Integracija CoAP posrednika u IPv4 i IPv6 mrežama interneta stvari

Ninčević, Ivan

Sveučilište Jurja Dobrile u Puli
Tehnički fakultet u Puli

Ivan Ninčević

Integracija CoAP posrednika u IPv4 i IPv6 mrežama interneta stvari

Diplomski rad

Pula, rujan 2024. godine

Sveučilište Jurja Dobrile u Puli
Tehnički fakultet u Puli

Ivan Ninčević

Integracija CoAP posrednika u IPv4 i IPv6 mrežama interneta stvari

Diplomski rad

JMBAG: 0036517556, redoviti student

Studijski smjer: Diplomski sveučilišni studij Računarstvo

Predmet: Komunikacijski protokoli

Znanstveno područje: Tehničke znanosti
Znanstveno polje: Računarstvo
Mentor: doc. dr. sc. Marko Kršulja
Komentor: asistent Dalibor Fonović

Pula, rujan 2024. godine

izv. prof. dr. sc. Marko Kršulja
asistent Dalibor Fonović

(Ime i prezime nastavnika)

Komunikacijski protokoli

(Predmet)

Tehnički fakultet u Puli

Sveučilište Jurja Dobrila u Puli
Tehnički fakultet u Puli

ZADATAK TEME DIPLOMSKOG RADA

Pristupniku **MBS:**

Ivan Ninčević **0036517556**

Studentu Tehničkog fakulteta u Puli, izdaje se zadatak za diplomski rad – tema rada pod nazivom:

Integracija CoAP posrednika u IPv4 i IPv6 mrežama interneta stvari

Sadržaj zadatka:

Zadatak diplomskog rada najprije je objasniti koncept Interneta stvari i pregled komunikacijsih protokola koji se koriste u sklopu IoT-a sa posebnim naglaskom na analizi CoAP protokola. Zatim je potrebno objasniti koncept ostvarivanja komunikacije između internetske mreže i mreže IoT uređaja integracijom posrednika (proxya). U praktičnom dijelu rada biti će prikazana komunikacija između Interneta i mreža IoT uređaja korištenjem CoAP protokola i korištenjem CoAP posrednika (CoAP proxy).

Rad obraditi sukladno odredbama Pravilnika o završnom/diplomskom radu Sveučilišta u Puli.

Ivan Ninčevič
(Ime i prezime studenta):

Redovni
(status izvanredni/redovni)

Diplomski sveučilišni studij Računarstvo
(studij)

Datum: 15. travnja 2024.

Potpis nastavnika Fonović D

Marko Kršulja

Sadržaj

1. Uvod

1. **Hipoteza:** Uspostavljanje efikasne komunikacije između internetske mreže i mreže IoT uređaja putem posredničke arhitekture i specijaliziranih protokola, poput CoAP-a, omogućit će optimiziranu razmjenu podataka i integraciju IoT sustava s internetom.
2. **Predmet istraživanja:** Predmet istraživanja je pregled i analiza komunikacijskih protokola koji se koriste u Internetu stvari, s posebnim fokusom na Constrained Application Protocol (CoAP) te implementacija mreže IoT uređaja i posredničkog rješenja za komunikaciju s internetom.
3. **Problem istraživanja:** Kako postići učinkovitu i pouzdanu komunikaciju između IoT uređaja i šire internetske mreže uzimajući u obzir ograničene resurse IoT uređaja i specifičnosti protokola prilagođenih njihovim potrebama, poput CoAP-a?
4. **Ciljevi rada:**
    - Detaljno objasniti koncept Interneta stvari i relevantne komunikacijske protokole.
    - Napraviti pregled ključnih protokola, uključujući različite verzije Internet protokola i CoAP.
    - Prikazati arhitekturu i realizaciju mreže IoT uređaja koja koristi CoAP za komunikaciju.
    - Analizirati izazove integracije IoT mreže s internetom putem posrednika.
5. **Metodologija rada:** Metodologija rada uključuje teorijski pregled ključnih koncepata Interneta stvari i komunikacijskih protokola te praktičnu implementaciju mreže IoT uređaja koristeći CoAP. Rad koristi komparativnu analizu različitih verzija Internet protokola u kontekstu IoT komunikacije. Također, prikazana je integracija IoT uređaja s internetom putem posrednika.
6. **Struktura rada:**
    - Uvod: Objašnjenje Interneta stvari i važnosti komunikacijskih protokola.
    - Pregled Interneta stvari: Primjeri upotrebe i komunikacijski tokovi.
    - Internet protokol: Pregled verzija i izazova prijelaza.
    - Komunikacijski protokoli u IoT: Detaljan pregled CoAP-a i drugih relevantnih protokola.

- Praktični dio: Implementacija i analiza mreže IoT uređaja i posrednika.
- Zaključak: Sažetak rezultata i preporuke za daljnji razvoj komunikacijskih rješenja u IoT sustavima.

2. Internet stvari

Kako bi bilo moguće objasniti način komunikacije unutar mreže interneta stvari i komunikaciju te mreže sa ostatkom interneta, potrebno je pojasniti što je, u svojoj suštini, Internet stvari.

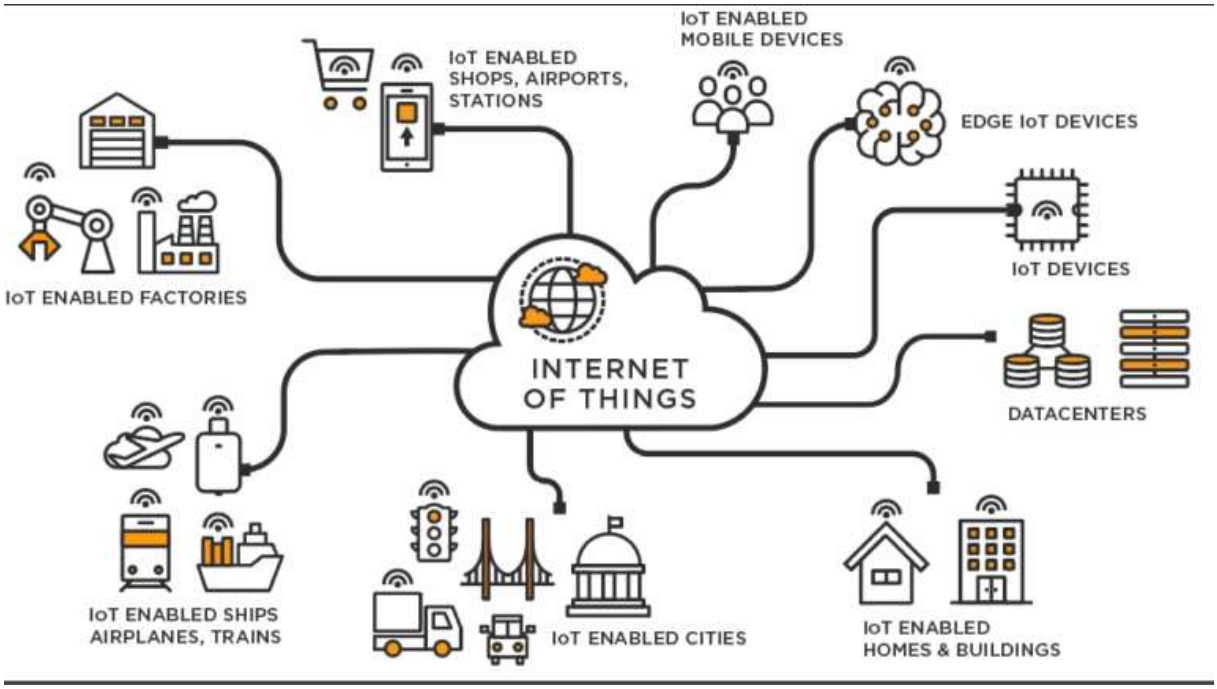
Kako je Internet stvari široko primjenjiv koncept, jedinstvena definicija nije dogovorena, već ovisno o dijelu interneta stvari koji je potrebno definirati postoje različite definicije. Definicije navedene u nastavku su u svojemu radu nazvanom „Internet of Things: A Comprehensive Overview on Protocols, Architectures, Technologies, Simulation Tools, and Future Directions“ sakupili Mohammad Mansour, Amal Gamal i ostali. Prva od definicija IoT uređaje definira kao sve uređaje koji su međusobno povezani i aktivno uključeni u ono što se može smatrati budućim internetom. [1] Druga definicija se više oslanja na same protokole, definirajući IoT uređaje kao sve uređaje koji su međusobno povezani unutar mreže i koji unutar te mreže ovise o istome protokolu, pritom opisujući Internet kao globalnu mrežu više mreža. [1] Zadnja navedena definicija definira sami koncept interneta stvari, opisujući Internet stvari kao svaki uređaj kojemu se uvijek može pristupiti od strane bilo koga, u bilo kojemu trenutku, sa bilo koje lokacije, korištenjem bilo koje aplikacije i putem bilo koje mreže. [1]

Potrebno je naglasiti da navedene definicije ne pokrivaju sva moguća shvaćanja interneta stvari, zato što sam pojam Internet stvari sadrži sve uređaje koji, neovisno o njihovoj primjerni, odrađuju operacije koje su postale ključni dio svakodnevnog života. Slika 1 ilustrira neka od područja života u kojima se pojavljuju IoT uređaji.

Ideja koja je dovela do razvoja interneta stvari bila je poprilično jednostavna, kombiniranje svakodnevnih uređaja sa jednostavnim računalima. Ovim konceptom, svakodnevni uređaji pretvoreni su u pametne uređaje koji međusobno mogu komunicirati i koordinirati kako bi njihovi zadatci bili proizvedeni što učinkovitije.

Dobar primjer kombiniranja uređaja sa jednostavnim računalima su suvremeni rendgenski uređaji. Prije samo nekoliko godina, nakon izrade rendgenske snimke, te snimke su bile pohranjene na specijaliziranom filmu, kojega je potom bilo potrebno razviti kako bi bile dobivene kvalitetne slike iz kojih je moguće napisati nalaz, odnosno dijagnosticirati pacijenta. Danas, zahvaljujući razvoju interneta stvari, suvremeni rendgenski uređaji su također i IoT uređaji koji nakon obavljenog snimanja snimku automatski šalju u bazu podataka, time izbacujući potrebu za razvijanjem filma i ubrzavajući cijeli proces dijagnostike.

Nešto više o primjenama Interneta stvari i IoT uređaja biti će rečeno u nastavku.



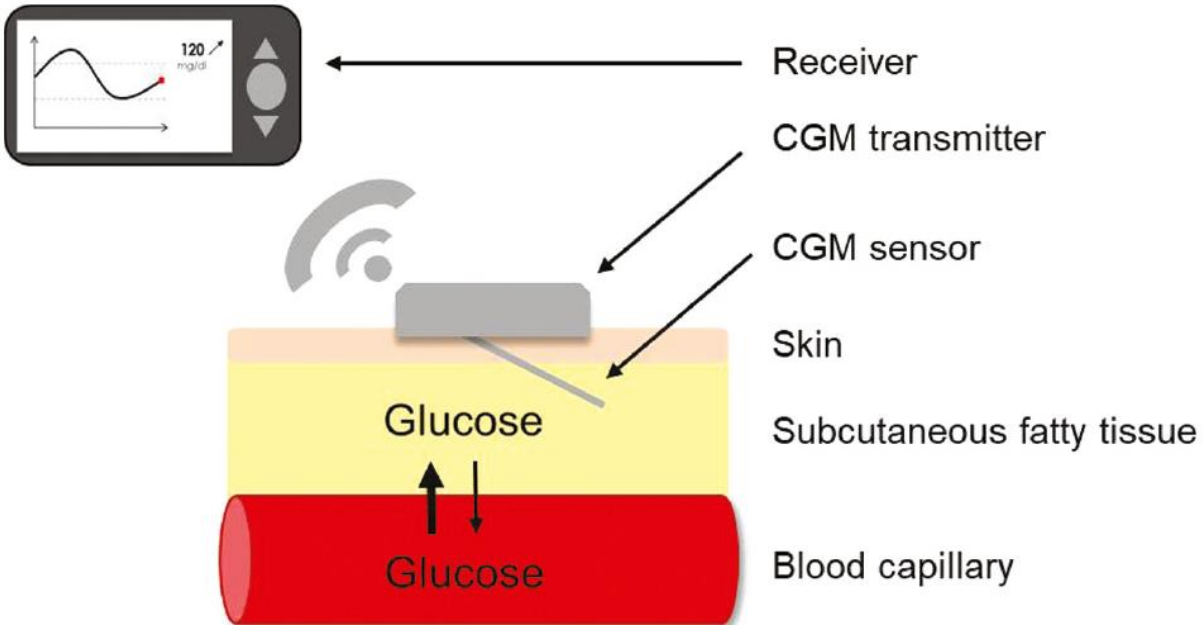
Slika 1. Ilustracija područja svakodnevnog života u kojima se koriste IoT uređaji.

2.1 Primjene Interneta stvari

Kao što je ranije navedeno, Internet stvari i IoT uređaji se danas primjenjuju u skoro svim aspektima svakodnevnog života, od medicine i industrije, sve do kućanstava. Detaljni primjeri ovih aspekata navedeni su u nastavku.

Jedno od područja primjene IoT uređaja u kojemu su ti uređaji imali najveći utjecaj je medicina, gdje su revolucionirali način na koji se izvršavaju postupci dijagnostike i praćenja stanja pacijenata u svakome trenutku. U postupcima dijagnostike, više nije potrebno na uređajima očitavati podatke te ih prepisivati u bolničke informatičke sustave, već IoT uređaji za dijagnostiku samostalno te podatke šalju u sustav, te su podatci spremni za očitavanje i pisanje nalaza bez potrebe za ručnim radom prepisivanja. Sve više su korišteni i IoT uređaji sustava za daljinsko praćenje stanja pacijenata, koji omogućavaju praćenje stanja pacijenata bez da ti pacijenti direktno leže u krevetu povezani na uređaje za praćenje. Ovakvi uređaji koriste bežičnu tehnologiju, najčešće WiFi ili bluetooth, za slanje podataka u bazu podataka nad kojom se provodi analiza stanja pojedinog pacijenta. Jedan od prvih i najviše nošenih primjera uređaja koji mogu pratiti stanje pacijenta su pametni satovi, koji svakodnevno prate stanje čovjeka, te ukoliko se pojave nekakvi zdravstveni problemi, poput nepravilnog

kucanja srca, mogu odmah upozoriti pacijenta, ali, ukoliko se pojave ozbiljniji problemi, ti uređaji mogu upozoriti i pružatelja zdravstvene njege. U sklopu primjera primjene IoT uređaja u zdravstvu, potrebno je spomenuti i pojednostavljena života koja su IoT uređaji pridonijeli ljudima koji boluju od dijabetesa. Nekada, dijabetičari su morali periodički provjeravati količinu šećera u krvi, ali danas to više nije potrebno. Danas dijabetičari stalno nose uređaje koji samostalno periodički provjeravaju količinu šećera u krvi i upozoravaju pacijenta kada dođe do potrebe za dodavanjem inzulina. [5][6][7][8]



Slika 2. Komponente sustava za kontinuirano praćenje glukoze

Internet stvari jednu od svojih najraširenijih primjena također nalazi i u gradovima, kao baza pametnih gradova. Pametni grad može se opisati kao gradsko područje koje u svojoj infrastrukturi koristi informacijsko-komunikacijsku tehnologiju, s ciljem poboljšanja kvalitete života građana. Neki od najprimjetljivijih primjera primjene IoT uređaja u ostvarivanju pametnih gradova je primjena senzora, i stvarnih podataka, kako bi se upravljalo semaforima i smanjilo vrijeme provedeno čekajući. Sličan primjer ovome, uveden u mnogim gradovima u kasnijim godinama 20. stoljeća, je daljinsko upravljanje semaforima, gdje su centri za hitne službe mogli regulirati promet kako bi vozila hitnih službi neometano mogla doći do lokacije gdje su potrebni. Također vezano uz promet i vozila, česta upotreba IoT uređaja su i senzori na parkirnim mjestima. Njima se jednostavno prikazuje koliko slobodnih parkirnih mjesta postoji na određenom parkiralištu, omogućavajući vozačima da prije samog ulaska na parkiralište znaju imali mjesta ili ne. Pored primjera vezanih uz promet, IoT uređaji se u gradovima koriste i

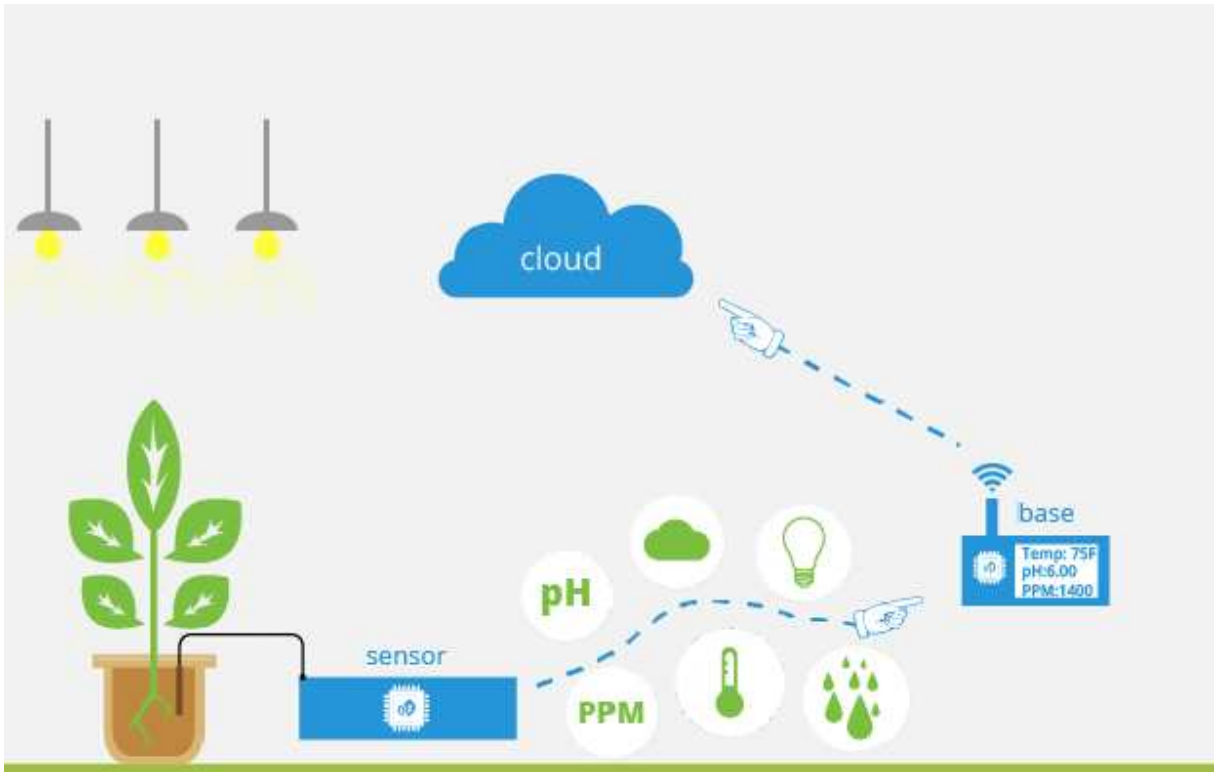
prilikom gospodarenja otpadom. Kako se u današnje doba sve više potiče građane na reciklažu, mijenjaju se stari kontejneri. Novi kontejneri u sebi imaju tehnologiju koja prati koliko tko baca otpada, te na osnovu toga izračunava koliko tko plaća odvoženje otpada. Ti kontejneri u sebi mogu sadržavati i detektor količine otpada prisutne u njima koji javlja kada je kontejner pun, odnosno kada ga je potrebno isprazniti. Uz IoT uređaje koji pojednostavljuju upravljanje gradovima, postoje i drugi IoT uređaji na mnogim lokacijama, najčešće svrhe tih drugih uređaja su u meteorološke svrhe. Meteorološki IoT uređaji, jedan od kojih je prikazan na slici 3., često su detektori temperature, vlage, naoblake i količine oborina, te zahvaljujući tim uređajima, moguće je imati podatke o trenutnom vremenu, ali je i jednostavnije prognozirati vrijeme u budućnosti. Primjer takvih detektora su i detektori kvalitete zraka koji su nedavno postavljeni i na nekoliko lokacija u Puli. [9][10][11][12][13]

Slika 3. Prikaz meteorološkog IoT uređaja sa svim modulima koje može sadržavati

Široka je primjena IoT uređaja i u industriji, gdje najveći utjecaj ima na proizvodne

procese i manufakturu. IoT uređaji u tvornicama nadgledaju ponašanje strojeva, predviđaju potrebe održavanja i prate metrike proizvodnje. Uz samo nadziranje strojeva i toka proizvodnje, IoT uređaji imaju ključnu ulogu u automatizaciji proizvodnje kompleksnih proizvoda, samostalno izvršavajući provjere nad napravljenim dijelovima prije nadograđivanja na te dijelove. Time se smanjuje mogućnost propusta koju se može očekivati prilikom ručne provjere od strane čovjeka. Sve u svemu, IoT uređaji u industriji imaju veliku ulogu u smanjenju troškova i povećanje efikasnosti proizvodnog procesa. [14][15]

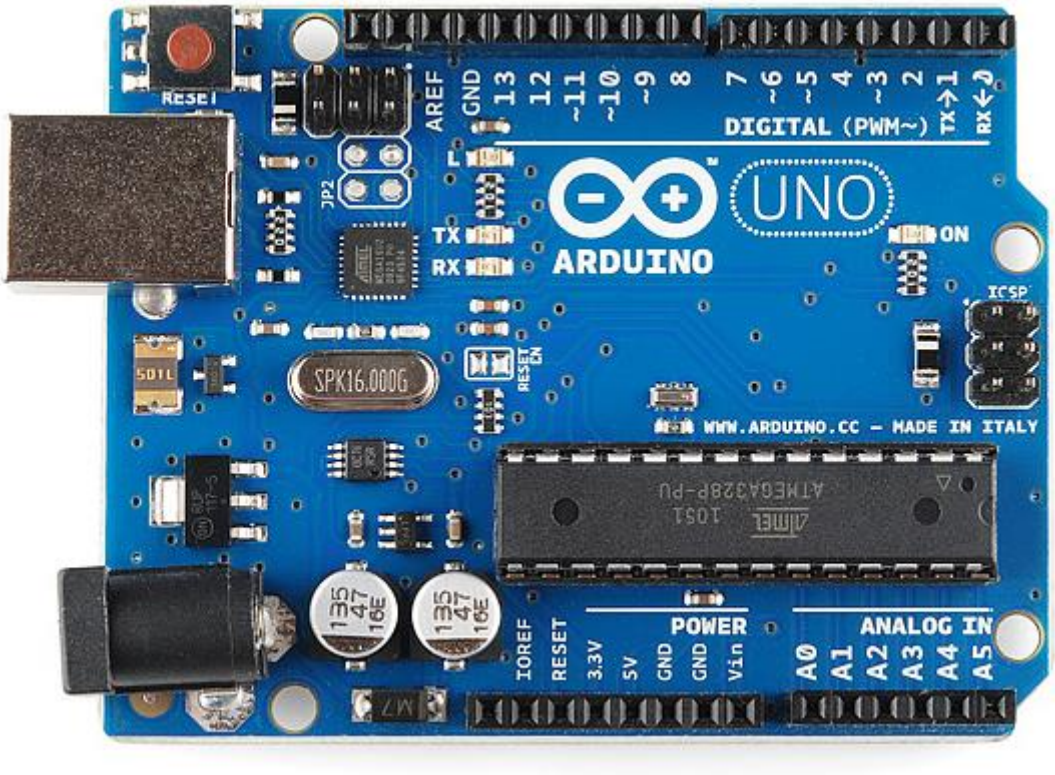
Iako ne toliko široko poznata, upotreba interneta stvari u poljoprivredi danas je sve raširenija. IoT uređaji se u kontekstu poljoprivrede koriste u sklopu nadziranja stanja tla, specifično temperature i vlage, primjer ovakvog uređaja prikazan je na slici 4. Također, korištenjem IoT uređaja može se pratiti i zdravlje sadnica, analizirajući fotografije i uspoređujući ih sa poznatim bolestima određene vrste biljke. Zahvaljujući svim informacijama koje IoT uređaji mogu prikupiti, poljoprivrednici mogu veoma jednostavno optimizirati navodnjavanje, količinu gnojiva i količinu pesticida koje je potrebno iskoristiti. Potrebno je naglasiti da IoT uređaji mogu pomagati i u zaštiti poljoprivrednih područja od štetnih životinja, nadziranju stada domaćih životinja, te i u lovu. [16][17][18][19][20]



Slika 4. Prikaz IoT uređaja za nadziranje stanja tla

2.2 Komunikacija unutar interneta stvari

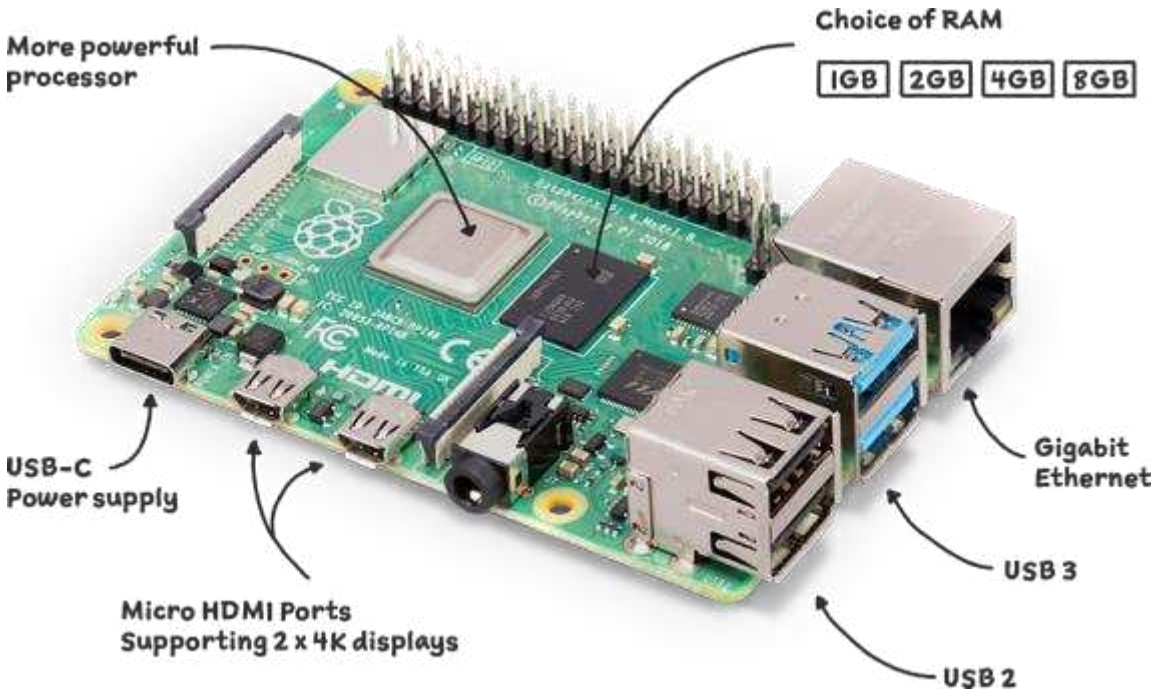
Ključ cijelog koncepta interneta stvari je komunikacija između IoT uređaja, s krajnjim ciljem omogućavanja komunikacije, dijeljenja podataka ili koordinacije bez ikakvih poteškoća u prijenosu. IoT uređaji često se pokreću na malim i jednostavnim uređajima koji nisu osposobljeni za provođenje zahtjevnih procesa, već su specijalizirani za točno određene operacije. Uređaji koji se najčešće koriste kao baza za IoT programe su mikro kontroleri i tako zvana jednoploča računala (eng. single-board computers), najbolji primjeri ovih skupina su Arduino pločice i Raspberry Pi uređaji. Primjer Arduino pločice prikazan je na slici 5., a primjer Raspberry Pi uređaja prikazan je na slici 6.



Slika 5. Prikaz Arduino pločice

Kako ranije navedeni uređaji nisu opremljeni za izvođenje kompleksnih zadataka, razvijeno je mnogo različitih oblika komunikacije, ovisno o zahtjevima pojedinog IoT uređaja, neki od zahtjeva koji su doveli do novih tehnologija su doseg, brzina razmjene podataka i potrošnja energije. Iako IoT uređaji imaju mogućnost raditi i preko žične veze, češća je upotreba bežičnih tehnologija poput Wi-Fi tehnologije, Bluetooth tehnologije, pa čak i telefonskih mreža. Jedna od posebno razvijenih tehnologija za IoT uređaje je Zigbee, bežična tehnologija razvijena 2005. godine, ova tehnologija se često koristi u pametnim kućanstvima zahvaljujući niskoj potrošnji energije i poprilično visokoj pouzdanosti. Tehnologija patentirana tek 2014. godine, LoRa, također se često

koristi u komunikaciji s IoT uređajima. Ova tehnologija je stvorena specifično s potrebama komuniciranja preko velikih udaljenosti, s minimalnom potrošnjom energije, što čini LoRa tehnologiju jednom od najpovoljnijih za komunikaciju s IoT uređajima koji se primjenjuju u poljoprivredi ili za IoT uređaje koji moraju nadgledati neki nepristupačan dio prirode.



Slika 6. Prikaz Raspberry Pi 4 uređaja

Ranije navedene tehnologije bazirane su na fizičkom sloju, te imaju svoja pravila i ograničenja, veće probleme uzrokuju veličine paketa koje protokoli stvaraju na mrežnom i aplikacijskom sloju. Ti problemi se uglavnom pojavljuju u propusnosti mreže, pojavljivanju latencije unutar mreže, pojavljivanju zagušenja mreže i u većoj potrošnji energije. Smanjenjem paketa koji se šalju putem mreže moguće je izbjeći ili smanjiti utjecaj svih navedenih problema te su u tu svrhu stvoreni novi protokoli aplikacijskog sloja koji će biti navedeni i pojašnjeni kasnije u ovome radu.

Kako bi kasnije u radu bilo moguće pokrenuti mreže na različitim verzijama Internet protokola, potrebno je prvo napraviti pregled nastanka tog protokola i pojasniti zašto postoje različite verzije tog protokola, što će biti napravljeno u nastavku.[1][9][14][15][21][22]

3. Internet protokol

Internet protokol je osnovna komponenta svih računalnih mreža te služi kao ključni protokol među svim protokolima koji se koriste na internetu. Primarna funkcija ovog protokola je dostaviti pakete od pošiljatelja do primatelja na način da promatra IP adrese koje se nalaze u zaglavlju svih paketa koji putuju internetom. Internet protokol svoju funkciju vrši na mrežnom sloju cijelog interneta, preusmjerujući pakete preko mnogih čvorova i mreža, time osiguravajući efikasan prijenos paketa.

Internet protokol radi bez potrebe za stvaranjem komunikacijskog tunela između pošiljatelja i primatelja, već paketi nezavisno putuju kroz mrežu i, u slučaju zagušenja mreže, ti paketi mogu imati različite putanje prije dolaska do primatelja. Ovakva struktura dopušta mreži da dinamički preusmjeruje pakete i efikasno rješava probleme grešaka u slanju, time smanjujući broj zagušenja do kojih može doći prilikom velike količine prometa unutar mreže. Ovakvim pristupom slanju paketa također se postiže i skalabilnost mreže, omogućavajući jednostavno dodavanje korisnika i uređaja u mrežu.

Adrese su ključna stvar Internet protokola, te svaka verzija ima svoj način stvaranja tih adresa. Verzija 4 Internet protokola (IPv4) koristi 32-bitnu arhitekturu adresa, dok verzija 6 Internet protokola (IPv6) koristi 128-bitnu arhitekturu adresa. Više o IPv4 i IPv6 biti će pojašnjeno kasnije u ovome radu.

Jedan od razloga za uspjeh Internet protokola je činjenica da je u razvoju istoga surađivalo više institucija za istraživanje i standardizaciju. Najpoznatije među njima su Internet Engineering Task Force i Institute of Electrical and Electronics Engineers, poznatiji kao IEEE. Kako su institucije za standardizaciju surađivale u razvoju, protokol je odmah prilagođavan postojećim standardima, osiguravajući da se slaže sa svim već postojećim najboljim praksama u struci. Ovakav razvoj je osigurao da protokol po svom nastavku odmah bude standardiziran, te su kasniji standardi i mrežne tehnologije rađeni na način da unaprjeđuju mogućnosti Internet protokola.

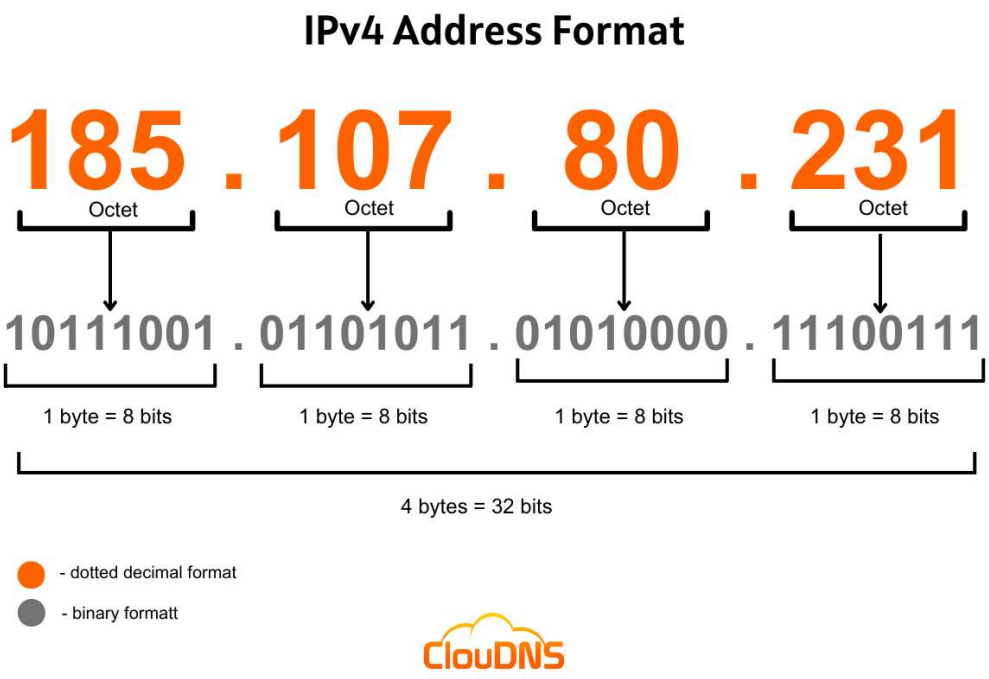
Prilagodljivost Internet protokola na promjene unutar mreže i na skalabilnost mreže bila je jedna od olakšavajućih okolnosti prilikom prelaska interneta sa akademske i istraživačke mreže na globalnu i komercijalnu upotrebu. [24][25][26][27][28]

3.1 IPv4

Verzija 4 Internet protokola, odnosno IPv4, je trenutno korištena verzija Internet

protokola, prvi put implementirana na ARPANET-u, 1983. godine. Nakon niza testiranja, ova verzija je postala baza komunikacije cijelog interneta, te je ona ključ koji je omogućio brzo širenje interneta i komercijalizaciju interneta.

Stvar koja IPv4 čini zasebnom verzijom je način na koji se adresiraju uređaji spojeni u mrežu. IPv4 koristi 32-bitni niz, odnosno niz od četiri okteta, koji se u svakodnevnoj upotrebi prikazuje u decimalnoj notaciji, kao niz od četiri decimalna broja odvojenih točkama. Primjer IPv4 adrese prikazan je na slici 7.



Slika 7. Prikaz IPv4 adrese

Kako su adrese u 32-bitnom formatu i odvojene na oktete, svaki od tih okteta može prikazivati decimalnu vrijednost u rasponu od 0 do 255, tako da je početna adresa u IPv4 0.0.0.0, a završna adresa 255.255.255.255.

Potrebno je naglasiti da nisu sve adrese unutar raspona IPv4 za javnu upotrebu, već su adrese raspodijeljene u pet klasa. Klasa A ima raspon adresa mreža od adrese 1.0.0.0 do adrese 127.0.0.0, te se koristi isključivo za jako velike mreže, odnosno za mreže koje imaju jako puno računala povezanih zajedno, kojih u ovom rasponu postoji samo 126. Klasa A, postavljena sa ovim pravilima može podržavati 16 777 214 računala spojenih na svaku mrežu ove klase. Klasa B, korištena za mreže srednjih veličina, ima raspon adresa mreža od adrese 128.0.0.0 do adrese 191.255.0.0. S tim rasponom, klasa B podržava 16 382 mreže i 65 534 računala spojenih na svaku od tih mreža.

Klasa C podržava isključivo male, lokalne, mreže, te u svom rasponu ima adrese od

192.0.0.0 do 223.255.255, što omogućava približno dva milijuna računalnih mreža, svaka od kojih može sadržavati najviše 254 računala. [29][30] Većina javnih računala prilikom spajanja na Internet automatski dobiva neku adresu unutar klase C, lokalno u Puli adrese unutar 192.168.0.0 mreže.

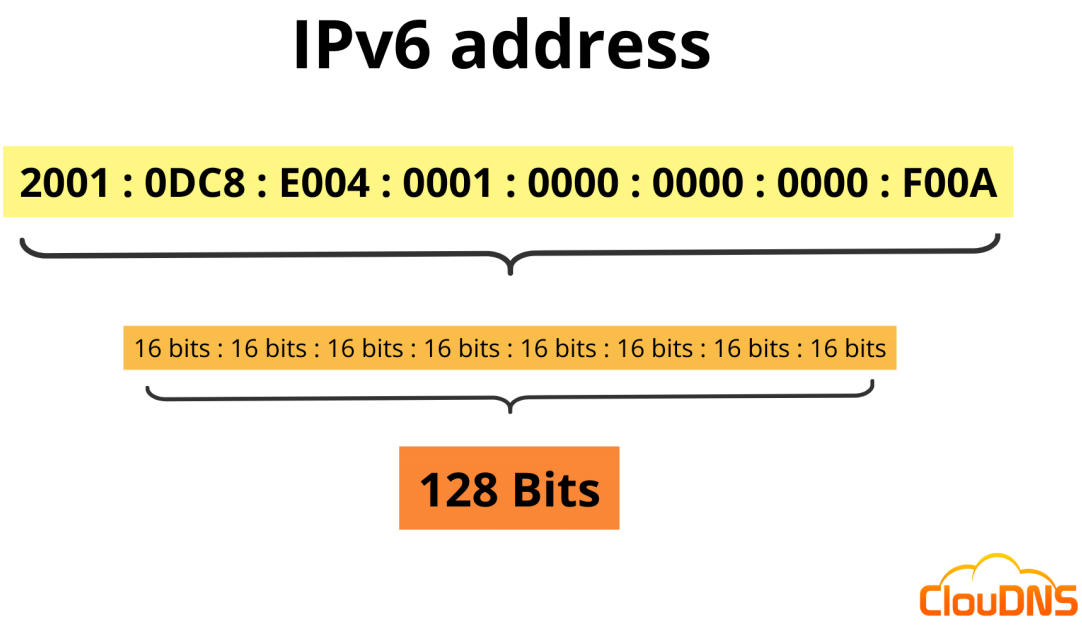
Uz klase A, B i C koje su podijeljene s obzirom na veličinu računalnih mreža koje podržavaju, klase D i E podijeljene su s obzirom na upotrebu, te nisu dostupne javnosti na korištenje. Klasa D koristi se za slanje velike količine podataka sa jednog računala na veliki broj drugih računala, ove adrese se često koriste za prijenos zvuka ili videa u stvarnom vremenu, za prijenos kabelskih televizijskih programa putem Internet protokola ili za prijenos podataka sa burze u stvarnom vremenu. Raspon adresa koje se smatraju klasom D kreće od adrese 224.0.0.0 do adrese 239.255.255.255. Klasa E u svom rasponu sadrži sve adrese od 240.0.0.0 do 255.255.255.255, te se ovaj raspon dodjeljuje isključivo u svrhe istraživanja i pokušaje unaprjeđenja rada ovog protokola. [30]

U svojoj cijelosti, 32-bitna struktura IPv4 omogućava 4 294 967 296 različitih adresa. Početkom 1980-ih, kada je ova verzija Internet protokola tek bila osmišljavana, smatralo se da nikada neće biti potrebe za iskorištavanjem svih ovih adresa, ali danas su već skoro sve dostupne adrese popunjene, te prijelaz na IPv6 počinje biti sve nužniji.[29][30][31][32][33]

3.2 IPv6

Kako je počelo biti očito da 32-bitna struktura nije dovoljno opširna, Internet Engineering Task Force je u dokumentu RFC 2460, 1998. godine, predstavio verziju Internet protokola koja bi trebala naslijediti IPv4, odnosno šestu verziju Internet Protokola, IPv6.

IPv6 u strukturi svoje adrese koristi 128 bitova, što povećava broj dostupnih adresa na 2^{128} adresa, što je broj adresa koje neće uskoro biti popunjene. Za razliku od IPv4, gdje se adrese svakodnevno prikazuju u decimalnom obliku, IPv6 adrese se prikazuju u heksadecimalnom obliku, kao osam grupa po četiri heksadecimalna znaka. Struktura IPv6 adrese prikazana je na slici 8.



Slika 8. Prikaz strukture IPv6 adrese

Uz povećanje broja adresa, IPv6 je pojednostavio format zaglavlja koje sam protokol dodaje na pakete, učinio preusmjeravanje paketa efikasnijim te dodao podršku za ekstenzije. Za razliku od IPv4, IPv6 ne koristi klase, već koristi hijerarhijsku strukturu adresa, odvajajući IPv6 adrese na tri glavna tipa: adrese koje šalju pakete točno jednom primatelju (eng. unicast), adrese koje šalju pakete na više različitih primatelja (eng. multicast), te adrese koje označavaju više različitih uređaja, ali paket se šalje najbližem (eng. anycast). [36]

Opširan raspon IPv6 adresa podijeljen je u nekoliko raspona, gdje svaki ima određenu svrhu. Neki od najbitnijih su raspon globalnih unicast adresa, odnosno raspon 2000::/3, što je ekvivalentno javnim IPv4 adresama, koje se koriste za direktnu komunikaciju. Raspon fe80::/10 koristi se za komunikaciju unutar jedne mreže, slično privatnim adresama unutar IPv4.

Unutar IPv6, zahvaljujući značajci Stateless Address Autoconfiguration-a, uređaji mogu automatski postavljati svoje IP adrese, bez potrebe za ikakvim serverom. Također, unutar IPv6 ugrađena je potpora za IPsec, što čini IPv6 sigurnijim od IPv4. [29][31][33][34][35][37]

3.3 Poteškoće prelaska sa IPv4 na IPv6

S obzirom na sve postojeće razlike između IPv4 i IPv6, količinu dostupnih adresa, efikasnost prosljeđivanja paketa, i samu sigurnost, postavlja se pitanje zašto se IPv4

uopće još uvijek koristi.

Prijelaz sa IPv4 na IPv6 predstavlja jako puno izazova, jedna od glavnih poteškoća je u nekompatibilnosti tih dvaju protokola. IPv6 je napravljen kao skroz novi protokol izrađen na Internet protokolu, a ne nadogradnja na IPv4, što znači da IPv4 i IPv6 ne mogu mijenjati jedan drugoga. Kako bi uređaji mogli raditi sa obje verzije protokola, potrebno je imati obje tehnologije na tom uređaju, te verzija protokola koja se koristi ovisi o mreži na koju je uređaj spojen. Na žalost, ovakav način rada mora se ugrađivati u sve nove uređaje kako bi ti uređaji mogli raditi za vrijeme prijelaznog perioda i nakon samog prijelaznog perioda.

Ključni problem sa ovakvim pristupom je činjenica da istovremeno pokretanje oba protokola zahtjeva iskorištavanje puno računalnih resursa, kao što su količina iskorištene memorije i količina snage procesiranja. Uz same računalne resurse, ovaj pristup zahtjeva pažljivo nadziranje sustava, kako ne bi došlo do problema sa prosljeđivanjem paketa ili konflikata prilikom adresiranja.

Veliki izazov u prijelazu sa IPv4 na IPv6 su sama cijena, količina potrebnog vremena i kompleksnost povezane sa unaprjeđivanjem infrastrukture što mnoge kompanije smatraju nepotrebnom investicijom, osobito kada se sagleda činjenica da IPv4 mreže još uvijek rade dovoljno kvalitetno, te se pojavljuju tehnologije koje produljuju vijek trajanja IPv4, poput Network Address Translation (NAT) koje omogućuju privatnim IPv4 mrežama pristup širem internetu putem jedne adrese preko koje se zajednički predstavljaju. Zadržavanje postojećih mrežnih administratora, i treniranje novih administratora, koji znaju kako pravilno upravljati IPv6 mrežama predstavlja još jedan razlog zbog kojega kompanije odgađaju prelaženje na IPv6, dodatno otežavajući i usporavajući postupak prijelaza.

Proces prelaska na IPv6 predstavlja poteškoće i u području sigurnosti, neovisno o ugrađenim svojstvima kao što je IPsec, koji omogućava kriptiranu komunikaciju, mnogi alati za sigurnost unutar mreža, kao i mnoge sigurnosne prakse, su još uvijek fokusirani na IPv4 te nisu prilagođeni, ili dovoljno testirani, na IPv6. Naravno, nedostatak alata ostavlja IPv6 mreže otvorenima za već poznate oblike sigurnosnih napada od kojih su IPv4 mreže zaštićene dodatnim alatima, ali također IPv6 ostavlja prostora za nove, ne otkrivene, oblike sigurnosnih napada, što može uzrokovati velike probleme prilikom masovnog prelaska na novi protokol.

Jedna stvar koja dodatno komplicira prelazak na IPv6 je globalnost interneta. Kako je Internet globalan, postoje mnogi pružatelji usluga koji nisu u mogućnosti primijeniti

IPv6 istom brzinom kojom to mogu neki drugi pružatelji usluga, što opet dovodi do problema gdje IPv6 mreže moraju moći surađivati sa IPv4 mrežama, za što postoje mnogi pretvarači prometa ili posrednici, koji primaju komunikaciju sa IPv4 mreže, te je prenose na IPv6 mrežu do pravog primatelja. Posrednici će biti korišteni kasnije u ovome radu, prilikom komunikacije između mreže koja koristi IPv4, odnosno IPv6, i mreže IoT uređaja.

Potrebno je naglasiti da je ovakva tehnologija potrebna zato što postoji veliki broj starijih uređaja koji ne podržavaju IPv6 tehnologiju, te će uređaji koji podržavaju i IPv4 i IPv6 biti potrebni do trenutka u kojemu više ne postoje uređaji koji ne podržavaju IPv6.

Prelazak sa IPv4 na IPv6 je kompleksan proces koji je već započeo pred više godina, i vjerojatno neće brzo biti gotov. Taj proces uključuje velike tehnološke i operacijske promjene, a s njima i potrebu za velikim financijskim sredstvima. Usprkos svim tim izazovima, prelazak na IPv6 je neizbježan zbog svakodnevnog povećanja uređaja koji u povezani sa internetom, odnosno sve manjim brojem dostupnih IPv4 adresa. [38][39][40][41][42][43]

4. Protokoli interneta stvari

Kao što je navedeno ranije u ovom radu, IoT uređaji često imaju manje mogućnosti i komponente slabijih karakteristika nego što je to kod uređaja koji uobičajeno imaju mogućnosti rada s internetom. Neke od slabijih karakteristika su često ograničena moć procesiranja, ograničena memorija ili slaba baterija. Stoga, bilo je potrebno stvoriti nove, specijalizirane, komunikacijske protokole za takve uređaje. Cilj ovih protokola bio je optimizirati poslove koje inače komunikacijski protokoli rade, odnosno optimizirati efikasnost, pouzdanost i količinu potrošene energije prilikom obavljanja tih poslova.

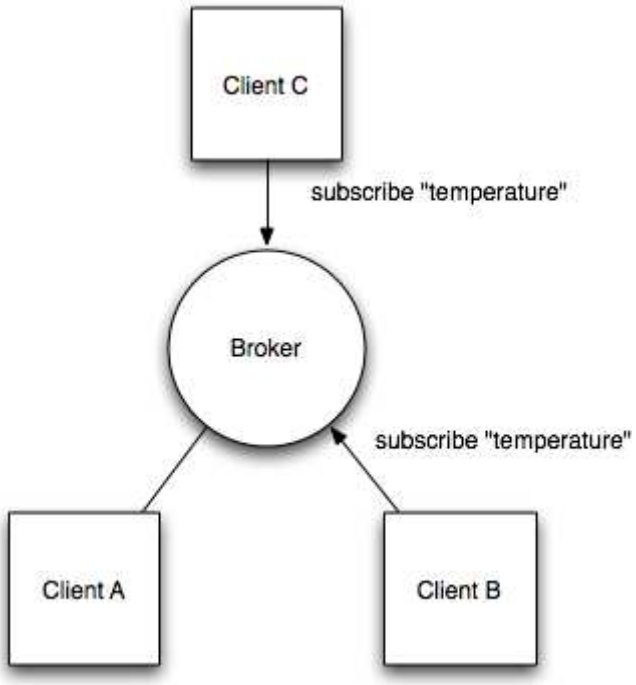
Prilikom dizajniranja IoT protokola, posvećena je pažnja ranije navedenim potrebama ovih uređaja, te su uz te potrebe postignuta i poboljšanja u efikasnosti propusnosti mreže i u smanjenju latencije. Dodatne izazove prilikom stvaranja takvih protokola napravila je širina upotrebe IoT uređaja, koja je dovela do potrebe za slanjem velikog proja različitih tipova podataka i do potrebe za kvalitetnom sigurnosti.

Najkorišteniji protokoli u internetu stvari i IoT uređajima su Hypertext Transfer Protokol (HTTP), Message Queuing Telemetry Transport (MQTT) i Constrained Application Protocol (CoAP), o kojemu će nešto više biti rečeno kasnije. [45][46][47]

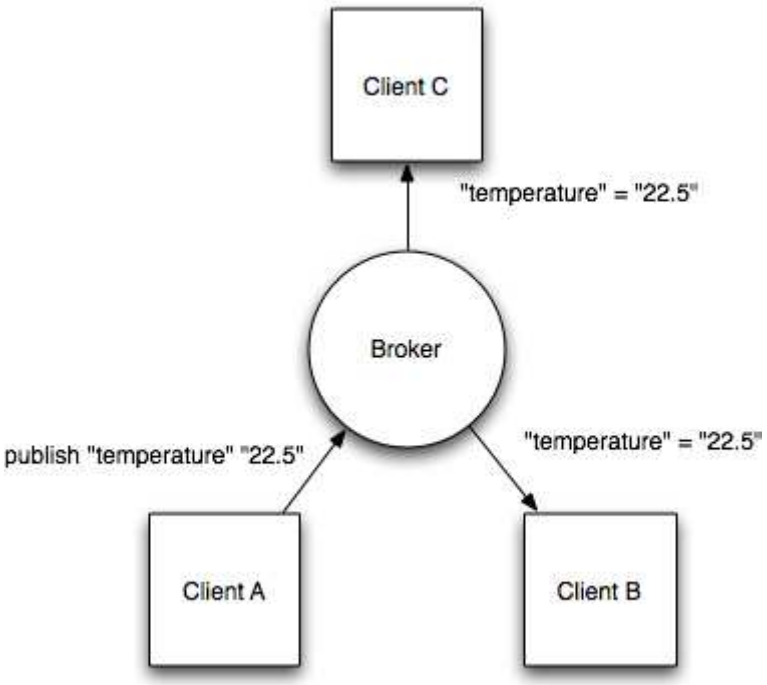
HTTP nije jedan od protokola dizajniranih za IoT uređaje, ali se svejedno često koristi u IoT uređajima i aplikacijama. Razlog tome je širina upotrebe samog HTTP i količine uređaja koji su kompatibilni s njime, ali HTTP uz podatke koje mora poslati šalje poprilično veliko zaglavlje, koje loše utječe na ranije navedena ograničenja IoT uređaja. Usprkos tim ograničenjima, HTTP se jako često koristi u IoT aplikacijama, osobito u onima gdje je potrebna suradnja, odnosno komunikacija, s drugom informatičkom opremom koja nije dio interneta stvari. Kasnije u radu, biti će prikazan način na koji je moguće zaobići ovakvu upotrebu HTTP protokola.

MQTT, kojega je kasnih 1990-ih napravio IBM, je protokol kojemu je glavni cilj slanje poruka sa malenim zaglavljem, bez uzrokovanja dodatne latencije. MQTT protokol je posebno dobar u uvjetima u kojima su propusnost mreže i latencija ključni, a te prednosti proizlaze iz činjenice da je ovaj protokol dizajniran na principu objavljivanja i pretplata na teme. Odnosno, svaki uređaj objavljuje podatke nekom posredniku na određenu adresu, koja se u ovome kontekstu naziva temom, a svi uređaji kojima ta komunikacija mora biti prenesena se na tom istom posredniku pretplaćuju na određenu temu. U trenutku kada je neki podatak objavljen na tu temu, posrednik će podatke prenijeti na sve pretplaćene IoT uređaje.[44] Primjer rada MQTT protokola prikazan je

na slikama 9 i 10, gdje je prikazano pretplaćivanje na temu, te objavljivanje podataka na tu temu. [48][49]



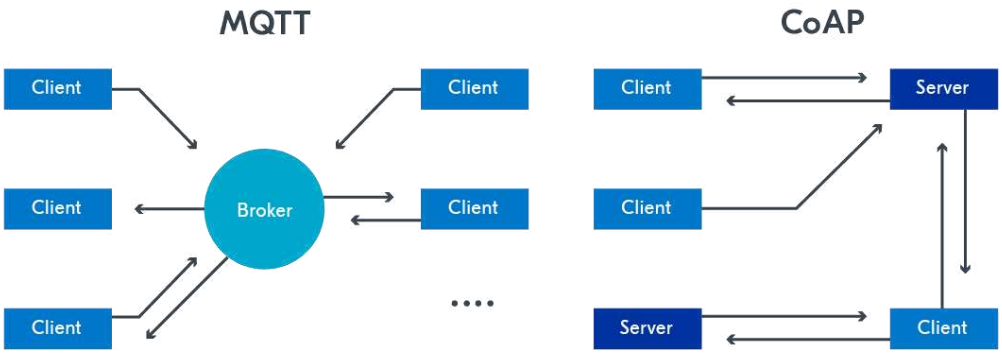
Slika 9. Primjer pretplaćivanja IoT uređaja na temu „temperature“ unutar MQTT protokola



Slika 10. Primjer objave i prosljeđivanja podataka unutar MQTT protokola

Constrained Application Protocol (CoAP), dizajniran je sa ciljem omogućavanja svih mogućnosti HTTP-a, ali bez potrebe za velikim zaglavljem, što ga čini idealnim za IoT

uređaje. Slika 11 prikazuje razliku između MQTT mreže, koja ovisi o posredniku, i CoAP mreže, koja omogućuje komunikaciju između servera i klijenata.



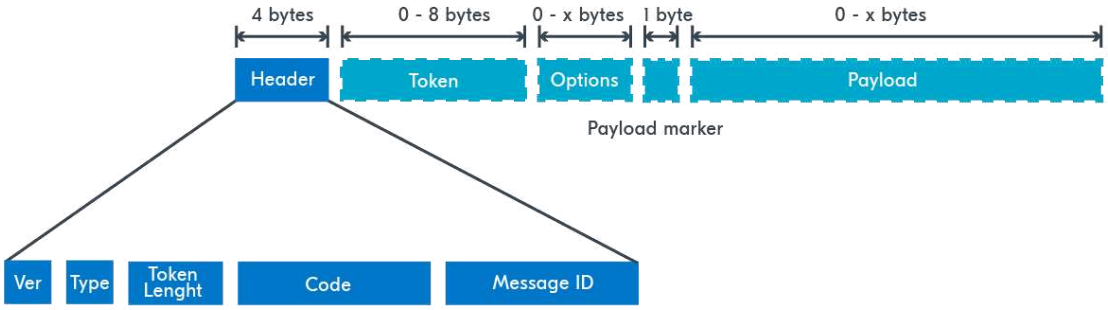
Slika 11. Razlika u konstrukciji MQTT i CoAP mreža

Kako će se prilikom pokretanja IoT mreže s posrednikom kasnije u radu koristiti CoAP, u nastavku će taj protokol biti detaljnije objašnjen.

4.1 Constrained Application Protocol (CoAP)

Gledajući povijest ranije navedenih protokola, CoAP, je relativno mlad protokol, prvi puta predstavljen 2014. godine od strane Internet Engineering Task Force-a (IEEE), te je napravljen s namjerom korištenja u mrežama s ograničenim čvorovima unutar interneta stvari. U usporedbi s HTTP-om, CoAP je dizajniran kako bi radio efikasno smanjujući zaglavlje paketa što je više moguće, bez gubitka ikakvih funkcionalnosti kao što su pouzdana komunikacija ili jednostavna integracija u postojeće mrežne tehnologije.

Kao i HTTP, CoAP radi na principu zahtjeva (eng. request) i odgovora (eng. response), ali ne koristi Transmission Control Protocol (TCP), već koristi User Datagram Protocol (UDP), što samo po sebi već smanjuje količinu informacija koja se upisuje u zaglavlje paketa. Značajka koja čini CoAP jako dobrim za IoT uređaje je mogućnost asinkrone komunikacije, koja dopušta da IoT uređaji rade sa nižom potrošnjom ili ograničenom vezom prema mreži, što omogućava Representational State Transfer (REST) arhitekturu. Uz sam prijelaz sa TCP na UDP, dodatan doprinos efikasnosti, odnosno veličini paketa, ima i činjenica da CoAP pakete kodira u binarni format, kojega je jednostavnije slati. Slika 12 prikazuje strukturu poruke unutar mreže koja koristi CoAP.



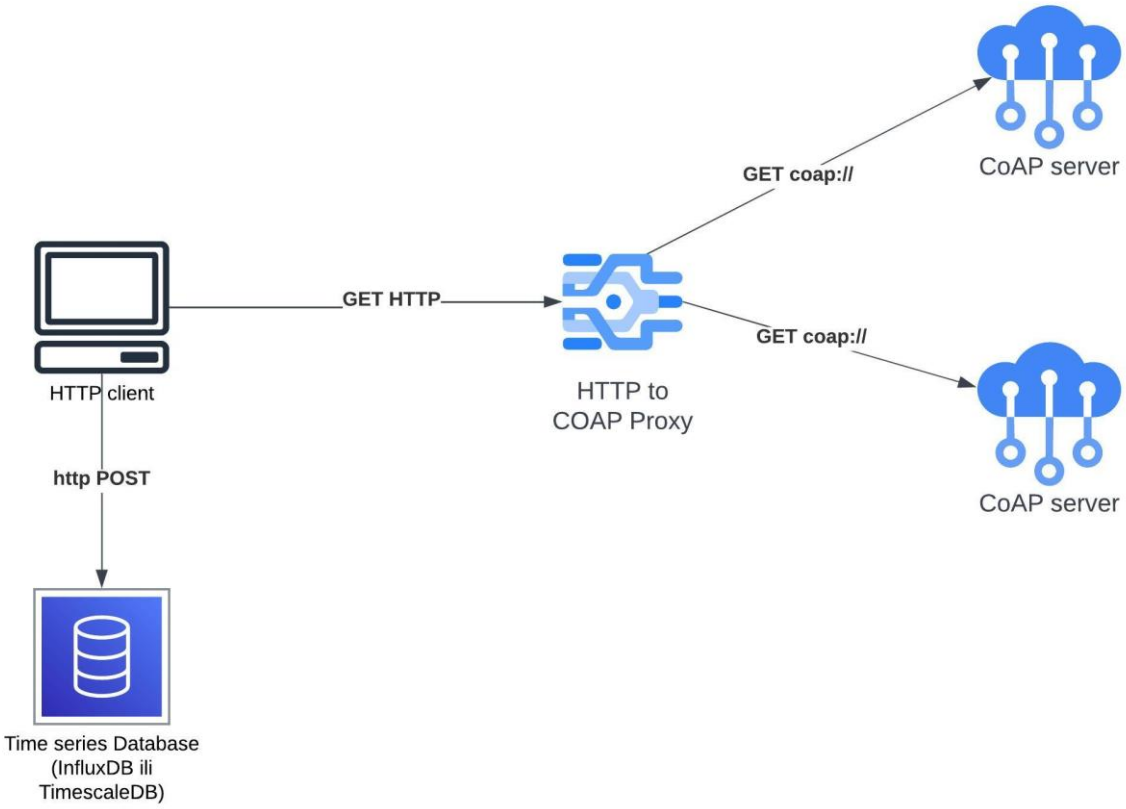
Slika 12. struktura poruke unutar Constrained Application Protocol-a

Ranije spomenuta REST arhitektura, koja se često koristi prilikom izrade aplikacija koje rade korištenjem HTTP-a, sadrži četiri metode GET, POST, PUT i DELETE, CoAP unutar sebe sadrži i nekoliko jedinstvenih metoda, primijenjenih isključivo internetu stvari. Najvažnija među tim metodama je OBSERVE, koja omogućava da IoT uređaj promatra određeni resurs te bude obavješten kada dođe do neke promjene, bez potrebe za stalnim slanjem upita.

Primjena CoAP-a je poprilično široka u internetu stvari, od kućnih uređaja sve do infrastrukture pametnih gradova, činjenica da CoAP troši malo energije i ne utječe negativno na propusnost mreža čine ga idealnim za široku primjenu. CoAP je uspješno ugrađen i u suradnju sa sustavima u oblaku (eng. cloud) što pojednostavljuje integraciju u mrežne sustave što je dodatno pojednostavljeno činjenicom da je REST arhitektura već postojeća unutar CoAP arhitekture. Zahvaljujući korištenju UDP-a u svojoj strukturi, CoAP također omogućava i istovremeni prijenos na više drugih uređaja, odnosno multicast, koji postaje sve popularniji u modernoj komunikaciji.[44][48][51][52]

5. Pokretanje mreže

Arhitektura mreže, zajedno sa protokolima koji se koriste na određenom dijelu mreže, prikazana je na slici 13. Mreža se sastoji od dva CoAP servera, jednog posrednika sa HTTP na CoAP, klijenta koji komunicira sa posrednikom i sa bazom podataka u koju zapisuje kada je poslan zahtjev za evidencijom temperature, koji uređaj je evidentirao temperaturu, i kolika je ta temperatura bila.



Slika 13. Arhitektura mreže napravljene u sklopu ovog rada

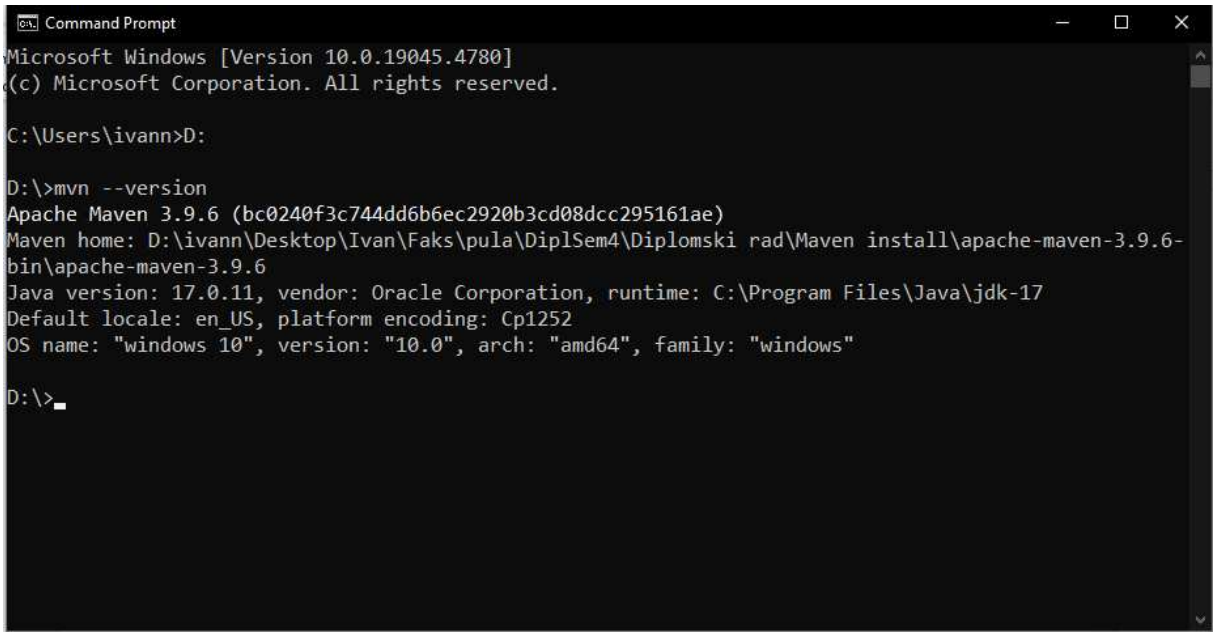
U ovome nastavku biti će prikazani koraci učinjeni za pokretanje mreže nad kojom će biti provedena analiza kasnije u ovome radu.

5.1 Pokretanje testnog okruženja

Kako bi bilo moguće napraviti IoT uređaje koji ispravno komuniciraju korištenjem CoAP protokola, potrebno je imati testno okruženje namijenjeno za CoAP komunikaciju. U svrhu testiranja rada samog CoAP protokola i aplikacija za svrhu rada nad istim protokolom, korišten je Eclipse Californium[53], koji služi kao okvir za komunikaciju pozadinskih usluga unutar IoT uređaja. Eclipse Californium je baziran na Javi, ali ta činjenica ne ograničava mogućnost komunikacije s aplikacijama napisanim u drugim

programskim jezicima.

Kako bi Eclipse Californium bilo moguće pokrenuti, potrebno je prvo instalirati Maven na računalu na kojem želimo pokretati Eclipse Californium. Maven je alat koji služi za stvaranje i pokretanje projekata pisanih u Javi, stvaranje Maven-a započelo je kada je bilo potrebno pojednostaviti izgradnju Java projekata za vrijeme stvaranja Jakarta projekta.[54] Sam proces instalacije Maven-a je poprilično jednostavan, nakon preuzimanja datoteke sa službene Apache Maven stranice, tu datoteku je potrebno raspakirati na željenu lokaciju lokalnog računala. Instalacija Maven-a zahtjeva dodavanje putanje do Maven mape u sistemske varijable i sistemske putanje. Dodavanjem putanje do Maven mape na te lokacije, omogućava se korištenje Maven naredbi u naredbenom retku (eng. command prompt) kao što je prikazano na slici 14.[56]

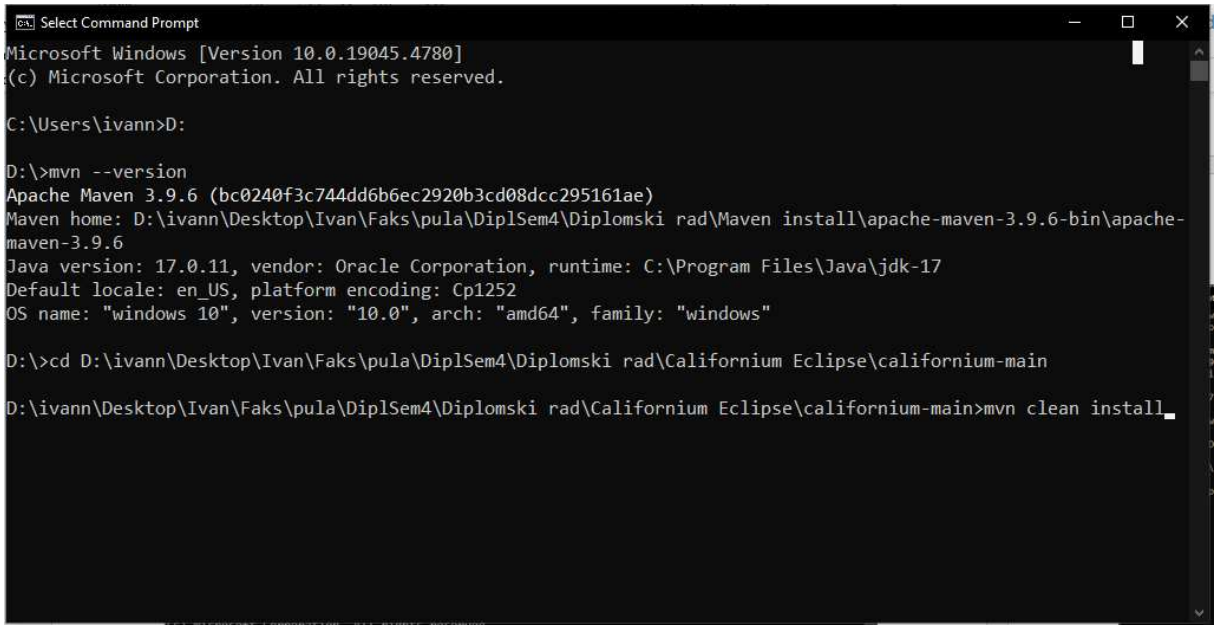


Slika 14. Prikaz korištenja Maven naredbe u command prompt-u

Nakon što je Maven instaliran, moguće je instalirati Eclipse Californium. Kako bi bilo moguće napraviti Maven instalaciju Eclipse Californium-a, potrebno je preuzeti trenutne resurse sa GitHub-a[56], te se unutar naredbenog letka pozicionirati u tu mapu i pokrenuti Maven proces izgradnje projekta, naredba za izgradnju Eclipse Californium projekta prikazana je na slici 15.

Nakon što je sam Eclipse Californium projekt izgrađen, moguće je pokretati dijelove testnog sučelja. Ključni dio testnog sučelja je alat zvan Cf-Browser, uloga tog alata je ponašati se kao klijent koji se spaja na CoAP servere. Kako bi bilo moguće instalirati Cf-Browser, potrebno je preuzeti alate za programiranje Java projekata (eng. Java

Development Kit) iz paketa javafx[57], te ga staviti na željenu lokaciju na lokalnom računalu. Nakon što je javafx preuzet potrebno je preuzeti kompajliran program sa Cf-Browser GitHub-a[57].

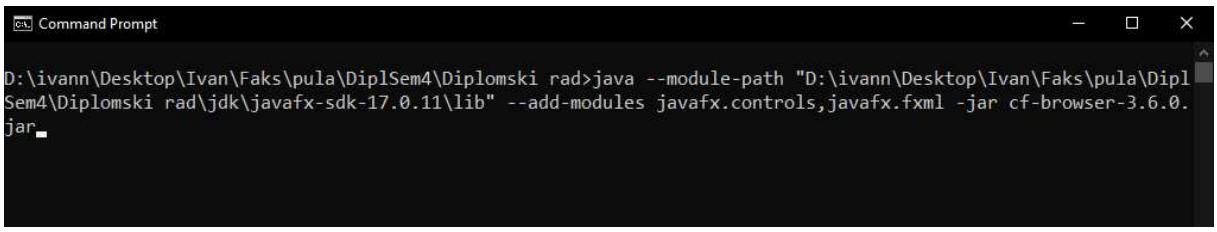


Slika 15. Prikaz naredbe za izgradnju Eclipse Californium projekta

Nakon što je kompajliran program preuzet i postavljen u željenu lokaciju, moguće je pokrenuti Cf-Browser. Kako bi se moglo pokrenuti Cf-Browser, potrebno je pokrenuti naredbu čija struktura je prikazana na slici 16., a primjer naredbe prikazan je na slici 17.

```
java --module-path <path to javafx> --add-modules javafx.controls,javafx.fxml -jar cf-browser-3.6.0.jar
```

Slika 16. Struktura naredbe za pokretanje Cf-Browser-a

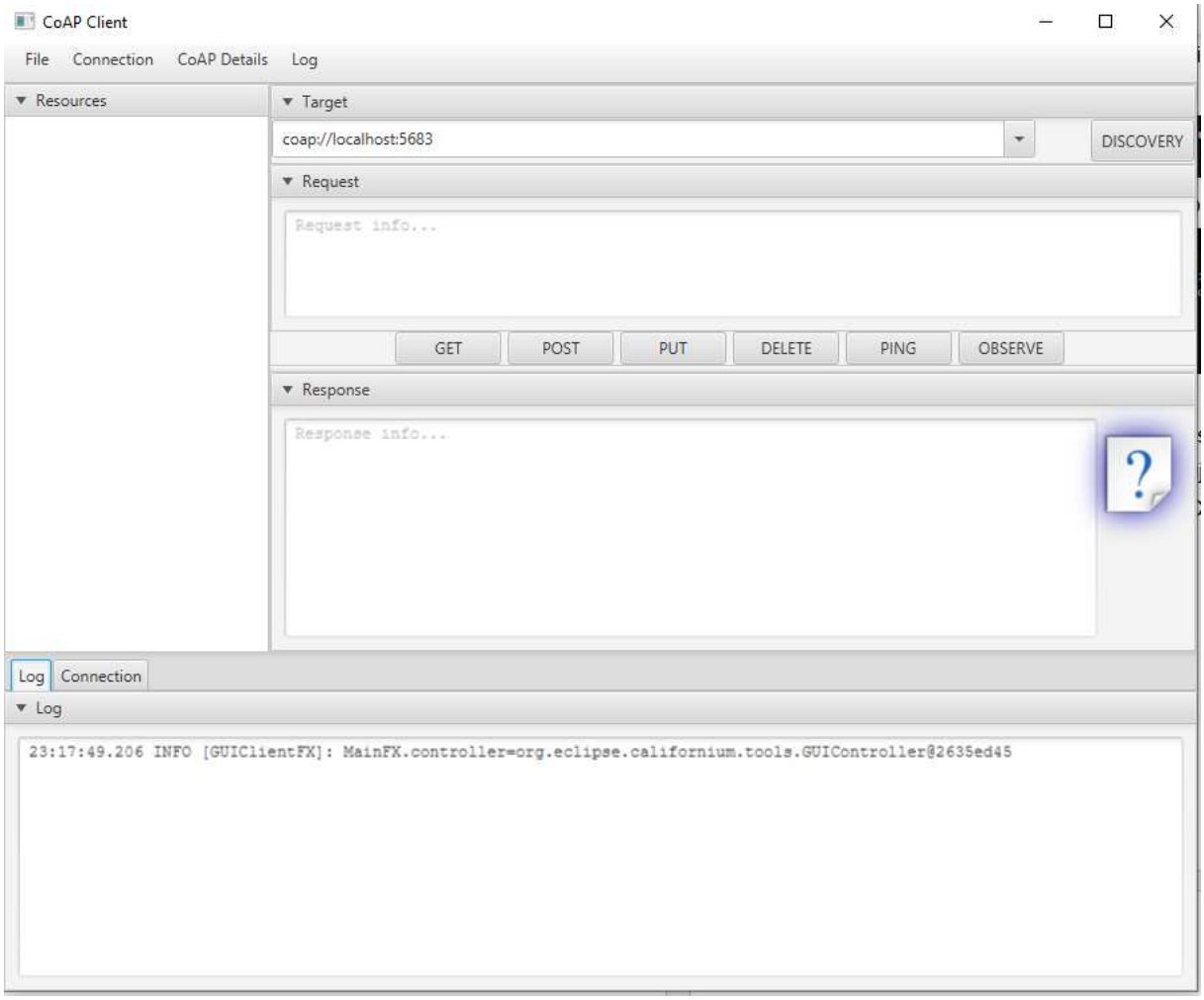


Slika 17. Primjer narede za pokretanje Cf-Browser-a

Nakon pokretanja naredbe prikazane na slici 17., pokreće se Cf-Browser, te se otvara grafičko sučelje klijenta za rad s aplikacijama koje komuniciraju putem CoAP-a. Ovo grafičko sučelje ključno je za testiranje CoAP aplikacija na lokalnom računalu, prikaz grafičkog sučelja nalazi se na slici 18.

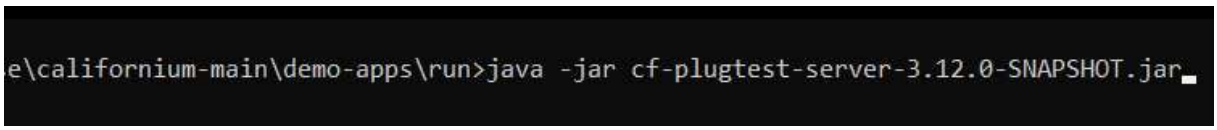
Kako bi bilo moguće testirati rad samog Cf-Browser CoAP klijenta, potrebno je pokrenuti i testni CoAP server, Eclipse Californium instalacija koju smo ranije izgradili

korištenjem Maven-a sadrži testni CoAP server. Kako bi bilo moguće pokrenuti testni CoAP server, potrebno je pozicionirati se u „.\californium-main\demo-apps\run“ mapu unutar lokalne Californium Eclipse mape.

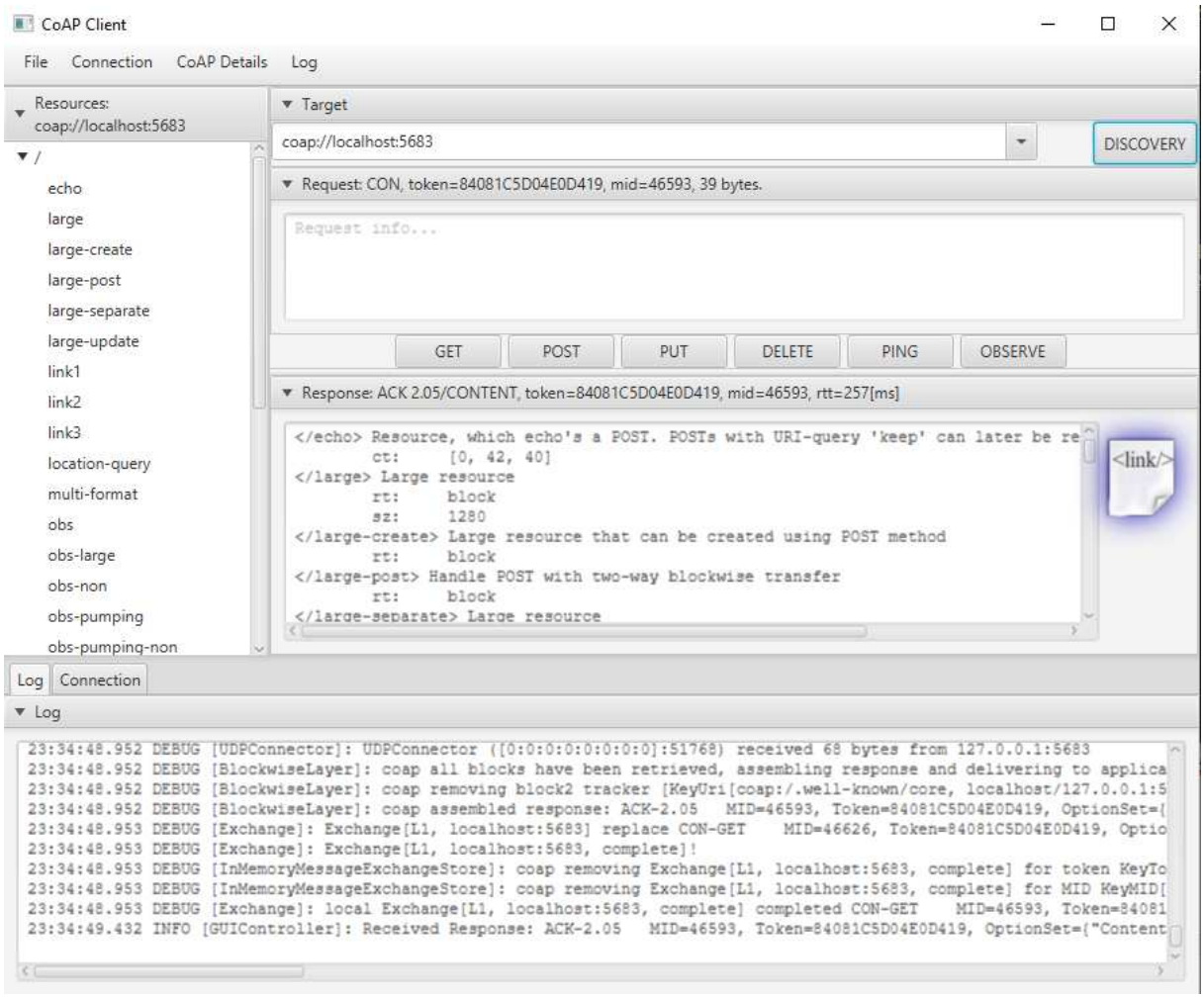


Slika 18. Prikaz Cf-Browser grafičkog sučelja bez aktivnih CoAP aplikacija

Kako bi bilo moguće pokrenuti testni CoAP server, potrebno je pokrenuti naredbu prikazanu na slici 19. Nakon pokretanja testnog CoAP servera, potrebno je pritisnuti „Discovery“ gumb, nakon čega će se prikazati svi resursi pronađenog lokalnog CoAP servera. Nakon što je server pronađen, korištenjem REST metoda, moguće je s njim komunicirati. Na slici 20. je prikazan CoAP klijent nakon što je pronašao lokalni testni CoAP server.



Slika 19. Naredba za pokretanje testnog CoAP servera, koji je dio Eclipse Californium paketa



Slika 20. Prikaz Cf-Browser grafičkog sučelja nakon pronađenog testnog CoAP servera

Nakon provjere ispravnosti rada Cf-Browser CoAP klijenta, moguće je započeti pisanje kodova, odnosno CoAP servera, koji će raditi na IoT uređajima unutar naše mreže. Kodovi tih CoAP servera biti će prikazani u nastavku.

5.2 Kodovi stvoreni u radu

U nastavku biti će prikazani i pojašnjeni kodovi za pokretanje CoAP servera, za pisanje ovih kodova korišten je Python programski jezik. Kodovi za CoAP server, posrednik prometa iz HTTP u CoAP i kod za klijent biti će prikazani u zasebnim poglavljima. Potrebno je naglasiti da je za potrebe ovog rada potrebno napisati kod koji koristi i IPv4 i IPv6, te će u nastavku biti naglašene razlike u kodovima.

5.2.1 Kodovi CoAP servera

U ovome dijelu, prikazani su kodovi CoAP servera, ključni paket korišten prilikom rada sa CoAP protokolom unutar Python-a je aiocoap paket.[58]

Kao što je ranije navedeno, kako bi bilo moguće simulirati stvarno okruženje, temperature će biti nasumično generirane u rasponu od 25.0 i 33.0. CoAP server napisan u svrhu ovog projekta ima dio koji će generirati te temperature svaki puta kada je uređaju poslan zahtjev za praćenjem temperature, dio koji će vratiti i zapisati temperaturu, te dio koji će ispisati sve evidentirane temperature. Na slici 21. prikazan je dio IPv4 koda koji pokreće cijeli server.

```
80  async def main():
81
82      # Creating the write file if it doesn't already exist
83  >   if not os.path.exists(write_file_name):···
86
87      # Resource tree creation
88      root = resource.Site()
89
90      root.add_resource(
91          [".well-known", "core"], resource.WKCResource(root.get_resources_as_linkheader)
92      )
93
94      root.add_resource(["test"], Test())
95      root.add_resource(["recordtemp"], RecordTemperature())
96      root.add_resource(["alltemperatures"], ListTemperatures())
97
98      # On Windows bind is necessary because the code can't pick up the localhost address and port by itself
99      await aiocoap.Context.create_server_context(bind=('127.0.0.1',5683), site = root)
100
101     # Run forever
102     await asyncio.get_running_loop().create_future()
103
104
105 if __name__ == "__main__":
106     parser = argparse.ArgumentParser(description="Parser is used for the name of the device that is running.")
107     parser.add_argument("name", help="Please put in the name of this device")
108     args = parser.parse_args()
109     device_name = args.name
110     asyncio.run(main())
```

Slika 21. Prikaz koda koji pokreće CoAP server.

U kodu prikazanom na slici 21. mogu se vidjeti ključni dijelovi koda koji dodaju resurse samog servera i definiraju kod koji se pokreće prilikom poziva tog resursa, odnosno te rute. U istome kodu također je moguće vidjeti liniju 99, u kojoj je moguće vidjeti 'bind' argument, kojega je potrebno dodati na Windows sustavima, zato što na tim sustavima nije moguće automatski preuzeti lokalni poslužitelj i port na kojemu će se pokretati CoAP server. Na ovoj liniji pojavljuje se razlika između IPv4 u IPv6 koda, specifično u 'bind' argumentu, potrebno je IPv4 adresu lokalnog poslužitelja s IPv6 adresom lokalnog poslužitelja, prikazanom na slici 22.

```
97      # On Windows bind is necessary because the code can't pick up the localhost address and port by itself
98      await aiocoap.Context.create_server_context(bind=('::1',5683), site = root)
99
```

Slika 22. Prikaz razlike u kodu, specifično 'bind' argumentu između IPv4 i IPv6

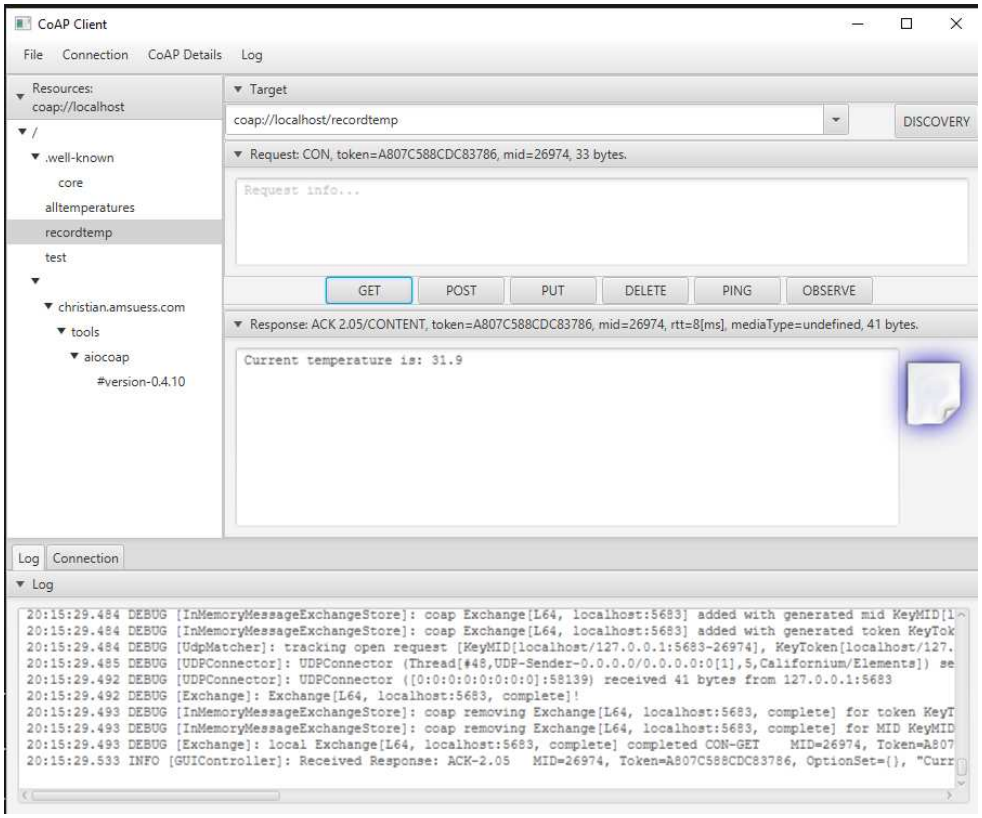
Na liniji 109 prikazan je kod koji iz argumenata pokretanja servera preuzima ime servera, koje će biti korišteno prilikom zapisivanja temperature. Linije 94, 95 i 96 prikazuju dodane resurse i metode, odnosno klase, koje te rute pozivaju. Potpuni kod klasa RecordTemperature() i ListTemperature() prikazan je na slici 23.

```
32  class RecordTemperature(resource.Resource):
33      async def render_get(self, request):
34          global device_name
35          current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M")
36
37          # In CoAP, body of a request is called "payload"!
38          current_temperature = await generate_random_temperature()
39
40          # Crafting the message to send
41          message_to_write = ""
42          message_to_write = message_to_write + ("Recording device: %s" % device_name)
43          message_to_write = message_to_write + ("\n Time of recording: %s" % current_time)
44          message_to_write = message_to_write + ("\n Recorded temperature: %s " % current_temperature)
45          message_to_write = message_to_write + ("\n-----------------------------------------------\n")
46
47          # Establishing the number of lines in the file, so that writing into the file can be confirmed.
48          with open(write_file_name, "r") as file:
49              lines = file.readlines()
50              write_file_name_start_lenght = len(lines)
51
52                  # Writing the message into the file
53          with open(write_file_name, "a") as file:
54                  file.write(message_to_write)
55
56          # Checking the current number of lines in the file.
57          with open(write_file_name, "r") as file:
58              lines = file.readlines()
59              write_file_name_end_lenght = len(lines)
60
61          # Checking if the message was added and returning an apropriate reply.
62          if(write_file_name_end_lenght > write_file_name_start_lenght):
63              text = "Current temperature is: %s" % current_temperature
64          else:
65              text = "An error ocurred while writing into file."
66
67          return aiocoap.Message(payload = text.encode("utf8"))
68
69  # Class where the CoAP device returns all recorded temperatures
70  class ListTemperatures(resource.Resource):
71      async def render_get(self, request):
72          with open(write_file_name, "r") as file:
73              lines = file.readlines()
74          recorded_temperatures = ""
75          for line in lines:
76              recorded_temperatures = recorded_temperatures + line
77          #print(recorded_temperatures)
78          return aiocoap.Message(payload = recorded_temperatures.encode("utf8"))
79
```

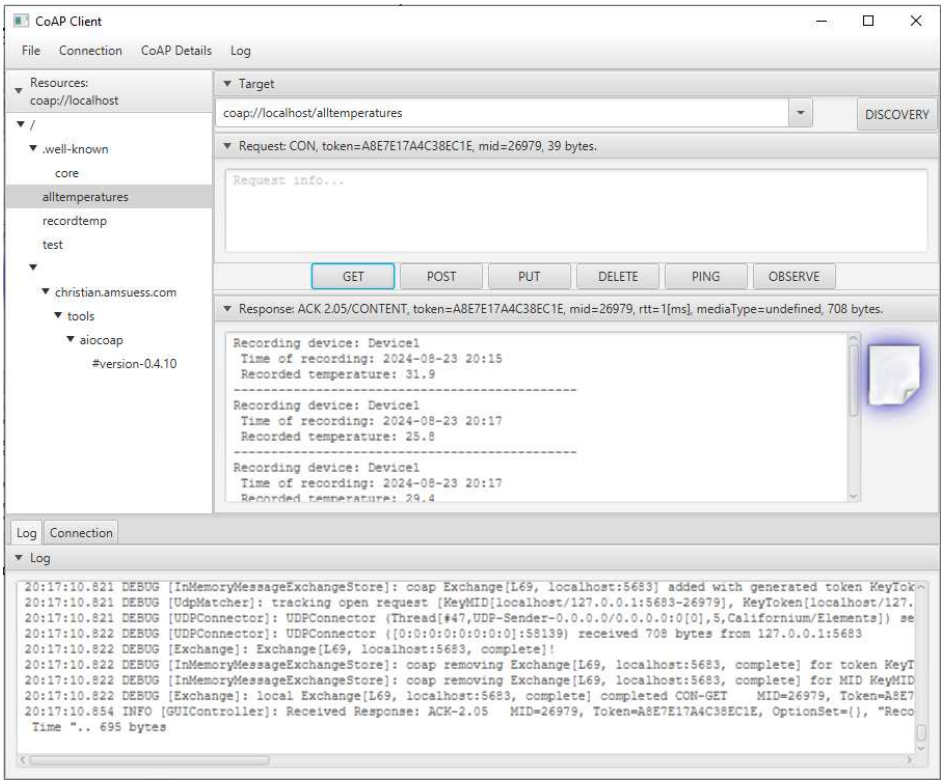
Slika 23. Prikaz koda CoAP servera u kojemu su opisane rute za evidentiranje temperature i ispis svih evidentiranih temperatura

Klasa RecordTemperature(), preuzima lokalno vrijeme, ime uređaja i nasumičnu temperaturu koje potom zapisuje u lokalnu datoteku te potom vraća poruku sa generiranom temperaturom kao odgovor. ListTemperature() klasa dohvaća sve temperature koje je uređaj evidentirao, te vraća taj popis.

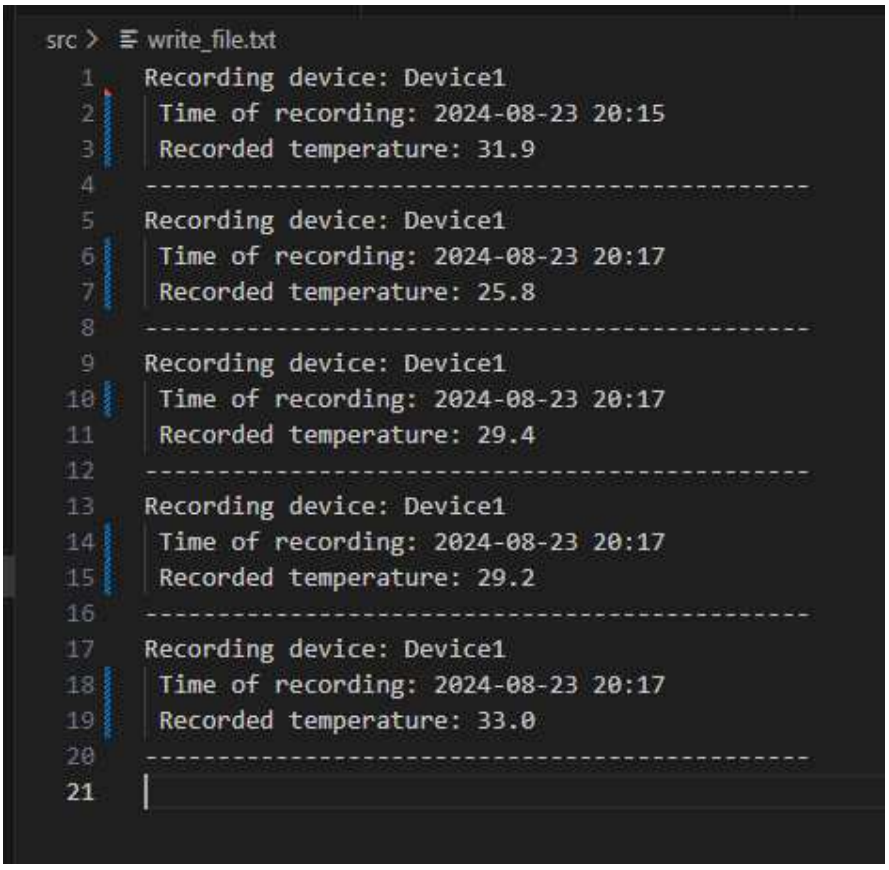
Prilikom pokretanja ovih kodova, potrebno je navesti i ime uređaja koji će biti zadano tom uređaju. Na slikama 24 i 25 prikazan je izgled Cf-Browser CoAP klijenta prilikom komunikacije s CoAP serverom. Na slici 26, prikazan je izgled datoteke u koju CoAP server upisuje evidentirane temperature.



Slika 24. izgled Cf-Browser CoAP klijenta prilikom dohvaćanja temperature sa CoAP servera



Slika 25. izgled Cf-Browser CoAP klijenta prilikom dohvaćanja svih evidentiranih temperatura sa CoAP servera



Slika 26. izgled datoteke u koju CoAP server upisuje temperature

5.2.2 Kodovi posrednika

Kao što su u prethodnom dijelu bili prikazani kodovi CoAP servera, u nastavku su prikazani kodovi posrednika iz HTTP u CoAP. Posrednik se koristi kako bi se napravila pregrada između vanjskih klijenata i unutarnje CoAP mreže, time postoji samo jedna pristupna točka preko koje postoji komunikacija prema unutarnjoj mreži, odnosno samo jedna točka koju je potrebno osigurati. Uz same sigurnosne prednosti, posrednik omogućava komunikaciju između dva različita protokola, HTTP i CoAP, na način da sa klijentom komunicira putem HTTP, a sa CoAP serverom putem CoAP protokola.

Kako bi posrednik bio uvijek dostupan, potrebno ga je pokrenuti sa nekim servisom koji će ga uvijek držati pokrenutim, za što je u ovome kodu korišten FastAPI modul.

Kako je posrednik uređaj koji komunicira sa oba CoAP servera, potrebno je imati rute za kontaktiranje oba servera. Rute za tu komunikaciju prikazane su na slici 27., te je razlika prilikom korištenja IPv6 prikazana na slici 28.

```
33  # There will be different routes for taking temperatures from different devices
34  # For the purposes of simplicity, there will be two known devices: CoAPdevice1, and CoAPdevice2
35  # There will be two routes made accordingly for eahc of these.
36  @app.get("/temperature_device1")
37  async def get_temperature_device1():
38      global context
39      # CoAP server URL
40      server_url = "coap://127.0.0.1:5683/recordtemp"  ### presently, as both devices are running
41
42      # Create a request message
43      request = Message(code=GET, uri=server_url)
44      response = "No response yet"
45
46      # Send the request and get the response
47      response = await context.request(request).response
48      print('Response code:', response.code)
49      print('Response payload:', response.payload.decode("utf8"))
50      temperature = response.payload.decode("utf8").split(":")[1].strip()
51
52      return temperature
53
54  @app.get("/temperature_device2")
55  async def get_temperature_device2():
56      global context
57      # CoAP server URL
58      server_url = "coap://127.0.0.1:5683/recordtemp"  ### presently, as both devices are running
59
60      # Create a request message
61      request = Message(code=GET, uri=server_url)
62      response = "No response yet"
63
64      # Send the request and get the response
65      response = await context.request(request).response
66      print('Response code:', response.code)
67      print('Response payload:', response.payload.decode("utf8"))
68      temperature = response.payload.decode("utf8").split(":")[1].strip()
69
70      return temperature
```

Slika 27. prikaz ruta za komunikaciju sa oba CoAP servera korištenjem IPv4

```
# There will be different routes for taking temperatures from different devices
# For the purposes of simplicity, there will be two known devices: CoAPdevice1, and CoAPdevice2
# There will be two routes made accordingly for eahc of these.
@app.get("/temperature_device1")
async def get_temperature_device1():
    global context
    # CoAP server URL
    server_url = "coap://[::1]:5683/recordtemp"  ### presently, as both devices are running on th

    # Create a request message
    request = Message(code=GET, uri=server_url)
    response = "No response yet"

    # Send the request and get the response
    response = await context.request(request).response
    print('Response code:', response.code)
    print('Response payload:', response.payload.decode("utf8"))
    temperature = response.payload.decode("utf8").split(":")[1].strip()

    return temperature

@app.get("/temperature_device2")
async def get_temperature_device2():
    global context
    # CoAP server URL
    server_url = "coap://[::1]:5683/recordtemp"  ### presently, as both devices are running on th

    # Create a request message
    request = Message(code=GET, uri=server_url)
    response = "No response yet"

    # Send the request and get the response
    response = await context.request(request).response
    print('Response code:', response.code)
    print('Response payload:', response.payload.decode("utf8"))
    temperature = response.payload.decode("utf8").split(":")[1].strip()

    return temperature
```

Slika 28. prikaz ruta za komunikaciju sa oba CoAP servera korištenjem IPv6

Također, prilikom pokretanja koda koji koristi IPv6, potrebno je na komandu dodati '--host' argument, u kojemu se naglasi IPv6 adresa, ova naredba je prikazana na slici 29. [61]

```
\src> python -m uvicorn HTTP_to_CoAP_proxy_IPv6:app --reload --host '::1'
```

Slika 29. prikaz naredbe kojom se pokreće FASTAPI kod korištenjem IPv6

5.2.3 Postavljanje baze podataka

Kako klijent preuzima evidentirane temperature te ih zapisuje u bazu podataka, potrebno je postaviti bazu podataka. U svrhu korištenja baze podataka korišteno je Timescale[59] online sučelje, preko kojega je postavljena baza podataka.

Timescale stvara baze podataka korištenjem PostgreSQL relacijske baze podataka, kojom je moguće upravljati kroz Python, korištenjem psycopg2 paketa. Kako bi klijent mogao zapisivati u bazu, stvorene su dvije tablice, jedna za IPv4, i jedna za IPv6, za što je korišteno Python sučelje, prikazano na slici 30. Iz sigurnosnih razloga, sigurnosni kod za povezivanje s bazom uklonjen je sa slike.

Svaka od stvorenih tablica sadrži polje ključa, polje za ime uređaja koji je evidentirao temperaturu, polje u koje se upisuje vrijeme kada je podatak evidentiran u bazu podataka, te polje u koje se upisuje temperatura.[60]

```python
import psycopg2


# Connecting to the database using my connection string
CONNECTION = #connection string goes here
conn = psycopg2.connect(CONNECTION)
cursor = conn.cursor()

# Creating tables
table_name = "temperatures_IPv4"
cursor.execute(f"""
    CREATE TABLE {table_name} (
        id SERIAL PRIMARY KEY,
        device VARCHAR(100),
        time TIMESTAMP,
        temperature FLOAT(1)
        );
        """)
conn.commit()
print(f"Table '{table_name}' created successfully.")

table_name = "temperatures_IPv6"
cursor.execute(f"""
    CREATE TABLE {table_name} (
        id SERIAL PRIMARY KEY,
        device VARCHAR(100),
        time TIMESTAMP,
        temperature FLOAT(1)
        );
        """)
conn.commit()
print(f"Table '{table_name}' created successfully.")

# Closing the connection
if conn:
    cursor.close()
    conn.close()
```

Slika 30. prikaz Python koda za stvaranje tablica u PostgreSQL bazi podataka

5.2.4 Pokretanje klijenta

Kako klijenti simuliraju stvarne korisnike koji će komunicirati sa CoAP serverima, potrebno je pokrenuti ih kao web aplikacije, te su u tu svrhu pokrenuti korištenjem FastAPI modula. Kao i sa ranije prikazanim kodovima, napisana su dva klijenta, jedan koji koristi IPv4 i jedan koji koristi IPv6, ali u ovom kodu razlika je samo u jednoj liniji gdje se upisuje IPv4, ili IPv6, adresa posrednika prema CoAP mreži.

Ključni dio koda klijenta je onaj koji se pokreće pri svakom pokretanju koda. Taj dio koda spaja aplikaciju na bazu podataka, te kada aplikacija završi s izvođenjem postoji drugi dio koda koji zatvara vezu aplikacije prema kodu. Ovi dijelovi koda prikazani su na slici 31, s koje je iz sigurnosnih razloga uklonjen kod za povezivanje s bazom.

```
6   app = fastapi.FastAPI()
7   proxy_url = "http://127.0.0.1:8000" ### change this to [::1] for IPv6
8   conn = ""
9   cursor = ""
10  table_name = ""
11
12
13  # Startup tasks
14  @app.on_event("startup")
15  async def startup_method():
16      global conn, cursor, table_name
17
18      # Connecting to the database using my connection string
19      CONNECTION = #connection string removed for safety reasons
20      conn = psycopg2.connect(CONNECTION)
21      cursor = conn.cursor()
22      table_name = "temperatures_IPv4"
23
24  # Shurdown tasks
25  @app.on_event("shutdown")
26  async def shutdown_method():
27      global conn, cursor
28      if conn:
29          cursor.close()
30          conn.close()
```

Slika 31. prikaz koda za spajanje na bazu podataka i za prekid veze prema bazi podataka

Klijent u svojem kodu sadrži tri rute, dvije koje dohvaćaju podatke sa CoAP uređaja, korištenjem posrednika, i jednu koja dohvaća i ispisuje sve temperature zapisane u bazi podataka. Kod za povezivanje na jedan od uređaja i kod za preuzimanje svih

zapisa iz baze podataka prikazan je na slici 32. Iz slike 32 vidljiv je način stvaranja putanje za komunikaciju s CoAP mrežom, gdje se na IP adresu posrednika prikazanu na slici 31 dodaje potrebna ruta, te se, nakon dohvata ta temperatura, trenutno vrijeme i ime uređaja zapisuju u bazu podataka.

```
66  @app.get("/temperature_dev2")
67  async def get_temperature_device2():
68      url = proxy_url + "/temperature_device2"
69
70      print("Asking Device 2 for the recorded temperature.")
71
72      async with httpx.AsyncClient() as client:
73          response = await client.get(url, timeout = 25)
74          recorded_temperature = json.loads(response.text)
75
76      current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
77
78      # Inserting recieved temperature into the database
79      cursor.execute(f"""
80          INSERT INTO {table_name} (device, time, temperature)
81          VALUES (%s, %s, %s);
82      """, ("Device 2", current_time, float(recorded_temperature)))
83      conn.commit()
84      print("Data inserted successfully.")
85
86      return recorded_temperature
87
88  @app.get("/all_recored_temperatures")
89  async def get_all_temperatures():
90      all_recorded = []
91      # Fetching all temperatures from the table
92      cursor.execute(f"SELECT * FROM {table_name};")
93      rows = cursor.fetchall()
94      for row in rows:
95          print(row)
96
97      # Parsing the result of the query so that ID is not returned
98      all_recorded = [(item[1], item[2], item[3]) for item in rows]
99
100     return all_recorded
101
```

Slika 32. prikaz koda za dohvaćanje temperature s klijenta i zapis iste u bazu, te koda za dohvaćanje svih zapisanih mjerenja iz baze podataka

Prilikom pokretanja ove aplikacije korištenjem IPv6, potrebno je samo zamijeniti adresu posrednika s IPv4 adresom posrednika, te je potrebno pokrenuti klijenta, slično kao što je prikazano s posrednikom na slici 29.

5.2.5 HTTP kodovi

Za potrebe rađenja usporedbe između CoAP-a i HTTP-a, potrebno je imati sličan način komunikacije i u HTTP protokolu, te su napisani kodovi koji komuniciraju isključivo tim protokolom. Kao što je ranije prikazan CoAP server, napisan je i HTTP server koji vrši jednaku funkciju, ali koristi drugi protokol. Na slici 33. prikazana je ruta HTTP servera koji služi za evidenciju temperature, dio koda koji vrši istu funkciju u CoAP serveru prikazan je ranije u radu, na slici 23.

```
@app.get("/record_temperature")
async def get_temperature_device1():
    global device_name
    current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M")

    current_temperature = await generate_random_temperature()

    # Crafting the message to send
    message_to_write = ""
    message_to_write = message_to_write + ("Recording device: %s" % device_name)
    message_to_write = message_to_write + ("\n Time of recording: %s" % current_time)
    message_to_write = message_to_write + ("\n Recorded temperature: %s " % current_temperature)
    message_to_write = message_to_write + ("\n-----------------------------------------------\n")

    # Establishing the number of lines in the file, so that writing into the file can be confirmed.
    with open(write_file_name, "r") as file:
        lines = file.readlines()
        write_file_name_start_lenght = len(lines)

    # Writing the message into the file
    with open(write_file_name, "a") as file:
            file.write(message_to_write)

    # Checking the current number of lines in the file.
    with open(write_file_name, "r") as file:
        lines = file.readlines()
        write_file_name_end_lenght = len(lines)

    # Checking if the message was added and returning an apropriate reply.
    if(write_file_name_end_lenght > write_file_name_start_lenght):
        text = "Current temperature is: %s" % current_temperature
    else:
        text = "An error ocurred while writing into file."

    return text
```

Slika 33. prikaz dijela koda HTTP servera u kojemu se vrši evidencija temperature

Kako kod HTTP servera nema potrebu za spajanjem na posebnu adresu, kao što je bio slučaj u kodu CoAP servera, nije potrebno pisati dvije verzije koda, već je dovoljno prilikom pokretanja naglasiti verziju Internet protokola koja se koristi.

U sklopu sučelja za komunikaciju isključivo putem HTTP-a napisan je i posrednik iz HTTP komunikacije u HTTP komunikaciju. Kao što je ranije navedeno, uloga posrednika je zaštita mreže servera tako da je postojanje posrednika bez promjene komunikacijskog protokola čest slučaj. Kako se HTTP server može pokretati na obje

verzije Internet protokola, potrebno je napisati dvije verzije koda posrednika, dio koda za komunikaciju s HTTP serverima prikazan je na slici 34.

```
30  @app.get("/temperature_device1")
31  async def get_temperature_device1():
32      # HTTP server URL (replace with your server's URL)
33      server_url = "http://[::1]:8002/record_temperature"
34
35      # Sending a HTTP request
36      async with httpx.AsyncClient() as client:
37          response = await client.get(server_url, timeout = 25)
38          recorded_temperature = json.loads(response.text)
39
40      # Returning the response
41      print(recorded_temperature)
42
43      return recorded_temperature
44
45  @app.get("/temperature_device2")
46  async def get_temperature_device2():
47      # HTTP server URL (replace with your server's URL)
48      server_url = "http://[::1]:8002/record_temperature"
49
50      # Sending a HTTP request
51      async with httpx.AsyncClient() as client:
52          response = await client.get(server_url, timeout = 25)
53          recorded_temperature = json.loads(response.text)
54
55      # Returning the response
56      print(recorded_temperature)
57
58      return recorded_temperature
59
```

Slika 34. dio koda za komunikaciju posrednika s HTTP serverima

Na slici 34., prikazan je isječak koda za komunikaciju korištenjem IPv6. Razlika između komunikacije s verzijom 4 Internet protokola nalazi se u linijama 33 i 38, gdje se nalaze adrese Internet protokola servera kojima posrednik šalje zahtjeve.

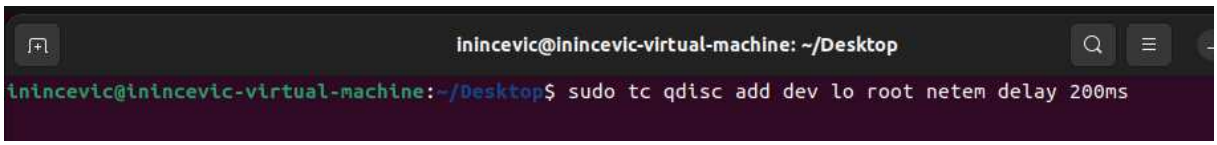
U nastavku rada biti će prikazan način praćenja prometa te će biti analizirane razlike u različitim oblicima komunikacije.

6. Analiza komunikacije

Cilj ovoga rada je analiza integracije CoAP posrednika prilikom korištenja različitih verzija Internet protokola, ali uz tu analizu potrebno je usporediti sam CoAP s HTTP-om. Kako bi bilo moguće, potrebno je postaviti stabilno testno okruženje i testne parametre koji će biti objašnjeni u nastavku rada. Kako bi bilo moguće dohvatiti što povjerljivije podatke praćen je promet na samo jednome dijelu mreže, odnosno praćena je razmjena podataka između klijenta, posrednika i jednog servera.

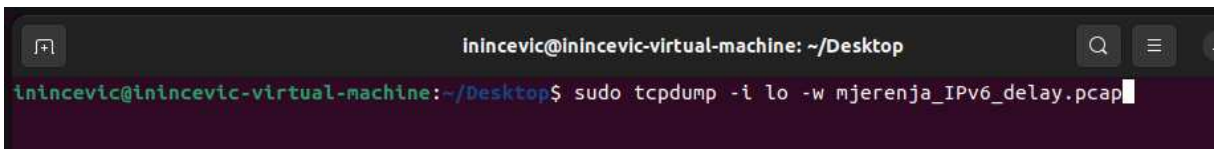
6.1 Testno sučelje

Za postavljanje testnog sučelja, odabran je virtualan Ubuntu Linux sustav[67] pokrenut na VMWare Workstation hipervizoru[68]. S obzirom na izoliranost komunikacije i komunikaciju unutar jednog virtualnog računala, potrebno je postaviti određeno usporenje prometa na mreži. Za postavljanje usporenja, korišten je „traffic control“ Linux alat, te je usporenje mreže postavljeno na 200ms. Naredba za postavljanje usporenja prilikom komunikacije unutar računala prikazana je na slici 35.



Slika 35. prikaz naredbe za usporenje komunikacije unutar računala

Također, kako bi bilo moguće pratiti svu komunikaciju unutar računala i kasnije ju analizirati, potrebno je sam promet snimiti i spremiti u određenu datoteku. Za svrhu snimanja i spremanja prometa, korišten je „tcpdump“ Linux alat, naredba za pokretanje kojega je prikazana na slici 36.



Slika 36. prikaz naredbe za snimanje komunikacije unutar računala

Nakon što je postavljeno usporenje mreže i pokrenuto snimanje komunikacije, potrebno je samo pokrenuti server i posrednika. U nastavku biti će opisani postupci testiranja.

6.2 Provođenje testiranja

Testiranje je provedeno nad četiri slučaja koja će biti uspoređena kasnije u ovome radu. Prvi od tih četiri slučaja je komunikacija korištenjem četvrte verzije Internet protokola, HTTP servera i posrednika prometa s HTTP na HTTP, drugi slučaj također u komunikaciji koristi IPv4, ali koristi CoAP server i posrednik prometa s HTTP na CoAP.

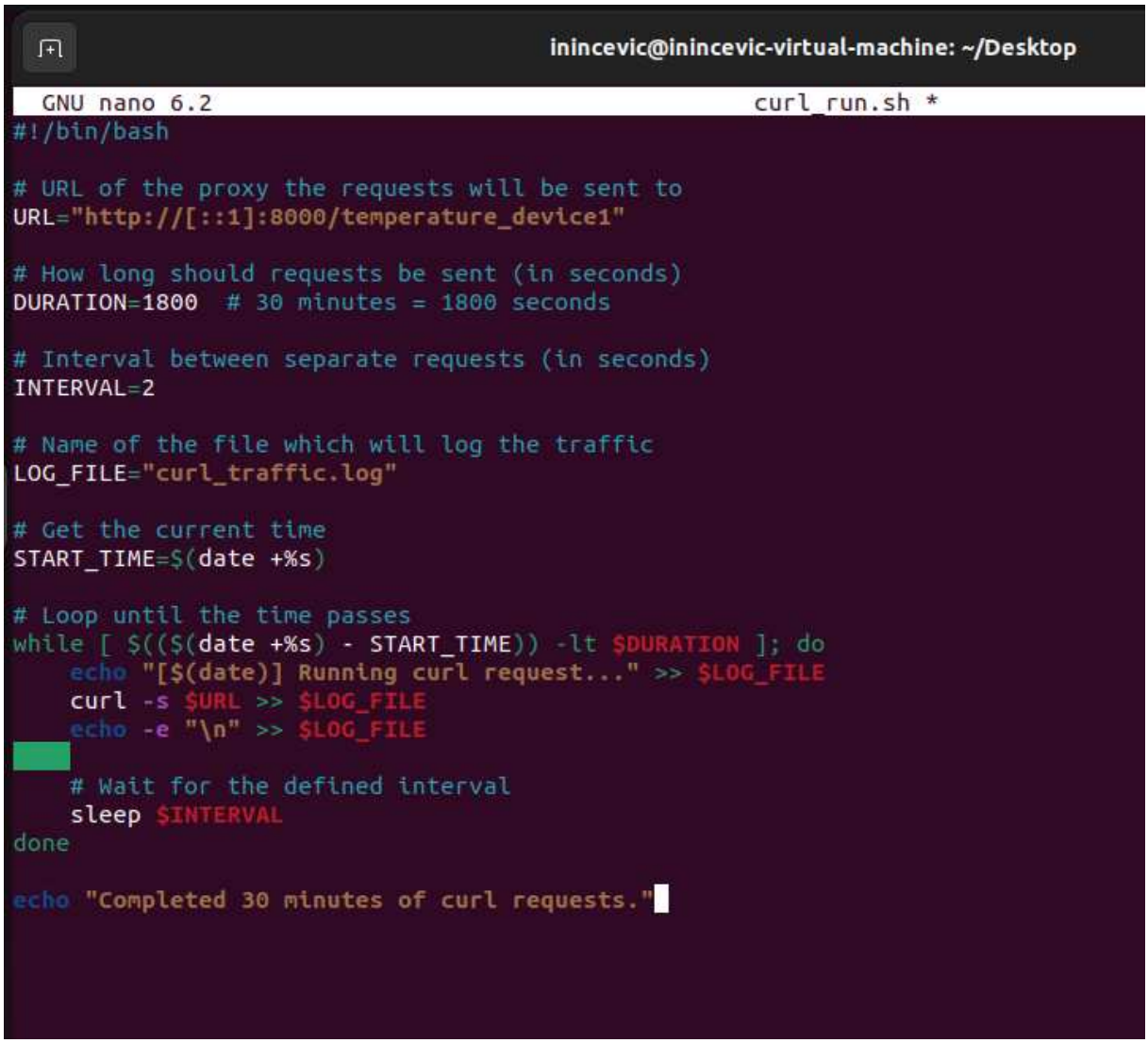
Treći slučaj koristi šestu verziju Internet protokola, odnosno IPv6, zajedno s HTTP serverom i posrednikom prometa s HTTP na HTTP, a četvrti slučaj uz korištenje IPv6 koristi CoAP server i posrednik prometa s HTTP na CoAP.

Prilikom testiranja, kao klijent, korišten je Linux terminal pomoću kojega su slani zahtjevi na posrednike korištenjem „curl“ naredbe. Ta naredba šalje zahtjev na određenu URL adresu koja je zadana kao argument, prikaz „curl“ naredbe prikazan je na slici 37.

Slika 37. prikaz „curl“ naredbe unutar Linux terminala

Naravno, slanjem jednog zahtjeva nije moguće postignuti realne rezultate, tako da je potrebno poslati veliku količinu zahtjeva. Kako bi broj zahtjeva bio dovoljno velik, odabrano je slanje zahtjeva približno svake dvije sekunde, u razdoblju od trideset minuta te je u svrhu ovakvog slanja zahtjeva napravljena skripta. Kod skripte za slanje periodičkih zahtjeva posredniku prikazan je na slici 38.

Nakon završetka izvođenja skripte, odnosno nakon obrade svih poslanih zahtjeva, i prekida snimanja komunikacije stvorena je datoteka s nastavkom .pcap, koja će u nastavku rada biti korištena prilikom analize prometa.



Slika 38. prikaz skripte za slanje periodičkih zahtjeva posredniku

6.3 Analiza prometa

Za analiziranje prikupljenih podataka korišteno je Wireshark programsko sučelje[69], unutar kojega je moguće pratit pojedine komunikacije, veličinu paketa tih komunikacija, vrijeme potrebno za te komunikacije, i slične podatke.

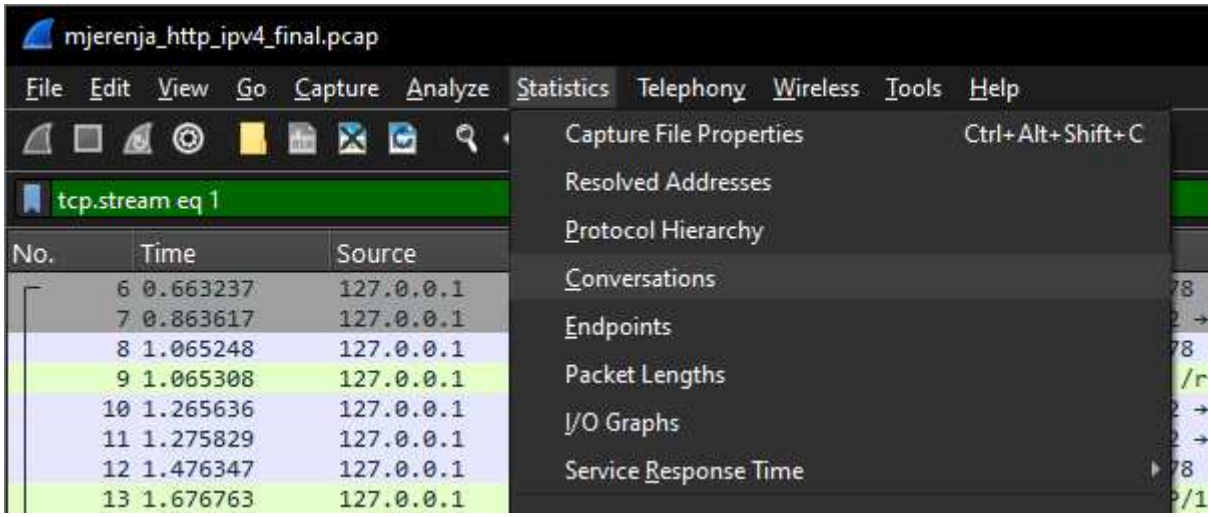
Kako bi analize bile što preciznije, u nastavku će biti uspoređena komunikacija korištenjem HTTP-a s komunikacijom korištenjem CoAP-a nad istom verzijom Internet protokola, te će biti provedena usporedba komunikacije korištenjem CoAP-a nad različitim verzijama Internet protokola.

Sve snimke prometa korištene za provođenje ove analize dostupne su na GitHub repozitoriju ovog rada.[66]

6.3.1 Analiza HTTP i CoAP komunikacije

U svrhu ove analize, biti će korištene snimke prometa korištenjem IPv4 i mreža koje se pokreću korištenjem HTTP-a, odnosno CoAP-a. Glavne mjere koje će se koristiti za usporedbu ova dva protokola su razlike u veličini paketa poslanog od servera prema posredniku i brzina prijenosa tih paketa.

Za potrebe izrade analize, korišten je Wireshark-ov statistički alat za praćenje razgovora, čija lokacija na sučelju je prikazana na slici 39.



Slika 39. Wireshark-ov statistički alat za praćenje razgovora unutar snimljene komunikacije

Korištenjem ovoga alata na snimci komunikacije HTTP servera s posrednikom s HTTP na HTTP i posrednika s klijentom, isječak dobivenih zapisa razgovora prikazan je na slici 40.

Address B	Port B	Packets	Bytes	Stream ID	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration
127.0.0.1	8000	12	1 kB	0	7	567 bajtovi	5	493 bajtovi	0.000000	2.8836
127.0.0.1	8002	12	1 kB	1	7	630 bajtovi	5	493 bajtovi	0.663237	1.6171
127.0.0.1	8000	12	1 kB	2	7	567 bajtovi	5	493 bajtovi	4.577490	2.8886
127.0.0.1	8002	12	1 kB	3	7	630 bajtovi	5	493 bajtovi	5.254697	1.6088
127.0.0.1	8000	12	1 kB	4	7	567 bajtovi	5	493 bajtovi	9.137336	2.8819
127.0.0.1	8002	12	1 kB	5	7	630 bajtovi	5	493 bajtovi	9.803736	1.6129
127.0.0.1	8000	12	1 kB	6	7	567 bajtovi	5	493 bajtovi	13.673207	3.0288
127.0.0.1	8002	12	1 kB	7	7	630 bajtovi	5	493 bajtovi	14.486600	1.6138
127.0.0.1	8000	12	1 kB	8	7	567 bajtovi	5	493 bajtovi	18.424612	2.8997

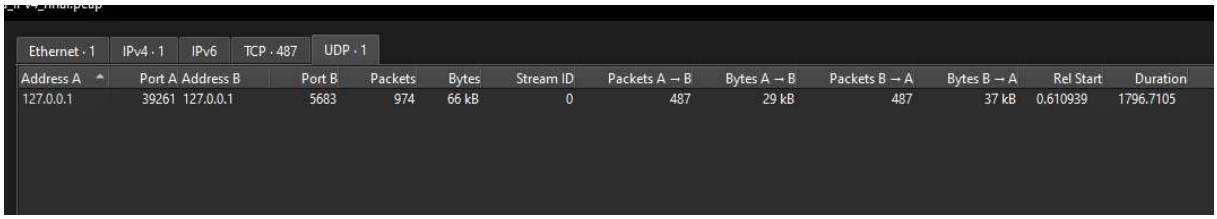
Slika 40. pregled isječka zapisa komunikacije klijenta s posrednikom s HTTP na HTTP, te komunikacije posrednika s HTTP serverom

Kako bi bilo moguće usporediti protokole, potrebno je u istom analitičkom alatu prikazati i komunikaciju CoAP servera s posrednikom s HTTP-a na CoAP i komunikaciju posrednika s klijentom. Kao što je ranije navedeno, CoAP protokol komunicira korištenjem UDP-a, tako da su zapisi komunikacije između klijenta i posrednika razdvojeni od komunikacije posrednika i servera. Isječak zapisa HTTP komunikacije prikazan je na slici 41., dok su zapisi CoAP komunikacije prikazani na

slici 42.

Address B	Port B	Packets	Bytes	Stream ID	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration
127.0.0.1	8000	12	1 kB	0	7	567 bajtovi	5	468 bajtovi	0.000000	2.0390
127.0.0.1	8000	12	1 kB	1	7	567 bajtovi	5	468 bajtovi	3.666010	2.0346
127.0.0.1	8000	12	1 kB	2	7	567 bajtovi	5	468 bajtovi	7.354952	2.0312
127.0.0.1	8000	12	1 kB	3	7	567 bajtovi	5	468 bajtovi	11.036276	2.0295
127.0.0.1	8000	12	1 kB	4	7	567 bajtovi	5	468 bajtovi	14.733682	2.0308
127.0.0.1	8000	12	1 kB	5	7	567 bajtovi	5	468 bajtovi	18.414973	2.0332
127.0.0.1	8000	12	1 kB	6	7	567 bajtovi	5	468 bajtovi	22.103488	2.0402
127.0.0.1	8000	12	1 kB	7	7	567 bajtovi	5	468 bajtovi	25.855580	2.0271
127.0.0.1	8000	12	1 kB	8	7	567 bajtovi	5	468 bajtovi	29.541797	2.0361
127.0.0.1	8000	12	1 kB	9	7	567 bajtovi	5	468 bajtovi	33.214014	2.0587

Slika 41. prikaz isječka zapisa komunikacije klijenta s posrednikom s HTTP na CoAP



Slika 42. pregled zapisa komunikacije posrednika s HTTP na CoAP i CoAP servera

Kao što je moguće vidjeti, svi HTTP zapisi su razdvojeni te nije moguće vidjeti ukupan promet ili prosječnu brzinu prijenosa, dok postoji samo jedan CoAP zapis koji zadrži ukupne podatke. Kako bi bilo moguće dobiti podatke potrebne za usporedbu, potrebno je podatke kopirati kao JSON te provesti analizu nad tim podatcima.

Prije provođenja same analize JSON-a, potrebno je naglasiti podatke koji su navedeni u zapisima i način na koji se stvarna komunikacija događa.

Polje „Port B“ označava port na kojemu je pokrenut servis s kojim je izvršena komunikacija. Kako se klijent uvijek pokreće na portu 8000, sigurno je da je sva komunikacija koja na poziciji „Port B“ ima zapis 8000 izvršena između klijenta i posrednika, dok je sva komunikacija koja na poziciji „Port B“ ima zapis 8002 izvršena između posrednika i HTTP servera. U pregledu komunikacije posrednika i CoAP servera, na polju „Port B“ zapisan je port 5683, što je unaprijed zadani port za svu CoAP komunikaciju.

Polje „Bytes“ označava ukupnu količinu podataka razmijenjenu u zapisu komunikacije, u grafičkom sučelju ovaj zapis je zaokružen, ali prilikom uvida u JSON ispis dostupna je točna vrijednost. Samo polje „Bytes“ je suma podataka prikazanih u poljima „Bytes A -> B“ i „Bytes B -> A“, odnosno suma podataka razmijenjenih u oba smjera. Zadnje bitno polje u ovom prikazu je polje „Duration“, odnosno polje u kojemu je prikazano koliko dugo je trajala određena komunikacija. Prilikom promatranja HTTP komunikacije vidljivo je koliko je određeni tok komunikacije trajao, dok se u prikazu CoAP komunikacije vidi samo ukupno trajanja. Za usporedbu trajanja komunikacije, odnosno

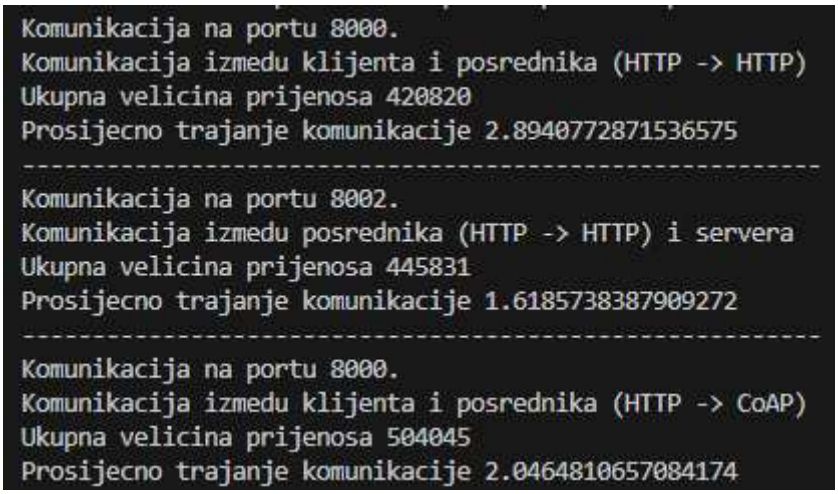
razmatranja koji je protokol brži, biti će potrebno matematički izračunati duljinu trajanja CoAP komunikacije. Postupak ovog izračuna biti će prikazan nešto kasnije u ovome radu.

Kako bi bilo moguće raditi analizu i usporedbu protokola, potrebno je imati srednje vrijednosti izračunate iz JSON oblika snimki razgovora. Kod za odabir korištenih podataka i izračun srednjih vrijednosti dostupan je na GitHub repozitoriju ovog rada[66] te je dio tog koda prikazan na slici 43. Podaci dobiveni analizom JSON datoteke prikazani su na slici 44., te je na toj slici moguće vidjeti ukupnu količinu podataka razmijenjenu HTTP komunikacijom te prosječno trajanje HTTP komunikacije.

capture_analysis > code_for_analysis.py > ...

```
1   import json, statistics
2
3   # opening the JSON data file
4   with open('http_IPv4_capture_data.json', 'r') as file:
5       full_http_traffic = json.load(file)
6   with open('coap_IPv4_capture_data_client_side.json', 'r') as file:
7       coap_http_traffic = json.load(file)
8
9
10  total_data_size = 0
11  all_durations = []
12
13  # analyzing data for client -> proxy communication
14  for record in full_http_traffic:
15      if record['Port B'] == "8000":
16          total_data_size = total_data_size + int(record['Bytes'])
17          all_durations.append(float(record['Duration']))
18          #print(record['Port B'])
19          pass
20
21  print("Komunikacija na portu 8000.")
22  print("Komunikacija izmedu klijenta i posrednika (HTTP -> HTTP)")
23  print("Ukupna velicina prijenosa", total_data_size)
24  print("Prosijecno trajanje komunikacije", statistics.mean(all_durations))
25  print("------------------------------------------------------------")
26
```

Slika 43. isječak koda za analizu JSON ispisa snimke komunikacije



Slika 44. prikaz podataka dobivenih analizom HTTP komunikacije

Podaci prikazani na slici 44., zajedno sa podatcima o CoAP prijenosu prikazani su u tablici 1. Potrebno je naglasiti da nedostaje podatak o prosječnoj duljini trajanja CoAP komunikacije, koja će biti izračunata naknadno.

Snimana komunikacija	Ukupno razmijenjenih podataka (u bitovima)	Prosječno trajanje komunikacije (u sekundama)
Klijent i posrednik (HTTP | HTTP)	420 820	2.8941
Posrednik (HTTP | HTTP) i HTTP server	445 831	1.6186
Klijent i posrednik (HTTP | CoAP)	504 045	2.0465
Posrednik (HTTP | CoAP i CoAP server	66 232	

Tablica 1. Prikaz podataka dobivenih analizom JSON oblika snimki komunikacije

Bez izračuna prosječnog trajanja komunikacije, u tablici 1. jasno je vidljiva razlika u količini podataka prenesenoj prilikom komunikacije između posrednika s HTTP-a na HTTP i HTTP servera te komunikacije između posrednika s HTTP-a na CoAP i CoAP servera. Prije nego što je moguće odrediti točne razlike u prosječnom trajanju komunikacije, potrebno je matematički izračunati prosječno trajanje komunikacije između posrednika s HTTP-a na CoAP i CoAP servera.

Kako bi bilo moguće napraviti ovaj izračun, potrebno je naglasiti da u asinkronim sustavima, odgovor na poslani zahtjev nije vraćen pošiljatelju sve dok postoji neka

komunikacija koju je potrebno izvršiti u procesu stvaranja tog odgovora. To je slučaj u ovoj komunikaciji, što znači da komunikacija između klijenta i posrednika unutar svog trajanja sadrži i period čekanja izvršenja komunikacije posrednika i servera.

Također, potrebno je naglasiti da, kako je testiranje i praćenje komunikacije rađeno u izoliranim i stabilnim uvjetima i kako je veličina zahtjeva poslanog od klijenta na posrednika uvijek jednaka, moguće je pretpostaviti da je prosječno vrijeme potrebno za slanje poruke od klijenta do posrednika jednako, neovisno o protokolu putem kojega se odvija komunikacija između posrednika i servera.

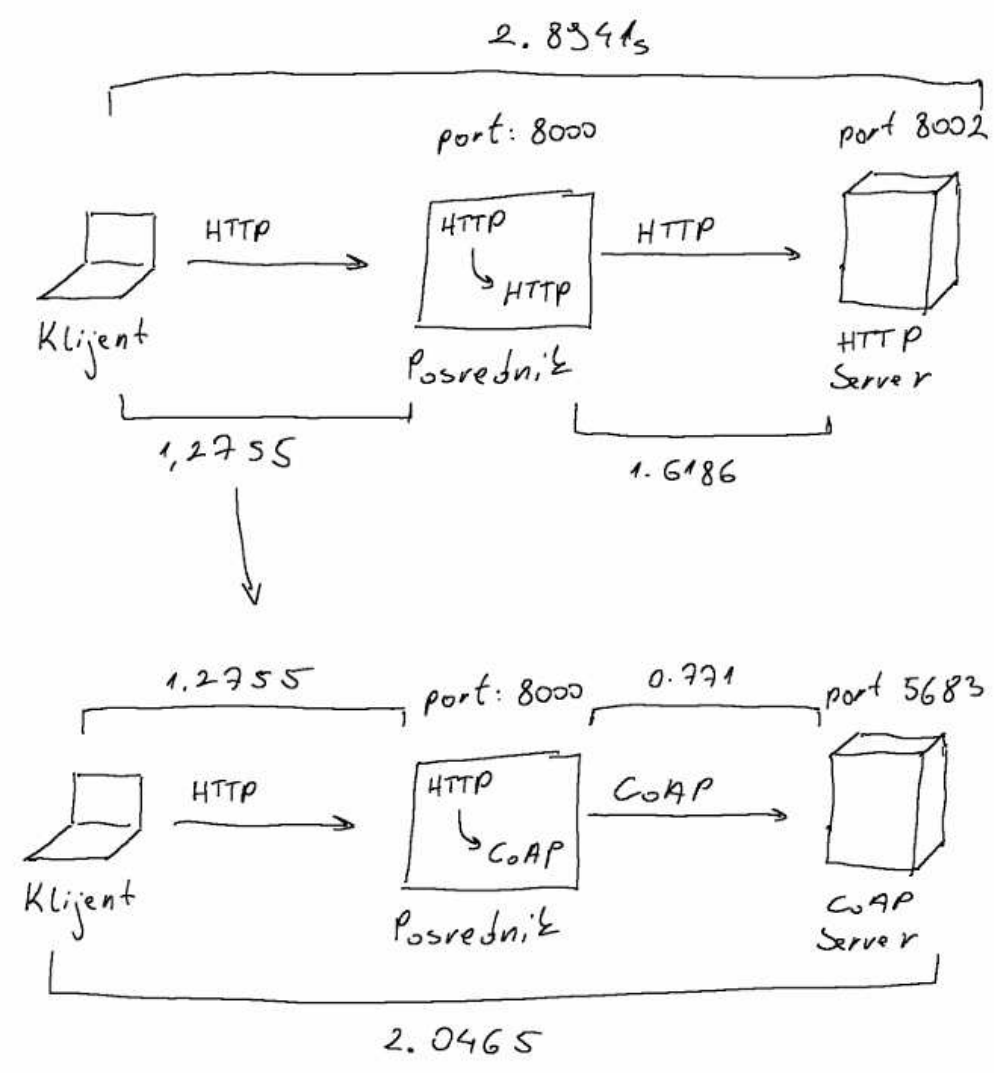
Korištenjem prosječnih vrijednosti prikazanih u tablici 1. i navedene pretpostavke, moguće je izračunati trajanje komunikacije između klijenta i posrednika oduzimanjem prosječnog trajanja komunikacije u sekundama posrednika iz HTTP-a u HTTP i HTTP servera od prosječnog trajanja komunikacije u sekundama između klijenta i posrednika iz HTTP-a u HTTP, odnosno oduzimanjem 1.6186 od 2.8941. Ovim izračunom dobiveno je prosječno trajanje komunikacije između klijenta i posrednika, bez vremena koje je provedeno čekajući komunikaciju posrednika i servera, te to trajanje iznosi 1.2755s.

Korištenjem izračunatog prosječnog vremena komunikacije klijenta s posrednikom, moguće je odrediti trajanje komunikacije korištenjem CoAP-a između CoAP servera i posrednika iz HTTP-a u CoAP oduzimanjem prosječnog trajanja komunikacije između klijenta i posrednika, bez vremena koje je provedeno čekajući komunikaciju posrednika i servera, od prosječnog trajanja komunikacije između klijenta i posrednika iz HTTP-a u CoAP. Odnosno oduzimanjem 1.2755 od 2.0465, čime se dobije prosječno trajanje CoAP komunikacije između posrednika iz HTTP-a u CoAP koje iznosi 0.771s.

U tablici 2. prikazani su svi rezultati dobiveni analizom i izračunom te je na slici 45. prikazana skica dvaju testnih okruženja sa zapisanim rezultatima.

Snimana komunikacija	Ukupno razmijenjenih podataka (u bitovima)	Prosječno trajanje komunikacije (u sekundama)
Klijent i posrednik (HTTP | HTTP)	420 820	2.8941
Posrednik (HTTP | HTTP) i HTTP server	445 831	1.6186
Klijent i posrednik (HTTP | CoAP)	504 045	2.0465
Posrednik (HTTP | CoAP i CoAP server	66 232	0.771

Tablica 2. Prikaz podataka dobivenih analizom JSON oblika snimki komunikacije i izračunom prosječnog trajanja CoAP komunikacije.



Slika 45. skica HTTP i CoAP testnih okruženja sa prikazanim prosječnim vremenima trajanja komunikacije između komponenti okruženja

Iz podataka u tablici 2., jasno je vidljivo da je CoAP bolji izbor protokola za potrebe IoT uređaja zato što zahtjeva mnogo manju količinu podataka za prijenos jednakih informacija, također je trajanje komunikacije manje od pola komunikacije prilikom korištenja HTTP protokola što je pozitivna stvar za IoT uređaje.

6.3.2 Analiza IPv4 i IPv6 komunikacije

Kao što je ranije navedeno, provedeno je snimanje komunikacije na obje verzije Internet protokola, kako bi bilo moguće usporediti koja je verzija protokola kvalitetnija za komunikaciju s IoT uređajima.

U svrhu ove analize, korištene su snimke komunikacije između klijenta, posrednika s HTTP-a na CoAP i CoAP servera kako bi bilo što jednostavnije prikazati razliku u verzijama Internet protokola. Kako bi bilo moguće usporediti protokole, korišten je isti alat kao i u ranije navedenoj usporedbi CoAP-a i HTTP-a, odnosno Wireshark-ov statistički alat za praćenje razgovora unutar snimljene komunikacije prikazan na slici 39. Snimljeni promet korištenjem IPv4, prikazan je ranije u ovome radu na slikama 41. i 42. Slika 41. prikazuje HTTP komunikaciju između klijenta i posrednika, dok slika 42. prikazuje CoAP komunikaciju između posrednika i servera. Isječak snimljenog prometa HTTP komunikacije korištenjem IPv6 prikazan je na slici 46., te je snimljeni promet CoAP komunikacije korištenjem iste verzije Internet protokola prikazan na slici 47.

Address B	Port B	Packets	Bytes	Stream ID	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration
::1	8000	12	1 kB	0	7	703 bajtovi	5	568 bajtovi	0.000000	2.0681
::1	8000	12	1 kB	1	7	703 bajtovi	5	568 bajtovi	3.708486	2.0384
::1	8000	12	1 kB	2	7	703 bajtovi	5	568 bajtovi	7.382277	2.0381
::1	8000	12	1 kB	3	7	703 bajtovi	5	568 bajtovi	11.048849	2.0450
::1	8000	12	1 kB	4	7	703 bajtovi	5	568 bajtovi	14.737521	2.0484
::1	8000	12	1 kB	5	7	703 bajtovi	5	568 bajtovi	18.425488	2.0640
::1	8000	12	1 kB	6	7	703 bajtovi	5	568 bajtovi	22.160868	2.0539
::1	8000	12	1 kB	7	7	703 bajtovi	5	568 bajtovi	25.861717	2.0570
::1	8000	12	1 kB	8	7	703 bajtovi	5	568 bajtovi	29.555650	2.0843
::1	8000	12	1 kB	9	7	703 bajtovi	5	568 bajtovi	33.337707	2.0493
::1	8000	12	1 kB	10	7	703 bajtovi	5	568 bajtovi	37.049710	2.0454
::1	8000	12	1 kB	11	7	703 bajtovi	5	568 bajtovi	40.719752	2.0354
::1	8000	12	1 kB	12	7	703 bajtovi	5	568 bajtovi	44.374986	2.0487

Slika 46. prikaz isječka zapisa komunikacije klijenta s posrednikom korištenjem IPv6

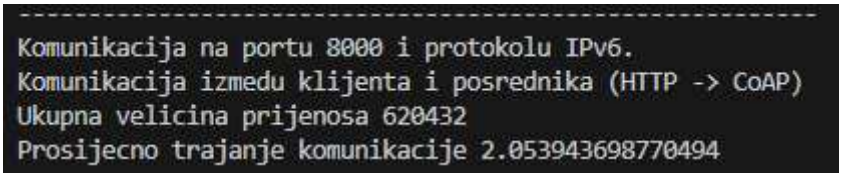
Ethernet · 1 | IPv4 · 1 | IPv6 · 1 | TCP · 488 | UDP · 2

Address A	Port A	Address B	Port B	Packets	Bytes	Stream ID	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration
127.0.0.1	5353	224.0.0.251	5353	2	169 bajtovi	1	2	169 bajtovi	0	0 bajtovi	1034.079469	159.4201
::1	40306	::1	5683	976	86 kB	0	488	39 kB	488	47 kB	0.623696	1799.0212

Slika 47. pregled zapisa komunikacije posrednika sa serverom korištenjem IPv6

Snimka HTTP komunikacije je kopirana u JSON formatu, te analiziran kodom prikazanim na slici 43., ranije u ovome radu. Rezultati analize HTTP komunikacije

između klijenta i posrednika prikazani su na slici 48., a usporedba podataka o komunikaciji korištenjem IPv4 i korištenjem IPv6 prikazani su u tablici 3.



Slika 48. prikaz podataka dobivenih analizom HTTP komunikacije korištenjem IPv6

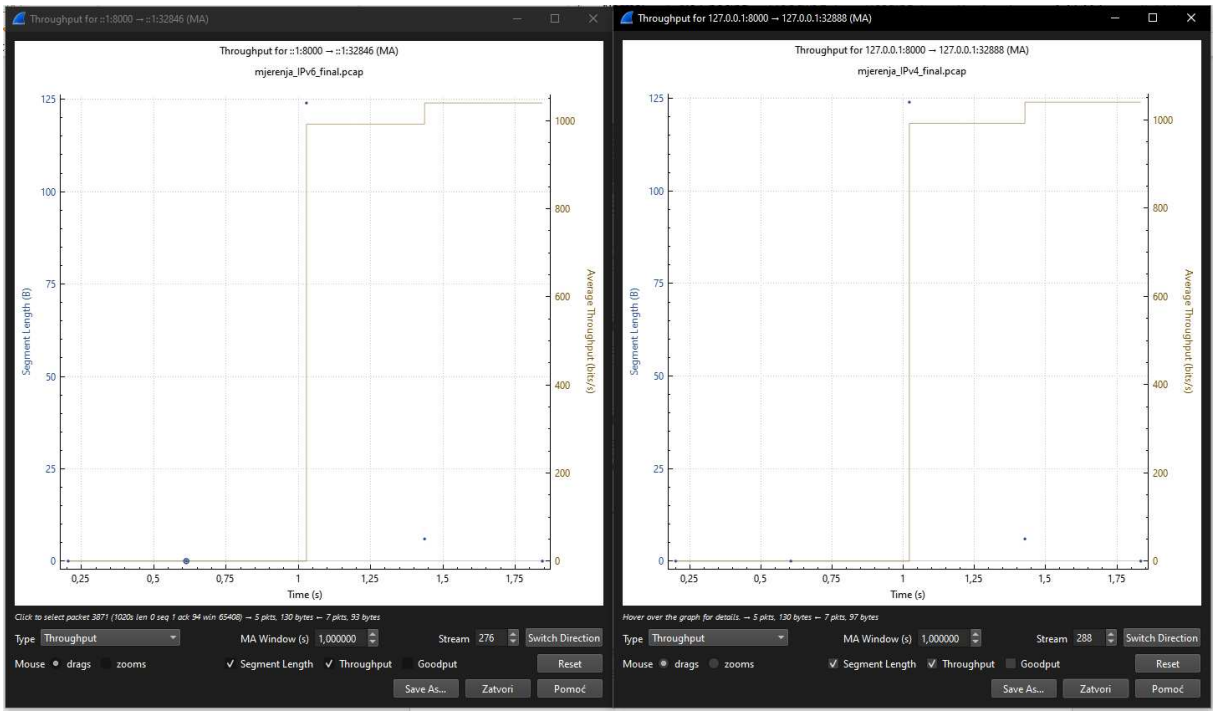
Protokol	IPv4	IPv6
Količina poslanih podataka putem HTTP-a (u bitovima)	504 045	620 432
Prosječno trajanje komunikacije unutar HTTP-a (u sekundama)	2.0465	2.0539
Količina poslanih podataka putem CoAP-a (u bitovima)	66 232	85 888
Ukupno trajanje komunikacije unutar CoAP-a (u sekundama)	1796.7105	1799.0212

Tablica 3. usporedba količine poslanih podataka i trajanja komunikacije korištenjem IPv4 i IPv6

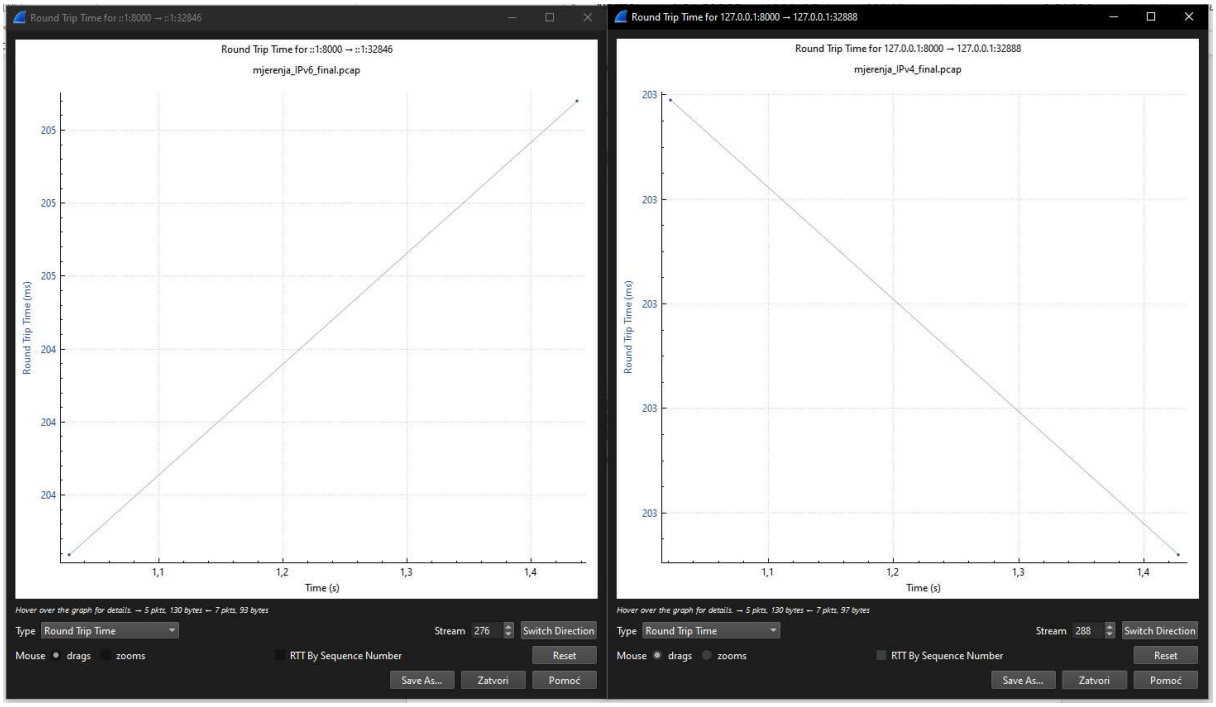
Usporedbom podataka prikazanih u tablici 3., rezultati analize prikazuju manju količinu podataka potrebnu za prijenos jednake količine informacija, te prikazuju da je IPv4 protokol nešto brži u prijenosu tih podataka u odnosu na IPv6. Kako su rezultati brzine prijenosa veoma slični, u razmatranje su uzeta još dva mjerenja, a to su propusnost mreže (eng. throughput) i povratno vrijeme komunikacije.

Prilikom razmatranja propusnosti mreže, vidljivo je da su rezultati identični, odnosno propusnost mreže je jednaka korištenjem obje verzije Internet protokola. Grafovi propusnosti mreže prilikom korištenja IPv4 i IPv6 prikazani su na slici 49.

Promatranjem povratnog vremena komunikacije vidljivo je da je povratno vrijeme skoro identično, odnosno približno 203ms korištenjem IPv4 te u rasponu od 204ms do 205ms korištenjem IPv6. Grafovi povratnog vremena komunikacije mreže prilikom korištenja IPv4 i IPv6 prikazani su na slici 50.



Slika 49. Prikaz propusnosti podataka unutar komunikacije putem IPv6 (lijevo) i IPv4 (desno)



Slika 50. Prikaz povratnog vremena komunikacije u IPv6 (lijevo) i IPv4 (desno)

Razmatranjem svih navedenih analiza, moguće je zaključiti da je IPv4 pogodniji protokol za komunikaciju s IoT uređajima i CoAP protokolom.

7. Zaključak

Internet stvari (IoT) postaje neizostavni dio svakodnevice, s primjenama u pametnik kućanskim uređajima, automobilima, medicinskoj opremi i industriji. Kako bi se osigurao daljnji razvoj IoT sustava, ključno je uspostaviti efikasne mreže u kojima ti uređaji djeluju. U radu se istražuju komunikacijski protokoli za IoT, s posebnim naglaskom na integraciju posredničkih sustava za olakšavanje komunikacije između IoT mreža i interneta.

IoT uređaji prisutni su u raznim industrijama, a njihov razvoj ovisi o učinkovitoj komunikaciji s internetom. Zbog sve većeg broja uređaja i ograničenih resursa, nužno je prilagoditi protokole komunikacije potrebama ovih sustava, uzimajući u obzir specifične zahtjeve IoT okruženja poput optimizacije potrošnje energije i resursa.

Internet protokol, odnosno prelazak s IPv4 na IPv6 postaje neizbježan zbog nedostatka adresa u IPv4, što zahtijeva prilagodbu IoT uređaja novijoj verziji protokola. Međutim, testiranja u ovom radu pokazuju da IPv4 još uvijek daje bolje rezultate u nekim segmentima komunikacije u testnim uvjetima, iako IPv6 donosi dugoročne prednosti u skalabilnosti.

IoT uređaji koriste različite aplikacijske protokole, no zbog ograničenih resursa potrebno je odabrati one koji su najefikasniji. CoAP se u ovom radu pokazao kao bolji izbor od HTTP-a jer koristi manje resursa i omogućuje bržu razmjenu podataka. U usporedbi s HTTP-om, CoAP omogućava smanjenje veličine poslanih paketa i optimizira potrošnju energije, što je od ključne važnosti za IoT uređaje.

Integracija posrednika u IoT mreže olakšava prilagodbu između različitih protokola. Uloga posrednika je presretanje zahtjeva koji dolaze putem HTTP-a i njihovo preusmjeravanje prema IoT uređajima putem CoAP-a. Ovo ne samo da smanjuje količinu podataka koji se šalju, već povećava sigurnost jer skrivene adrese IoT uređaja nisu izložene klijentima. Time se omogućava bolje upravljanje mrežom, gdje posrednik predstavlja jedinu točku komunikacije s IoT uređajima, što pojednostavljuje osiguravanje mreže.

Integracija posrednika i korištenje protokola poput CoAP-a omogućava učinkovitiju komunikaciju u IoT sustavima. Ovaj pristup smanjuje potrošnju resursa, poboljšava sigurnost i olakšava prijelaz na novije verzije Internet protokola, što je ključno za daljnji razvoj IoT tehnologije. Korištenje posrednika pojednostavljuje osiguranje mreže jer se komunikacija odvija putem jedne točke, čime se poboljšava sigurnost i održavanje

mreže.

Izvori

[1] Mansour, M. et al., „Internet of Things: A Comprehensive Overview on Protocols, Architectures, Technologies, Simulation Tools, and Future Directions“, Brno University of Technology. Dostupno na https://www.mdpi.com/1996-1073/16/8/3465 (pristupljeno 23. srpnja 2024.)

[2] Kumar, G. et al., „IOT enabled Intelligent featured imaging Bone Fractured Detection System“, Journal of Intelligent Systems and Internet of Things. Dostupno na https://www.researchgate.net/publication/373962652_IOT_enabled_Intelligent_featured_imaging_Bone_Fractured_Detection_System (pristupljeno 23. srpnja 2024.)

[3] Alghofaili, Y. i M. A. Rassam, „A Trust Management Model for IoT Devices and Services Based on the Multi-Criteria Decision-Making Approach and Deep Long Short-Term Memory Technique“, Faculty of Engineering and Information Technology, Taiz University. Dostupno na https://www.mdpi.com/1424-8220/22/2/634 (pristupljeno 23. srpnja 2024.)

[4] Duggal, N. „What Are IoT Devices? Definition, Types, and 5 Most Popular for 2024“, simplilearn, ljeto 2024., https://www.simplilearn.com/iot-devices-article (pristupljeno 23. srpnja 2024.)

[5] Griggs, K. et al., „Healthcare Blockchain System Using Smart Contracts for Secure Automated Remote Patient Monitoring“, Journal of Medical Systems. Dostupno na https://www.researchgate.net/publication/325605811_Healthcare_Blockchain_System_Using_Smart_Contracts_for_Secure_Automated_Remote_Patient_Monitoring (pristupljeno 23. srpnja 2024.)

[6] Onasanya, A. i M. Elshakankiri, „Smart integrated IoT healthcare system for cancer care“, Wireless Networks. Dostupno na: https://www.researchgate.net/publication/330097522_Smart_integrated_IoT_healthcare_system_for_cancer_care (pristupljeno 23. srpnja 2024.)

[7] Kumar, M., Kumar. R. i P. Kaur, „A healthcare monitoring system using random forest and internet of things (IoT)“, Multimedia Tools and Applications. Dostupno na: https://www.researchgate.net/publication/331286547_A_healthcare_monitoring_system_using_random_forest_and_internet_of_things_IoT (pristupljeno 23. srpnja 2024.)

[8] Freckmann. G „Basics and use of continuous glucose monitoring (CGM) in diabetes therapy“, Journal of Laboratory Medicine. Dostupno na: https://www.degruyter.com/document/doi/10.1515/labmed-2019-0189/html?lang=en

(pristupljeno 23. srpnja 2024.)

[9] Chourabi, H. et al., „Understanding Smart Cities: An Integrative Framework“. Dostupno na: https://www.researchgate.net/publication/254051893_Understanding_Smart_Cities_An_Integrative_Framework (pristupljeno 23. srpnja 2024.)

[10] Kok, I., Simsek, M. U. i S. Ozdemir, „A deep learning model for air quality prediction in smart cities“, 2017 IEEE International Conference on Big Data (Big Data). Dostupno na: https://www.researchgate.net/publication/322512343_A_deep_learning_model_for_air_quality_prediction_in_smart_cities (pristupljeno 24. srpnja 2024.)

[11] Li, Y. et al., „Smart Choice for the Smart Grid: Narrowband Internet of Things (NB-IoT)“, IEEE Internet of Things Journal. Dostupno na: https://www.researchgate.net/publication/321690295_Smart_Choice_for_the_Smart_Grid_Narrowband_Internet_of_Things_NB-IoT (pristupljeno 24. srpnja 2024.)

[12] Liu, Y. et al., „Intelligent Edge Computing for IoT-Based Energy Management in Smart Cities“, IEEE Network. Dostupno na: https://www.researchgate.net/publication/331951759_Intelligent_Edge_Computing_for_IoT-Based_Energy_Management_in_Smart_Cities (pristupljeno 24. srpnja 2024.)

[13] NiuBol, „What is the IoT weather station?”, NiuBol, jesen 2023., https://www.niubol.com/Product-knowledge/What-is-the-IoT-weather-station.html (pristupljeno 24. srpnja 2024.)

[14] Zhong, Y. R. et al., „Intelligent Manufacturing in the Context of Industry 4.0: A Review“, Engineering. Dostupno na: https://www.researchgate.net/publication/321280578_Intelligent_Manufacturing_in_the_Context_of_Industry_40_A_Review (pristupljeno 24. srpnja 2024.)

[15] Li, L., Ota, K. i M. Dong, „Deep Learning for Smart Industry: Efficient Manufacture Inspection System With Fog Computing“, IEEE Transactions on Industrial Informatics. Dostupno na: https://www.researchgate.net/publication/325516518_Deep_Learning_for_Smart_Industry_Efficient_Manufacture_Inspection_System_With_Fog_Computing (pristupljeno 25. srpnja 2024.)

[16] Nóbrega, L. et al., „Animal monitoring based on IoT technologies“, IEEE Xplore. Dostupno na: https://www.researchgate.net/publication/325635623_Animal_monitoring_based_on

IoT_technologies (pristupljeno 25. srpnja 2024.)

[17] Keswani, B. et al., „Adapting weather conditions based IoT enabled smart irrigation technique in precision agriculture“, Neural Computing and Applications. Dostupno na: https://www.researchgate.net/publication/327896453_Adapting_weather_conditions_based_IoT_enabled_smart_irrigation_technique_in_precision_agriculture_mechanisms (pristupljeno 25. srpnja 2024.)

[18] Karim, F., Fathallah, K. i A. Frihida „Using Cloud IOT for disease prevention in precision agriculture“, Procedia Computer Science. Dostupno na: https://www.researchgate.net/publication/324738503_Using_Cloud_IOT_for_disease_prevention_in_precision_agriculture (pristupljeno 25. srpnja 2024.)

[19] Electronics Media „Internet of Things (IoT) Applications in Agriculture“, Electronics Media, proljeće 2017., https://www.electronicsmedia.info/2017/04/26/internet-things-iot-applications-agriculture/ (pristupljeno 25. srpnja 2024.)

[20] Ivić, S. „Primjena tehnologije Internet stvari (IoT) za udaljeno praćenje mjerenja senzora“, Sveučilište Josipa Jurja Strossmayera u Osijeku. Dostupno na: https://dabar.srce.hr/islandora/object/etfos%3A2389 (pristupljeno 25. srpnja 2024.)

[21] sparkfun „What is an Arduino?“, sparkfun, https://learn.sparkfun.com/tutorials/what-is-an-arduino/all (pristupljeno 26. srpnja 2024.)

[22] opensource.com „What is a Raspberry Pi?“, opensource.com, https://opensource.com/resources/raspberry-pi (pristupljeno 26. srpnja 2024.)

[23] Connectivity Standards Alliance „Zigbee The Full-Stack Solution for All Smart Devices“, connectivity standards alliance, https://csa-iot.org/all-solutions/zigbee/ (pristupljeno 26. srpnja 2024.)

[24] Cloudflare „What is the Internet Protocol?“, Cloudflare, https://www.cloudflare.com/learning/network-layer/internet-protocol/ (pristupljeno 26. srpnja 2024.)

[25] Littlejohn Shinder, D., „Reviewing TCP/IP Basics“ U: Littlejohn Shinder, D. et al., *MCSA/MCSE (Exam 70-291) Study Guide*. Dostupno na: https://www.sciencedirect.com/topics/computer-science/internet-protocol (pristupljeno 28. srpnja 2024.)

[26] Kerner, S. M. „Internet Protocol (IP)“, TechTarget, https://www.techtarget.com/searchunifiedcommunications/definition/Internet-Protocol (pristupljeno 28. srpnja 2024.)

[27] Cerf, V. G. i R. E. Kahn „A Protocol for Packet Network Intercommunication“, Princeton. Dostupno na: https://www.cs.princeton.edu/courses/archive/fall06/cos561/papers/cerf74.pdf (pristupljeno 28. srpnja 2024.)

[28] Kessler, G. C. „An Overview of TCP/IP Protocols and the Internet“, InterNIC. Dostupno na: https://garykessler.net/library/tcpip.html (pristupljeno 28. srpnja 2024.)

[29] Babatunde, O. i O. Al-Debagy „A Comparative Review Of Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6)“, International Journal of Computer Trends and Technology. Dostupno na: https://www.researchgate.net/publication/263856140_A_Comparative_Review_Of_Internet_Protocol_Version_4_IPv4_and_Internet_Protocol_Version_6_IPv6 (pristupljeno 29. srpnja 2024.)

[30] MeridianOutpost „The Five IPv4 Classes - Quick Reference“, MeridianOutpost, https://www.meridianoutpost.com/resources/articles/IP-classes.php (pristupljeno 29. srpnja 2024.)

[31] Ancheta, J. B. S. „An Overview of IP Addressing“. Dostupno na: https://www.researchgate.net/publication/361053333_An_Overview_of_IP_Addressing (pristupljeno 29. srpnja 2024.)

[32] Sisat, S. N. „IP Subnetting“, International Journal of Knowledge Engineering and Soft Data Paradigms. Dostupno na: https://www.researchgate.net/journal/International-Journal-of-Knowledge-Engineering-and-Soft-Data-Paradigms-1755-3229 (prsitupljeno 30. srpnja 2024.)

[33] Huangxu, J. „IP Addresses and Subnetting“ U: Huawei Technologies Co., Ltd. *Data Communications and Network Technologies*. Dostupno na: https://www.researchgate.net/publication/364656568_IP_Addresses_and_Subnetting (pristupljeno 30. srpnja 2024.)

[34] Deering, S. i R. Hinden „Internet Protocol, Version 6 (IPv6) Specification“, RFC 2460. Dostupno na: https://datatracker.ietf.org/doc/html/rfc2460 (pristupljeno 31. srpnja 2024.)

[35] Malone, D. „Observations of IPv6 Addresses“, Passive and Active Network Measurement, 9th International Conference. Dostupno na: https://www.researchgate.net/publication/220850283_Observations_of_IPv6_Addresses (pristupljeno 31. srpnja 2024.)

[36] Cisco support „IPv6 Anycast Address“, Cisco support,

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6_basic/configuration/xe-3se/5700/ip6-anycast-add-xe.html (pristupljeno 31. srpnja 2024.)

[37] Netgate Docs „IPv6 Subnetting“, Netgate Docs, https://docs.netgate.com/pfsense/en/latest/network/ipv6/subnets.html (pristupljeno 31. srpnja 2024.)

[38] Ordabayeva, G. et al., „A Systematic Review of Transition from IPV4 To IPV6“, ICEMIS'20: The 6th International Conference on Engineering & MIS 2020. Dostupno na: https://www.researchgate.net/publication/343820813_A_Systematic_Review_of_Transition_from_IPV4_To_IPV6 (pristupljeno 1. kolovoza 2024.)

[39] Al-Ani, M. i R. A. A. Haddad „IPv4/IPv6 Transition“, International Journal of Engineering Science and Technology. Dostupno na: https://www.researchgate.net/publication/269810379_IPv4IPv6_Transition (pristupljeno 1. kolovoza 2024.)

[40] Ashraf, Z. et al., „Challenges and Mitigation Strategies for Transition from IPv4 Network to Virtualized Next-Generation IPv6 Network“, The International Arab Journal of Information Technology. Dostupno na: https://www.researchgate.net/publication/366258347_Challenges_and_Mitigation_Strategies_for_Transition_from_IPv4_Network_to_Virtualized_Next-Generation_IPv6_Network (pristupljeno 3. kolovoza 2024.)

[41] Shah, J. L. i A. I. Khan „Towards IPv6 Migration and Challenges“. Dostupno na: https://www.researchgate.net/publication/339912713_Towards_IPv6_Migration_and_Challenges (pristupljeno 3. kolovoza 2024.)

[42] Issac, B. „Analysis of IPv6 through Implementation of Transition Technologies and Security Attacks“, International Journal of Business Data Communications and Networking. Dostupno na: https://www.researchgate.net/publication/312317912_Analysis_of_IPv6_through_Implementation_of_Transition_Technologies_and_Security_Attacks (pristupljeno 4. kolovoza 2024.)

[43] Sharma, G. „Implementation of IPv6“. Dostupno na: https://www.academia.edu/89672132/Implementation_of_IPv6?uc-g-sw=32784961 (pristupljeno 4. kolovoza 2024.)

[44] Eclipse foundation „MQTT and CoAP, IoT Protocols“, Eclipse foundation, https://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php#:~:t

ext=Like%20HTTP%2C%20CoAP%20is%20a,used%20extensively%20to%20save%20space (pristupljeno 4. kolovoza 2024.)

[45] Choudhary, S. i G. Meena „Internet of Things: Protocols, Applications and Security Issues“, Procedia Computer Science. Dostupno na: https://www.sciencedirect.com/science/article/pii/S1877050922021019 (pristupljeno 5. kolovoza 2024.)

[46] Bandyopadhyay, D. i J. Sen „Internet of Things: Applications and Challenges in Technology and Standardization“, Wireless Personal Communications. Dostupno na: https://www.researchgate.net/publication/51890865_Internet_of_Things_Applications_and_Challenges_in_Technology_and_Standardization (pristupljeno 5. kolovoza 2024.)

[47] Sinha, R.S, Yiqiao, W. i S. Hwang, „A survey on LPWA technology: LoRa and NB-IoT“, ICT Express. Dostupno na: https://www.researchgate.net/publication/315505158_A_survey_on_LPWA_technology_LoRa_and_NB-IoT (pristupljeno 6. kolovoza 2024.)

[48] Thangavel, D. et al., „Performance evaluation of MQTT and CoAP via a common middleware“, 2014 IEEE Ninth International Conference on Intelligent Sensors. Dostupno na: https://www.researchgate.net/publication/267636202_Performance_evaluation_of_MQTT_and_CoAP_via_a_common_middleware (pristupljeno 6. kolovoza)

[49] Soni, D. i A. Makwana „A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT)“, International Conference on Telecommunication, Power Analysis and Computing Techniques (ICTPACT - 2017). Dostupno na: https://www.researchgate.net/publication/316018571_A_SURVEY_ON_MQTT_A_PROTOCOL_OF_INTERNET_OF_THINGSIOT (pristupljeno 7. kolovoza 2024.)

[50] DevAcademy „CoAP protocol“, https://academy.nordicsemi.com/courses/cellular-iot-fundamentals/lessons/lesson-5-cellular-fundamentals/topic/lesson-5-coap-protocol/ (pristupljeno 8. kolovoza 2024.)

[51] Shelby, Z., Hartke, K. i C. Bormann, „The Constrained Application Protocol (CoAP)“, RFC 7252. Dostupno na: https://datatracker.ietf.org/doc/rfc7252/ (pristupljeno 8. kolovoza 2024.)

[52] Bormann, C., Castellani A. P. i Z. Shelby, „CoAP: An Application Protocol for Billions of Tiny Internet Nodes“, IEEE Internet Computing. Dostupno na: https://www.researchgate.net/publication/254061984_CoAP_An_Application_Protoco

l_for_Billions_of_Tiny_Internet_Nodes (pristupljeno 9. kolovoza 2024.)

[53] Eclipse Californium, „Eclipse Californium™“, https://eclipse.dev/californium/ (prsitupljeno 10. srpnja 2024.)

[54] Apache Maven Project, „What is Maven?“, https://maven.apache.org/what-is-maven.html (pristupljeno 10. srpnja 2024.)

[55] phoenixNAP, „How to Install Maven on Windows“, https://phoenixnap.com/kb/install-maven-windows (pristupljeno 10. srpnja 2024.)

[56] Eclipse Californium GitHub repozitorij, https://github.com/eclipse-californium/californium (pristupljeno 10. srpnja 2024.)

[57] GitHub repozitorij za Eclipse Californium Browser alat, https://github.com/eclipse-californium/californium.tools/tree/main/cf-browser (pristupljeno 11. srpnja 2024.)

[58] Christian Amsüss and aiocoap contributors „{aiocoap}: Python CoAP Library“. Dostupno na: https://aiocoap.readthedocs.io/en/latest/ (pristupljeno 15. srpnja 2024.)

[59] Timescale online sučelje PostgreSQL baze podataka, https://www.timescale.com/ (pristupljeno 21. kolovoza 2024.)

[60] Timescale documentation, „Python quick start“, https://docs.timescale.com/quick-start/latest/python/ (pristupljeno 21. kolovoza 2024.)

[61] Uvicorn dokumentacija, „Uvicorn Settings“, https://www.uvicorn.org/settings/#:~:text=IPv6%20addresses%20are%20supported%2C%20for,%3A%20'127.0.0.1 (pristupljeno 19. srpnja 2024.)

[62] Michigan Ross Business Tech, „Tech 101: Internet of Things“, https://businesstech.bus.umich.edu/uncategorized/tech-101-internet-of-things/ (pristupljeno 23. kolovoza 2024.)

[63] Raspberry Pi, „Raspberry Pi 4“, https://www.raspberrypi.com/products/raspberry-pi-4-model-b/ (pristupljeno 23. kolovoza 2024.)

[64] ClouDNS, „What is an IPv6 address? [Fully explained]“, https://www.cloudns.net/blog/what-is-an-ipv6-address/ (pristupljeno 27. kolovoza 2024.)

[65] ClouDNS, „What is IPv4? Everything you need to know“, https://www.cloudns.net/blog/what-is-ipv4-everything-you-need-to-know/ (pristupljeno 27. kolovoza 2024.)

[66] Ninčević, I., GitHub repozitorij ovog Diplomskog rada, https://github.com/inincevic/DiplomskiRad (pristupljeno 3. rujna 2024.)

[67] Canonical Ubuntu, „Get Ubuntu Server“, https://ubuntu.com/download/server

(pristupljeno 4. rujna 2024.)

[68] VMWare by Broadcom, „VMware Workstation Pro: Now Available Free for Personal Use“, https://blogs.vmware.com/workstation/2024/05/vmware-workstation-pro-now-available-free-for-personal-use.html (pristupljeno 4. rujna 2024.)

[69] Wireshark, „About Wireshark“, https://www.wireshark.org/about.html (pristupljeno 4. rujna 2024.)

Popis slika

Privremeno normalno pisano, kasnije će biti pravilno formatirani

Popis tablica

Sažetak

Uređaji interneta stvari sve su raširenij u svojoj upotrebi, te je potrebno pripremiti cjelokupni Internet za povećanje broja tih uređaja. Ključni problem koji trenutno postoji na globalnom internetu je smanjen broj dostupnih adresa korištenjem IPv4 te je prelazak na IPv6 neizbježan u bliskoj budućnosti. Kako bi uređaji interneta stvari bili spremni za taj prelazak, potrebno je provesti analizu ponašanja tih uređaja i aplikacijskih protokola koje ti uređaji koriste na obje verzije Internet protokola.

Također, kako su uređaji interneta stvari mali uređaji kojima je glavni cilj efikasnost, potrebno je provesti analizu nad protokolima koji se koriste u aplikacijama koje uređaji interneta stvari koriste. U ovome radu provedena je komparativna analiza Constrained Application Protokol-a s Hypertext Transfer Protokol-om, koji je najčešće korišten protokol u aplikacijama koje rade putem mreža. Kako bi bilo moguće komunicirati s uređajima interneta stvari koji se pokreću na CoAP-u, potrebno je postaviti posrednik, čija uloga je prenošenje prometa s jednog protokola na drugi. Posrednik je korišten i prilikom komunikacije s HTTP-om, kako bi se moglo postaviti jednake uvjete testnog okruženja i kako analiza prijenosa podataka ne bi imala više komponenti u jednom obliku komunikacije nego u drugome.

U rezultatima analize ovih protokola vidljiva je razlika između rada mreže korištenjem CoAP-a i HTTP-a, vidljiva je prednost implementacije posrednika, te je jasno vidljiva sličnost u komunikaciji putem IPv4 i IPv6.

Ključne riječi: Uređaji interneta stvari, IoT uređaji, IPv4, IPv6, HTTP, CoAP, posrednik

Abstract

Internet of things devices are becoming more widespread in their use so it has become necessary to prepare the entirety of the Internet for the increased number of these devices. The key issue currently affecting the global internet is the lack of available addresses using the IPv4, making the transition to IPv6 inevitable in the near future. In order for internet of things devices to be ready for this transition, it is necessary to conduct the analysis of those device's behaviour and the analysis of the application protocols used by those devices across both versions of the Internet Protocol.

Since Internet of things devices are small devices which have efficiency as their main

goal, it’s necessary to conduct an analysis of the protocols used by Internet of things applications so that their efficiency can be measured. In this paper, a comparative analysis of the Constrained Application Protocol and the Hypertext Transfer Protocol, which is the most commonly used protocol in web applications, is conducted. In order to communicate with Internet of Things devices that use CoAP for their communication, it's necessary to implement a proxy responsible for transferring the traffic between the protocols. This proxy, although unnecessary, was used even when there was exclusively HTTP communication, this was done so that the test environment would be equal when using both protocols, and so that the analysis of the data transfer wouldn’t have more components in one form of communication than it does in the other.

Results of these analyses show a difference in CoAP and HTTP communications, they show an advantage in implementing a proxy, and they clearly show a similarity in IPv4 and IPv6 communication.

Keywords: Internet of Things, IoT, Constrained Application Protocol, CoAP, Hypertext Transfer Protocol, HTTP, IPv4, IPv6, proxy