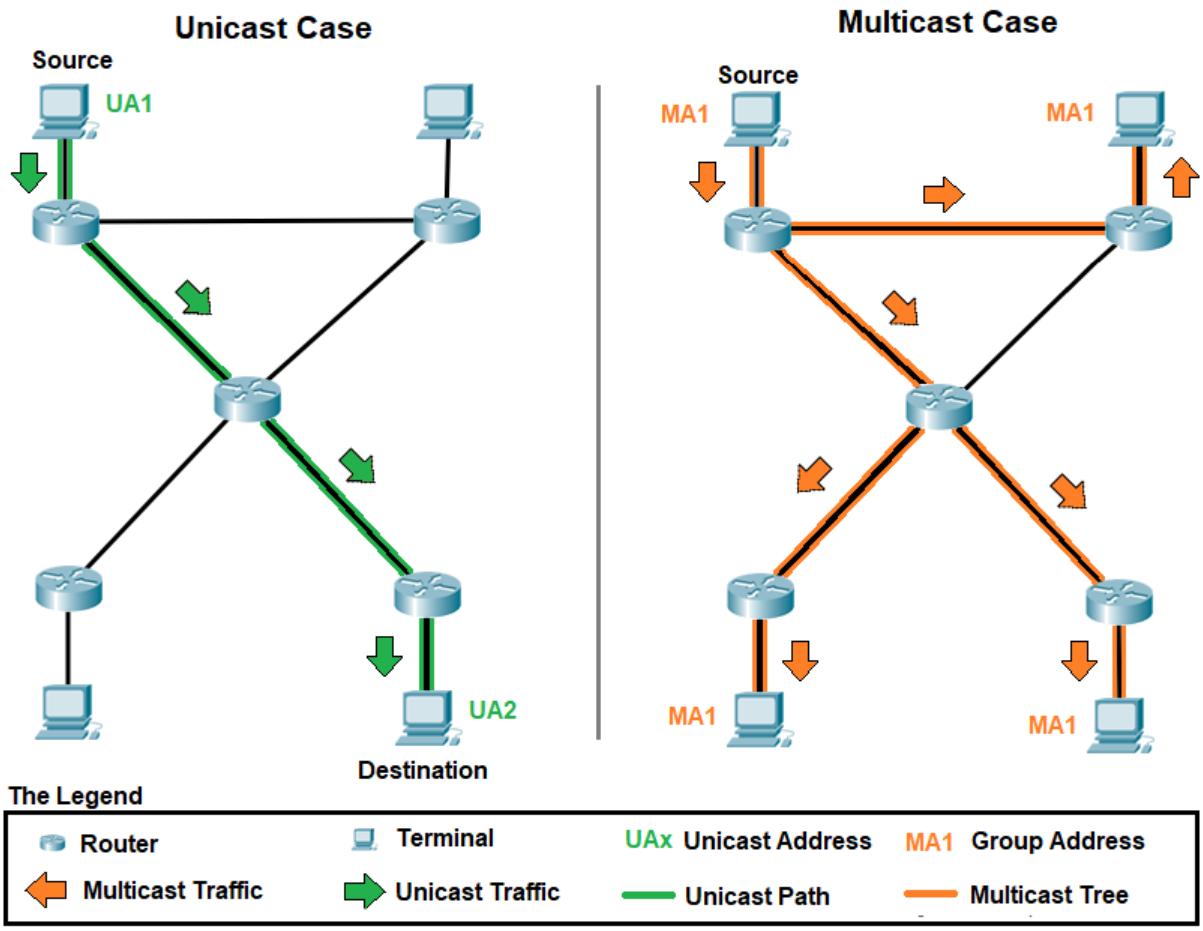
Obr. 13 OPC UA PubSub –komunikácia s broker-om

- **Bez broker-a –** Vďaka tejto možnosti sa OPC UA PubSub spolieha na sieťovú infraštruktúru, ktorá sa využije na dourčenie správy jednému alebo viacerým príjemcom. Zvyčajne sa na tieto účely používajú router-e, switch-e alebo bridge. Na obrázku 14 je príklad využitia komunikácie bez broker-a, kde sa môže využiť UDP unicast alebo multicast na doručenie správ (Obr. 15). Výhody daného riešenia:
    - Vyžaduje len štandardné sieťové vybavenie a žiadne ďalšie softvérové komponenty, ako napríklad broker.
    - Protokol UDP podporuje viacerých subscriber-ov, ktorí používajú multicast adresovanie.
    - Doručenie správ môže byť rýchlejšie, nakoľko správy neprechádzajú cez pridané softvérové komponenty. Režijné náklady môžu byť taktiež nižšie, pretože odpadá nutnosť starať sa o pridané softvérové - hardvérové komponenty.

  Nevýhodu riešenia s UDP spočíva v tom, že daný protokol je stateless, t.j. nezaručenie doručenie všetkých správ. Riešením problému môže byť zmena protokolu, poslanie správy viackrát alebo prípadne správy baliť do väčších dátových setov a tie odosielať (Pri každom odoslaní by sa z balíka najstaršia správa odobrala a pridala by sa nová.).



Obr. 14 OPC UA PubSub – komunikácia bez broker-u



Obr. 15 Unicast vs Multicast

Záver

Popularita IoT a IIoT stále narastá, pričom medzi najpálčivejšie problémy patrí zabezpečenie komunikácie medzi rôznymi zariadeniami. Vďaka presunu informácií sa zariadenia dokážu navzájom ovplyvňovať a reagovať na rôzne požiadavky z okolia.

Protokol MQTT predstavuje vhodné riešenie pre komunikáciu so zariadeniami, ktoré majú obmedzené napájanie a prenosová rýchlosť je nízka. MQTT využíva na smerovanie správ broker, pričom takýmto protokolom je AMQP. AMQP má komplexnejšiu štruktúru správ ako MQTT, ktorá umožňuje nastaviť rôzne dodatočné parametre a preto je vhodný aj na komunikáciu medzi väčšími celkami. Nakoľko veľkosť správy je väčšia, vyžaduje sa kvalitnejšie pripojenie na internet.

Vďaka rozšíreniu PubSub a STN pre ethernet sa môže stať OPC UA štandardom pre komunikáciu v priemysle, nakoľko bude schopné pokryť komunikáciu v celej podnikovej automatizačnej pyramíde, t. j. od senzorov, aktuátorov až po ERP systém.

Smer pre budúci výskum a štúdium vidím v komplexnejšom naštudovaní protokolu AMQP, ktorý je možné využiť pre komunikáciu založenú na PubSub. Ďalší smer pre pokračovanie v práci vidím aj v možnosti reálne implementovať architektúru OPC UA PubSub a otestovať nezávislosť komponentov.

Zdroje

[1] Ashton, K. 2009. *(That 'Internet of Things' Thing)*. [on-line]. [cit: 2018-26-10]. Dostupné na: <https://www.rfidjournal.com/articles/view?4986>

[2] MQTT 101 – How to Get Started with the lightweight IoT Protocol. [on-line]. [cit: 2018-27-10]. Dostupné na: <https://www.hivemq.com/blog/how-to-get-started-with-mqtt>

[3] MQTT Essentials Part 4: MQTT Publish, Subscribe & Unsubscribe. [on-line]. [cit: 2018-27-10]. Dostupné na: <https://www.hivemq.com/blog/mqtt-essentials-part-4-mqtt-publish-subscribe-unsubscribe>

[4] AMQP Advanced Message Queuing Protocol. [on-line]. [cit: 2018-29-10]. Dostupné na: <http://www.amqp.org>

[5] Products and Success Stories. [on-line]. [cit: 2018-28-10]. Dostupné na: <http://www.amqp.org/about/examples>

[6] OPC Unified Architecture Specification Part 14: PubSub. [on-line]. [cit: 2018-31-10]. Dostupné na: <https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-14-pubsub>