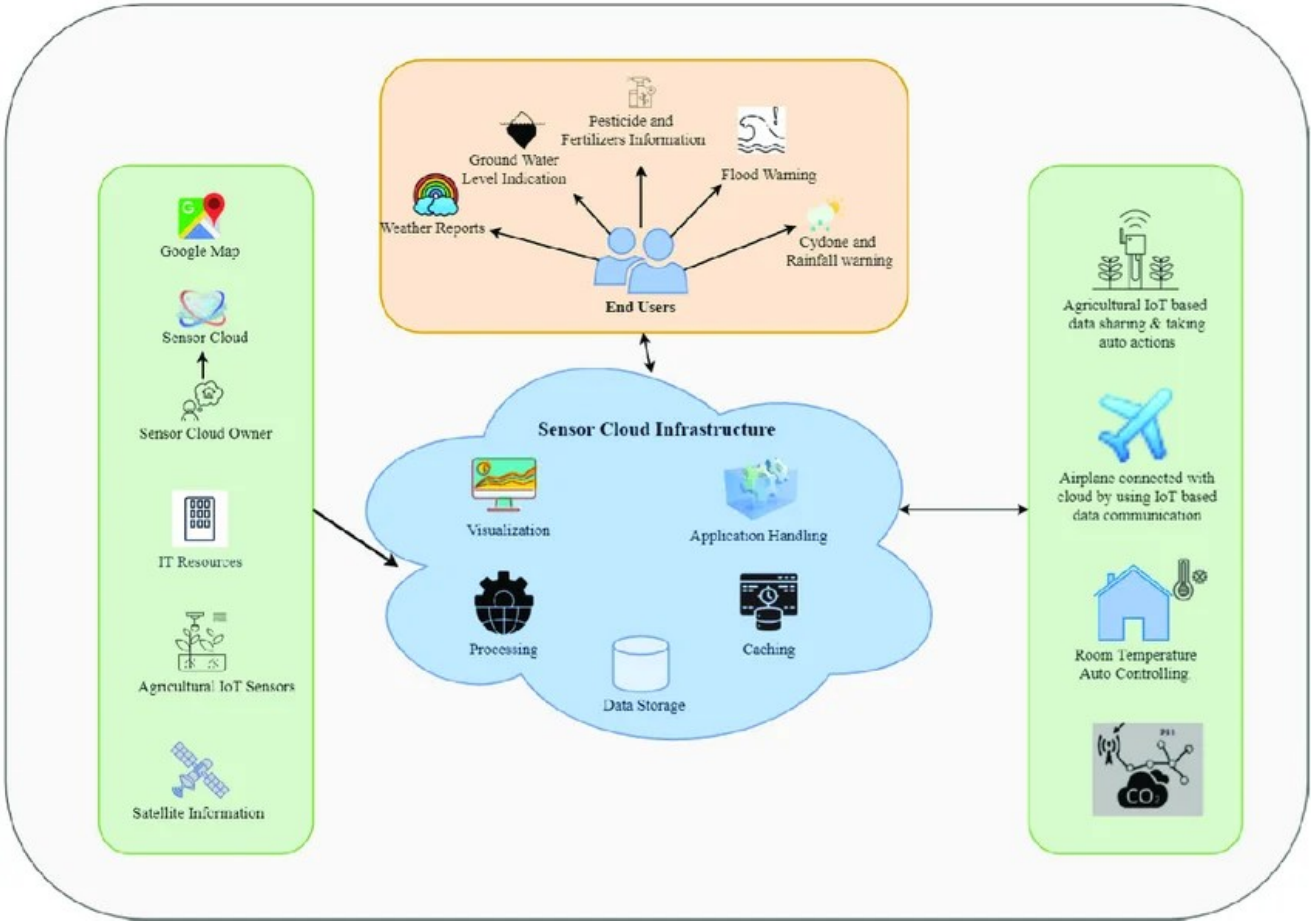
IoT Komunikacioni protokoli

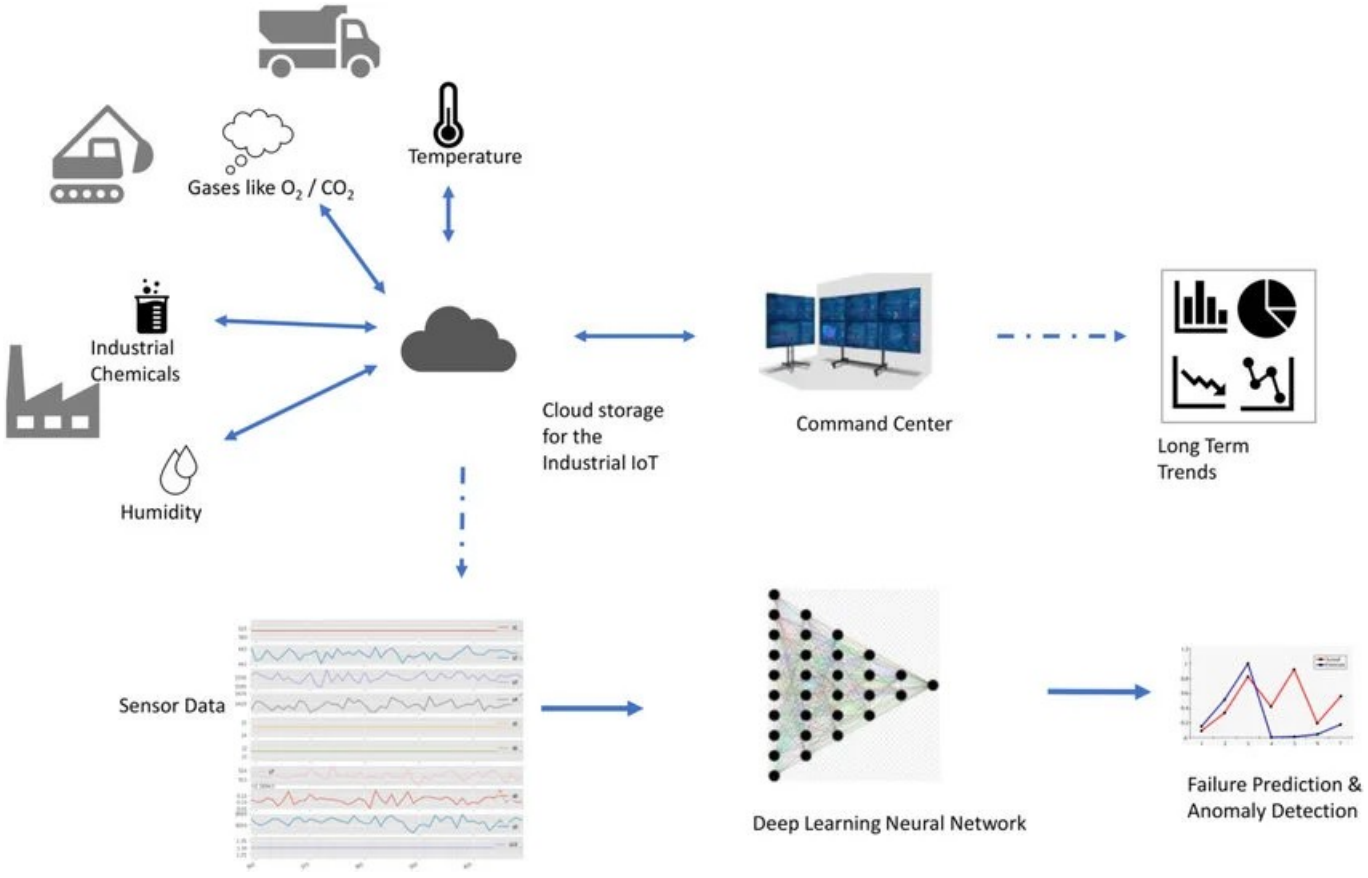
IoT Komunikacioni protokoli

- Internet of Things (IoT) = mreža fizičkih uređaja
- Uređaji imaju:
- senzore
- procesiranje
- komunikaciju
- Cilj: prikupljanje i razmena podataka

IoT Komunikacioni protokoli



IoT Komunikacioni protokoli



Zašto su komunikacioni protokoli ključni?

- Uređaji moraju da komuniciraju pouzdano

  Ograničenja:
- mala potrošnja energije
- mala memorija
- nestabilna mreža

  Protokoli rešavaju:
- kako se šalju podaci
- koliko često
- koliko pouzdano

Zašto su komunikacioni protokoli ključni?

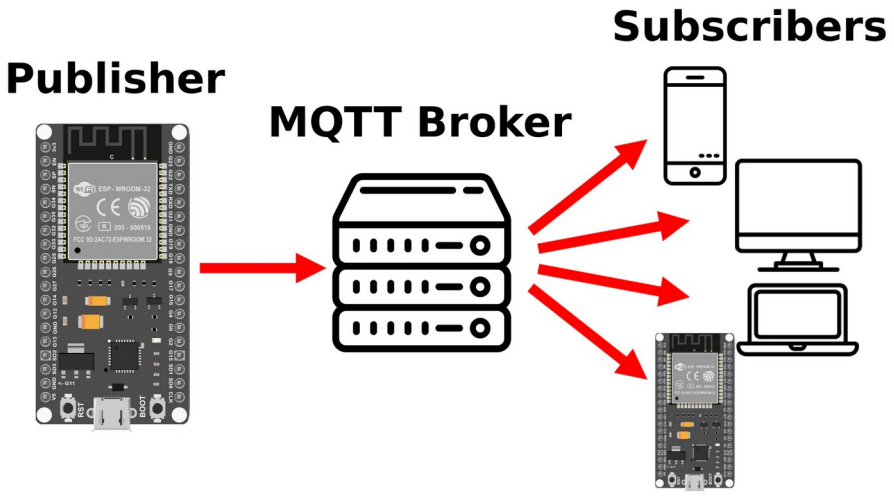
Za razliku od klasičnog interneta:

- IoT uređaji imaju ograničene resurse
- mala potrošnja energije
- nestabilna ili spora mreža
- potreba za pouzdanom komunikacijom

Zato ne možemo koristiti uvek standardne protokole kao HTTP.

Moramo imati:

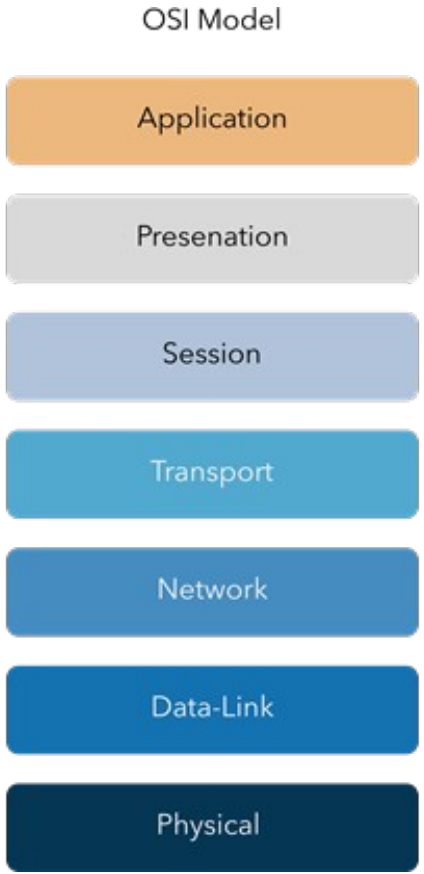
- lagane protokole (MQTT, CoAP)
- koji troše malo energije
- i rade i kada mreža nije stabilna.



Komunikacioni slojevi (OSI model – pojednostavljeno)

- Application layer (MQTT, HTTP, CoAP)
- Transport layer (TCP, UDP)
- Network layer (IP, LoRaWAN, NB-IoT)
- Physical layer (WiFi, BLE, Ethernet)

Komunikacioni slojevi (OSI model – pojednostavljeno)



Komunikacioni slojevi

- Komunikacija u mrežama se organizuje kroz slojeve.
- Mi nećemo koristiti svih 7 OSI slojeva detaljno — za IoT nam je bitno da razumemo 4 glavna nivoa.
- Na vrhu je application layer — tu su protokoli kao MQTT, HTTP i CoAP.
- Ispod toga je transport layer — TCP ili UDP, koji određuju kako se podaci prenose.
- Zatim ide network layer — IP ili specifične IoT mreže kao LoRaWAN.
- Na kraju je physical layer — WiFi, Bluetooth, Ethernet.

Modeli komunikacije: Client–Server vs Publish–Subscribe

Client–Server:

- direktna komunikacija
- klijent šalje zahtev
- server odgovara

Publish–Subscribe:

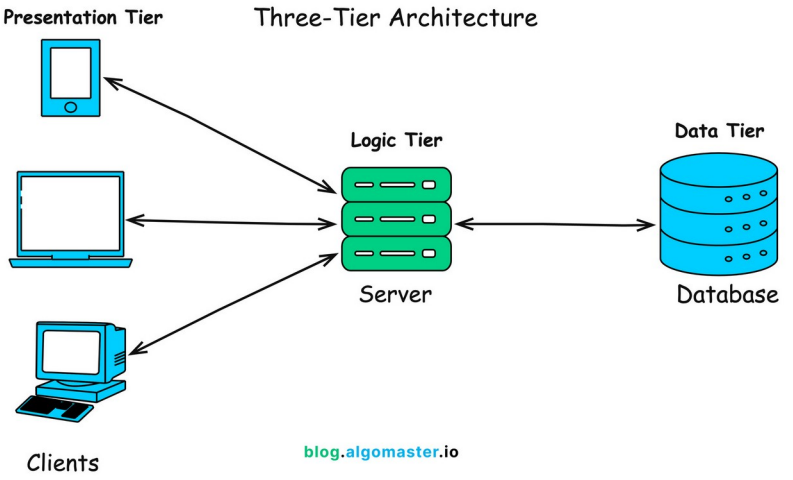
- komunikacija preko brokera
- uređaji ne komuniciraju direktno
- poruke se šalju na "topic"

Client–Server

Client–Server — klasičan internet model.

Na primer:

- browser → šalje zahtev
- server → vraća odgovor
- HTTP radi na ovom principu.



Publish–Subscribe

Drugi model je Publish–Subscribe, koji se često koristi u IoT-u.

- Ovde uređaji ne komuniciraju direktno.
- Umesto toga:
- svi komuniciraju sa jednim centralnim entitetom — brokerom
- uređaj može da objavi poruku (publish)
- drugi uređaji se pretplate (subscribe) na određenu temu (topic)

i automatski dobijaju poruke“

Publish–Subscribe

Na primer:

senzor objavljuje temperaturu na topic: home/temperature

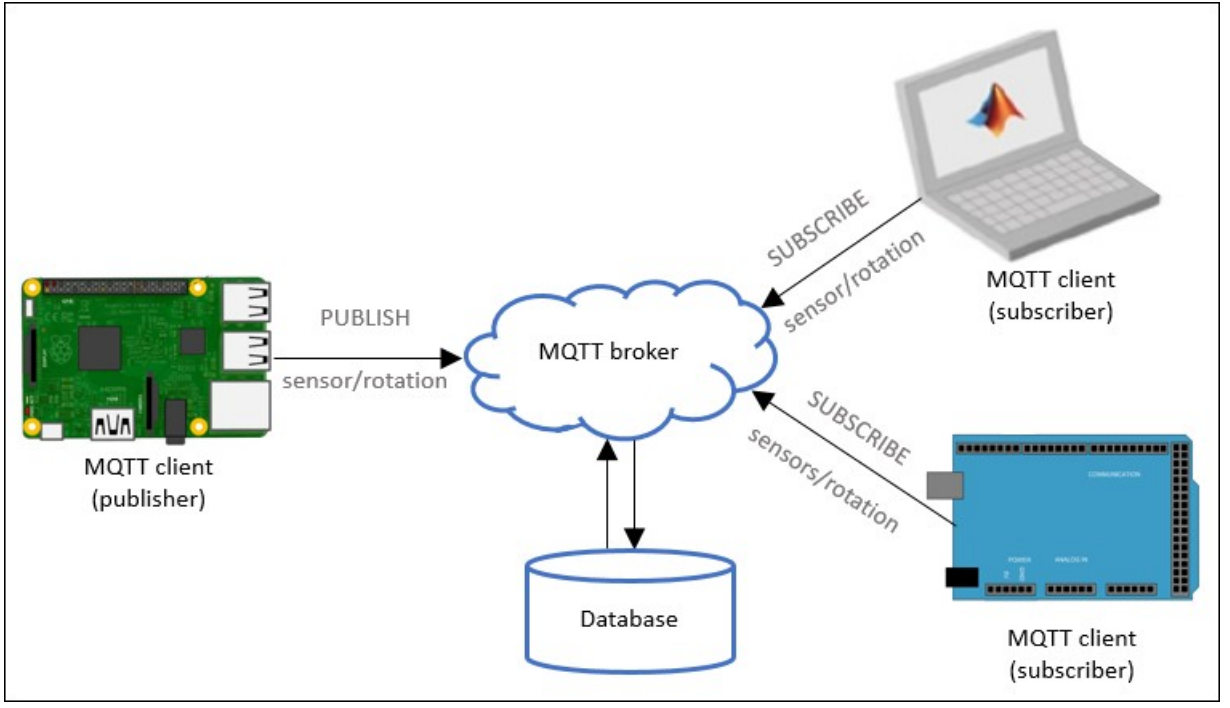
telefon je subscribe na taj topic

čim se pojavi nova vrednost → telefon je dobija

U Publish–Subscribe modelu:

uređaji NE moraju da znaju jedni za druge

komunikacija je fleksibilnija



Client–Server vs Publish–Subscribe

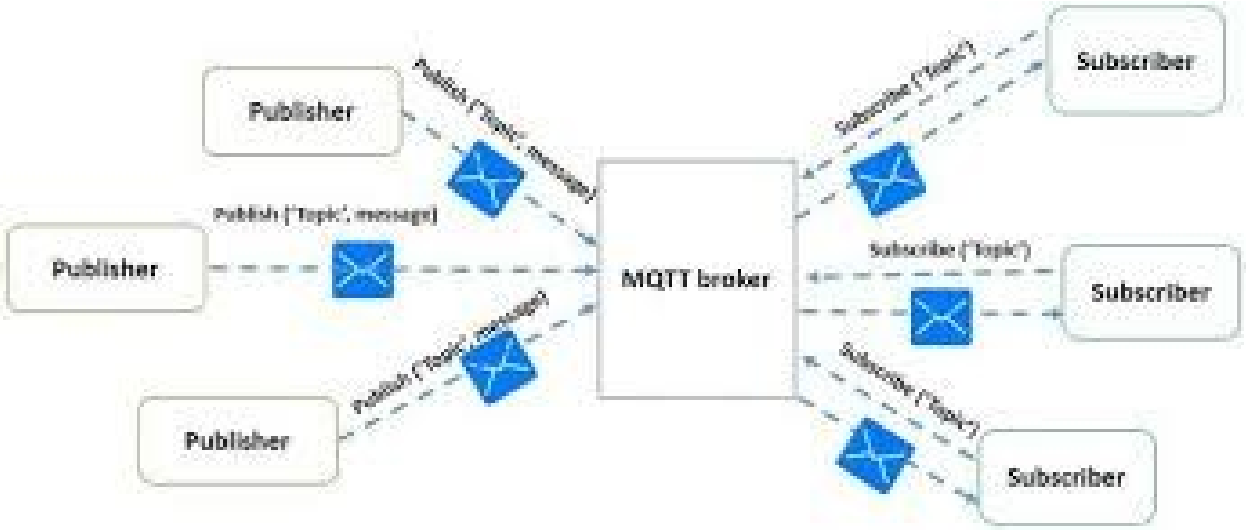
- HTTP = Client–Server
- MQTT = Publish–Subscribe
- broker = centralna tačka

MQTT – osnovni koncept

- lagan (lightweight) protokol
- koristi Publish–Subscribe model
- radi preko TCP
- koristi broker
- idealan za IoT

MQTT – osnovni koncept

- lagan (lightweight) protokol
- koristi Publish–Subscribe model
- radi preko TCP
- koristi broker
- idealan za IoT

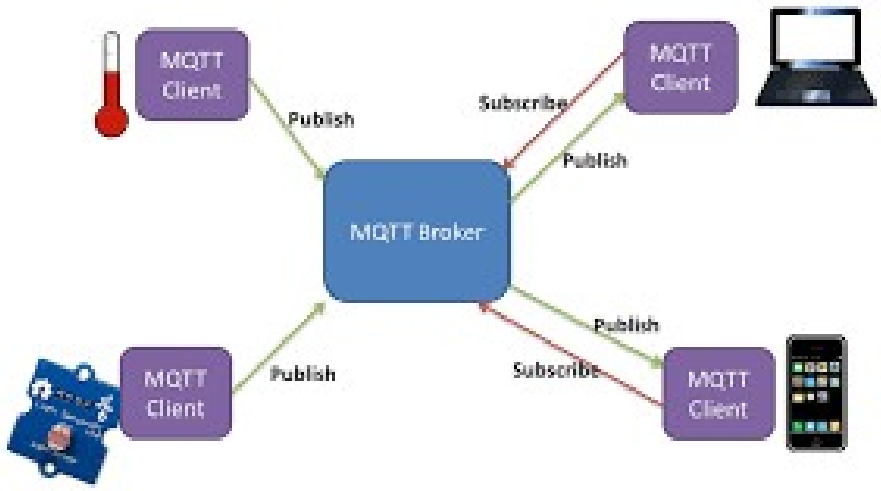


MQTT – osnovni koncept

- MQTT je jedan od najvažnijih protokola u IoT sistemima.
- Razvijen je kao lagan protokol, što znači da koristi vrlo malo resursa — što je idealno za mikrokontrolere kao što su STM32 ili ESP32.
- Radi po Publish–Subscribe modelu, što znači da uređaji ne komuniciraju direktno, već preko brokera.
- MQTT koristi TCP kao transportni protokol, što znači da obezbeđuje pouzdanu komunikaciju.
- Zbog svoje jednostavnosti i efikasnosti, MQTT je danas standard u mnogim IoT sistemima.

MQTT – kako funkcioniše

- Topic (npr. home/temperature)
- Publisher šalje poruke
- Subscriber prima poruke
- QoS (Quality of Service):
- QoS 0 – bez garancije
- QoS 1 – najmanje jednom
- QoS 2 – tačno jednom



MQTT – kako funkcioniše

- MQTT se zasniva na konceptu topic-a.
- Topic je kao kanal komunikacije.
- Na primer: home/temperature.
- Uređaj koji meri temperaturu objavljuje (publish) podatke na taj topic.
- Svi uređaji koji su pretplaćeni (subscribe) na taj topic dobijaju poruku.
- Važan deo MQTT-a je QoS — Quality of Service.
- To određuje koliko je pouzdana isporuka:
- QoS 0 – poruka može da se izgubi
- QoS 1 – stići će bar jednom
- QoS 2 – stići će tačno jednom (najsigurnije, ali najsporije)

MQTT – kako funkcioniše

- Na primer:
- STM32 meri temperaturu
- šalje na home/temperature
- mobilna aplikacija je subscribe
- Svaki put kada se pošalje nova vrednost → aplikacija je dobija

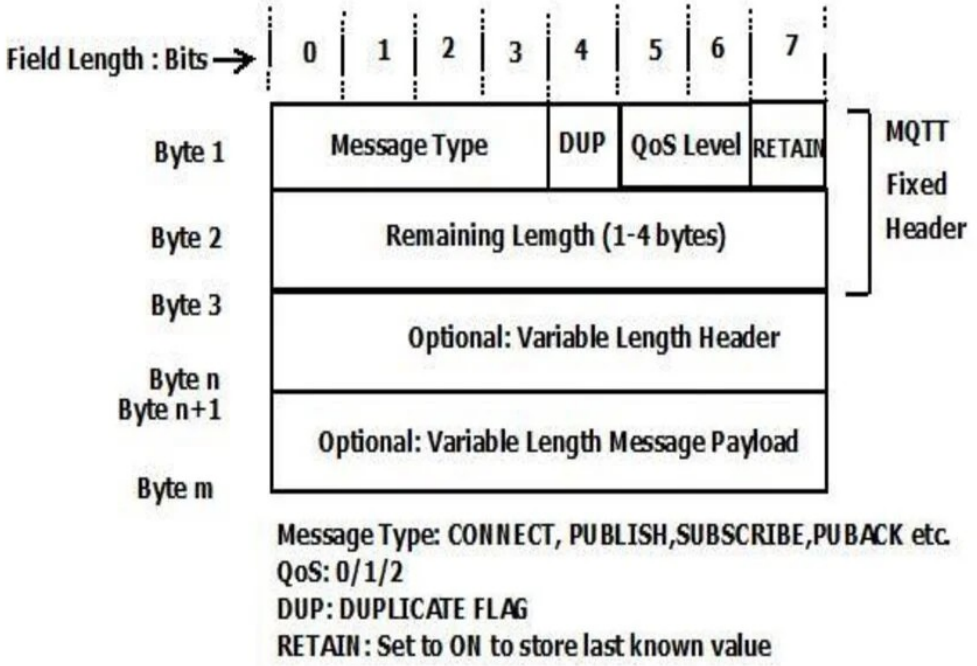
MQTT paket i komunikacija

Struktura MQTT paketa:

- Fixed header
- Variable header
- Payload

Osnovne poruke:

- CONNECT / CONNACK
- PUBLISH
- SUBSCRIBE / SUBACK
- DISCONNECT

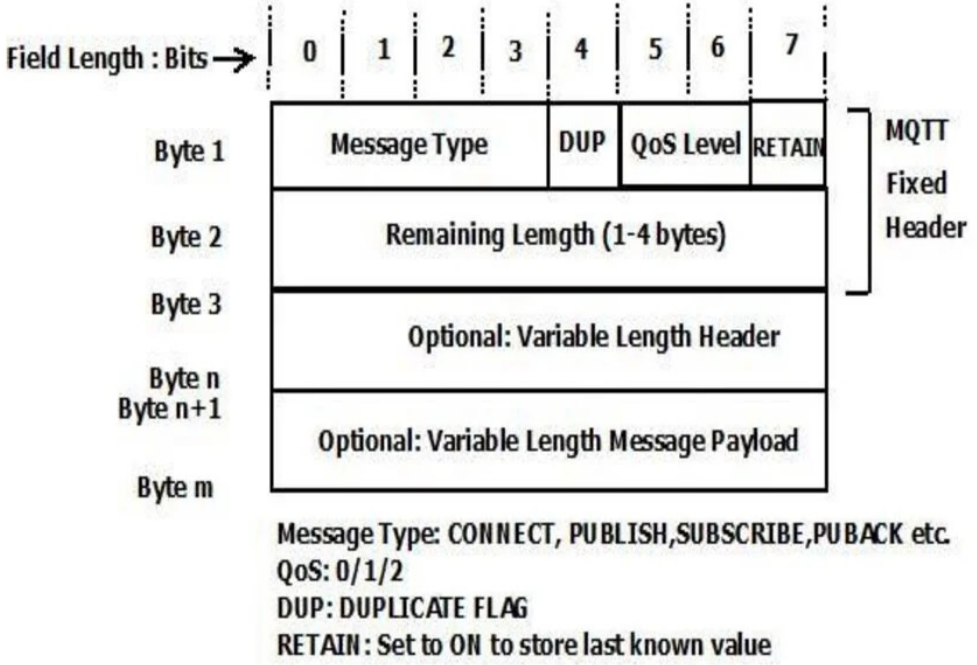


MQTT paket i komunikacija

MQTT komunikacija se zasniva na vrlo jednostavnom formatu poruke.

- Svaka MQTT poruka ima tri dela:

- Fixed header
- tip poruke (npr. PUBLISH, CONNECT)
- QoS, retain flag
- Variable header
- zavisi od tipa poruke
- npr. topic ime
- Payload
- stvarni podaci (npr. temperatura 24.7)



Tok komunikacije (korak po korak)

- uređaj šalje CONNECT
- broker odgovara CONNACK
- uređaj šalje PUBLISH poruke
- klijent šalje SUBSCRIBE
- broker šalje SUBACK
- komunikacija traje dok se ne pošalje DISCONNECT

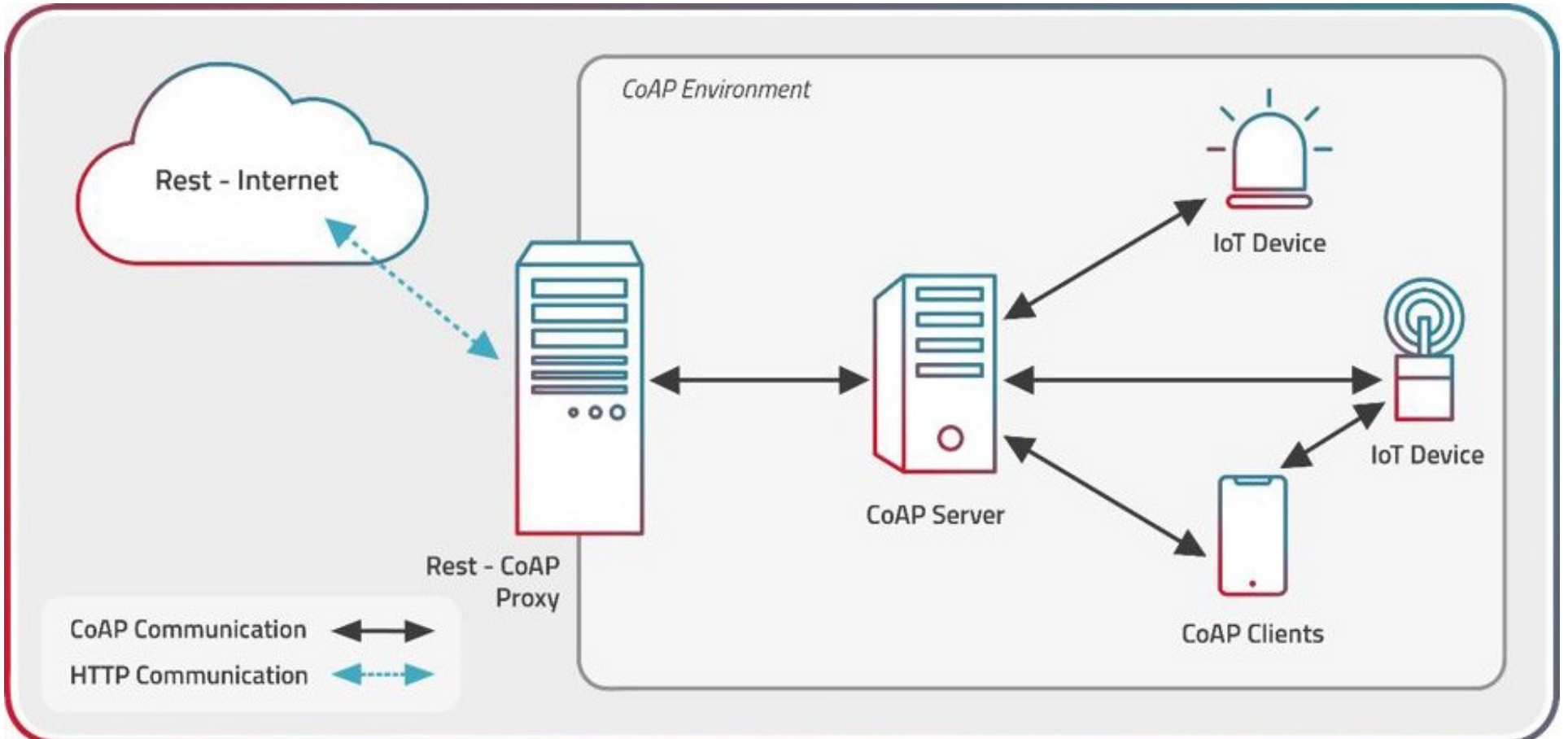
Zašto je MQTT lagan

- header je mali (2 bajta minimalno)
- nema kompleksnog zaglavlja kao HTTP
- payload može biti vrlo mali

CoAP protokol (Constrained Application Protocol)

- lagan protokol za IoT
- radi preko UDP
- koristi REST model (GET, POST, PUT, DELETE)
- dizajniran za uređaje sa ograničenim resursima
- koristi request/response model

CoAP protokol (Constrained Application Protocol)



CoAP protokol

- CoAP je takođe lagan IoT protokol, ali za razliku od MQTT-a koristi drugačiji model komunikacije.
- On koristi request/response model, sličan HTTP-u

  Na primer:
- klijent šalje zahtev: GET temperatura
- server vraća odgovor: 24.7
- Dakle, komunikacija je direktna — nema brokera

CoAP

MQTT:

- publish/subscribe
- broker

CoAP:

- client/server
- direktna komunikacija

CoAP

MQTT:

- publish/subscribe
- broker

CoAP:

- client/server
- direktna komunikacija

CoAP-UDP

CoAP koristi UDP, što znači:

- manji overhead
- brža komunikacija
- ali nema garantovane isporuke kao TCP

CoAP-UDP

CoAP koristi UDP, što znači:

- manji overhead

Overhead su svi dodatni podaci koje šaljemo zajedno sa pravim podatkom

Što je overhead manji:

- manje se troši mreža
- manje se troši energija
- komunikacija je brža
- Zato IoT protokoli teže da imaju što manji overhead.

CoAP-REST koncept

CoAP koristi iste operacije kao HTTP:

- GET
- POST
- PUT
- DELETE
- Zato je vrlo pogodan za integraciju sa web sistemima

CoAP

CoAP je kao ‘lakši HTTP za IoT

- CoAP koristi UDP
- request/response model
- nema brokera
- sličan HTTP-u

MQTT vs CoAP

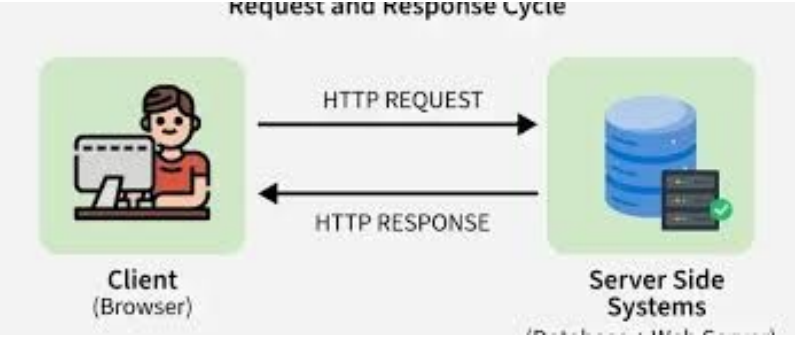
Karakteristika	MQTT	CoAP
Model komunikacije	Publish–Subscribe	Client–Server
Transport	TCP	UDP
Broker	DA	NE
Overhead	mali	veoma mali
Pouzdanost	visoka (TCP + QoS)	niža (UDP)
Latencija	srednja	niska
Pogodno za	real-time streaming	brze zahteve

MQTT vs CoAP

- MQTT koristi publish–subscribe model i radi preko TCP-a, što znači da je komunikacija pouzdana, ali ima nešto veći overhead.
- Sa druge strane, CoAP koristi client–server model i radi preko UDP-a, što ga čini bržim i sa manjim overhead-om, ali manje pouzdanim.
- MQTT je idealan kada želimo kontinuirano slanje podataka — na primer, senzorski podaci u realnom vremenu.
- CoAP je bolji kada želimo brze zahteve — na primer, da očitamo vrednost senzora na zahtev

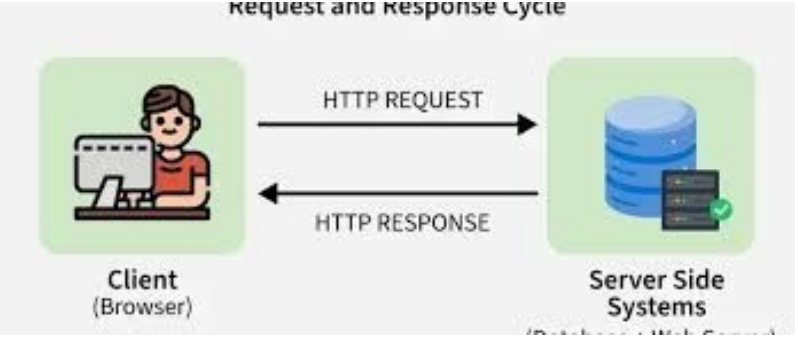
HTTP u IoT

- zasnovan na Client–Server modelu
- koristi TCP
- REST komunikacija (GET, POST…)
- veliki overhead
- široko podržan



HTTP u IoT

- HTTP je najpoznatiji protokol na internetu i koristi se u skoro svim web aplikacijama.
- Radi po client–server modelu — klijent šalje zahtev, server odgovara



Problem sa HTTP-om u IoT-u je

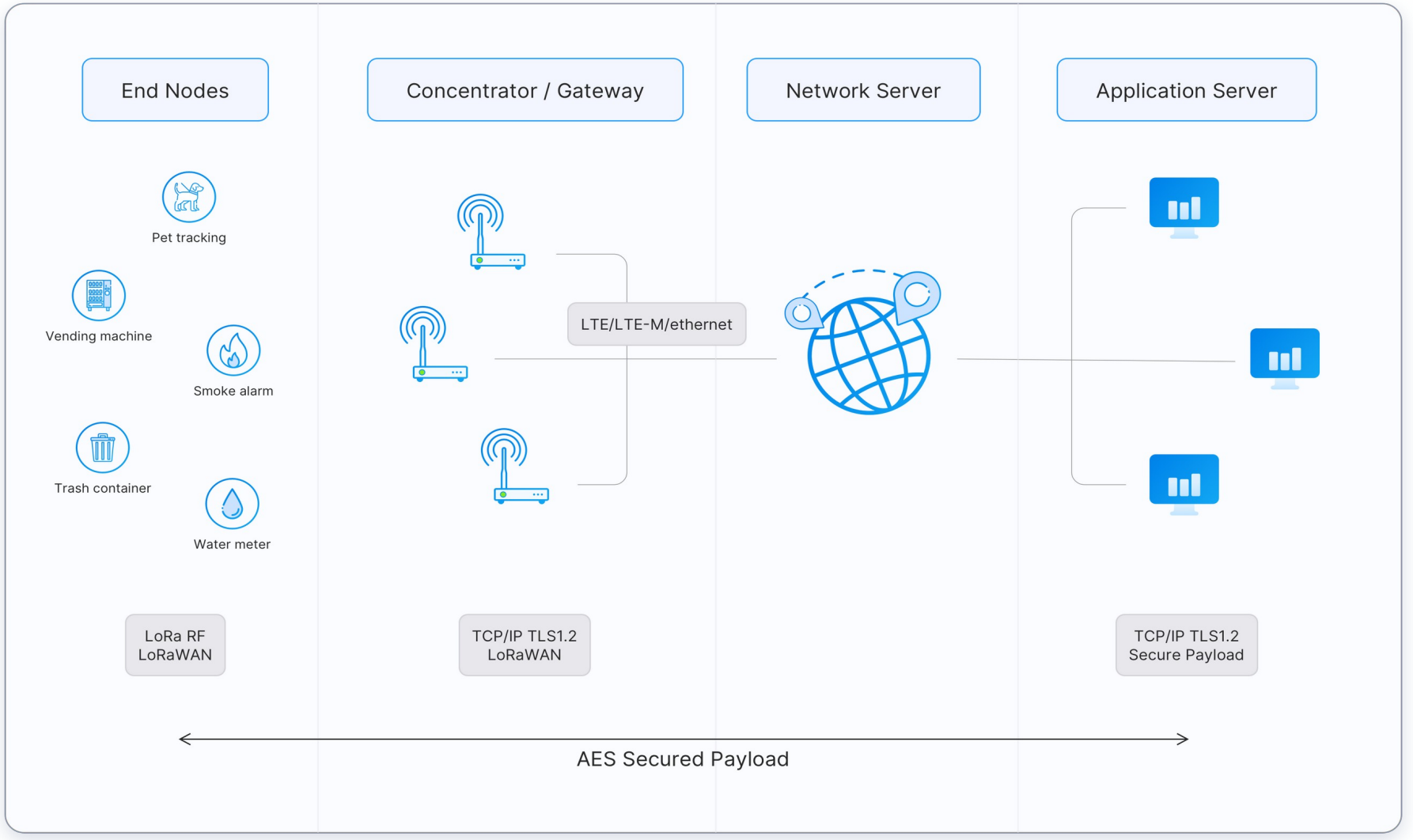
- veliki overhead
- tekstualni format
- zahteva više energije
- nije efikasan za male poruke

Problem sa HTTP-om u IoT-u je

- I pored toga, HTTP se često koristi jer:
- je jednostavan
- svi ga znaju
- lako se integriše sa web sistemima
- cloud servisi ga podržavaju
- HTTP je praktičan, ali nije optimizovan za IoT

LoRaWAN (IoT mreža)

- long-range komunikacija (km domet)
- mala potrošnja energije
- mali protok podataka
- koristi gateway i network server
- pogodan za smart city i industriju


End Nodes
Concentrator / Gateway
Network Server
Application Server
Pet tracking
Vending machine
Smoke alarm
Trash container
Water meter
LTE/LTE-M/ethernet
LoRa RF
LoRaWAN
TCP/IP TLS1.2
LoRaWAN
TCP/IP TLS1.2
Secure Payload
AES Secured Payload

LoRaWAN (IoT mreža)

- LoRaWAN je mrežni protokol namenjen za IoT uređaje koji treba da rade na velikim udaljenostima uz vrlo malu potrošnju energije
- Uređaji ne komuniciraju direktno sa internetom.
- Oni šalju podatke ka gateway-u, koji ih prosleđuje network serveru, a zatim cloud sistemu.

LoRaWAN (IoT mreža)

Prednosti LoRaWAN-a su:

- domet od nekoliko kilometara
- veoma mala potrošnja energije
- mogućnost rada na baterije godinama

Mana:

- mali protok podataka

LoRaWAN (IoT mreža)

Ovakvi sistemi se koriste u smart city projektima.

Na primer:

- merenje kvaliteta vazduha
- senzori na autobusima
- monitoring okoline
- LoRaWAN = mala brzina + veliki domet + mala potrošnja

LoRaWAN (IoT mreža)

nije isto što i MQTT/CoAP

- ovo je mreža, ne aplikacioni protokol
- koristi gateway
- idealno za smart city

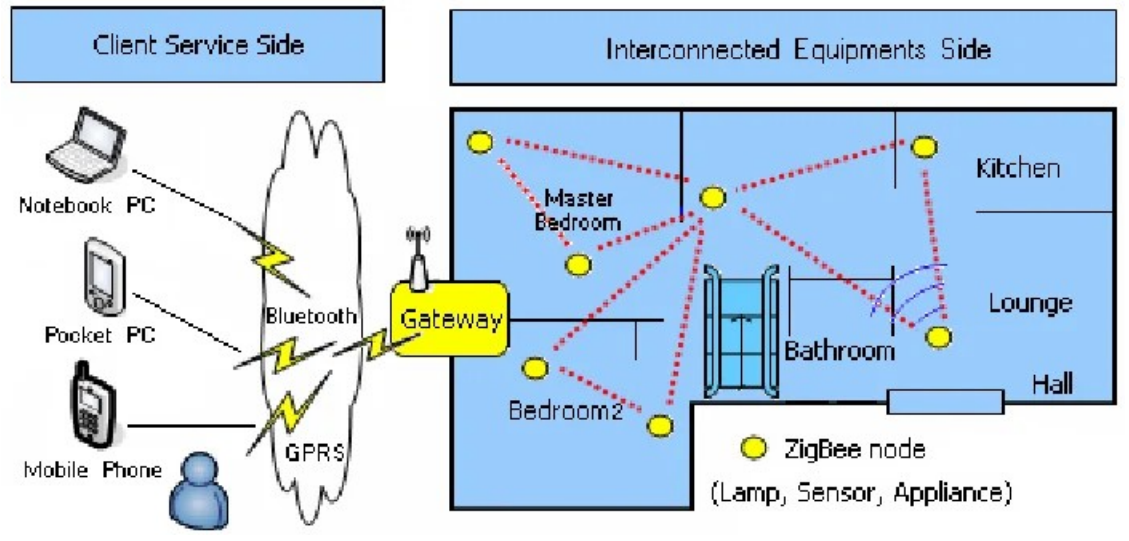
BLE i Zigbee – kratkodometne IoT mreže

BLE (Bluetooth Low Energy):

- mali domet (do ~100 m)
- vrlo mala potrošnja energije
- direktna komunikacija (telefon ↔ uređaj)

Zigbee:

- mesh mreža
- mali domet po čvoru
- niska potrošnja
- koristi se u pametnim kućama



BLE i Zigbee – kratkodometne IoT mreže

Zigbee je drugačiji — koristi mesh mrežu.

To znači da:

- uređaji mogu prosleđivati podatke jedni drugima
- mreža se može proširiti

Koristi se u:

- pametnim kućama
- sistemima automatizacije
- industriji

BLE i Zigbee – kratkodometne IoT mreže

BLE:

- direktna veza (telefon ↔ uređaj)

Zigbee:

- mreža uređaja (mesh)

BLE i Zigbee – kratkodometne IoT mreže

BLE → pametni sat šalje podatke telefonu

Zigbee → senzori u kući komuniciraju međusobno

BLE = point-to-point

Zigbee = mreža (mesh)

Koji protokol koristiti?

MQTT:

- real-time podaci
- publish/subscribe
- monitoring sistemi

CoAP:

- brzi zahtevi
- direktna komunikacija
- kontrola uređaja

Koji protokol koristiti?

HTTP:

- web integracija
- jednostavna implementacija

LoRaWAN:

- veliki domet
- mala potrošnja
- smart city

Koji protokol koristiti?

BLE / Zigbee:

- kratke distance
- pametne kuće / uređaji

Koji protokol koristiti?

Ne postoji jedan univerzalan protokol za sve IoT sisteme:

Izbor zavisi od:

- vrste uređaja
- udaljenosti
- količine podataka
- zahteva za pouzdanošću

Na primer:

- MQTT koristimo za kontinuirano slanje podataka
- CoAP za brze zahteve
- LoRaWAN za velike udaljenosti
- BLE za komunikaciju sa telefonom

Ključ je izabrati pravi alat za pravi problem.

Primer Python

- publisher.py = pošiljalac
- subscriber.py = primalac

- Oni ne komuniciraju direktno jedan sa drugim.
- Oba se povezuju na MQTT broker:

- To je test.mosquitto.org

- Broker je centralna tačka koja:
- prima poruke od publisher-a
- prosleđuje ih svim subscriber-ima koji slušaju isti topic

Primer Python

- Subscriber se poveže na broker
- Kada se pokrene subscriber.py, on uradi:
- konekciju ka brokeru
- prijavi se na topic
- Topic je:
- demo/iot/prof_peulic_temp

Primer Python

- publisher.py, se takođe poveže na isti broker.
- Posle toga u petlji:
- generiše slučajnu temperaturu
- šalje je na isti topic

Primer Python

- Broker vidi:
- stigla je poruka
- topic je demo/iot/prof_peulic_temp
- On proveri:
- ko je subscribe na taj topic?
- Pošto je subscriber pretplaćen, broker mu prosledi poruku.

Primer Python

Topic

- Topic je kao naziv kanala.

Na primer:

- home/temp
- lab/sensor1
- demo/iot/prof_peulic_temp

Publisher šalje poruku na topic.

- Subscriber prima samo ako sluša taj isti topic.

Primer Python

Payload

- Payload je stvarni podatak.
- To je npr. temperatura

Primer Python

Broker

- Broker je posrednik.
- Bez njega MQTT :
- publisher ne zna ko prima poruke
- subscriber ne zna ko ih šalje
- Oni znaju samo:
- broker
- topic
- To je suština publish/subscribe modela.

Zašto je ovo dobro za IoT?

Zato što uređaji ne moraju da znaju jedni za druge.

- Na primer:
- jedan senzor šalje temperaturu
- telefon prima
- dashboard prima
- drugi server prima

A publisher i dalje šalje samo jednu poruku brokeru.

- Broker radi distribuciju dalje.

client.connect("test.mosquitto.org", 1883, 60)

- Poveži se na MQTT broker
- na adresi test.mosquitto.org
- preko porta 1883
- i održavaj konekciju aktivnom na svakih 60 sekundi

client.connect("test.mosquitto.org", 1883, 60)

- Ovo je javni broker
- svi ga koriste
- nije bezbedan
- nije za produkciju

Praksa

- SVOJ SERVER (self-hosted)
- Instalacija Mosquitto ili npr:
- EMQX
- HiveMQ
- na:
- VPS (DigitalOcean, AWS EC2…)
- ili firmni server

Praksa

- CLOUD MQTT (najčešće danas)

- Koristi se gotova platforma:
-
- AWS IoT Core
- Azure IoT Hub
- HiveMQ Cloud