UNIVERZITET SINGIDUNUM
BEOGRAD
DEPARTMAN ZA POSLEDIPLOMSKE STUDIJE

DOKTORSKA DISERTACIJA

KOMPARATIVNA ANALIZA UTICAJA SISTEMA ZA SERIJALIZACIJU NA ENERGETSKU EFIKASNOST KLIJENTSKIH MOBILNIH UREĐAJA

MENTOR: Dr. Vladimir Matić

KANDIDAT: Predrag Sibinović

BEOGRAD, 2019.год.

Apstrakt

Zahvaljujući društvenim mrežama i brzom protoku vesti mobilni uređaji postali su neraskidivi deo svakodnevice. Međutim, pre samo desetak godina mobilne aplikacije bile su namenjene isključivo specijalnim potrebama. Operativne funkcionalnosti tih uređaja i aplikacija bile su daleko ispod mogućnosti postojećih. Preduslov za ovakav razvoj mobilnih uređaja stvorile su naučne discipline koje naizgled nisu povezane ali su svoju sinergiju evaluirale kroz razvoj mobilnih uređaja i njihovih aplikacija. Prvenstveno se misli na telekomunikacije, elektroniku, softversko inženjerstvo i internet tehnologije.

Skokoviti razvoji ovih oblasti su omogućili da mobilni uređaji dostignu funkcionalnosti klasičnih PC-jeva, ali i da u budućnosti očekujemo dalje napretke. Jedan od ograničavajućih faktora jeste autonomnost i energetska nezavisnost uređaja. Drugim rečima - sa postojećom tehnologijom baterija se ne može produžiti neka značajna autonomnost uređaja. Samo ekonomsko tržište diktira potrebu za brzim procesorima, manjim dimenzijama uređaja (debljina) ali i što većim dimenzijama ekrana. Veoma bitna je i funkcionalnost mobilnog interneta bez koga mobilni uređaji ne bi imali svoju današnju funkciju. Jedan od ciljeva proizvođača mobilnih uređaja jeste kako da zadrže sve ove funkcionalnosti a da povećaju energetsku efikasnost a samim time i autonomiju rada uređaja. Povećavanje energetske efikasnosti uređaja moguće je postići putem hardverskih inovacija: proizvodnje komponenata sa manjim gubicima, procesorima koji bi radili na manjim naponima (manjim od 3.3 v) itd. Takođe, modemi za komunikaciju na baznim stanicama koji obezbeđuju mobilni internet uređaju, kreiraju se tako da imaju što veće iskorišćenje. Analizom rada mobilnih uređaja ustanovljeno je da je najveći potrošač energetskih resursa baterije ekran a zatim slede internet servisi. Pored hardverskih optimizacija kojima se može postići veća efikasnost, moguće je programskim putem omogućiti znatne uštede u potrošnji.

Upotreba internet servisa odgovorna je za malu autonomnost mobilnih uređaja pa se može doći do zaključka da je ovo pravo mesto za optimizaciju. Naravno, internet servisi na mobilnom uređaju su širok pojam. Jedan od odgovora se krije u trenutnom konceptu mobilnih servisa. Većina aplikacija na mobilnom uređaju funkcioniše po principu *master slave* komunikacije sa nekim od mnogobrojnih servera. Drugim rečima: bilo da se radi o običnom čitanju poruka, aplikaciji za vremensku prognozu, nekoj socijalnoj mreži ili *online* strategiji, aplikacija i server razmenjuju značajne količine podataka u oba pravca. Aplikacija šalje i prima složene strukture podataka. U

eri objektno orijentisanih programskih jezika ti podaci se mogu tretirati kao liste i nizovi instanci kasa ili objekti. Sam proces pakovanja i raspakivanja složenih struktura podataka u nizove bajtova pogodnih za slanje naziva se serijalizacija podataka.

Ovim radom su dati rezultati energetske efikasnosti prenosa serijalizovanih podataka između mobilnih uređaja i servera. Realizovano je test okruženje u kome je najpre merena potrošnja električne energije tokom komunikacije sa serverom. Rezultati merenja su upotrebljeni za razvoj matematičkog modela u kome su kasnije kao parametri uvedene funkcije koje opisuju vreme potrebno za serijalizaciju i veličinu serijalizovanog zapisa. Podaci dobijeni ovim istraživanjem primenjeni su za definisanje eksperimentalnog okruženja API – *CBSaver*, gde se na osnovu matematičkih modela bira optimalan način serijalizacije i kompresije u realnom vremenu. Rezultati dobijeni tokom testa API – *CBSaver-a* ukazuju na smer u kome treba ići u cilju poboljšanje energetske efikasnosti.

Abstrakt

We are witnessing that mobile devices have become an inseparable part of our everyday life. First of all, thanks to social networks and fast news flow. However, in the past just about a decade ago, mobile applications were designed exclusively for special purposes. Also, the operational functionality of these devices and applications was far below the existing ones. Such a development of mobile devices led to the development of scientific disciplines that did not seem to be in a relationship, but evaluated their synergy through the development of mobile devices and their applications. It is primarily meant for telecommunications, electronics, software engineering and Internet technology.

The high speed developments in these areas have enabled mobile devices to achieve the functionality of classic PCs, but also to continue to make progress in the future. One of the limiting factors is the autonomy and energy independence of the device. In other words, with the existing battery technology, some significant autonomy of the device cannot be achieved. The economic market dictates the need for fast processors, smaller dimensions of the device (thickness), but also the larger dimensions of the screen. Also, there is a very important functionality of the mobile cell Internet without which mobile devices would not be so useable. One of the goals of the mobile device maker is how to keep all these functionalities and increase the energy efficiency, and energy autonomy of the device. Increasing the energy efficiency of the device can be achieved through hardware and software innovations.

Very well know is that the usage of internet is one of the biggest power drain reasons, so it lead to the conclusion that this is the right place for optimization. One of the answers is covered in the current concept of mobile services. Most applications on the mobile device function according to the master slave communication principle with some of the many servers. In other words, the application and server exchange significant amounts of data in both directions. Whether it's a simple reading of messages or an application for weather forecasts, some social network or an on-line strategy. The application sends and receives complex data structures. Currently in the era of object-oriented programming languages, all these data can be treated as lists and series of class instances, objects

The process of packing and unloading of complex data structures into a series of bytes suitable for sending is called data serialisation.

This paper deals with research results into energy-efficient transmission of serialised data between servers and mobile devices. A test environment was created in which the research authors primarily measured the electricity consumption during the communication with a server. The measurement results have been used to create a mathematical model which was later introduced with functions, serving as parameters, which depict the time necessary for serialisation and the size of a serialised file. The data collected through this research have been used for an experimental API-CBSaver, which on the basis of mathematical models chooses the most favourable manner of serialisation and compression in real time. The results obtained through the API-CBSaver test indicate the direction which one should take for the purposes of improving energy efficiency.

Sadržaj

Spisak slika

Spisak tabela

Spisak skraćenica

1 I Deo

1.1 Uvod

Oblast mobilnih tehnologija u informatici i računarstvu trenutno se najbrže razvija. Zbog mnogobrojnih tržišnih zahteva i operacija koje obavljaju, mobini uređaji bivaju limitirani energetskom autonomijom. Shodno tome postoji mnogo radova na temu povećanja autonomnosti energetske efikasnosti. Gledano sa stanovišta pristupa autora postoje dve grupacije radova. Prva se bavi hardverskim rešenjima uređaja dok druga grupa autora pokušava da softverskim optimizacijama postojećih metoda ostvari značajne uštede električne energije tokom rada uređaja. Istraživanja su pokazala da je internet komunikacija kao i aplikacije koje se oslanjaju na razmenu podataka sa serverima jedan od vodećih uzročnika potrošnje električne energije.

Sama upotreba mobilnih telefona i tableta prevazišla je namenu koju je imala početkom veka, a to je prost prenos govora i pisanih poruka. Mobilni telefon je za osamnaest godina postao uređaj koji je neraskidivi deo naše svakodnevice, jer gotovo da ne postoji oblast života za koju ne postoji posebna aplikacija. Prema [1] statistici iz 2014. godine prosek instaliranih aplikacija po jednom korisniku iznosio je 26,2. Treba napomenuti da skoro sve aplikacije imaju svoje veb servere sa kojima razmenjuju podatke. Sa serverom se razmenjuju podaci složene strukture a pritom su platforme različite, potom se podaci serijalizuju i kao takvi transportuju. U većini slučajeva koristi se REST oblik komunikacije a podaci se serijalizaciju JSON formatom. Pored JSON-a koji je opšteprihvaćen postoje desetine drugih sistema za serijalizaciju. Ova doktorska teza bavi se upravo uticajem pojedinih sistema za serijalizaciju na potrošnju baterije na mobilnom uređaju.

U nauci postoje radovi na temu serijalizacije i upoređivanja performansi ovih sistema. Međutim, nema informacija o tome kako sam proces koji se odigrava na serveru utiče na potrošnji električne energije mobilnog uređaja. Tokom ovog istraživanja se mora pripremiti hardversko i softversko okruženje kojim se može izmeriti potrošnja električne energije na modemu mobilnog uređaja

tokom zahteva za prijem podatka. Pored toga podaci se moraju podeliti na segmente tokom REST zahteva i lokalizovati delovi na koje indirektno utiču performanse proces serijalizacije podataka koji se odvija na serveru. Nakon toga moguće je izvršiti uporedni test više različitih tipova serijalizacije podataka u odnosu na potrošnju električne energije tokom zahteva.

Analizom postojeće literature možemo izdvojiti sledeće osobine koje su bitne za ovaj naučno istraživački rad. Postoje mnoge metode za snimanje potrošnje električne energije na mobilnom uređaju ali ih možemo podeliti u dve grupe, one koje koriste gotova programska rešenja za snimanje poput *PowerTutora, Nokia Profing Tools, LittleEye* i drugog pristupa koji podrazumeva postavljanje eksternih senzora na sam uređaj i pojedine komponente od interesa. Tokom transporta podataka vidi se da postoje pojedina elektroenergetska staja na koja proces serijalizacije može indirektno da utiče; a tim uticajem se utiče na potrošnju energije. Takođe analizirani rezultati merenja će nam poslužiti za kontrolu i verifikaciju dobijenih rezultata (da li su rezultati merenja u okviru vrednosti koje su i drugi autori dobili).

Analizirani radovi daju nam motivaciju za dalji rad jer postoji veliki broj radova o povećavanju energetske efikasnosti. Oni su bazirani na softverskim modifikacijama u određenim segmentima komunikacije mobilnih uređaja sa serverom, što opravdava uticaj istraživanja serijalizacije na energetski efikasan prenos podataka od servera do mobilnog uređaja. Takođe, kompresija podataka dokazano utiče i menja karakteristiku prenosa podataka jer smanjuje veličinu podataka.

Podaci koji bi se dobili ovim analizama mogli se iskoristiti za kreiranje novih hibridnih tipova serijalizacije koji bi bili energetski efikasniji od postojećih ili da ukazu na drugačije pristupe tokom projektovanja savremenih mobilnih aplikacija.

Rezultati ovog naučno istraživačkog rada su sažeti i publikovani u naučnom časopisu na SCI listi. *Technical Gazette 26, 3(2018), 955-962* pod naslovom *An Analysis of Energy Efficient Data Transfer Between Mobile Device and Dedicated Server* .

1.2 Hipoteza

Metode koje se koriste u naučnim istraživanjima tokom ovog rada mogu se kategorisati u sledeće delove:

- Analiza postojećih radova drugih autora na: potrošnju električne energije i energetsku optimizaciju
- Teorijska razmatranja iz oblasti prenosa podataka i serijalizacije istih
- Laboratorijska merenja uticaja serijalizacije na potrošnju baterije na realnom uređaju
- Korišćenje metoda za uspostavljanje korelacije između serijalizacije i potrošnje električne energije
- Statističke metode u cilju poboljšanja rezultata merenja

Naučni doprinos ovog rada ogleda se u istraživanju oblasti koja ima veliku primenu u savremenom softverskom inženjerstvu i mobilnim aplikacijama a kojom su se uglavnom bavile velike komercijalne kompanije poput *Google–a, Facebook-a, Microsoft*-a . Ovo je prvi od radova koji na precizan način laboratorijskim merenjima, teorijskom analizom ali i praktičnim primenama pokušava da utvrdi zavisnost potrošnje električne energije na mobilnim uređajima u zavisnosti od tipova serijalizacije, i da ukaže na nove smernice za što optimalnije korišćenje alata za serijalizaciju. Praktični doprinos leži u raščlanjenju sistema za serijalizaciju i deserijalizaciju koji se svakodnevno primenjuju u programiraju, ali i u ukazivanju na posledice po energetsku efikasnost koje može da proizvede neoptimalna upotreba.

1.3 Struktura rada

Rad se sastoji iz sedam poglavlja. Prvo poglavlje predstavlja opšti uvod o problemu kojim se rad bavi, kao i definisanje hipoteze i metodologije istraživanja. U drugom delu su analizirana postojeća dostignuća i radovi drugih autora, prvenstveno na oblastima optimizacije potrošnje električne energije i eksperimentalnog merenja potrošnje na mobilnim uređajima. Treće poglavlje se bavi

analizom postojećih sistema za serijalizaciju, podeli i načinu implementacije. Uloga sistema za serijalizaciju u mobilnim sistemima današnjice obrađena je u četvrtom poglavlju. Ono se takođe bavi analizom sistema koji su u masovnoj upotrebi i opštom slikom koja je prisutna u tehnologiji posmatrana kroz prizmu sistema za serijalizaciju. Nakon ovih razmatranja sledi peto poglavlje koje se sastoji od postavke opreme za merenje. Definisanje scenarija eksperimenata i pisanje programa za testiranje. Analiza REST profila potrošnje, dobijanje opšteg matematičkog modela u koji se postepeno uključuju matematički modeli svakog od testiranih sistema za serijalizaciju. Najbitniji deo ovog poglavlja jeste simulator koji može na osnovu varijabilnih parametara tokom analize uporediti efikasnost pojedinih sistema i scenarija. Iz zaključka dobijenog u petom poglavlju tokom šestog dela rada prezentuje se energetski efikasni sistem za serijalizaciju *CBSaver*, a takođe i vrši implementacija u praksi. Prikazuju se uporedni rezultati *CBSaver* API i klasičnih metoda koje se trenutno primenjuju. Sedmom poglavlju pripada sveopšti zaključak i komentar rezultata dobijenih tokom celokupnog istraživanja. Nakon njega slede pregled literature, spisak slika , skraćenica i tabela. Pored rada u pisanom obliku uz rad se prilažu i izvorni kodovi programa koji su korišćeni tokom rada.

2 II Deo

2.1 Analiza postojećih metoda za smanjenje i merenje potrošnje električne energije

Ovo poglavlje rada se odnosi na teorijsku analizu pređašnjih radova i postojeće literature kroz analizu povezanih oblasti. Prva oblast su softverske metode za smanjenje potrošnje baterije putem optimizacije sistema i aplikacija u smeru veće ekonomičnosti. Druga oblast se odnosi na radove koji se bave analizom i metodama za merenje potrošnje. Oba segmenta su jako bitna za ovu ovaj rad. Analiziranjem radova prve oblasti dolazimo do bitnih saznanja o ponašanju komponenata mobilnih sistema gledanih kroz prizmu energetske efikasnosti i primenljivih ideja za poboljšanje performansi. Shodno tome većina autora čije smo radove analizirali se odlučila za kombinaciju više sistema komunikacije i promenu vremenskih intervala rada modema za mobilni mrežni prenos podataka i WiFi modema, kako bi se smanjila potrošnja. U pojedinim radovima autori su se bavili izmenama postojećih protokola u cilju smanjene aktivnosti modema i procesora. Pokazana je i potrošnja u trenucima komunikacije uređaja putem mobilnog interneta. Utvrđena zavisnost i matematički modeli utroška energije u zavisnosti od veličine podataka.

Naučni radovi pomenute druge oblasti koji se bave eksperimentalnim merenjem potrošnje energije pojedinih komponenata pružaju nam uvid u metode i opremu koje su nam bile potrebne za eksperimentalno merenje tokom istraživanja. Na osnovu tih podataka dobijeni su materijali i ideje za nastavak analize, jer je polazna ideja utvrđivanje zavisnost potrošnje od tipa i načina serijlizacije podataka i izolovanje tog uticaj od drugih faktora.

2.1.1 Operativni sistemi i potrošnja

Pre nego što se krene sa analizom performansi samih kompetentan mobilnih uređaja treba obratiti pažnju na koncepte energetske svesti samih operativnih sistema koji su predmet ovog rada. Početkom devedesetih godina za potrebe prvih prenosivih računara javila se ideja o sistemima koji su svesni svoje potrošnje radi veće energetske autonomije [2]. Početkom novog milenijuma neki autori su postavili koncept da je energetska nezavisnost mobilnih uređaja resurs prvog reda i da mu treba prilagoditi ostale resurse [3]. U narednom periodu ovaj koncept je došao u drugi plan na

račun performansi komponenata, sve do masovne pojave pametnih mobilnih telefona sa zahtevnim operativnim sistemima i dodatnim komponentama poput senzora bežičnog interneta i velikih ekrana. A. Roy i drugi u radu [4] dali su primer *Cinder* operativnog sistema za mobilne uređaje koji je pratio rad aplikacija i omogućavao da se pojedine radnje inhibiraju kako bi se zadržala dragocena energija za radnje većeg prioriteta baziran je bio na Hi Star operativnom sistemu na kome su uvedene dve nove paradigme: *slavine i rezervoari*. Cilj je bio da se pojedinim aplikacijama ograniči potrošnja energije tako da kada one upadnu u agresivni režim u pogledu potrošnje mogu da dobiju samo onoliko energije koliki je protok slavine na koju su vezani. Pored *Cindera* pomenućemo i druge operativne sisteme slične namene. *CondOS* [5], *NemesisOS* koji slično kao i kod *Cindera* dodeljuje energiju na osnovu njihove korisnosti i troškova ali razmatrajući trenutno opšte stanje sistema. [6]. Potpuno drugačiji pristup ima *ErdOS* koji je zamišljen kao podrška *Android OS*-u. [7] Taj pristup za razliku od svojih prethodnika nije koristio unapred definisane krute algoritme kao prethodno pomenuti OS, već je učio o navikama korisnika i na osnovu tih obrazaca raspolagao energijom komponenata.

Danas nam je opštepoznato da *Android* i *IOS* operativni sistemi inhibiraju pozadinske aplikacije kada baterija padne na određeni nivo.

2.1.2 Potrošnja komponenti

U [8] Carroll i Aaron su dali sveobuhvatnu analizu potrošnje komponenta na mobilnom uređaju. Oni su na otvorenoj platformi *Neo Freeruner* analizirali rad svake komponente ponaosob tako što su postavili spoljni merač. Otvorena platforma, kako hardverska tako i softverska, ovog telefona im je to omogućila. Bile su im dostupne šeme hardverskih komponenti pa su mogli da postave spoljne merače na svaku od komponenta koju su merili. Analizama su došli do zaključka da se samo energija potrošena na osvetljenje ekrana kreće u rasponu od 7.8 mW pa do 414 mW. Oni su svoj rad bazirali na *Android* operativnom sistemu. Na osnovu njihovog eksperimenta zaključeno je takođe da RAM memorija nije veliki potrošač ali da u nekim okolnostima može da bude ispred samog procesora po potrošnji. Još jedan od zaključaka je da se sve komponente koje nisu trenutno u funkciji trebaju isključiti i odvojiti od napajanja. Autori predvođeni G. P. Perrucci-jem [9] njihovim eksperimentalni rezultatima prijavljuju veću potrošnju energije u 3G mrežama za

tekstualne poruke (SMS) i govorne usluge u poređenju sa 2G mrežama. Potrošnja energije slanja tekstualnih poruka povećava se linearno sa dužinom poruke, dok jačina signala jasno utiče na vreme potrebno za prenos poruke u oba tipa mreža

2.1.3 Neaktivnost i potrošnja

Performanse *PowerSaveMode* su analizirali X. Chuah i drugi u radu [10]. Ispitivali su performanse i odredili maksimalni interval za *listening* tako da maksimizuju vreme koje uređaj provede u stanju mirovanja tj. (*sleep state*) i na taj način su smanjili potrošnju energije ali tako da zadovolje performanse kašnjenja podataka u toku komunikacije. Ovo je jedna od metoda koja na bazi promene vremenskih intervala u mašini stanja komunikacije protokola IEEE 802.11 između AP (Pristupne tačke *Access Point*) i mobilnog uređaja, povećava vreme kada uređaj ostaje u pasivnom modu ali tako da ne utiče bitno na brzinu protokola. Autori su u radu [10] pokazali da WiFi za skeniranje mreže koristi 1.4260 W a za prosečnu komunikaciju samo 0.89 W. U pasivnom modu njegova potrošnja je 0.256 W. Mobilni modem troši 2-4 mAh u pasivnom stanju ali u aktivnom 250 – 300 mAh. Potrošnja BT (Bluetooth) modula se ogleda u 0.12w u trenutku skeniranja i 0.01W u pasivnom stanju.

2.1.4 Bežične komunikacije

Moderni mobilni uređaji se oslanjaju na bežične komunikacije za prenos glasa i SMS-a, i skoro svi servisi su zavisni od konekcije na internet. Mobilni mrežni prenos podataka postao je sve češći kao i upotreba onlajn aplikacija socijalnih medija, pa mreža ima tendenciju rasta u budućnosti. Već sada, veliki deo potrošnje čini bežična komunikacija. Tipični set bežičnih interfejsa na mobilnom uređaju su *WIFI*, *BlueTooth*, i mobilni mrežni prenos podataka (GPRS, 2G, 3G, 4g). Postoje različiti načini za smanjenje potrošnje energije tokom rada bežične komunikacije. Oni se ogledaju u optimizaciji jednog interfejsa pojedinačno, kombinaciji više interfejsa sa ciljem da im se podele zadaci gde oni imaju bolje performanse, zatim vremenska ograničenja i ograničenja veličine podataka.

2.1.4.1 Bluetooth

Bluetooth (IEEE 802.15.1) definiše fizički sloj i MAC (Medium Access Control) podsloj za bežičnu komunikaciju i najčešće se koristi za povezivanje i prenos podataka između uređaja na kraćem rastojanju. Bluetooth tehnologija je od značaja kada se prenose informacije između dva ili više uređaja koji su blizu jedan drugom, u slučaju kada brzina prenose nije od velike važnosti, kao što je slučaj sa telefonima, štampačima ili slušalicama (na rastojanjima od 10 cm do 10 m) . Prvobitna ideja za razvoj Bluetooth tehnologije je koncipirana 1994. godine kada je kompanija Ericsson Mobile Communications počela da proučava sisteme sa niskom potrošnjom energije koji mogu zameniti kablove pri povezivanju mobilnih uređaja i dodatne opreme na kraćim rastojanjima. Mobilna telefonija je jedno od prvih tržišta gde je počela da se upotrebljava Bluetooth tehnologija, najviše zbog toga što se Bluetooth čipovi na lak način mogu dodati u mobilni uređaj. Međutim, Bluetooth tehnologija je našla svoju primenu i kod drugih uređaja, kao što su miš, štampači, PC, Hi-Fi sistemi, ali i kod aplikacija razvijenih za automobile. Uređaj koji komunicira putem Bluetooth tehnologije može da radi ili u master ili u slave režimu rada. Maksimalan broj uređaja koji može biti povezan ovom tehnologijom je osam – sedam aktivnih slave uređaja i jedan master uređaj, i oni obrazuju tzv. Piconet, koji predstavlja najjednostavniju konfiguraciju Bluetooth mreže. Ovakve konfiguracije se mogu međusobno povezati obrazujući tzv. Scatternet. Dva slave uređaja ne mogu da komuniciraju direktno jedan sa drugim u bilo kom trenutku osim u fazi otkrivanja. Raspodela kanala i uspostavljanje komunikacije su u nadležnosti master uređaja. Master i slave uređaji mogu da zamene uloge u slučaju kada jedan uređaj treba da se poveže u više od jedne Piconet mreže. Komunikacija upotrebom Bluetooth tehnologija nema nikakvu zavisnost od IP adrese korisnika. Ovakvim dizajnom se olakšava raspoređivanje uređaja zbog toga što ne mora da se vodi računa o raspodeli adresa, podrazumevanom ruteru, itd.

Wi-Fi (IEEE 802.11) protokol definiše fizički sloj i MAC (Medium Access Control) podsloj za bežičnu komunikaciju i uglavnom se koristi za povezivanje više računara, kao proširenje ili zamena za žičnu mrežu - LAN (Local Area Network). Standard IEEE 802.11 je poznatiji pod nazivom Wi-Fi. On omogućava bržu konekciju, veću udaljenost od bazne stanice i bolju sigurnost u odnosu na Bluetooth. Mobilnost uređaja je jedna od bitnijih stavki u bežičnim mrežama. Klijent može da menja svoju lokaciju slobodno, bilo u okviru jednog segmenta mreže, bilo između

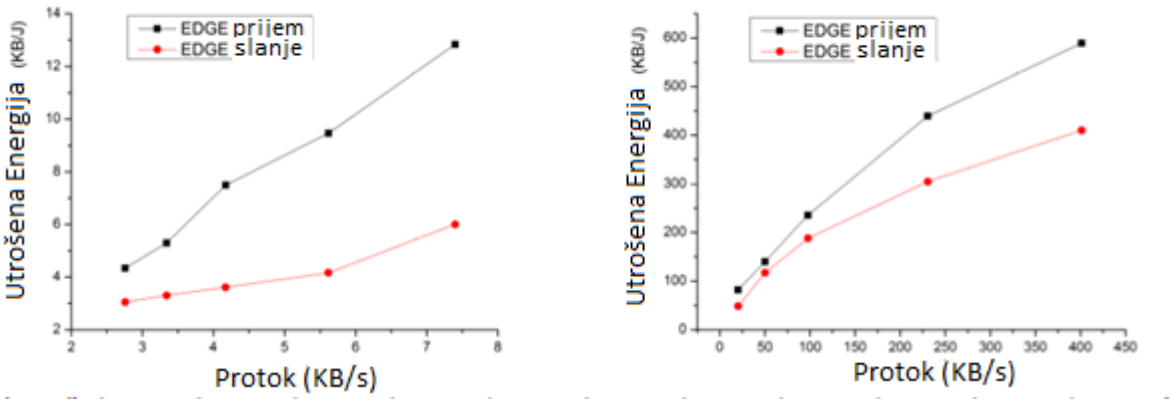
različitih segmenata mreže. IEEE je 1997. godine odobrio standard za bežične lokalne mreže - WLAN (Wireless Local Area Network) pod nazivom 802.11 kojim se specificiraju karakteristike uređaja sa brzinom prenosa podataka od 1 do 2 Mb/s. Nakon zadovoljavajućih rezultata dobijenih od strane kompanija Lucent Technologies i Harris Semiconductors, IEEE je napravio proširenje standarda sa boljim performansama pod nazivom IEEE 802.11b, koji dodatno podržava brzinu protoka od 5,5 i 11 Mb/s: najveći broj trenutno dostupnih uređaja na tržištu su bazirani na upotrebi ove tehnologije. Standardom 802.11b specificirane su neke od modifikacija u kodiranju. Tokom 1997. godine IEEE je takođe objavio specifikaciju standarda 802.11a. Takva specifikacija se i dalje odnosila na MAC i fizički sloj sa radom u opsegu od 5 GHz. Tokom 2003. godine IEEE je odobrio upotrebu standarda 802.11g, najnovije verzije 802.11 standarda. Standard 802.11g obezbeđuje iste performanse kao i 802.11a, sa tom razlikom što 74 koristi opseg od 2,4 GHz. Kompatibilnost sa 802.11b uređajima je zagarantovana. Wi-Fi bežična mreža je zasnovana na ćelijskoj arhitekturi, a osnovna ćelija se naziva osnovni servisni skup (BSS - Basic Service Set). Osnovni servisni skup predstavlja skup mobilnih ili fiksnih stanica. Ukoliko se stanica pomeri tako da se nalazi izvan svog osnovnog servisnog skupa, ona više nije u mogućnosti da direktno komunicira sa ostalim članovima tog skupa. BSS je osnova za nezavisni osnovni servisni skup (IBSS - Independent Basic Service Set) i za prošireni servisni skup (ESS – Extended Service Set). IBSS predstavlja najjednostavniju konfiguraciju u kojoj stanice mogu da komuniciraju direktno, bez upotrebe pristupne tačke - AP (Access Point). Više osnovnih servisnih skupova može da obrazuje proširenu formu mreže. Komponenta koja se koristi za povezivanje više osnovnih servisnih skupova u tom slučaju se naziva distributivni sistem (DS). Distributivni sistem sa pristupnom tačkom omogućava kreiranje mreže proširenog servisnog skupa proizvoljne veličine i složenosti. Raspoloživi propusni opseg je podeljen na 14 delimično preklapajućih kanala, a širina svakog od njih iznosi 22 MHz. Jedanaest ovih kanala je dostupno u Sjedinjenim Američkim Državama, 13 je dostupno u Evropi, a samo jedan je dostupan u Japanu. Svi uređaji u okviru istog osnovnog servisnog skupa koriste isti kanal.

3G (*third generation*) je treća generacija mobilne telekomunikacione tehnologije. To je skup standarda za mobilne uređaje i mobilne servise propisane specifikacijama od strane *International Mobile Telecommunications-2000 (IMT-2000)* u okviru Međunarodne Telekomunikacione unije (*International Telecommunication Union*). 3G tehnologija je našla primenu u bežičnoj telefoniji,

mobilnom internet pristupu, video pozivima i mobilnoj televiziji. Određeni broj telekomunikacionih kompanija promovišu bežični mobilni internet pod nazivom 3G, što znači da su date usluge obezbeđene kroz 3G bežičnu mrežu. Takve usluge dostupne su korisnicima najčešće na nivou mesečne pretplate. 3G tehnologija podržava brzine prenosa podataka od 384 Kb/s do 2 Mb/s, iako se u komercijalnoj upotrebi najčešće koristi brzina prenosa podataka od oko 100 Kb/s . 4G (*fourth generation*) je četvrta generacija mobilne komunikacione tehnologije i predstavlja naslednicu 3G tehnologije. Ova tehnologija obezbeđuje veće brzine protoka podataka u odnosu na 3G . Početak upotrebe 4G tehnologije seže u 2007. godinu kada je u Južnoj Koreji predstavljen WiMAX standard ove tehnologije. U martu 2008. godine, sektor za komunikacije *International Telecommunications Union-Radio* (ITU-R) je dao specifikaciju propisa koje mora da zadovoljava 4G standard pod nazivom *International Mobile Telecommunications Advanced (IMT-Advanced).* Propisi u pogledu brzine prenosa podataka postavljaju se na oko 100 Mb/s za komunikaciju u stanju visoke mobilnosti (na primer iz voza ili automobila) i 1 Gb/s za stanja niske mobilnosti (na primer u toku šetnje ili mirovanja). Pokrivenost signalom kod 4G mreže je jedan od značajnijih problema, čak i u razvijenim zemljama. Zbog toga su korisnici ove tehnologije često ograničeni na urbane sredine i glavne putne pravce. Bazne stanice 4G mreže su uglavnom locirane u gusto naseljenim područjima, a izvan tih oblasti prenos podataka se može ostvariti upotrebom 3G mreže.

Jako bitna studija na osnovu koje smo zasnovali polazne pretpostavke za dalji naučno istraživački rad jeste rad [11]. Autori su snimali profil potrošnje tokom prenosa podataka preko mobilnog interneta i prikazali zanimljivu pojavu repne potrošnje, koja se ne javlja kod istog prenosa podatka putem IEE801.11 prenosa. Ovaj efekat je mnogo manji u GPRS nego u 3G mrežama. Pored toga, oni su aproksimovali matematički model prenosa podataka za svaki od tri tipa mreže gde su precenili da je repni efekat srazmeran veličini prenesenih podataka. Na osnovu ovih istraživanja oni su dali predlog *TailEnder* protokola koji ima ulogu da rasporedom zahteva za prenos podataka u realnom vremenu, započne sleći zahtev pre nego što se repni efekat prethodnog zahteva završi. Na taj način su izgubili repnu energiju ali narušili performanse sistema. Tako da su svoj princip primenili na aplikacijama koje su tolerantne na mala vremenska kašnjenja. Na ovaj način su pod određenim okolnostima autori postigli i do 40% uštede.

Na osnovu rada Xia i Fenga [12] možemo dobiti jako korisne informacije o profilu potrošnje u toku razmene podataka pute *WiFi* modema i *Edge* modela bežičnog interneta .



Slika 1 Uticaj intenziteta protoka podataka na potrošnju električne energije ***preuzeto do*** *[9]*

Na osnovu njihove analize vidi se da je kod *Edge* mobilne telefonije nekoliko puta veća potrošnja prilikom slanja podatka nego kod prijema. Takođe, što je propusni opseg veći to je i efikasnost veća. Ovi testovi su rađeni na *Android* OS I uređaju ZTE V880.

Pojedini autori su se opredelili da urade poboljšanja na samom protokol steku to jest u samoj strukturi *MAC* protokola [13]. Pokazalo se da se optimizacijama može uštedeti energija na svakom od slojeva. Zapravo, važno je naglasiti da optimizacija za IEEE 802.11 standarde nije obično primenjiva za mobilne mreže zbog različitih načina i karakteristika snage. Ipak, ovo istraživanje se uglavnom fokusira na rešenja za energetsko efikasno upravljanje IEEE 802.11 interfejsima koji se mogu implementirati na nivou softvera.

Autori projekta *CoolSpots* [14] krenuli su od pretpostavke da je *WLAN* mnogo brzi od *Bluetooth* protokola ali i da je potrošnja El energije u stand by stanju mnogo manja kod *Blutootha* nego kod *WLAN*-a. Na ovaj način predlog autora je da se *Bluetooth* koristi za uključivanje *WLAN*-a kada je to potrebno i njegovo isključivanje kada to nije. Kod eksperimentalnog sistema autori tvrde da su došli do smanjenja električne energije za 50% u odnosu na klasični *WLAN* sistem.

Na ovom mesto je svrsishodno analizirati metode optimizacije drugih autora vezane za bežični prenos podataka Ono što je za naš istraživački rad bitno je da se u pojedinim radovima uočavaju pokušaji da se određena energetska stanja u kojima se nalazi mrežni modem a nisu esencijalna za prenos podatka eliminišu metodama optimalnog planiranja akcija, metodama gašenja modema u trenucima neaktivnosti, izmenama u pojedinim vremenskim rasporedima aktivnosti samog

protokola. Ovo je bitno jer polazimo sa pretpostavkom da sam proces serijalizacije ima dvojaki uticaj na rad modema: preko veličine podataka koje kreira i vremenom koje mu je potrebno da serijalizuje podatke. Ovim veličinama utiče se na energetska stanja samog modema. Ovo će biti dalje razmatrano u poglavlju IV.

2.1.5 Kompresija podataka za prenos

Kompresija podataka ili kompresija je proces smanjenja fizičkog prostora potrebnog za čuvanje podataka korišćenjem određenih metoda za snimanje podataka. Osnovi način za kompresiju podataka se ogleda u tome da se u okviru jednog zapisa traže sekvence koje se ponavljaju, zatim se pravi mapa mesta na kojima se ta sekvenca ponavlja. Na taj način je moguće značajno smanjiti potreban prostor za skladištenje u zavisnosti od strukture i tipa datoteke, zapisa.

Kompresijom podataka se naziva njihovo preslikavanje iz jednog načina predstavljanja (jedne grupe simbola) u drugi (drugu grupu simbola) tako da se dobije koncizniji niz simbola.

Dve osnovne vrste kompresije: kompresija bez gubitaka podataka – ekspanzija komprimovanih podataka daje podatke identične originalnim (pogodno za proizvoljne podatke): Huffman-ovi kodovi i LZW algoritam i kompresija sa gubitkom dela podataka – ekspanzija ne uspeva da tačno reprodukuje originalne podatke, ali je stepen kompresije veliki (pogodno za podatke zvuka i slike): JPEG algoritam, MPEG algoritam. Tehnike kompresije se koriste za: prenos podataka preko sporih (npr. modemskih) veza i arhiviranje podataka

2.1.5.1 Gzip

Gzip je format datoteke i softverska aplikacija koja se koristi za kompresiju datoteka i dekompresiju. Program su kreirali Jean – loup Gailli i Mark Adler kao besplatni softver za program kompresije koji se koristi u ranim Unix sistemima. Verzija 0.1 je prvi put objavljena 31. oktobra 1992. godine, a verzija 1.0 sledi u februaru 1993. godine. *Gzip* se bazira na DEFLATE algoritmu, što je kombinacija LZ77 i Huffman kodiranja. DEFLATE je bio zamišljen kao zamena za LZV i druge algoritme za kompresiju podataka obrađenih patentom, koji su u to vreme ograničili upotrebljivost kompresa i drugih popularnih programa za arhiviranje. *Gzip* je format koji se

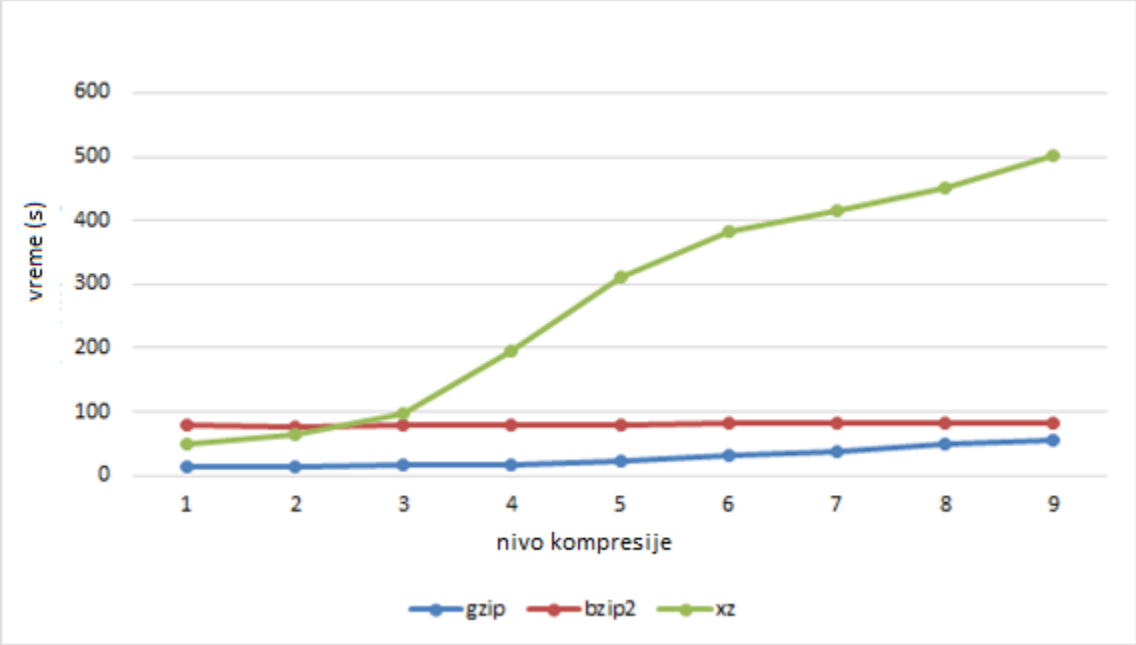
najčešće koristi kako bi smanjio format fajlova , ali i ubrzao prenos podataka time što će smanjiti njegovu veličinu.

2.1.5.2 bzip2

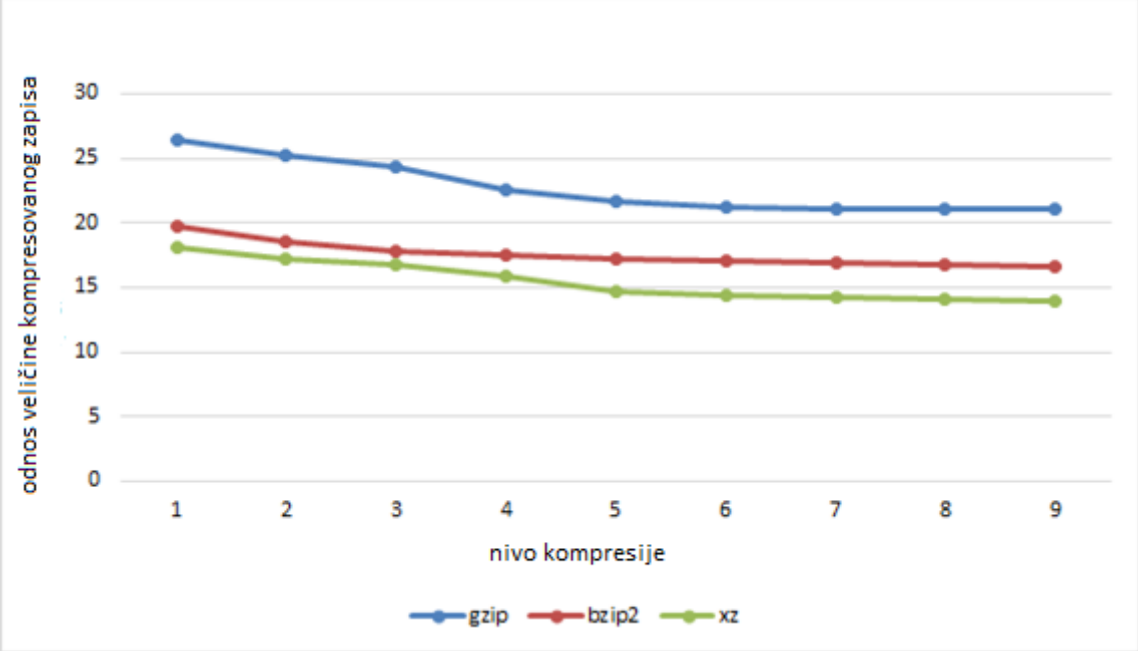
Naše istraživanje se odnosi na sisteme za kompresiju koji se najčešće koriste na serverima. Pored *gzip* kompresije postoje i *bzip* i *xy*. Program kompresije datoteka bzip2 je razvio Julian Sevard i započeo je 18. jula 1996. godine. Ostaje otvoreni izvorni program, dostupan svima besplatno, već dvadeset dve godine. Poslednje stabilno izdanje bilo je pre sedam godina. Verzija 1.0.6 je objavljena 20. septembra 2010. godine. Program kompresije *bzip2* baziran je na Burrovs-Vheeler algoritmu. Program može komprimirati datoteke, ali ih ne može arhivirati. Julian Sevard je i dalje zadužen za održavanje programa. Aplikacija za kompresiju radi na svim većim operativnim sistemima i dostupna je kao BSD licenca. Prema informacijama objavljenim od strane programera, program kompresije datoteka može kompresovati datoteke na 15% ili 10% drugih raspoloživih tehnika i radi na dvostruko većom brzinom kompresije i šest puta više od brzine dekompresije od *gzip*-a pod određenim okolnostima.

2.1.5.3 XZ

XZ Utils je besplatan softver za kompresiju opšte namene sa visokim odnosom kompresije. XZ Utils su napisani za POSIKS-like sisteme, ali takođe rade na nekim ne-so-POSIKS sistemima. XZ Utils su naslednik LZMA Utils. Jezgro kompresijskog koda XZ Utils baziran je na LZMA SDK-u, ali je prilično modifikovan da bi bio pogodan za XZ Utils. Primarni algoritam kompresije je trenutno LZMA2, koji se koristi unutar .kz kontejnerskog formata. Sa tipičnim datotekama, XZ Utils kreira 30% manju izlaznu vrednost nego gzip i 15% manju izlaznu vrednost od bzip2.



Slika 2Uporedni odnos brzina kompresije različitih alata ***preuzeto od*** *[15]*



Slika 3Uporedni prikaza odnosa kompresije kod različitih alata ***preuzeto od*** *[15]*

Gledano kroz zahteve ove disertacije kompresija podataka treba da upotpuni energetsku efikasnost prenosa serijalizovanih podataka. Prema tome, u nastavku sledi analiza postojećih radova vezanih za uticaj kompresije na potrošnju energije kod prenosa podataka REST komunikacijom.

2.1.5.4 Prethodni radovi

Kompresija podataka je ako bitna metoda u cilju smanjene potrošnje. U radovima [16] i [17] su objašnjene primene kompresije. Ona sa jedne strane smanjuje veličinu podataka kod transporta ali i energiju, ali sa druge algoritmi za kompresiju i dekompresiju zahtevaju povećanja procesorskog opterećenja kao i utrošak ram memorije. Navedeno dovodi do potrošnje električne energije tako da se tu mora pronaći ekvilibrijum. Međutim, i pored toga, autori [18] su zaključili da je kompresija jako povoljna kod konekcija gde je brzina protoka mala. Kod konekcija gde je brzina velika kompresija ima negativan uticaj.

Postoje radovi koji su razmatrali hibridne i adaptivne način kompresije. Kod takvog sistema kompresija se menja na osnovu stanja brzine prenosa. Ovaj način kompresije po autorima daje najbolje rezultate. Autori [18] su pobrojali sve tipove i algoritme kompresije i napravili poređenje koji je povoljniji za bateriju. Kompresija je zavisila od tipova podataka nad kojima je primenjivana. Oni su koristili različite tipove , slike i tekst. Testiranje je devet različitih programa rezultiralo je zaključkom da se pravilnim odabirom tipa kompresije može uštedeti do 50 % energije.

2.1.6 Izvršavanje obrade na server

Ovo poglavlje predstavlja osvrt na postojeće radove [3-5] usmerene ka poboljšavanju energetske efikasnosti mobilnog uređaja putem *Cloud Computing* metoda. Kod ovih autora je primenjena ideja korišćenja server i internet konekcije. Oni su odlučili da ogromne i masivne programske blokove, koji su bili vezani za neko izračunavanje i obračun nečega, prebace na udaljeni server ili veb server. *Cloud computing* je danas sveobuhvatno primenjena tehnologija koja i dalje poseduje potencijal obezbeđivanja mogućnosti za mobilne računare kao računske snage i energetske efikasnosti. Početkom ovog milenijuma javila se ideja, da energetska ušteda može biti ostvarena

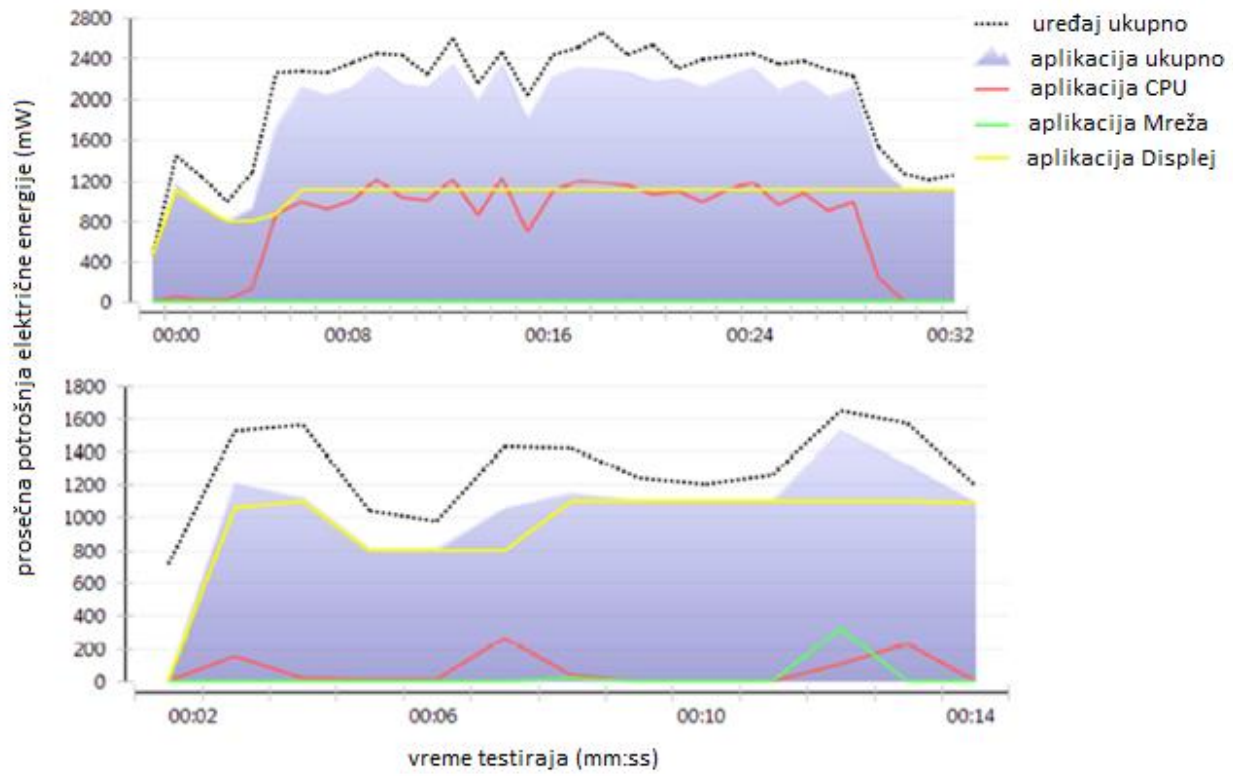
izvršavanjem obrade na udaljenom računaru [3]. Iskorišćavanje *cloud computing* servisa je ubrzo postiglo najveće interesovanje sa uslugama kao što su *Apple iCloud* ili *Microsoft Azure*, *Google drive*. Raniji radovi već su zamišljali mogućnost integrisanja klastera različitih nivoa, od programskih odluka [19], do kompajlera [20] i automatskih rešenja. Naime, određeni parametri aplikacije šalju se kao ulazni podaci na server gde će neki stacionarni računar sa stacionarnim napajanjem izračunati i uraditi sve te poslove koje bi inače trebala da radi aplikacija na telefonu. Taj sračunati materijal se vraća nazad mobilnom aparatu gde će on u okviru aplikacije prikazati rezultate. Na ovaj način omogućeno je da procesori sa manjim mogućnostima mogu izvršiti zahtevne operacije jer su lišeni ogromne potrošnje baterije i opterećenja procesora. Korisnik će koristiti aplikaciju a neće primetiti da se obrada odigrava na serveru. Međutim, faktori kao što su uslovi mreže mogu uticati na efikasnost procesa migracije. Naročito intenzitet protoka podataka može uticati na energetsku efikasnost, stoga je ta pojava ostavila dosta prostora za istraživanje i unapređivanje ovakvih sistema.

Među prvim ovakvim aplikacijama bilo je prepoznavanje govora. Navigacija i kreiranje ruta je radilo na tom principu. Danas se gotovo ne može zamisliti aplikacija koje ne vrši neku obradu podataka na serveru ili ne pretražuje neku udaljenu bazu podataka. Naravno, kod ovog pristupa postoje neka ograničenja. Ovaj pristup ne možemo koristiti ako stalno nešto akviziramo sa senzora telefona jer ćemo imati mnogo učestaliju komunikaciju sa serverom, što će dalje, dovesti do lošijih karakteristika nego da radimo obradu na samom uređaju.

Na osnovu rada [21] koji razmatra poboljšanje performansi mobilnog uređaja, mogu se doneti sledeći zaključci. Prilikom optimizacije aplikacije potrebno je odabrati funkcionalne blokove koji imaju malo ulaznih parametara, a prilikom izvršavanja koriste dosta procesora i procesorskog vremena. Pored toga, jako je bitno da ti procesi budu asinhroni tako da jedan ne može da utiče na drugi, odnosno, da zakasnelo izvršavanje jednog procesa usled okolnosti ne naruši rad drugog procesa koji se isto izvršava na serveru.

Autori su koristili *Little Eye* softver za određivanje protoka, potrošnje baterije i stanja memorije. Međutim, problem nastaje u bateriji zato što sami uređaji u telefonu ne računaju sa preciznom tačnošću samu potrošnju baterije već samo grubo mere kapacitet baterije. Ovo smo već napomenuli u prethodnom delu rada. Na osnovu analize dostupnih radova vezanih za merenje

utroška energije ovi podaci se moraju uzeti sa rezervom. Mogu da ocrtaju neke principe i obrise stvarnih potrošnji ali ne i da se nešto tačno prikaže. Za komunikaciju autori su koristili JSON format koji je tekstualnog tipa.



*Slika 4 Energetski profili potrošnje **preuzet od** [21]*

Eksperimenti u radu [21] su pokazali da migracija delova mobilnih aplikacija na udaljeni server može da doprinese čuvanju baterije na mobilnom uređaju. To je postignuto smanjenjem procesorskog opterećenja. Tu postoji i popriličan iznos energije potrebne da se podaci pošalju na server i da se prime sa servera. U pravilno optimizovanom sistemu oni su uvek manji nego ta obrada izvršena na mobilnom uređaju, što ne mora da bude uvek tako.

Ukoliko se koriste veliki podaci za transfer uz pomoć JSON stringova, može doći čak i do narušavanja ovog balansa, to jest da je ekonomičnije da se neka obrada vrši na samom uređaju i

da se ne šalje na server pa prima nazad. To je zapravo funkcija koja nam je najpotrebnija za naš rad.

2.1.7 Završne napomene poglavlja

Iz prethodne analize postojeće literature vidimo da su se naučnici intenzivno bavili usavršavanjem energetske efikasnosti mobilnih uređaja. Analizirani radovi imaju zajedničku osobinu: svi koriste izmene i projektovanja u programskom sloju. Tačnije pokušane su programske izmene kako u operativnim sistemima, tako i protokolima za komunikaciju. Pored citiranih radova postoje i drugi koji se bave temom optimizacije energetskih resursa, ali drugim hardverskim metodama i optimizacijama drugih komponenata (senzora, navika korisnika ..). Oni nisu obrađivani zbog nepovezanosti sa temom ovog istraživanja.

Iz ove analize proizilazi da je tema ovog istraživanja aktuelna, a da sam proces serijalizacije nije dovoljno istraživan u kontekstu savremenih aplikacija i energetske efikasnosti na mobilnim uređajima. Tokom istraživanja pronađeno je svega nekoliko radova na ovu temu na koje ćemo se pozivati.

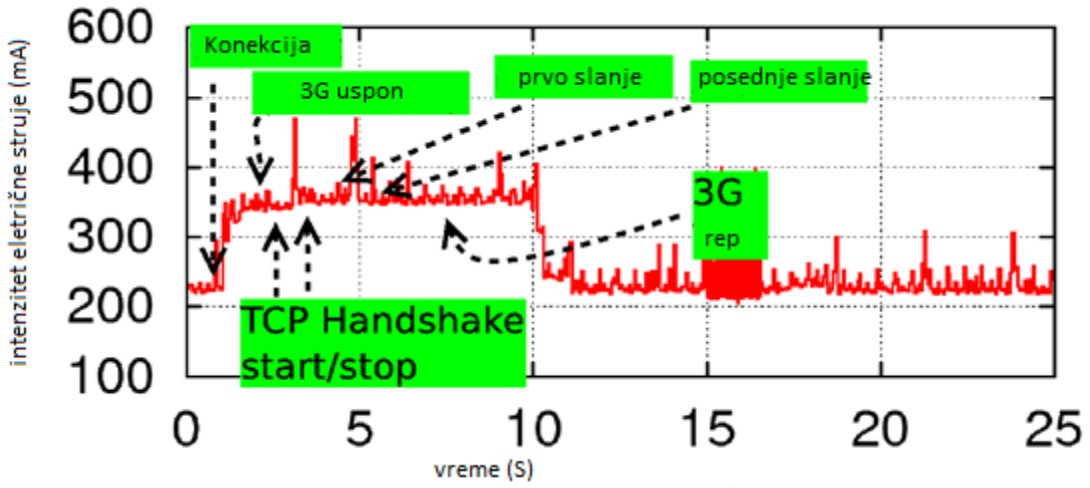
2.2 Merenje utroška energije analize prethodnih radova

Glavno pitanje na koje trebamo odgovoriti jeste: gde se energija potroši u okviru jedne aplikacije. Da bi na to pitanje dali valjan odgovor imamo tri celine na koje ćemo analizirati: razbijanje aplikacije u energetske entitete, praćenje potrošnje električne energije i aktivnosti svakog hardverskog dela na telefonu posebno. Najvažnije jeste mapiranje te potrošene energije na one entitete na koje smo već podelili program. Prvi deo je razbijanje programa na entitete. Takozvana granulacija ili segmentacija entiteta zavisi od preciznosti kojom programer hoće duboko da podeli svoj program da bi sračunao koje aktivnosti mu je najviše troše baterije. Ti entiteti se najčešće uzimaju kao četiri najpoznatija, dobro objašnjenja dela programa a to su: proces, *thread*-nit, potprogram i sistemski poziv. Generalni pristup koji će ovde biti primenjen je da se svaki sistematski poziv funkcije koji služi za IO (to jest komunikaciju sa perifernim uređajima) zapisuje. Isto se radi i sa svakim *thread-om* što nam omogućavaju sami Android operativni sistemi.

Na osnovu analize radova [22] [21] [23] da su napravili logove u samom kodu izvršnog programa sa nazivom aktivnosti i vremenom kada se ta aktivnost odigrala. Pored toga su merili ukupnu potrošnju energije.

Za razliku od desktop aplikacije ili servera na pametnim telefonima-mobilnim uređajima, potrošnja koju pravi svaki izlazno-uzlazni uređaj nije zanemarljiva i često može da bude veća od potrošnje koju pravi sam procesor. Svaka od tih komponenti može biti u nekoliko operativnih modova ili drugačije rečeno energetskih stanja.

Glavni problem sa kojim ćemo se susresti prilikom ovog merenja potrošnje, jeste što neke komponente mogu da imaju potrošnju u onom trenutku kad ih stvarno segment programa upotrebljava ali da mogu i da imaju neku repnu zakasnelu potrošnju koja je znatno manja od one kada se njihova aktivnost zahteva. Sama asinhronost može otežati analizu.

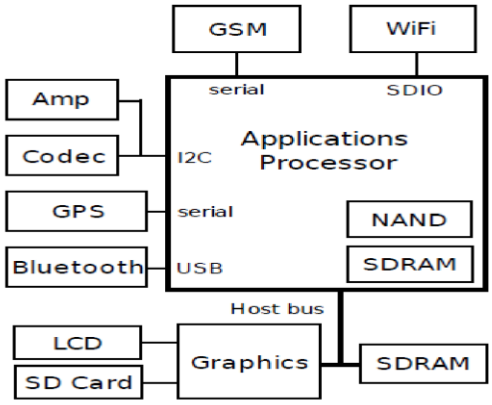


*Slika 5 Profil potrošnje energije tokom komunikacije **preuzeto od** [11]*

Još jedan od zanimljivih radova je rad Carroll-a i Aaron-a [8] jer su oni merili potrošnju pojedinačnih komponenti na mobilnom telefonu. Njihova ideja bila je da na svakoj od komponenti gdegod je bilo moguće postave jedan otpornik koji ima veoma malu otpornost. Na osnovu toga uspeli su da izmere napon (prosto merenje napona na tom otporniku) i tako su mogli da izračunaju protok struje. Tako su mogli da izmere stvarnu i realnu potrošnju energije u svakoj od tih komponenti. Ovo se razlikuje od pređašnjih radova gde smo videli da su autori koristili ili merili

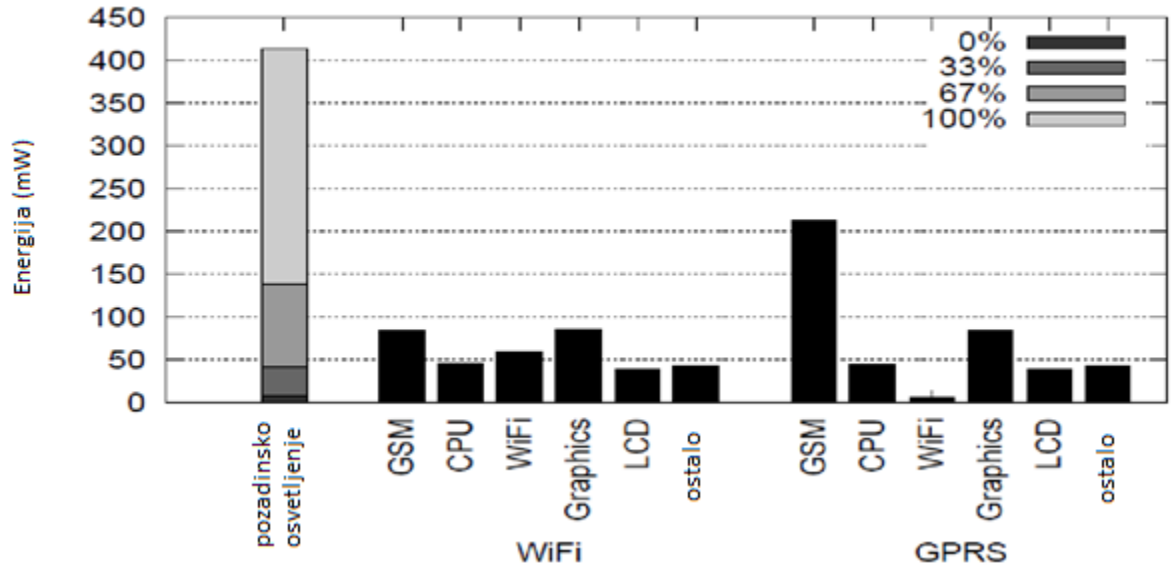
struju samo na ulazu u ceo uređaj, pa su softverskim putem i praćenjem logova procenjivali koliko koja komponenta troši .

U ovom delu rada nas isključivo zanima da analiziramo radove koji su se bavili načinim na koji se može precizno izmeriti potrošnja ali i metodama koje su autori navedeni u prethodnom poglavlju koristili. U radu [24] autori su pokušavali da prave matematički modeli na osnovu prostih testova komunikacije sa serverom na android telefonu. Testovi su se sastojali iz slanja i primanja podataka različitih veličina i preko različitih interfejsa od servera. Kod ovih autora je merena samo globalna potrošnja na osnovu kapaciteta baterije na telefonu, što se ne može smatrati preciznim.



Slika 6 Šematski prikaz savremenog mobilnog aparata

Autori [8] su možda imali najbolji i najtačniji način za merenje svake komponente ponaosob. Oni su koristili *Texas Instrumenst DAQ* uređaj za merenje koji je jako brz i precizan. Na taj način su mogli da izmere uspone i padove u potrošnji električne energije. Ovi autori nisu sproveli finu segmentaciju softvera na uređaju. Dakle, oni su pravili makro eksperimente koji su bili uglavnom vezani za slanje imejlova što je nama značajano jer se ova disertacija u osnovi bazirati na to kako olakšati komunikaciju između *cloud* servera i mobilnog uređaja.Jedan od zanimljivijih rezultata koji nam može biti od koristi je njihov prikaz merenja energije veb pretraživača. Veb pretraživač sam po sebi koristi REST oblik komunikacije. Urađen je uporedni test korišćenjem WIFI i GPRS modema. Kao što se očekivalo: potrošnja je bila mnogo veća kada je sve to na GPRS-u, ali su bitni i podaci koje mi ovde možemo da vidimo.



*Slika 7 Profil potrošnje pojedinačnih komponenti **preuzeto do** [8]*

Jedan od podataka na koji se mora obratiti pažnja je da 3G modem ima regulaciju snage. Autori [11] su radili eksperiment tokom kog su stavili mobilni uređaj u metalnu kutiju debljine 2 mm, nalik Faradejevom kavezu. Komunikacija sa baznom stanicom bila je jako loša. Sami modemi na mobilnim uređajima imaju mogućnost da menjaju jačinu signala u zavisnosti od kvaliteta konekcije. Ako su udaljeniji od bazne stanice onda oni emituju jači signal i suprotno. Ovo je bitno uzeti u obzir tokom ekperimentalnih merenja jer može bitno uticati na rezultate merenja. U različitim testovima mora se pratiti jačina signala.

Na kraju, autori su izneli ideju kojom možemo da upotpunimo naše istraživanje. Pokušali su da naprave energetski model. Njihov energetski matematički model predstavljen je u Tabeli 1. Model, do kojih su oni došli predstavlja konstante pomnožene sa vremenom izvršavanja.

		3G	GSM	WiFi
Energija prenosa podataka	$R(x)$	$0.025(x) + 3.5$	$0.036(x) + 1.7$	$0.007(x) + 5.9$
Repna energija	E	0.62 J/sec	0.25 J/sec	NA
Održavanje	M	0.02 J/sec	0.03 J/sec	0.05 J/sec
Repno vreme	T	12.5 sec	6 sec	NA
Energija po 50Kb prenosa u 20-sekundnom intervalu		12.5 J	5.0 J	7.6 J

*Tabela 1 Matematički modeli potrošnje **preuzeto od** [11]*

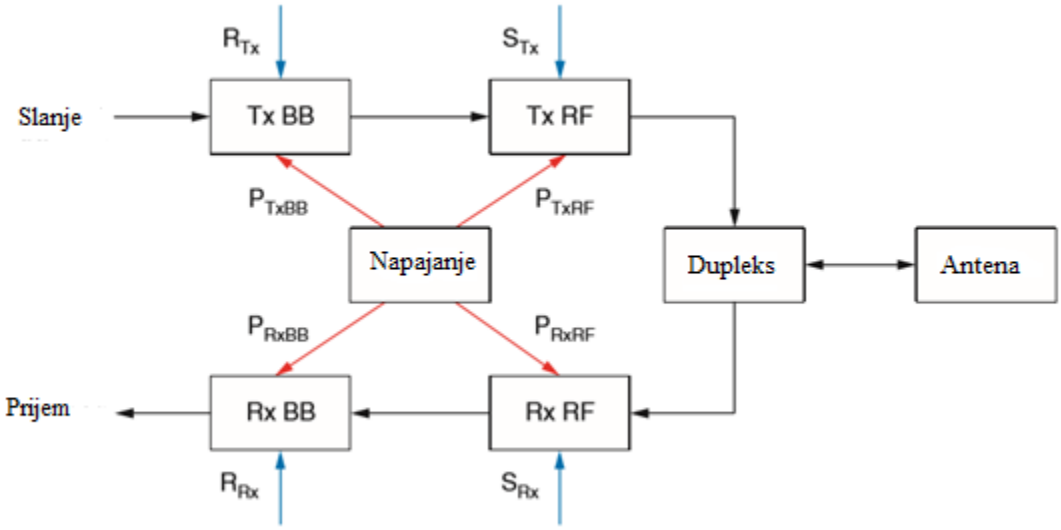
2.2.1 Metode merenja

Neki od autora su jednostavno koristili menadžment baterije na samom mobilnom uređaju, što se može smatrati da je solidan pokazatelj dobar za odnos ali ne i za toliko precizan. Drugi deo autora je koristio globalnu potrošnju mobilnih telefona. Direktno su merili potrošnju celokupnog uređaja električne energije sa napajanja, a zatim su softverskim putem beležili kada se koji uređaj aktivira. Došli su do interesantnih otkrića. Neki od perifernih uređaja ne prikazuju potrošnju, odnosno imaju zakasnili efekat. Njihova potrošnja se ne smanjuje odmah nakon izvršavanja programa nego postepeno opada. Ta grupa autora uspela je da napravi finu segmentaciju samog softvera unutar neke aplikacije. Treća grupa autora je otišla korak dalje u hardverskom smislu jer su merili potrošnju električne energije parcijalno na komponentama. Isti autori su napravili i korak unazad u softverskom pogledu jer nisi vršili finu segmentaciju softvera koji su testirali.

Više autora je koristilo model telefona sa Symbian operativnim sistemom, dok su kao alat koristili *Nokia Energy Profiler* (NEP) v1.1.

U [11] radu korišćen je softver za analizu potrošnje električne energije . *NEP v1.26* koja akvizira podatke na 250 ms i korišćen je telefon N95. Autori ovog rada su svako merenje ponavljali po 20 puta kako bi smanjili greške usled nepreciznosti merenja i dobili presek. Autori tvrde da njihovo merenje ima 95 % tačnosti. Inače *Nokia Profiler* je aplikacija koja radi na Symbian operativnim sistemima i ima ulogu merenja trenutne potrošnje struje u vatima, zatim merenja struje, opterećenja procesora i zauzeća ram memorije i ostalih parametara. Prednosti ovog sistema su laka upotreba i jednostavna analiza podatka, jer se sa lakoćom izvoze u *exel* tabele. Mane ovog sistema su nepreciznost i nemogućnost da se parcijalno izmeri potrošnja u komponentama.

Laurdsen Mads i Noel Laurent [25] su napravili jedan bitan eksperiment. Pokušali su da parcijalno mere potrošnju samih komunikacijskih interfejsa nezavisno od celog sistema. U tom pokušaju su ustanovili da je potrošnja pomoćnih komponenata (kao što su USB GPRS, 3G WiFi adapter), mnogo veća nego kod komponenti integrisanih u samim mobilnim uređajima. Razlog je što nisu predviđeni za to. Ovi autori su analizom došli do matematičkog modela uticaja sirove komunikacije na potrošnju baterije.



Slika 8 Blok šema modela potrošnje ***preuzeto od*** *[25]*

$$P=P_{on} + m_{rx} \times (P_{Rx} + P_{RxBB}(R_{Rx}) + P_{RxRF}(S_{Rx})) + m_{Tx} \times (P_{Tx} + P_{TxBB}(R_{Tx}) + P_{TxRF}(S_{Tx}))$$

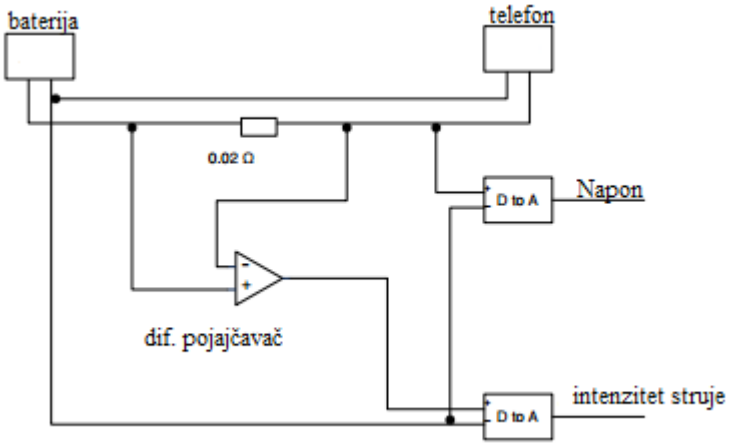
Gde su P_{on} , P_{Rx}, P_{Tx} energije koje opisuju sistem kada radi, kada prijemnik prima podatke, a transmiter šalje.

U literaturi se mogu naći radovi koji su svoju analizu bazirali na *Little Eye* [26]. Ovo je programski paket za analizu rada aplikacije na Android operativnom sistemu i grafički je dizajniran da na intuitivan način lako i brzo korisniku pruža mnoštvo informacija. *Little Eye* pomaže da se izmere zahtevane performanse i prikazuje koliko energije, ram memorije i procesora aplikacija troši u realnom vremenu. On je predviđen za duboku analizu rada aplikacije koja je u razvoju. Informacije se crpe sa samog uređaja na kome se aplikacija izvršava kroz *ADB* (*Android Debug Bridge*) što predstavlja instant rešenje.

Slika 9 Little Eye programski paket za praćenje potrošnje

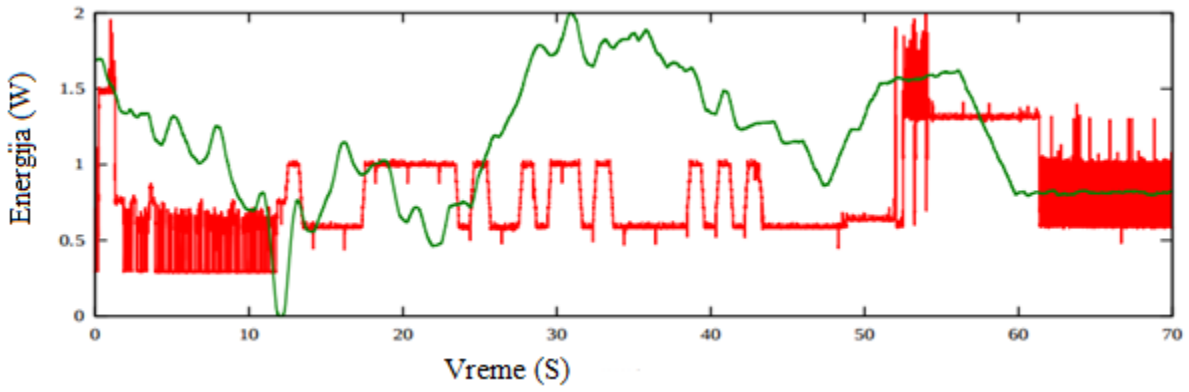
Ovakav sistem daje adekvatne rezultate, međutim oni se ne mogu uzeti za visoko precizne, jer merenja (konkretno električne energije) se baziraju na merenju kapaciteta baterije koje nije precizno i linearno. Prema tome, ovaj alat se može koristiti samo za početnu i plićу analizu.

U studiji [27] autori su se striktno posvetili samo temi merenja potrošnje i analizi. Oni su postavili nekoliko ciljeva koje su težili da zadovolje. Svi testovi su bili automatizovani, te nije bilo potrebe menjati okruženje ili opremu za posebnu seriju testova. Samo merenje potrošnje obavili su tako da imaju poseban hardver za merenje ali nisu vršili modifikacije na samom uređaju.



Slika 10 Električna šema uređaja za merenje potrošnje ***preuzeto od*** *[27]*

Jednostavnim električnim sklopom koji se sastoji od jednog pasivnog otpornika veoma male otpornosti i operacionog pojačivača oni su uspli da u realnom vremenu paralelno mere napon i jačinu struje koja iz baterije ide prema uređaju. Za digitalizaciju su koristili *National Instruments* PCI-MIO-16E-4 koji je mogao da radi akviziciju sa 25 Khz. Kako je sam mobilni uređaj sastavljen od komponenti sa velikom brzinom rada, autori su smatrali da je pre analize bitno osciloskopom proveriti prirodu potrošnje. Analizom su utvrdili da brzih pikova i promena nije bilo pa su zaključili da je moguće precizno meriti potrošnju i sa manjom brzinom akvizicije.U nameri da urade dobru segmentaciju i analizu potrošnje el. energije autori su test skriptu podelili na niže programske celine. Oni su napravili takter koji je neprekidno palio i gasio ekran telefona u ekvidistantnim trenucima. Na taj način su u finalnom trend dijagramu primećene povorke potrošnje. I tako su mogli da determinišu mesta gde počinju i završavaju se sekvence test programa.

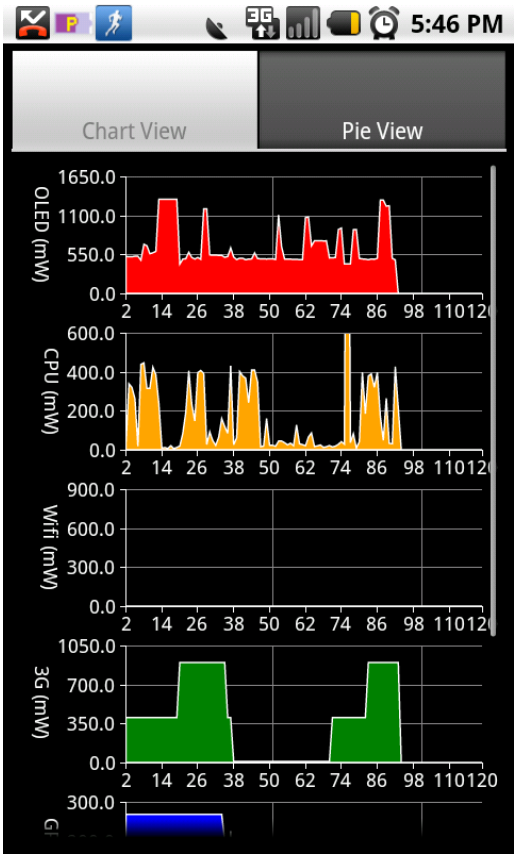


Slika 11 Grafik merenja potrošnje ***preuzeto od*** *[27]*

Način snimanja testa kod ovih autora se ne može svesti na mikro nivo gde se može analizirati svaka funkcija i instrukcija ponaosob i sa velikom preciznošću. Modernizacija ovog načina bi predstavljala korišćenje ADB poruka za sinhronizaciju mesto paljenja i gašenja ekrana. U nastavku se ne bi koristile makro skripte za test, već programi sa mnogo nižim setom instrukcija čije bi se izvršavanje beležilo.

Još jedan od radova koji je se bavi procenom potrošnja električne energije na mobilnim telefonima je rad grupe autora predvođenih Zangom sa Mičigenskog univerziteta. Ovaj rad objašnjava kreiranje i primenu sistema za procenu potrošnje baterije na Android operativnom sistemu

korišćenjem aplikacije koju su autori kreirali. Zanimljivo je da su autori ovog rada koristili senzor ugrađen u sam uređaj za merenje potrošnje. Njihova metoda predstavlja niz testova koji u određenom vremenskom intervalu, kroz više uzastopnih testova programskim putem opterećuju neku od analiziranih komponenti: od stanje neaktivnosti od maksimuma tako da prate utrošak energije. Kroz veliki broj iteracija i obradu rezultata i uklanjanja grešaka metodom najmanjih kvadrata, dolazi se do parametra potrošnje traženih komponenata i testiranog modela uređaja. Ovaj rad je testirao šest komponenta ponaosob : CPU , Ekran, WiFi , mobilni interfejs 3G , GPS i audio. Kroz rad napred navedenih autora predstavljen je interesantan način da se putem senzora u samom mobilnom uređaju izvrši merenje utroška električne energija na različitim komponentama i izdvoji uticaj traženog potrošača.



Slika 12 Prikaz aplikacije za merenje energije PowerTutor

2.3 Završne napomene poglavlja

Na osnovu gore pomenute analize možemo formirati sledeće zaključke bitne za ovaj rad :

- Veći paket koji se šalje stvara veću potrošnju
- Aktivan rad modema ostvaruje znatnu potrošnju
- Održavanje konstante konekcije sa serverom i konektovanje po zahtevu ostvaruju različite uticaje
- Distribuirani rad na serveru može da smanji potrošnju
- Sistem za precizno merenje mora biti fizički razdvojen za pojedine komponente
- Kompresija podataka može dvojako da utiče

Nakon ove analize možemo sa sigurnošću da tvrdimo da način serijalizacije itekako ima uticaj na potrošnju baterije. A razlozi za to su što različiti tipovi serijalizacije imaju različite veličine zapisa, proces serijalizacije na serveru može da takođe traje duže ili kraće i da utiče na vreme provedeno u održavanju konekcije.

3 III Deo

3.1 Serijalizacija

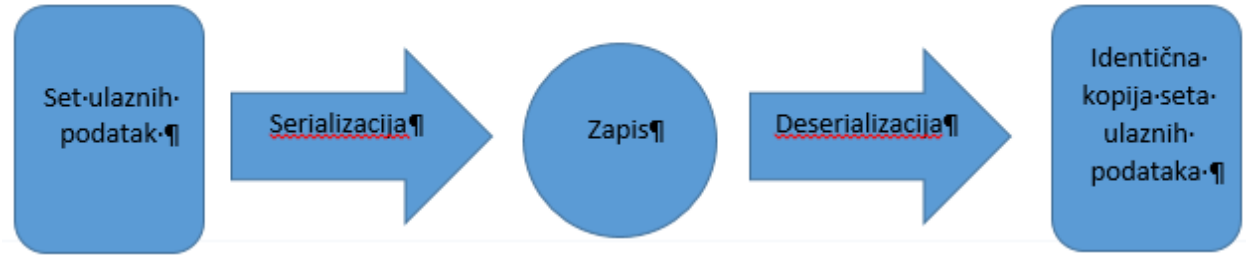
U sastavu savremenog softverskog inženjerstva nalaze se i moderne mobilne aplikacije spregnute sa veb tehnologijom bazirane na sveobuhvatnoj razmeni složenih struktura podataka. Posebnu pažnju treba posvetiti načinu razmene složenih struktura podataka. Jedan od vodećih problema sa kojima se projektanti kompleksnih informacionih sistema susreću je raznolikost platformi i programskih jezika koji se koriste kroz razvoj i implementaciju iste. Jedna savremena mobilna aplikacija ili sistem u najprostijem obliku se sastoji od aplikacije koja je grafički korisnički interfejs na samom uređaju, servera za razmenu podataka, relacione baze podataka i administrativnog interfejsa ako je to zahtevano. Ovakav koncept zahteva razmenu složenih struktura podataka između najmanje tri različite platforme. Treba uzeti u obzir da su podaci koji se razmenjuju u savremenom informacionom svetu jako složeni i da su objektno struktuirani. Gotovo da više nema razvoja softvera koji nije objektno orjentisan. Ne treba zapostaviti još jednu bitnu razmenu podataka koja postoji u samom programu. Sami programi bilo da su na mobilnoj platformi ili serveru zahtevaju trajno čuvanje nekih setova podataka i njihovo ponovo učitavanje. Većina njih tokom pokretanja sa lokalne trajne memorija učitava setove konfiguracionih podataka ili podatke o pređašnjem radu. Kod pojedinih programa potrebno je određene setove podataka proslediti na analizu razvojnom timu. Srazmerno svim ovih zahtevima u softverskom inženjerstvu postoji posebna oblast koja se naziva serijalizacija podataka. Skup programskih instrukcija tokom životnog veka jednog softvera u kojem se trenutna vrednost određenih: struktura podataka, promenljivih, instanci klasa ili stanja sesija može trajno sačuvati ili poslati kao meta zapis unapred definisane strukture a takođe i vratiti (rekonstruisati) iz tako trajno ili privremeno sačuvanog oblika (zapisa) u originalni format naziva se serijalizacija. Mnogi viši programski jezici sadrže već ugrađene biblioteke za serijalizaciju ali takav vid serijalizacije nije multiplatforman već se može koristiti samo u okviru istog programskog jezika. Takav način serijalizacije nije moguće koristiti u multiplatformnim sistemima kao sto su mobilne aplikacije.

Hronološki gledano prvi oblik serijalizacije se javlja kao *CSV* (*comma separated values*- vrednosti razdvojene zapetom) fajlovi. Iako su i danas veoma popularni za analizu nekih međurezultata ili krajnjih vrednosti, ipak poseduju nedostatke koji su ih potisnuli iz upotrebe u savremenom softverskom inženjerstvu. Naime, oni su tabelarne forme koje koriste zarez kao separator između dva segmenta od interesa. Ovo dovodi do toga da je komplikovano u njima čuvati precizne brojeve sa decimalom i složene tekstualne strukture koje imaju zapetu u svom tekstu. U toku evolucije softverskog inženjerstva stvorila se potreba da se kreira format koji može nadomestiti pomenute nedostatke. Jedan od prvih i dalje aktuelnih sistema za serijalizaciju predstavlja *XML*. On je dugi niz godina bio vodeći sistem za serijalizaciju. Prvi je ponudio moćne biblioteke za pakovanje i raspakivanje podataka koje su višejezične i višeplatformske, čime je omogućio brzi i efikasni transfer i razmenu podataka između različitih platformi. Međutim, kako je XML nastao u vremenu kada informatičke mogućnosti nisu bile ni približne današnjim ostao je uskraćen za brzi razvoj kakav je danas potreban. Njegova mana je nedostatak podrške za nizove. Veoma bitan nedostatak koji je usko povezan sa ovim naučnim radom je veličina njegovog fajla. Podaci koji se serijalizuju *XML* sistemom se prevode u zapis tekstualnog oblika koji je složene notacije. Veliki broj specijalnih znakova i tabulacija čini ga prevelikim. Ova karakteristika ima veliki uticaj na potrošnju baterije prilikom transfera tih zapisa (što će u narednom poglavlju biti dokazano). Pored toga, biblioteka koja obavlja raspakivanje podataka radi suviše sporo ako zapis poseduje podatke većeg obima. Jedan od novijih formata serializacije je *JSON*, nastao u *javascript* jeziku. *JSON* predstavlja prosto i brzo rešenje za potrebe serijalizacije, jer podržava nizove i četiri osnova tipa podataka. Međutim, kao i *XML* on je tekstualnog tipa. Samim tim sa stanovišta energetske efikasnosti on ima nedostatke. Veličina zapisa je manja nego kod *XML*-a zbog jednostavnije notacije i korišćenja manjeg broja tagova. Problem koji on poseduje je u brzini parsiranja tj povratka informacija iz zapisa u osnovni oblik. Bilo je pokušaja da se *JSON* format transformiše u binarni oblik ali nisu bili uspešni. Došlo do paradoksa gde su binarni fajlovi postali veći od odgovarajućih tekstualnih. S obzirom na ovaj problem i narastanjem potrebe za češćim pozivima funkcija za serijalizaciju i što većem obimu razmene podataka, velike softverske kompanije rešile su da se posvete ovom problemu. *Google* je 2012. predočio svoj novi sistem koji se zove *Protocol Buffer*. Facebook je razvio *Treef* a Appache *Apache Avro*. Sve njih odlikuje smanjeno vreme deserijalizacije i smanjen obim zapisa. Ovi sistemi su binarnog tipa ali su zbog toga morali da

definišu način i šabloni pakovanja podatka. U narednom delu ovog poglavlja pokušaćemo da detaljno pojasnimo proces serijalizacije.

3.1.1 Sistem za serijalizaciju

Pod sistemom za serijalizaciju podrazumevamo skup programskih instrukcija koje određeni složeni skup podataka (promenljivih zajedno sa njihovim trenutnim vrednostima) transformiše u zapis određenog formata a zatim definisanim procedurama iz zapisa vraća u identične složene strukture (Slika 11).



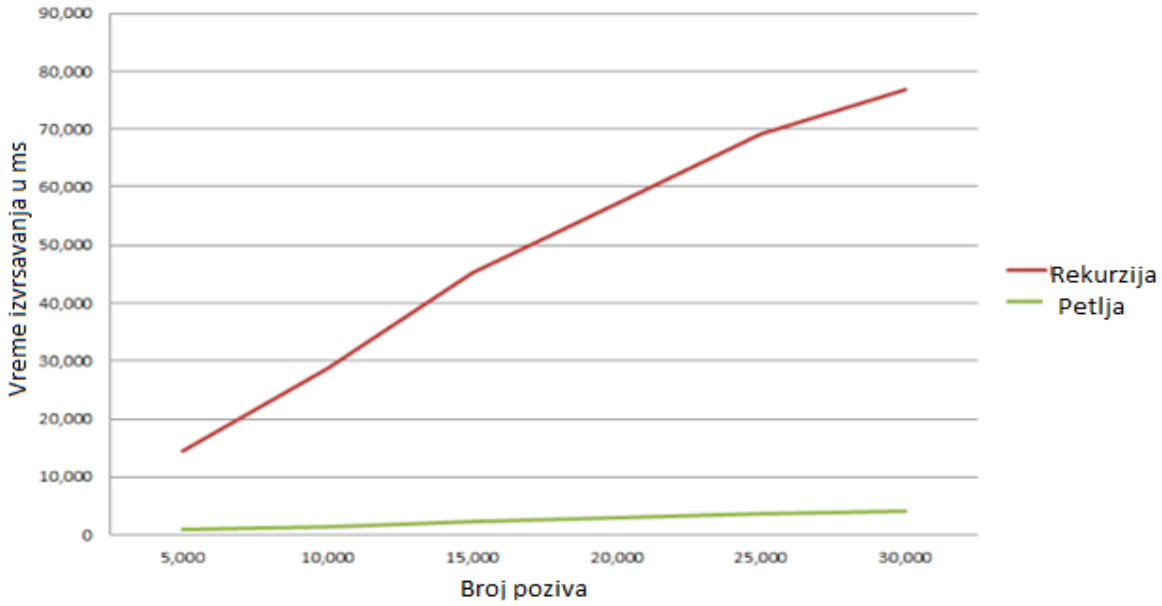
Slika 13 Šematski prikaz sistema serializacije

Jednom napisane funkcije za serijalizaciju i deserijalizaciju ne mogu se koristiti u drugim programskim jezicima kao takve, već se moraju formirati biblioteke koje se kasnije mogu dodati u kompatibilnim programskim jezicima. Drugačije i šire primenjeno rešenje je ponovno pisanje funkcija na drugim programskim jezicima (portovanje). Sistem za serijalizaciju tokom svog kreiranja mora da zadovolji sledeće kriterijume:

- prevod složenog seta podataka u zapis
- povratak seta podataka iz zapisa
- mogućnost čuvanja tipova podataka ulaznog seta na izlazni nakon procesa serijalizacije

Sisteme za serijalizaciju možemo podeliti prema tipu zapisa u dve kategorije: tekstualne (ljudima čitljive) i binarne (nečitljive). Ova podela predstavlja suštinsku razliku među sistemima jer tip zapisa utiče na performanse, načine implementacije i korišćenja sistema.

Složenost kod tekstualnih sistema predstavlja proces deserijalizacije zapisa. Njemu je potrebno posvetiti veoma veliki značaj jer on mora biti robustan na eventualne izuzetke u zapisu ali i dovoljno brz s obzirom da u većoj meri od njega zavise performanse sistema. Proces serijalizacije u najopštijem obliku možemo definisati kao iteraciju kroz ulazni set podataka i pakovanje naziva polja (varijabli) i njihovih vrednosti po unapred zadatom kriterijumu (šemi). Iz ove definicije proizilazi zaključak da proces serijalizacije u najopštijem obliku predstavlja jednosmerni proces koji nema potrebu za pamćenjem pređašnjih stanja u svom toku. Nasuprot procesu serijalizacije deserijalizacija zahteva prilično složeni sistem parsera koji na osnovu specijalnih karaktera rastavljaju zapis na pod segmente. Dalje se od tako dobijenih pod segmenata kreiraju složeni objekti ili strukture. Za razliku od serijalizacije proces deserijalizacije nije jednosmeran i nezavisan od prethodnog stanja parsera. Sam proces parsiranja je zahtevan jer iziskuje mnogo instrukcija da bi se jedan string podelio na podsegmente. Dakle, proces deserijalizacije može se svesti na traženje podsegmenata zapisa po unapred definisanim pravilima i od tih podsegmenata formirati ulaznu strukturu. Ovaj proces zahteva konstantno prisustvo zapisa. To znači da se tokom procesa moraju čuvati podaci o celom zapisu pa čak i oni koji su obrađeni. Ukoliko su zapisi velikog obima podataka iziskuje se poprilično zauzeće memorije. Kada se radi o složenom zapisu koji je nastao kao posledica serijalizacije nekog složenog objekta koji u sebi sadrži podobjekteI, onda je jedan od načina da se napravi algoritam za rekonstrukciju podataka rekurzija. Ovo je veoma komplikovan proces za procesor i RAM memoriju na kojima se izvršava, jer iziskuje veliki stek podataka. Nepravilno implementirani rekurzivni algoritmi mogu da naprave znatnu cenu obrade ukoliko je zapis većeg obima. Na sledećem grafiku se vidi odnos brzina rekurzivnih algoritama i klasičnih Slika 14.



Slika 14Odnos brzina rekurzivanih i iterativnih algoritama

Kako je prikazano, iterativni pristup je superioran u odnosu na rekurzivni čak i sa malim brojem funkcija poziva. Ovo pokazuje da je rekurzija skupa ne samo zato što uključuje dosta poziva funkcija, već i iz drugih razloga. Ovo je posebno izraženo za jako duboku rekurziju, jer opterećenje na procesoru postaje impozantno nakon samo hiljadu funkcija poziva, čak i sa funkcijom koja je prazna. Na uređaju sa slabijim procesorom ili mobilnom uređaju to bi lako moglo dovesti do vrlo loših performansi softvera.

Stanje zapisa je takođe bitno u procesu deserijalizacije. Problemi mogu nastati ako se zapis na neki način pokvari, odnosno izmeni. Već smo napomenuli da je za proces deserijalizacije potrebno raspolagati celim zapisom a ne samo njegovim segmentom. Zbog toga perser mora prekinuti rad i prijaviti grešku jer u protivnom može dati nevalidne podatke i ne može garantovati verodostojnost. Problemi mogu nastati ukoliko neki od podataka koje hoćemo serijalizovati sadrže kao vrednost neki od specijalnih znakova koje sistem koristi kao tagove u zapisu. Ipak, dobar sistem za deserijalizaciju ne može biti u punoj meri imun na takve manifestacije. Ulogu čuvara u slučajevima kvalitetnih sistema ima proces serijalizacije koji prepoznaje specijalne karaktere koji su kritični i menja ih drugim posebnim nizovima karaktera koji ne prave problem u toku rada parsera. Na kraju

procesa deserijalizacije sistem menja ove zamenske nizove u njihove originalne. Sve navedeno moguće je zahvaljujući tome što je u životnom veku podataka proces serijalizacije uvek stariji tj prethodi procesu deserijalizacije. Drugim rečima: podaci se moraju prvo serijalizovati da bi se deserijalizovali.

3.1.2 Postojeći tipovi serijalizacije

Danas je poznato preko dvadeset programa formata serijalizacije U tabli 2 su pobrojani neki od njih koji su u upotrebi. Naravno, ne koriste se svi podjednako. Najupotrebljiviji od njih je *JSON* [28]

Ime	Godina	Kreator	Baziran na	Šema/IDL	Čitljiv
CSV	1967	Yakov Shafranovich	n/a	delimično	da
XML	1998	W3C	SGML	da	da
XML-RPC	1998	Dave Winer	XML/SOAP	ne	da
JSON	2001	Douglas Crockford	JavaScript	delimično	da
YAML	2001	Clark Evans	C/Perl/XML	delimično	da
Candle	2005	Henry Luo	XML/JSON	da	da
OpenDDL	2013	Eric Lengyel	C/PHP	ne	da

*Tabela 2 Prikaz tekstualnih formata serijalizacije **preuzeto od** [29]*

Prvenstveno zbog ekspanzije veb tehnologije i *JavaScript* programskog jezika koji je postao okosnica razvoja veb *foregound* aplikacija. Ovaj trend se prelilo i na Android aplikacije Kao što je već rečeno opšteprihvaćeni XML je prethodio JSONu. Godine 1998. *XML* je uveden za asinhroni transfer strukturiranih podataka između klijenta i servera u *AJAX* Veb aplikacijama. U tom kontekstu, *XML* je definisan tekstualni format sa ulogom da prenese objekte na druge sisteme bez obzira na platformu ili programski jezik koji se koristi. Da bi prevazišli problem kompaktnosti, Binarni *XML* je predložen kao alternativa regularnom *XML*-u. Kada su podaci kodirani u *JSON*-u, rezultat je obično manji od ekvivalentnog kodiranja u *XML*-u. *JSON* je definisan kao alternativa *XML*-u i najčešće se koristi za komunikaciju klijent-server u Veb aplikacijama. *JSON* je zasnovan na sintaksi *JavaScript*-a, ali je takođe podržan i u drugim programskim jezicima. Takođe postoji binarno kodiranje za *JSON* (npr. *BSON*, , *UBJSON*). Još jedan čitljiv format za serijalizaciju je *YML* (super set JSON) [29].

Kao što smo već naveli postoje i binarni sistemi prenosa koji bi teorijski trebalo biti superiorniju u odnosu na tekstualne. U tabeli 3 su pobrojani savremeni formati .

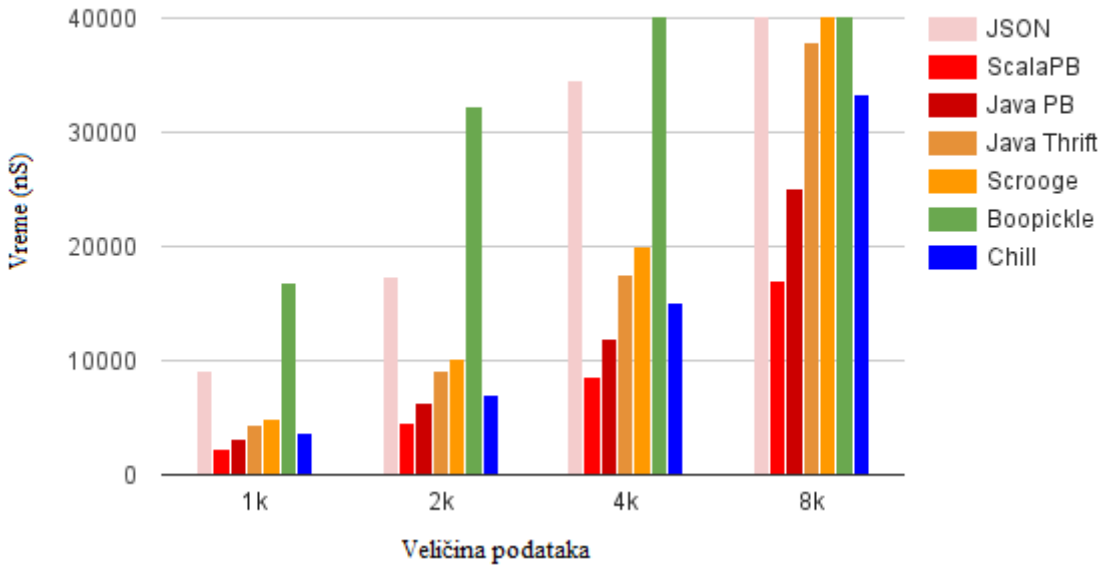
Ime	Godina	Kreator	Baziran na	Šema/IDL	Čitljiv
Avro	2009	ASF	n/a	da	ne
BSON	2003	MongoDB	JSON	ne	ne
Cap'n Proto	2013	Kenton Varda	protobuf	da	ne
Protocol Buffers	2008	Google	n/a	da	ne
Thrift	2007	Facebook/Apache		da	ne

*Tabela 3 Prikaz binarnih formata serijalizacije **preuzeto od** [29]*

Većina ovih protokola koristi *IDL* (Interface Description Language) koji služi da opiše strukturu serijalizovanih podataka. Samim tim, komplikovaniji su za razumevanje i implementaciju od tekstualnih, ali superiorniji u pogledu pakovanja podataka tj. veličine i brzine pakovanja.

3.1.3 Poređenje formata serijalizacije

Postoji dosta dostupnih istraživanja o komparaciji veličine zapisa i brzine rada pojedinih formata serijalizacije. Međutim, ono što je bitno je da sam dizajn nekog formata ne utiče svih 100% na performanse istog; konkretno na brzinu serijalizacije i deserijalizacije. Može se deseti sledeća manifestacija: da neki binarni formati za koje očekujemo da budu brži od tekstualnih u određenim programskim jezicima i pod određenim okolnostima bude slični ili sporiji. Dakle sama implementacija programskih biblioteka bitno utiče na brzinu serijalizacije. Konkretno tokom ovog istraživanja prilikom kreiranja *CBSaver API* (koji će biti razmatran u poslednjem poglavlju) primećeno je da Android *JSON* biblioteka deserijalizaciju podataka radi znatno brže nego msgPack Android biblioteka. S druge strane, kada je reč o c# bibliotekama situacija je drugačija. Sama struktura podataka koji se serijalizuje i deserijalizuje može da ima značajan uticaj na performanse..



Slika 15Prikaz odnosa vremena serijalizacija različitih tipova i biblioteka ***preuzeto od*** *[30]*

3.2 Završne napomene poglavlja

Možemo upotpuniti pretpostavke kako serijalizacija kao proces koji se izvršava na serveru može da utiče na energetsku efikasnost mobilnog uređaja koji komunicira sa njime. Dakle, različiti tipovi-sistemi za serijalizaciju imaju različite vrednosti parametara koji ih opisuju; u našem istraživanju to su vrednosti veličine zapisa koji nastaje procesom serijalizacije, ali i vreme potrebno da se podaci serijalizuju i deserijalizuju. U narednim poglavljima ove dve veličine pronaći će svoje mesto u celokupnom modelu potrošnje A iz ovog poglavlja prenosimo znanje o tome i pretpostavku kakve te osobine mogu biti kako bi ih dalje uvrstili u opšti model. Sledeće poglavlje se bavi arhitekturom mobilnih aplikacija i ulogom interneta, servera i serijalizacije.

4 IV Deo

4.1 Mobilni sistemi današnjice i uloga serijalizacije

Cilj ovog rada je da istraži uticaj korišćenja sistema za serijalizaciju na potrošnju baterije i predloži rešenja za poboljšanje, pa ćemo stoga analizirati potrebe mobilnih sistema današnjice. Većina mobilnih aplikacija današnjice ima ulogu da prenese informacije od korisnika do specijalizovanog servera i na osnovu tih informacija i zahteva vrati potrebne informacije ili rešenje problema. Čak i aplikacije poput raznih sistema za komunikaciju i socijalnih mreža rade po istom principu.

Na osnovu studije korporacije *AVG Techologies* o planu i razvoju mobilnih sistema [31] korišćenjem njihovih istraživanja i testova možemo da sagledamo koje su to aplikacije koji imaju najveći uticaj na bateriju. Pored toga, oni su takođe razmatrali koje aplikacije i u kolikoj meri koriste internet konekciju. Pomenuta studija može se smatrati verodostojnom jer je zasnovana na nasumičnom i anonimnom uzorku od tri miliona korisnika. Ova analiza je sprovedena počev od januara 2016. godine zaključno sa aprilom iste godine. U tabeli 4 su dati rezultati istraživanja.

1	SamsungAll Share	6	Facebook
2	Samsung Security	7	WhatsApp
3	Beaming Service	8	WeChat
4	ChatOn	9	AppLock
5	GoogleMaps	10	BBM

Tabela 4 Spisak mobilinih aplikacija sa najvećim utroškom baterije ***preuzeto od*** *[31]*

1	Facebook	6	Beaming Service
2	The Weather Channel	7	BBM
3	Instagram	8	WhatsApp
4	GoogleMaps	9	Facebook Messenger
5	Clean Master	10	BBM

Tabela 5 Spisak mobilnih aplikacija sa najvećim prenosom podataka preuzeto do [31]

Na osnovu ove analize se mogu videti i aplikacije koje su najveći konzumenti interneta u tabeli 5. Pregledom ove dve tabele možemo zaključiti da postoji velika korelacija između potrošnje baterije i upotrebe interneta. Opštepoznato je da je upotreba interneta drugi po redu uzrok prekomerne potrošnje baterije. Kako bi se bolje razumela ova korelacija pojašnjeni su principi rada nekih od aplikacija iz ove dve tabele.

Goole maps je aplikacija koja izdašno koristi *dedicate* server. Sama mapa (slika) terena se sastoji iz manjih delova takozvanih pločica (*tiles*) koje se po zahtevu korisnika za određenu regiju i određeni nivo uvećanja kreiraju na serveru i šalju korisniku. Pored slika šalju se i mnoge druge informacije, imena ulica, prikaza puteva i prikaz informacije o stanju saobraćaju. Pored samog preuzimanja informacija sa *Google maps* servera, aplikacija mora da uradi i dosta obrada na samom uređaju.

WhatsApp je aplikacija koja se koristi za razmenu tekstualnih poruka između dva klijenta tj. dva mobilna uređaja putem internet mreže. Ovakva komunikacija je moguća ako se koristi server kao

posrednik. U konkretnom slučaju koristi se *JABBER* (XMPP) [32] server a paketi se pakuju kao tekstualni nalik *XML*-u. Tačnije *XMPP* server podržava *JSON* i *XML* (ref XEP-0295).

Facebook messenger je aplikacija koja ima istu namenu kao i *WhatsApp.* Ona po potrošnji baterije nije među prvih deset ali je po zastupljenosti dva mesta ispred *WhatsApp*-a prema u tabeli 4. Činjenica je da je *Facebook* počeo da korist *Thrift* sistem za serijalizaciju koji je binaran. Ovo je jedan od pokazatelja koji može da motiviše naše istraživanje vezano za uticaj serijalizacije na performanse energetske efikasnosti. Dobro se zna na osnovu istraživanja koje je sprovedeno daje bolje rezultate u veličini fajla i načinu komunikacije.

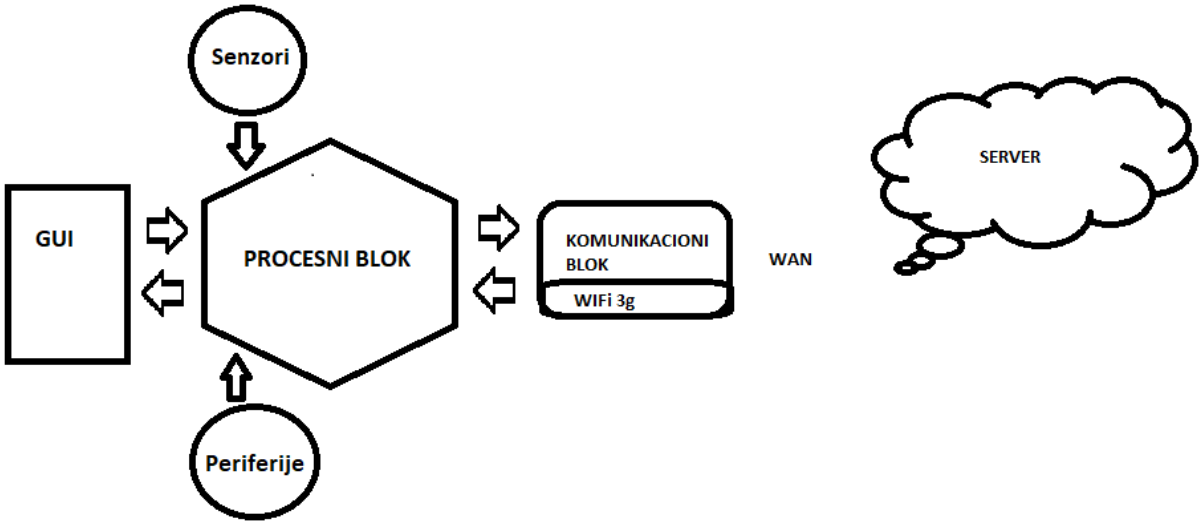
Dakle, iz prethodnog razmatranja o sistemima serijalizacije možemo zaključiti da na teret mobilnog uređaja padaju obimne deserijalizacije i neizbežni prijem podataka. Zbog same namene savremenog mobilnog uređaja jasno je da korisnik potražuje znatno viši obim podataka od servera nego što ga šalje. To je naročito vidljivo kod aplikacije kao što su *Google maps* ili bilo koje druge aplikacije za pomoć pri vožnji.

Ovakva analiza postojećih zahteva današnjice nam može ukazati da je za optimalniji rad mobilnih aplikacija potrebno obratiti pažnju na cenu deserijalizacije podataka. Kao sto se može zaključiti serijalizacijom je pretežno opterećen server. Eksperimentalnim metodama i analizom radova drugih autora u narednim poglavljima će se utvrditi da li postoji korelacija između opterećenosti servera nastale usled serijalizacije podatka na potrošnju baterije mobilnog uređaja.

4.1.1 Arhitektura mobilne aplikacije

Za ovaj naučnoistraživački rad smo pojednostavili arhitekturu jedne moderne mobilne aplikacije, na osnovu analize trenutnog stanja u razvoju mobilnih aplikacija i analize tržišta. Kao što se da primetiti na *App Store-u i Google Play-u* 80% aplikacija je besplatna. Prihod se ostvaruje kroz reklame koje se serviraju do korisnika. Takođe, prate se i navike korisnika koje se kasnije prodaju velikim trgovinskim lancima kako bi poboljšali prodaju. Dakle bez obzira šta je namena aplikacije ona u sebi ima razmenu podatka sa svojim dodeljenim serverom. Podaci koji se razmenjuju se serijalizuju i kao takvi razmenjuju. Tako gledano aplikaciju možemo podeliti u tri bloka: GUI blok (grafički korisnički interfejs), Procesni blok i Komunikacioni blok.

GUI blok ima namenu da prezentuje podatke i prikupi podate od korisnika. Procesni blok sve povezuje u jednu celinu, vrši obradu podataka i raspoređuje ih u druge blokove. Kao sto smo već rekli sama aplikacija može da komunicira isključivo sa dodeljenim serverom. Komunikacioni blok je zadužen za to. Pomoću nekog od modema (*WiFi* ili mobilni prenos podataka) on prima ili šalje podatke do servera. Ovako pojednostavljeni model aplikacije nam olakšava pristup u istraživanju i uticaju serijalizacije na energetsku efikasnost. Na Slici 16 se vidi blok šema.

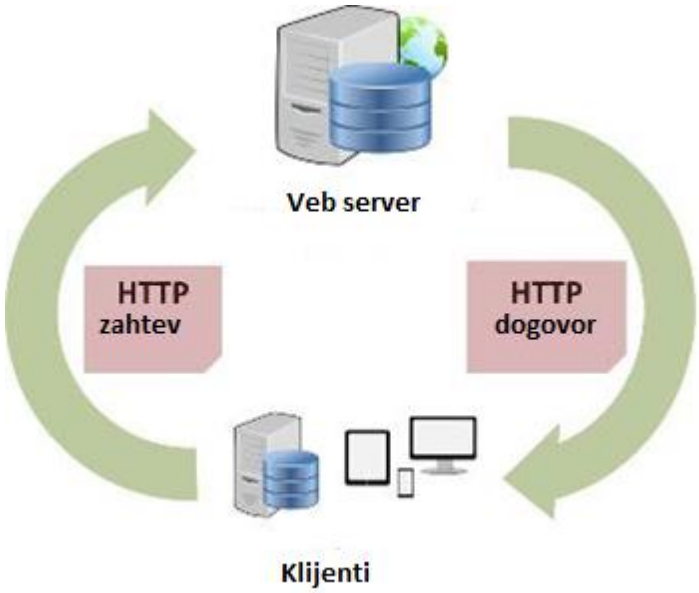


Slika 16 Blok šema mobilne aplikacije

4.2 REST komunikacija u savremenim aplikacijama

Način REST komunikacije postao je dominantan kod postojećih mobilnih aplikacija ali i *frontend* veb aplikacija koje su u ekspanziji. Može se reći da se sada REST izdvojio kao dominantan mrežni servis, koji je čak potisnuo SOAP i WSDL servise upravo zbog jednostavnosti korišćenja [33] [34]. Procena naučnika ali i velikih poslovnih kompanija je da će 2020. godine uređaji povezani sa internetom doživeti ekspanziju IoT (*Internet of Things*) [35]. Gledano sa stanovišta razvojnih inženjera i programera, REST komunikacija se nameće kao izuzetno rešenje za komunikaciju. Moraju se zadovoljiti zahtevi da komunikacija bude mala i brza, da se ne troši previše resursa na kodiranje i dekodiranje poruka i da sama programska implementacija ne troši prevelike resurse, kako RAM memorije tako i stalne memorije uređaja. Dakle, u ovom radu se podrazumeva da Komunikacioni blok uprošćenog modela aplikacije koristi upravo REST komunikaciju za razmenu

podataka sa serverom. Način razmene podataka je sledeći, klijentska strana (u našem slučaju mobilni uređaj) inicira komunikaciju prema serveru slanjem REST zahteva sa podacima na osnovu kojih će server po već otvorenom kanalu odgovoriti klijentu. U odgovoru se nalaze podaci koji su serijalizovani na serveru. Nakon prijema podataka veza se zatvara i po potrebi se proces ponavlja na isti način. Ovako podrazumevamo da komunikacioni blok radi i da za komunikaciju koristi 3G modem.



Slika 17 Dijagram toka REST komunikacije između klijenta i server

4.3 Završne napomene poglavlja

U ovom poglavlju smo pojasnili način funkcionisanja modernih mobilnih aplikacija pretežno u pogledu razmene podataka sa serverom koji je bitan za ovaj rad. Korišćenjem uprošćenog blok modela možemo se usredsrediti na sam tok komunikacije i analize uticaja serijalizacije na celokupnu energetsku efikasnost. Tako da, već u ovom trenutku imamo formiran zadatak koji trebamo izvršiti, trebamo meriti potrošnju 3G modema tokom REST zahteva i odgovora servera. Tako dobijene rezultate treba analizirati i locirati mesto na kome se oslikava uticaj serijalizacije

na potrošnju. U narednom poglavlju detaljno će biti opisani oprema i načini eksperimentalnog merenja.

5 V Deo

U ovom poglavlju ćemo se baviti postavkom opreme za testiranje, okruženjem i spiskom testova i načina na koje ćemo ih izvršiti. Na osnovu analize pređašnjih radova i teorijske analize u prethodnim poglavljima možemo postaviti okruženje koje će zadovoljiti uslove potrebne za dalje istraživanje :

- Obezbediti što preciznije merenje potrošnje 3G modema u toku testova
- Obezbediti što preciznije merenje potrošnje procesora u toku testova
- Obezbediti sistem za akviziciju merenja potrošnje usklađen sa trenutnim stanjem testa
- Obezbediti plan testova i bazu podataka za testiranje

Potrebe testa možemo podeliti u tri celine: fizičko merenje, sinhronizacija podataka i testovi.

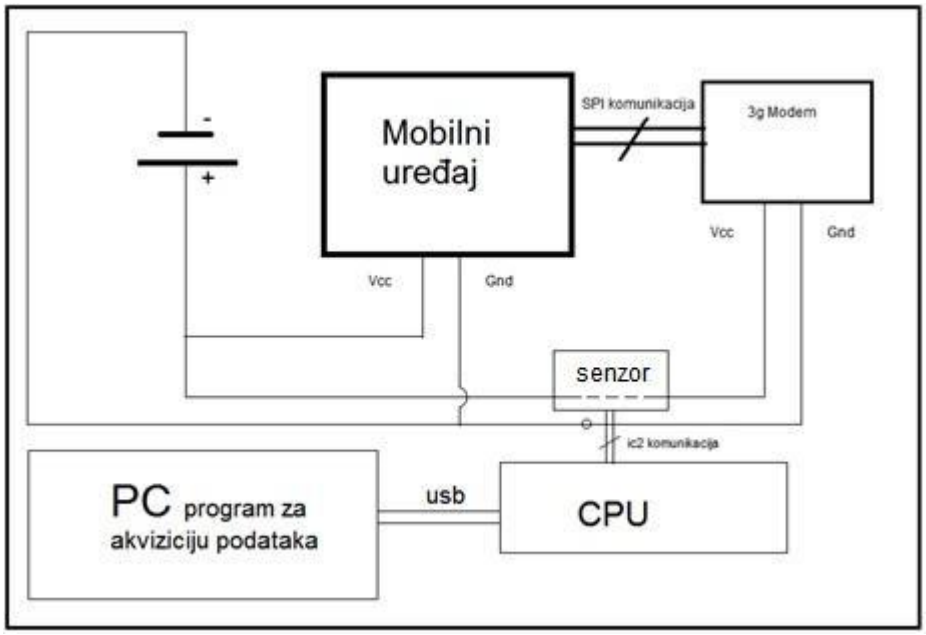
5.1 Fizičko merenje potrošnje

Kao što smo mogli videti autori prethodno analiziranih radova koristili su dva načina za merenje potrošnje:

- Senzor ugrađen u sam uređaj
- Eksterni senzor

Neki od autora su svoja istraživanja bazirali na merenju kapaciteta baterije. Na bazi karakteristike pražnjenja baterije koja nije linearna i nezavisna od grupe procesa i stanja koje nismo u stanju da kompenzujemo i izolujemo možemo sa sigurnošću tvrditi da je korišćenje eksternog senzora preciznije i merodavnije. Još jedna od specifičnih problema koji se moraju predvideti u okviru našeg istraživanja su izolovanje dve komponente koje su najzastupljenije u procesu razmene serijalizovanih struktura podataka između server i mobilnog uređaja. Već smo videli da su to procesor i mobilni mrežni modem. RAM memorija koja je takođe opterećena u ovom procesu nema veliki uticaj na potrošnju. Procesor mobilnog uređaja je aktivnostivan tokom serijalizacije i deserijlaizacije a modem tokom razmene podataka.

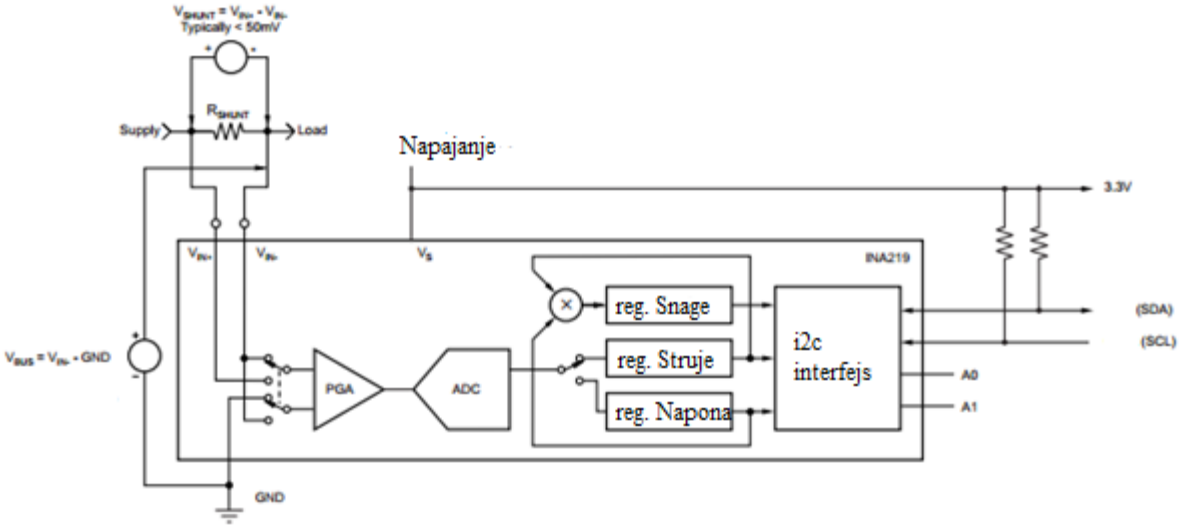
Problem sa kojim se srećemo tokom ovog istraživanja je merenje potrošnju mrežnog modema jer je hardverski upakovan u matičnu ploču uređaja i teško mu je prići. Kod novijih tipova mobilnih uređaja sa Android operativnim sistemom i Windows operativnim sistemom moguće je koristiti eksterne 3G modeme. Ideja je da se za komunikaciju koriste eksterni 3G modemi povezani putem USB OTG konekcije. Na ovaj način se na USB kablu za spajanje modema i uređaja može lakom intervencijom postaviti uređaj za merenje. Ovo je već viđeno rešenje u prethodno analiziranim radovima. Ovakvim konceptom dobijamo mogućnost da merimo potrošnju celog uređaja i potrošnju samog modema. Sa dva uređaja možemo lako izolovati potrošnju CPU-a sa ostalim komponentama i 3G modema i vršiti upoređivanje. Na sledećoj slici se vidi dijagram eksperimentalne konfiguracije



Slika 18 Šematski prikaz sistema za merenje

Na ovom mestu se treba definisati uređaj za merenje potrošnje. Analizom dostupnih uređaja na tržištu i mogućnostima odabran je uređaj baziran na *INA 219 IC* . Njega odlikuje preciznost od 0.1 mA i mogućnost merenja intenziteta struje u amperima ali i utrošene energije takođe preračunate u vatima. Ovaj uređaj pored merenja ima opciju komunikacije sa mikroprocesorskim jedinicama što nam je potrebno kako bi dobili sinhronizovane tabele potrošnje. Za razliku od drugih uređaja on ne radi na holovom efektu već ima otpornik sa operacionim pojačavačem što ga čini veoma

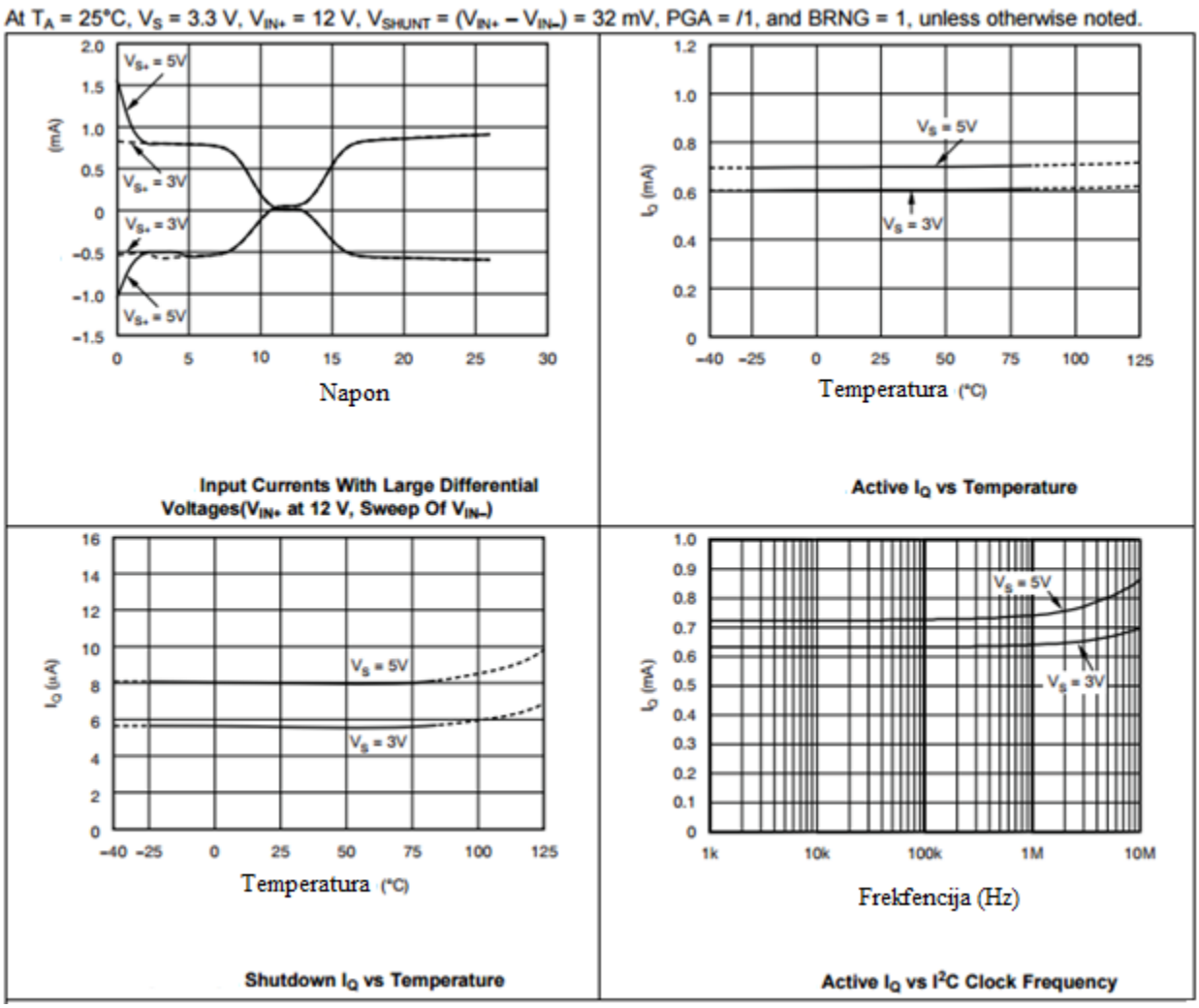
preciznim i osetljivim. Ovo je jako bitno u našem istraživanju kako bi smo dobili što validnije podatke.



Slika 19 Šematski prikaz INA 219 ***preuzeto od*** *[36]*

Specifičan je i po tome što struja koja se meri prolazi preko otpornika od 0.1 oma čiji se napon meri preciznim operacionim pojačavačem. Kako je razlika napona na operacionom pojačavaču +- 320 mV to znači da se može meriti struja u opsegu od 3.2 A. U okviru IC se nalazi i 12-to bitni analogno-digitalni konverter. Sličan način merenja je korišćen u nekim od pomenutih radova. [27] Oni su postavljali otpornik na kome su merili pad napona. Uređaj za merenje baziran na IC INA 291 predstavlja prilično komforno rešenje iz sledećih razloga :

- Visoka preciznost
- Smanjen uticaj šuma
- Analogno digitalna konverzija u okviru jednog uređaja
- Trenutno računanje utrošene snage
- Visoka brzina akvizicije 532 mikro sekunde što je više nego dovoljno za nas test jer sve bitnije operacije tokom testova koje će meriti preko 10 milili sekunde.



Slika 20 Karakteristika INA 219 ***preuzeto do*** *[36]*

Na osnovu dijagrama koje je dao proizvođač možemo videti da IC ima prilično konzistentne rezultate zavisno od uslova eksploatacije.

Primena INA 291 IC je kod :

- Servera
- Telekomunikacione opreme
- Punjača baterija
- Laptop računara
- Napajanja
- Opreme za testiranje

5.2 Očitavanje merenja

INA 291 IC poseduje I2C komunikaciju, tako da se preko ovog protokola čitanjem registra može trenutno pročitati merena vrednost: struja i naponi na otporniku-šantu. Sam uređaj INA 291 IC nije dovoljan da bismo mogli da prikupimo podatke. Dakle INA 291 IC će digitalizovati i sačuvati samo jednu - trenutnu vrednost. Za potrebne našeg eksperimenta potrebno je da imamo seriju snimaka koju moramo uskladiti sa trenutnim operacijama i stanjima u toku testa. Za to je potrebno da imamo još jedan uređaj koji će biti interfejs između uređaja za merenje i PC-ja na kome se nalazi program za analizu. S obzirom da INA 291 komunicira putem I2C protokola a sam PC nema tu opciju za eksperiment odabrali samo *Arduino* razvojno okruženje. Konkretno, upotrebili smo *Arduino* DUE kao najmoćnije razvojno okruženje iz familije Arduino. Svojim performansama ono prelazi potrebe našeg testa. *Arduino* Due na sebi poseduje 32- bitni procesor koji radi na 84 Mhz. Opremljen je sa više serijskih portova tako da može komunicirati sa više uređaja istovremeno.

Uloga ove komponente je da sa INA219 uređaja putem I2C očita vrednost struje koju koristi određena komponenta a zatim je skladišti u tabelu merenja koja se nalazi na PC-ju. Pored toga, uz svaku izmerenu vrednost, program koji se izvršava na *Arduninu* pakuje vreme od početa testa u ms, zatim tip merenja i broj kanala koje se meri. Potrebno je i da se sa mobilnog uređaja čita i trenutno stanje i programska komanda testa koja se trenutno izvršava. Uz pomoć ovako sakupljenih informacija možemo izvršiti sve potrebne analize.

Program na mikro kontroleru ima zadatke da:

- Na svaku 1ms meri vrednosti napona i struje sa uređaja INA 291
- Putem serijske veze zahteva stanje testa od test uređaja (trenutnu programsku instrukciju)
- Šalje kompletiranu informaciju na PC za analizu

5.3 Struktura test programa

Pod pojmom test programa podrazumevamo skup programskih instrukcija na veb serveru, test aplikaciju na mobilnom uređaju i odgovarajući set podataka nad kojima će se vršiti serijalizacija i deserijalizacija, kao i transport podataka od servera do mobilnog uređaja i obratno. Uloga test programa je da za zadatu komunikaciju tipa serijalizacije i seta podataka angažuje hardverske resurse mobilnog uređaja. Angažovanjem procesora i 3G modema u toku ovih testova putem sistema za merenje potrošnje struje možemo utvrditi uticaj tipa serijalizacije na potrošnju bateriju. Analizom tako dobijenih podataka donećemo zaključke i kreirati modele.

5.3.1 Test program na mobilnom uređaju

Test program na mobilnom uređaju ima dvojaku ulogu. Prva uloga je da angažuje resurse uređaja tokom serijalizacije kroz sledeći set operacija :

- Slanje zahteva serveru za prijem seta podataka
- Prijem serijalizovanih podataka od strane servera
- Raspakivanje deserijalizacija primljenog seta
- Slanje podataka na server

Za svaki od testova menjaće se tip serijalizacije i set podataka.

Druga uloga test programa na mobilnom uređaju je da pre početka svakog bloka instrukcija i njegovog završetka pošalje kod operacije na uređaj za snimanje potrošnje koji će sinhronizovati trenutno merenje i operaciju. Tokom implementacije test programa mora se voditi računa da operacije koje esencijalno nisu vezane za proces serijalizacije i deserijalizacije troše zanemarljivo male resurse u odnosu na one čiji uticaj merimo. To je i slučaj sa slanjem poruka za sinhronizuju sa uređajem za merenje. Zbog tehnoloških postupa koji se mogu izvesti a i malog uticaja na resurse mobilnog uređaja koristićemo serijsku komunikaciju. Putem serijske komunikacije i OTG opcije na mobilnom uređaju na mikroprocesorskoj Arduino ploči slaće se dvobajtna poruka o statusu naredbe koja se u tom trenutku izvršava. Na taj način merenje će biti potpuno sinhronizovano. Na sledećim slikama možemo videti pseudokod test programa:

```
For test type in [ ]  // svi tipovi testova
        While(testNum) // ponovalnje testova uzastopno zbog preciznosti
                SincMsg(01); //slanje poruke za sinhronizaciju
                SendHtppRequest(t); // slanje yahteva za serveru
                SincMsg(11);
                WaitForResponse();//
                SincMsg(02);
                Deserialise() ;// deserialiyzacija poruke dobijene od servera
                SincMsg(12);
                SerialiseData(); //pakovanje podataka
                SincMsg(03)
                SendDataToSrv() // slanje na server
                SincMsg(13)
```

5.3.2 Struktura test programa na veb serveru

Uloga test programa na veb serveru je da detektuje tip testa u zavisnosti od tipa istog koji se izvršava na mobilnom uređaju. Nakon detekcije tipa testa automatski pakuje određeni set podataka i koristi određeni tip serijalizacije kako bi opslužio test aplikaciju klijenta.

Pod tipom testa podrazumevamo tip serijalizacije koji se koristi u komunikaciji i set podataka nad kome se vrši serijalizacija. Tip testa takođe poseduje i podatke o veličini seta. Testovi su predviđeni tako da se veličina seta podataka sa svakim novim testom povećava. Takvim postupkom se dobije finija zavisnost merenja od veličine seta podataka.

Pod pojmom test programa na veb serveru se podrazumeva PHP skript i *MySql* baza sa setovima podataka. Kada test program sa mobilnog uređaja pošalje zahtev veb serveru za početak testa, kroz taj zahtev klijent predaje serveru parametar o tipu testa. Na osnovu tog parametra server zna koji set podataka da koristi tj. iz koje tabele u bazi da preuzme podatke. Pored toga u okviru tog

parametra se nalazi i veličina seta koji se traži. Dakle u trenutku kada program determiniše set i veličinu podataka php program vrši upit u određenu tabelu test baze i zahteva određeni broj redova iz tabele. Nakon serijalizacije podataka server šalje podatke nazad klijentu tj. test programu na test uređaju.

5.3.3 Test podaci

Na osnovu prethodno razmatranih sistema za serijalizaciju može se zaključiti da struktura i tipovi podataka koji se serijalizuju mogu imati uticaj na serijalizaciju i deserijalizaciju. Ta razlika posebno dolazi do izražaja kada se radi o brojevnim tipovima podataka. Kao bi se uticaji izmerili na najprecizniji način koristićemo tri seta podataka: tekstualni, mešoviti i brojevni.

Tekstualni tip set podataka predstavlja najčešće korišćeni set. Konkretno u našem testu ovo će biti baza sa tipičnim podacima korisnika. Dakle ime, prezime, broj telefona, adresa, e-pošta i datum rođenja. U ovakvom setu podataka većina je običan test –string. Čak i podaci koji su brojevnog karaktera kao što su broj telefona ili datum rođenja su stringovi jer se ne mogu predstaviti kao prirodni brojevi .

Mešoviti set podataka je set podatak gde polovina može biti predstavljena kao stringovi a druga polovina kao prirodni brojevi. Nas test će predstavljati tabelu geo lokacija. Dakle geografska širina, geografska dužina, naziv objekta i vreme kada je uočen.

Numerički set podataka je set u kome preovladavaju polja koja se mogu predstaviti putem prirodnih brojeva bilo celih bilo racionalnih brojeva.

5.3.4 Program za analizu podataka

Program za analizu podataka ima dve uloge. Prva je da uskladišti podatke dobijene merenjem i da ih sinhronizuje sa test programom u posebne tabele. Druga uloga je da na osnovu tih tabela kreira grafike i funkcije zavisnosti.

Svako merenje ima svoj identifikacioni broj, vreme i test kome pripada kao i segment programa u kome je izvršen. Podaci se unose u relacionu bazu podatka kako bi se lakše koristili nakon

prikupljanja. Na ovakav način bićemo u mogućnosti da koristimo sve matematičke metoda MySql paketa. Dobićemo automatizam prilikom analize podataka.

5.4 Analiza REST komunikacije

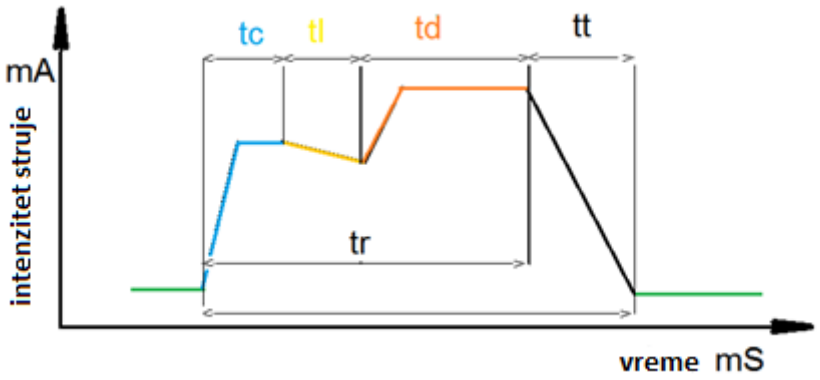
Kao uvod u analizu uticaja tipova serijalizacije na potrošnju električne energije u mobilnom uređaju potrebno je prvenstveno analizirati faktore koji utiču na potrošnju električne energije tokom REST komunikacije. Podsetićemo, REST komunikacija je odabrana kao vodeći vid komunikacije između mobilnih uređaja i veb servera. Analiza uticaja tipova serijalizacije se zasniva na pretpostavci da za razmenu podataka koristimo REST način u kome će se sadržati podaci serijalizovani nekim od vidova serijalizacije.

5.4.1 Softverski segmenti u toku REST komunikacije

Kada bi se REST komunikacija posmatrala kroz set instrukcija u najvišem programerskom sloju u okviru komunikacionog bloka, onda bi mogao da se izvede sledeći zaključak: segmenti u životnom veku jedne REST komunikacije su :

- period konekcije na server (t_c)
- period slanja zahteva (t_p)
- period čekanja na odgovor (t_l)
- period prijema podatka (t_d).

Grafički prikaz se može videti na sledećoj slici Slika 21



Slika 21 Segmentirani prikaz energetskog profila REST zahteva

Dakle vreme tc predstavlja vreme od otvaranja *socket*-a i uspostavljanja veze prema serveru koju je on prihvatio i slanje zahteva serveru. Pod zahtevom podrazumevamo sve preambule i segmente zaglavlja jednog HTTP zahteva.

```
GET /?a=1&b=2 HTTP/1.1
Host: localhost
User-Agent: my browser details
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Segment t_l predstavlja vreme koje prođe od trenutka kada server primi zahtev i početka slanja odgovora nazad klijentu. Konkretno ovo vreme može predstavljati vreme pretrage baze podatka, (obrade podatka) i vreme pakovanja podataka. U drugim slučajevima može prestavljati obradu i prepoznavanje govora, obradu slike ili rad neke navigacije. Segment t_d je vreme potrebno da klijent primi sve podatke koje mu je server prosledio. Ono je direktno srazmerno brzini komunikacije i veličini podataka.

Ovakav način podele je ustanovljen radi analize uticaja tipova serijalizacije na utrošak električne energije. Programer tokom kreiranja aplikacije i serverskog programa na ove pod segmente može imati direktan uticaj putem standardnih funkcija u većini programskih jezika. Shodno tome, na

osnovu ovih merenje može se doći do direktno primenljivih zaključaka u savremenom softverskom inženjerstvu.

5.4.2 Sistem za snimanje profila REST komunikacije

U prethodnom poglavlju je već objašnjen koncept aparature za merenje protoka struje tokom komunikacije mobilnog uređaja i servera na 3G modemu. Na ovom mestu biće objašnjen koncept test programa namenjen samo analizi REST komunikacije.

Pored već naglašenih problema fizičkog merenja i predloga potencijalnog rešenja prikazanog na slici Slika 18 , potrebno je formirati i softversko test okruženje. Pod pojmom test programa podrazumeva se skup instrukcija na veb serveru, test aplikacija na mobilnom uređaju, odgovarajući set podataka kao i izvršavanje transporta podataka od servera do mobilnog uređaja i obratno. Uloga test programa na mobilnom uređaju je mogućnost podešavanja određenih kombinacija parametara REST komunikacije. Nakon toga test program na mobilnom uređaju otpočinje REST komunikaciju sa definisanim serverom. U okviru slanja REST zahteva serveru se šalju prethodno zadati podaci, a to su podaci vezani za željenu veličinu podatka koji se povlače sa servera kao i vreme potrebno serveru da obradi zahtev. Vreme obrade servera je simulacija kašnjenja odgovora servera od trenutka kada server prihvati REST zahtev do trenutka kada odgovori. Server će za zadato vreme odložiti odgovor uspavljivanjem programske niti u kojoj je prihvatio pomenutu konekciju.

Test program na mobilnom uređaju ima zadatak da u lokalni CSV zapis upisuje vreme od početka testa, zatim šifru trenutne operacije koju izvodi i vrednost koju je pročitao sa mikro kontrolera koji putem senzora meri intenzitet protoka struje u napojnim vodovima 3G modema. Akvizicija se vrši na svakih 5 ms . Upisivanje u CSV zapis se ne vrši na svaku akviziciju već se podaci sakupljaju u listu, a nakon završenog REST zahteva se ta lista dodaje u lokalni zapis. Na taj način može se izbeći usporavanje i opterećivanje mobilnog uređaja tokom samog procesa REST komunikacije.

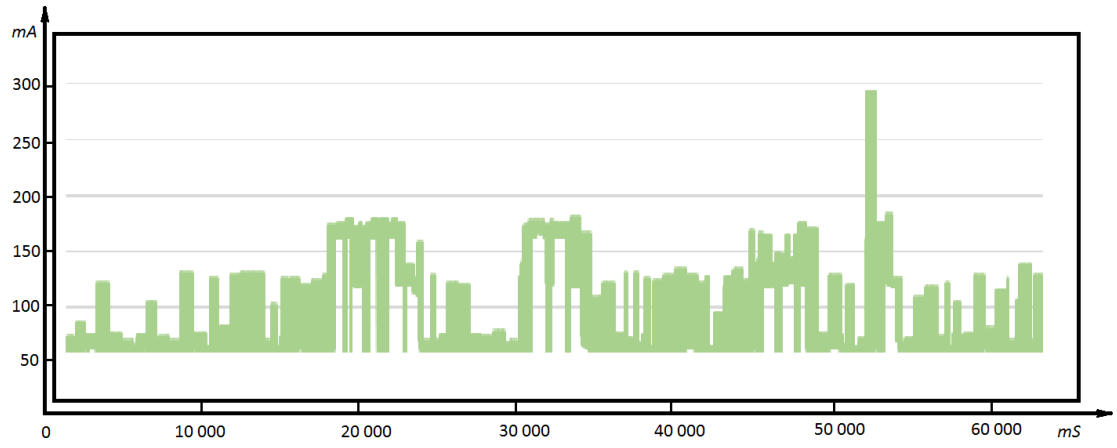
Test program mora posedovati sledeća podešavanja za promenu parametara testa: veličina podataka koja se zahteva od servera, vreme u sekundama između dva testa , broj ponavljanja REST zahteva tokom testa kao i vreme koje je serveru potrebno da obradi zahtev.

5.4.3 Rezultatu merenja

Naredni deo ovog poglavlja se bavi analizom rezultata i toka eksperimenta. Pre početka testova REST zahteva izmerena je potrošnja struje u stanju mirovanja kad je modem u aktivnom stanju ali su potisnuti svi servisi na klijentskom uređaju koji koriste internet. Nakon toga su testirani REST zahtevi sa promenom veličine podataka koju šalje server i simulacijama vremena obrade servera.

5.4.3.1 Stacionarno stanje

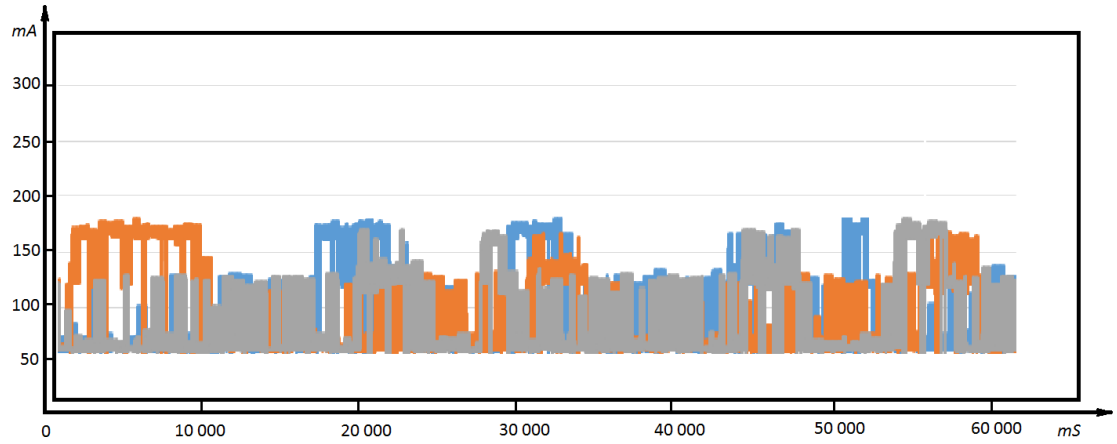
Kako bi se bolje upoznali sa prirodom 3G modema koji se koristi u istraživanju izvršena su merenja potrošnje električne energiju u stanju kada nema mrežne aktivnosti od strane klijentskog mobilnog uređaja. Ovo merenje je jako bitno jer predstavlja osnovu za analizu daljih merenja.



Slika 22 Snimak energetskog profila modema u stacionarnom stanju

Ovom prilikom je utvrđeno da potrošnja modema u takozvanom stacionarnom stanju ima intenzitet od 78 mA u proseku sa povremenim stanjima povišenog intenziteta gde se protok struje prema modemu kreće do 160 mA. Snimljeno je 30 sekvenci u različitim delovima dana sa iste lokacije u trajanju od 10 minuta. Na grafiku je prikazano više različitih sekvenci u trajanju od 60 s. Ova merenja pokazuju da modem u stanju bez aktivnosti, prouzrokovane nekom od korisničkih klijentskih aktivnosti, ima prosečnu potrošnju od 78mA i ima povremene skokovite promene

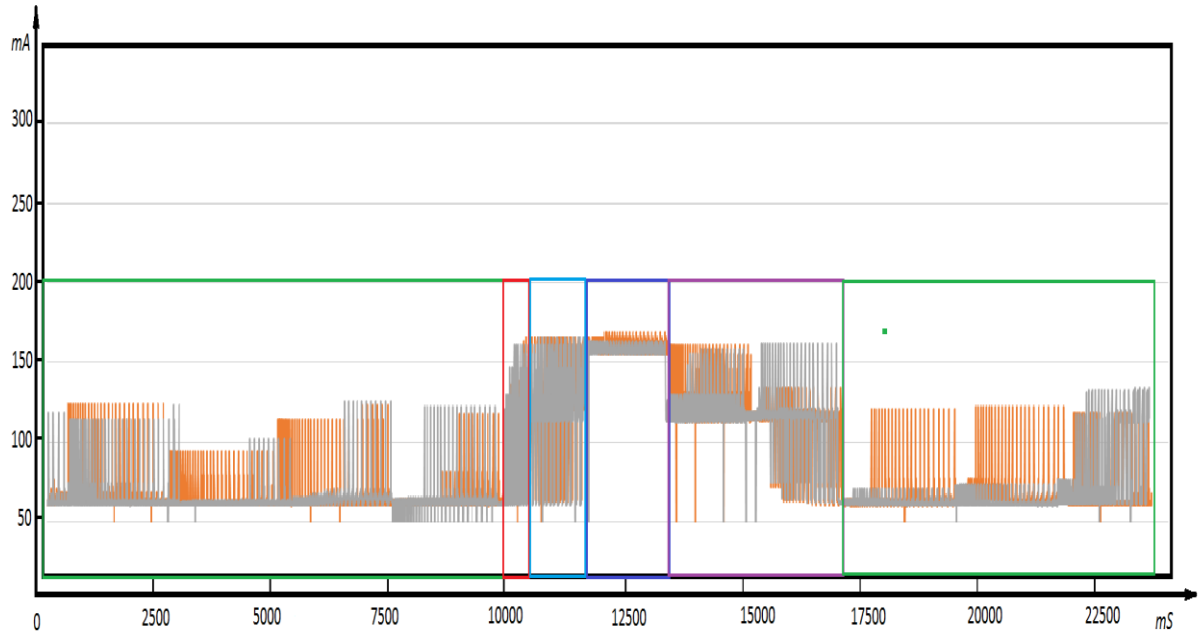
izazvane sopstvenom aktivnošću. Grafici ovih stacionarnih stajna služe u analizi profila komunikacije i upoređivanju aktivnih i neaktivnih stanja.



Slika 23 Prikaz energetskih nivoa stacionarnih stanja

5.4.3.2 Profil REST

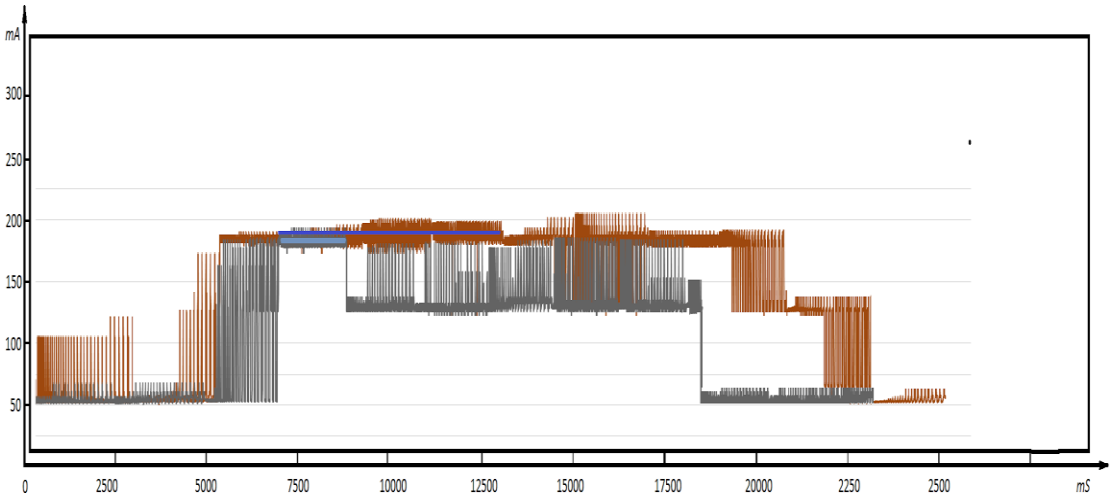
Snimanje energetskog profila REST zahteva predstavlja proces uporednog slanja test zahteva promenljivih parametara na serverski test program i akvizicije potrošnje električne energije modema u zavisnosti od stanja test programa klijenta.



Slika 24 Segmentirani energetski profil dve identična REST zahteva

Zahvaljujući gorepomenutom sistemu za sinhronizaciju merenja struje u napojnim vodovima modema i trenutne programske instrukcije koja se izvršava, sa lakoćom se mogu odvojiti segmenti REST zahteva. Slika 22 (zelena boja predstavlja stacionirano stanje, crvena početak konekcije, svetlo plava slanje podataka serveru, tamnoplava prijem podataka i ljubičasta zakasneli odziv ili repnu potrošnju [11]) Na grafiku se vide dva merenja identičnog REST zahteva koji su poslati u različitim trenucima. Na slici se nedvosmisleno vidi poklapanje svih segmenata i intenziteti struje koja napaja modem u tim trenucima. Ono što se razlikuje ova dva merenja se može pripisati ponašanju modema u stacionarnom stanju koje smo prethodno analizirali. Na grafiku se mogu jasno prepoznati segment konekcije, slanja zahteva, prijema zahteva ali i stanja koje ima povišenu energetsku vrednost, a da aplikacija nema u tom trenutku mrežne aktivnosti. Ovo stanje se javlja gotovo svaki put nakon aktivnosti 3G modema i traje do 5s. Po energetskom intenzitetu je manja od bilo kog segmenta aktivnog stanja ali je i znatno povišena u odnosu na stacionarno energetsko stanje.

U daljem toku istraživanja menjani su parametri testa tako da su od strane servera zahtevani postepeno obimniji setovi podataka.



Slika 25 Više različitih REST zahteva

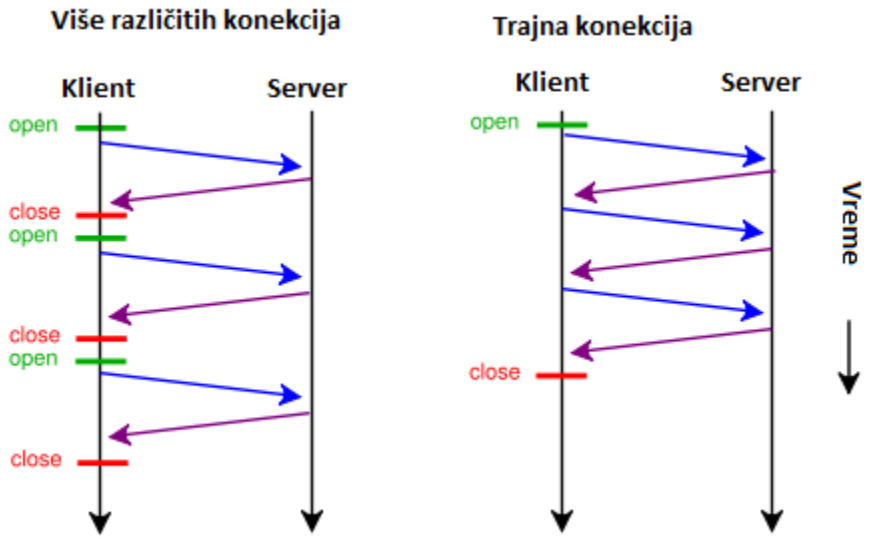
Na slici Slika 25 se nalazi uporedni prikaz dva energetska profila REST zahteva. Oba zahteva su generisana test programom sa mobilnog uređaja i poslata na isti test server. Razlika je u količini podataka koja je zahtevana od servera. Prvi zahtev je od servera potraživao 2kb podataka dok je drugi potraživao 20kb. Veličina podataka upućena kroz zahtev serveru je identična. S obzirom na mogućnosti test okruženja da pored akvizirane vrednosti prikaže i stanje (operaciju) koja se trenutno izvršava, na grafiku možemo razdvojiti već pomenute segmente i analizirati uticaj veličine podataka koje transportujemo na potrošnju energije. Shodno tome na grafiku vidimo da se početak komunikacije i slanje zahteva poklapaju, dok stanje nakon početak prijema podataka pravi razliku. Ta razlika se ogleda u dužini trajanja stanja prijema dok su energetski nivoi nepromenjeni. Odavde se može zaključiti da je potrošnja srazmerna veličini podataka. Matematički gledano potrošnja se može izraziti kao funkcija proizvoda vremena i eksperimentalno dobijene vrednosti za testirani uređaj. Gde je vreme provedeno u stanju prijema funkcija veličine podataka i intenziteta protoka podataka (stanja mreže).

5.4.3.3 Kašnjenje servera

U cilju analize utroška električne energije tokom komunikacije analiziran je i uticaj vremena obrade zahteva od strane servera na potrošnju električne energije klijenta. Krenulo se od pretpostavke da je server od trenutka prijema zahteva pa do trenutka slanja odgovora klijentu utroši

izvesno vreme. Ovaj vremenski period može biti u opsegu vrednosti, od nekoliko milisekundi pa do 10 i više sekundi zavisno od koncepcije sistema i prirode obrade koju server vrši. Par milisekundi ako se radi o prostom upisu u bazu koja je sa malo podataka do desetine sekundi ako se radi o nekom proračunu složenom upitu u veliku bazu i generisanju kompresovanih podataka. Na osnovu ovih pretpostavki i činjenica javlja se potreba za analizom utroška energije 3G modema tokom stanja održavanja konekcije prema serveru i osluškivanju akcije servera i slanja odgovora.

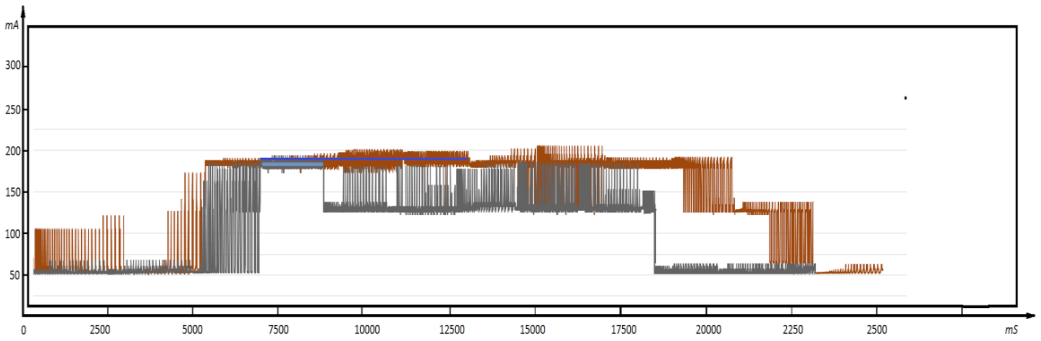
U ovom trenutku kao i tokom cele konekcije klijent i server na niskom nivou TCP protokola moraju da periodično razmenjuju podatke kako bi održali konekciju otvorenom. Ovi podaci služe za sinhronizaciju klijenta i servera. Tačnije, klijent je dužan da stalno proverava status servera slanjem malih paketa koje ne stižu do aplikativnog softvera već se obrađuju na niskim nivoima tcp protokola.



Slika 26 Aktivnost mreže tokom održavanja konekcije

Ova razmena kao i svaka druga podiže aktivnost 3G modema i on se ne vraća u stanje mirovanja. Kako bi se ovo ispitalo i dokazalo, pomoću pomenutog okruženja za testiranje i test programa, snimljeni su profili REST komunikacije sa varijabilnim kašnjenjem servera. Na slici Slika 27 se može videti energetski profil jedne REST komunikacije gde je vreme kašnjenja servera

postavljeno na 3s. Ovaj segment je označen na slici i nedvosmisleno se primećuje energetska aktivnost modema u periodu dok na izgled nema softverskih aktivnosti na aplikativnom nivou.



Slika 27Rest energetski profili sa velikim vremenom kašnjenja servera

U daljem istraživanju je nastavljeno analiziranje odziva i merenje potrošnje sa postepenim povećavanjem vremena kašnjenja odgovora servera. Na slici Slika 27 se vidi utrošak energije ovog segmenta u zavisnosti od vremena kašnjenja.

Sa dobijenih snimaka i grafika se zaključuje da je potrošnja u toku segmenta čekanja prilično izražena i da se, u kratkim intervalima do 3s, za testirane uređaje ne razlikuje puno od potrošnje u aktivnom stanju tokom prenosa. Nakon 3s potrošnja počinje da opada ali ne do nivoa stacionarnog stanja, već u ponovnim etapama od po dve sekunde gde u prvoj pada na 120mA a u sledećoj na potrošnju stacionarnog stanja. Dakle karakteristika potrošnje u stacionarnom stanju ima stepenasti profil.

Na osnovu ovog zapažanja o kašnjenju odgovora servera, dolazi se do zaključka da je sam server (pod time podrazumevamo softversko rešenje koje opslužuje korisnika) ima veoma veliku ulogu u potrošnji električne energije klijenta. Ovaj zaključak je bitan za istraživanje o uticaju serijalizacije na potrošnju električne energije mobilnog uređaja, podjednako kao i uticaj veličine podatka. Analiziranjem postojećih aplikacija koje imaju komunikaciju sa serverima vidi se da većina njih potražuje podatke po složenim bazama podataka i upisuje podatke. Dakle, ovo vreme kašnjenja je vreme pretrage baze ali pored toga i vreme koje će proces serijalizacije ali i deserijlizacije na serveru utrošiti. Samim tim algoritam serijalizacije i deserijalizacije ima uticaj

na klijentski uređaj. Pored serijalizacije tu je i proces kompresije podataka. O ovom je već bilo reči u prethodnim poglavljima.

5.4.3.4 Zakasnela aktivnost modema

Tokom eksperimenta je primećeno da je nakon svake od aktivnosti modema tokom REST komunikacije izvesno vreme postojala padajuća karakteristika energije modema. Svaka Rest komunikacija se određeno vreme smirivala i vraćala u stacionarno stanje. Na osnovu različitih merenja i eksperimenata, ali i korišćenja BLACK BOX modela za traženje korelacije između parametara testa i ove karakteristike, zaključeno je da se ona ponaša prilično konstantno i da nema uticaja na nju tokom konstante brzine interneta. Zbog toga je ona samo pridodata matematičkom modelu u vidu konstante.

Daljim eksperimentalnim radom je snimljeno po 30 REST energetskih profila gde se u svakoj seriji menjala količina podatak zahtevana od servera. Na taj način je dobijena prosečna energija i vreme koje se provede u stanju prijema podataka. Rezultati se mogu videti u tabeli Tabela 6. Pri svakom setu je vođeno računa da je stanje mreže konstantno i idealno. Drugim rečima: rezultati koji su snimljeni kada je brzina interneta odstupala od standardne nisu uzeti u razmatranje.

Naziv segmenta	Posečen intenzitet [mA]
Konekcija	123
Slanje zahteva	162
Čekanje na dogovor	140
Prijem podatka	164
Tail	123

Tabela 6 Prosečne vrednosti intenziteta struje u segmentima

5.4.5 Matematički model

Matematički model REST komunikacije se ogleda u matematičkoj funkciji koja u zavisnosti od parametara REST komunikacije: intenziteta protoka podataka, veličine podataka zahteva (slanja), veličine podataka prijema, vremena kašnjenja servera daje aproksimativnu energiju koju je modem utrošio tokom jedne uspešne razmene sa serverom. Na osnovu eksperimentalnih podataka o prosečnoj potrošnji tokom svakog od segmenata komunikacije možemo na osnovu trajanja svakog stanja i prirode potrošnje u tom stanju izračunati potrošnju. Pored prosečne potrošnje tokom tih stanja potrebna nam je i zavisnost vremena provedenog u stanjima od brzine protoka podataka i veličine podataka.

Iz prethodno navedenog, ukupni utrošak energije tokom REST zahteva može se predstaviti sledećim matematičkim modelom:

$$E[Ws] = (a_c * t_c + a_p * t_p + a_l * t_l + a_d * t_d) * U + E_t \qquad (1)$$

Gde je:

$t_d = \frac{D_s}{V_d}, t_p = H_s/V_u,$

a_p – Prosečni intezitet struje u napojnim vodovima modema tokom slanja zahteva serveru

a_c - Prosečni intezitet struje u napojnim vodovima modema tokom konekcije,

t_c - Vreme provedeno tokom konekcije,

H_s - Veličina POST zaglavlja u bajtovima,

V_u - Brzina slanja podatka u bajtovima po sekundi,

a_l - Prosečni intezitet struje u napojnim vodovima modema tokom čekanja na odgovor servera,

t_l - Vreme čekanja na odgovor od servera,

a_d - Prosečni intezitet struje u napojnim vodovima modema tokom prijema podatka,

D_s - količina podataka koja se prima od servera u bajtovima,

U - Napon napajanja,

V_d - Protok podataka u bajtovima po sekundi,

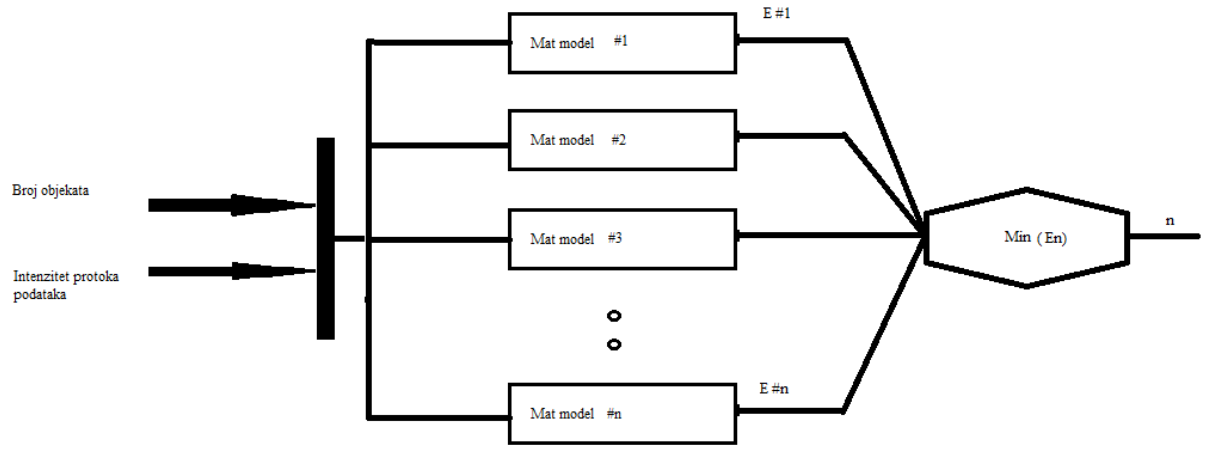
E_t – Energija koju modem troši nakon prekida komunikacije.

Na osnovu merenja i prosečnih vrednosti u izdvojenim segmentima jednačina (1) postaje:

$$E[Ws] = (0.123A * t_c + 0.163A * (t_p + t_d) + 0.140mA * t_l) * 3.7V + 2.875Ws \quad (2)$$

5.4.5.1 Uloga matematičkog modela

Uloga matematičkog modela u ovom istraživanju ima za cilj da olakša analizu uticaja serijalizacije na potrošnju baterije na dva načina. Prvi način se ogleda u tome što zahvaljujući matematičkom modelu možemo da analiziramo potrošnju u zavisnosti od parametara serijalizacije i performansi iste. Dakle, u ranoj fazi analize oslanjaćemo se na matematički model, što treba da donese vremensku uštedu i uštedu u resursima, a kasnije ćemo rezultate matematičkog modela upotrebiti u eksperimentalnom API-ju koji ima za cilj da poboljša energetsku efikasnost. Ideja ka kojoj stremimo, jeste da ceo proces prenosa objektno orijentisanih podataka predstavimo matematičkom jednačinom koja bi na osnovu stanja mreže (intenziteta protoka podatka), obima podataka, formata serijalizacije i stepena kompresije mogla da prikaže očekivanu potrošnju. Sam matematički model treba da predstavlja skup jednačina koje opisuju sve kombinacije serijalizacije i stepena kompresije gde su argumenti: broj objekata koji se serijalizuje i intenzitet protoka podataka od servera do klijenta. Vrednost koja se dobija predstavlja procenu utrošene energije modema klijentskog uređaja. Konkretno, u cilju poboljšanja energetske efikasnosti prenosa objektno orijentisanih podataka od severa do klijenta, ovaj skup jednačina treba da prikaže odnose energetske efikasnosti kod različitih tipova serijalizacije i stepena kompresije, kako bi se odabrao najpovoljniji scenario za klijentski uređaj.



Slika 28Uloga matematičkog modela u odabiru energetski efikasnog režima komunikacije

U nastavku je prikazano modelovanje jednačina koje opisuje vreme potrebno za serijalizaciju podatka i kompresiju i veličinu. Te jednačine će biti uvrštene u jednačinu (2).

5.5 Analiza procesa serijalizacije na serveru

Na ovom mestu bavićemo se performansama određenih tipova serijalizacije koje se najčešće koriste savremenom softverskom inženjerstvu. Na osnovu prethodnog poglavlja i analize REST komunikacije zaključili smo da dva parametra na koja imamo uticaja tokom komunikacije sa serverom mogu imati presudan uticaj na potrošnju energije na mobilnom uređaju. Parametri su veličina podataka koji se dovlače sa servera i vreme odziva servera.

Proces serijalizacije može da utiče na oba ova podatka. Prvenstveno, jer sirovi podaci koji se serijalizuju dobijaju dodatnu veličinu zbog specijalnih karaktera i tagova koje proces serijalizacije koristi kako bi od sirovih podataka napravio serijalizovani zapis. Postoje i tehnike kompresije koje mogu da smanje obim podata koji se prenosi. Sve pomenuto iziskuje neko vreme koje je potrebno serveru da on zapakuje podatke i pošalje ih do klijenta. To vreme jeste vreme kašnjenja servera koje je analizirano u prethodnom poglavlju. Kad se sve rečeno uzme u obzir dolazi se od zaključka da je potrebno utvrditi dve performanse. Te dve performanse su:

- stepen uvećavanja i kompresije podataka
- vremenom potrebno za izvršavanje istih.

Ovo poglavlje će obraditi najčešće formate serijalizacije koji se koriste u savremenim aplikacijama u kombinaciji sa REST komunikacijom. Na test serveru su izvršeni testovi i merenja performansi tako da se te vrednosti mogu kasnije uključiti u prethodni matematički model na osnovu kog će se prikazati indirektan uticaj procesa serijalizacije na potrošnju baterije mobilnog uređaja.

5.5.1 Testiranje

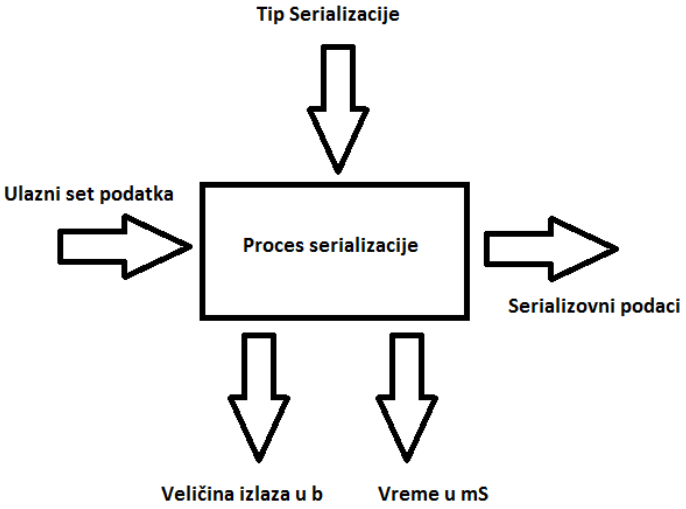
Tokom ovih testova usredsredićemo se na kombinaciju xml, jsona, php serializaicije, yaml i *msgPacket* i paketa za kompresiju *gzcompress* (programskih paketa). Test ima ulogu da na serveru na kome se vrši testiranje utvrdi efikasnost svake od komandi. Cilj testova je utvrđivanje međuzavisnost performansi različitih programskih paketa od veličine podataka. Pod performansama programskih paketa podrazumevamo veličinu serijalizovanog objekta i vreme provedeno tokom tog procesa. Isti tipovi performansi se primenjuje i na programskim paketima za kompresiju podataka.

5.5.1.1 Algoritam za testiranje

Algoritam za analizu procesa serijalizacije se sastoji iz univerzalnog seta instrukcija napisanih u PHP programskom jeziku. Ove programske instrukcije se izvršavaju na veb serveru. Set programskih instrukcija se može podeliti, na osnovu uloge u testu, na sledeće celine:

- Dopremanje podatka iz baze podatka kao objekta, unapred zadane veličine
- Serijalizacija seta objekata jednom unapred zadanom metodom
- Merenje vremena u milisekundima potrebno za izvršavanje serijalizacije seta objekata
- Merenje veličine podatka (niza bajtova) dobijenih procesom serijalizacije
- Kompresija niza bajtova dobijenih serijalizacijom sa unapred zadanim stepenom kompresije
- Merenje vremena potrebno za proces kompresije
- Merenje veličine podatka dobijenih procesom kompresije

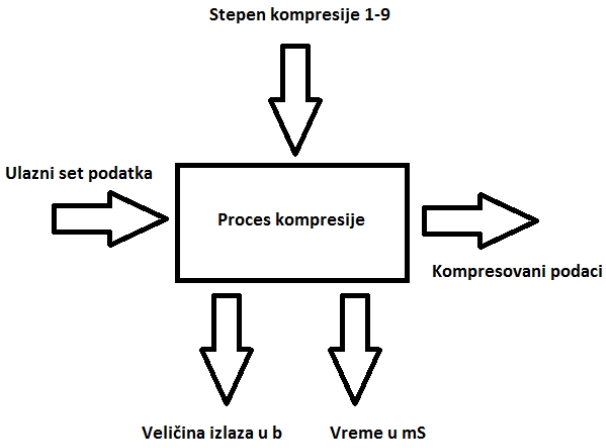
Pojedinačni test merenja performansi serijalizacije se može predstaviti blok dijagramom na sedećoj slici



Slika 29 Model programa za testiranje serijalizacije na serveru

Tipološki gledano, i u procesu serijalizacije i kompresije ulazni podaci su različiti. U jednom slučaju su objekti a u drugom su stringovi. Algoritamski gledano sa stanovišta testa u oba slučaja ulazni parametri za test su veličina podataka a izlazni veličina podataka i vreme.

Pojedinačni test merenja performansi kompresije se može prikazati sledećim dijagramom



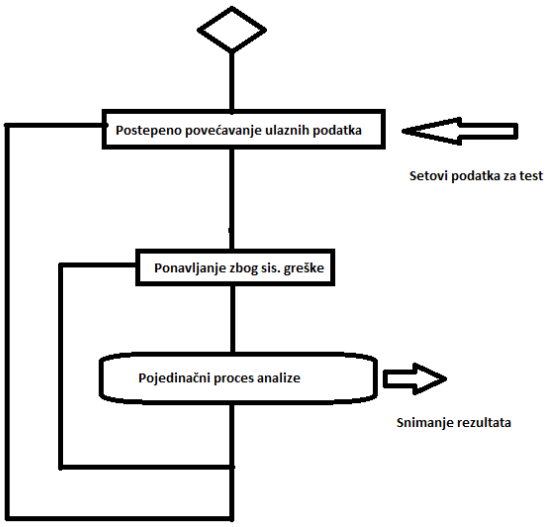
Slika 30 Model programa za testiranje kompresije na serveru

Iz ovakve perspektive programska struktura eksperimenta je ista za oba procesa. Tokom eksperimenta potrebno je uraditi testove sa različitim veličinama ulaznih podatka kako bi se dobila

valjana kriva zavisnosti performansi i veličine ulaznih podatka. Pored toga, treba obezbediti verodostojnost mernih podatka i otkloniti sistemske i stohastičke greške nastale tokom analize.

Proces merenja je isključivo softverske prirode i na njega mogu uticati samo stanja komponenti servera. Treba imati ovo u vidu pošto server koji se koristi u testu opslužuje i druge sisteme tako da u nekom od testova sam server može biti zauzet od strane nekog neželjenog procesa. Ovo se može odraziti na merenja tako da za neke od testova dobijemo neočekivano lošije performanse. Kako bi se ovo suzbilo svaki identični test treba ponoviti veliki broj puta. Pored toga za vreme trajanja testova treba nadgledati zauzetost procesora i memorije i odbaciti testove nastale u neidealnim uslovima.

S obzirom da je potrebno uraditi veoma veliki broj ponavljanja pojedinačnih testova, kako sa istim setovima podatka ali i sa postepenim povećavanjem obima ulaznih podataka, jasno je da se pojedinačni testovi moraju automatizovati. Automatizacija se ogleda u ponavljanju, automatskom pozivanju pojedinačnih testova više puta sa postepenim povećavanjem obima ulaznih podatka do definisanog opsega. Svaki od pojedinačnih testova se mora ponoviti više puta sa istim setom kako bi se suzbile greške.

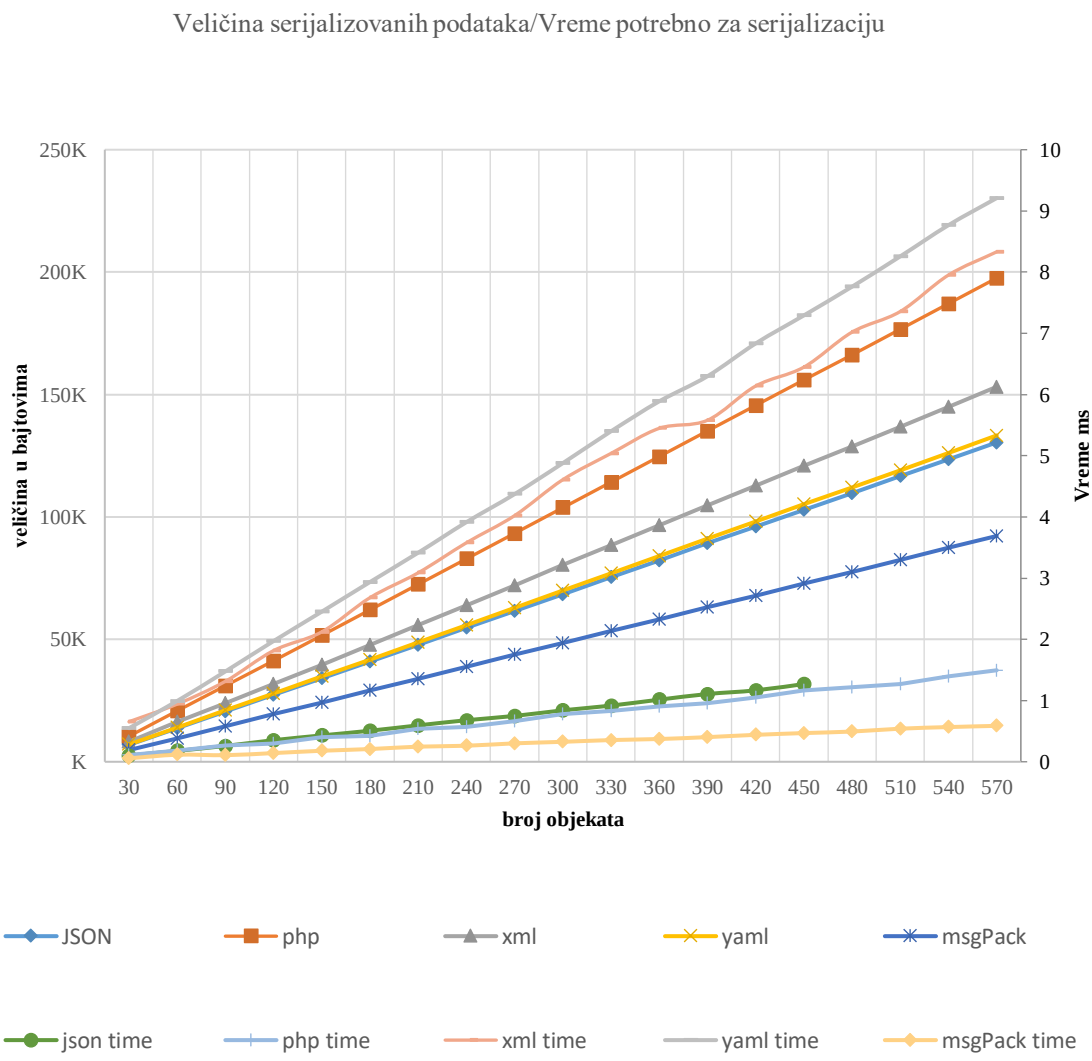


Slika 31 Model programa za testiranje

5.5.1.1.1 Rezultati eksperimenta

5.5.2 Serijalizacija

U gore opisanim testovima testirani su sledeći sistemi za serijalizaciju XML, JSON, PHP, YAML i msgPack. Veličina ulaznih podatka se kretala od 10 kb pa se postepeno povećavala do 2Mb. Svaki od testova je uzastopno ponovljen 100 puta u razmaku do 5s. Kao rezultat testova u okviru csv fajla se dobio set vrednosti o vremenu trajanja i veličini ulaza i izlaza. Analizom prikupljenih podatka nacrtani su grafici i izvedene matematički modeli za svaki od navedenih sistema serijalizacije. Tokom analize podatka odbačena su merenja koja su imala ekstremne vrednosti, koja su nastala usled preopterećenosti veb servera na kome su izvršavana merenja i testiranja. A od svih istorodnih setova merenja uzeta je srednja vrednost. Na sledećem grafiku mogu se videti odnosi veličina ulaznih sirovih objekta i veličine izlaznog serijalizovanog zapisa.



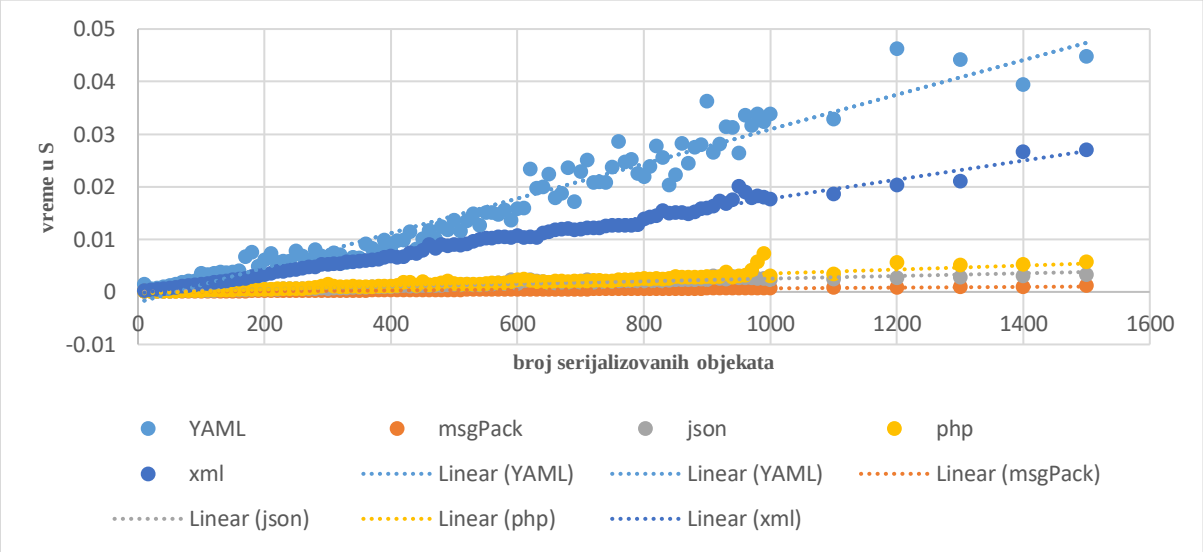
Slika 32 Odnos veličina serijalizovanog zapisa

Kao što se moglo i očekivati iz navedenog teorijskog razmatranja o pomenutim sistemima za serijalizaciju, u odnosu između veličina ulaza i izlaza se ispoljava nedvosmislena linearnost. Primećuje se da Php serijalizacija ima najnepovoljniji odnos, dok se msgPack nameće kao najbolje rešenje. MsgPack je hibridni sistem koji u sebi ima jedinstveni, i za razliku od čisto tekstualnih sistema kao što su PHP, XML i JSON, određeni stepen kompresije brojevnih vrednosti. Ovo merenje je u tom smislu potvrdilo teorijska razmatranja da su binarni sistemi za serijalizaciju superiorniji u pogledu odnosa veličina ulaza i izlaza. Bitno je naglasiti na ovim mestu da su za test korišćeni kombinovani setovi podatka koji u sebi imaju kako tekstualne tako i brojevne vrednosti.

Treba obratiti pažnju i na odnos performansi najčešće korišćenog JSON sistema i msgPacka sistema. Ovo dovodi u pitanje energetsku efikasnost veličine mobilnih aplikacija. O ovome će biti više reči u nastavku nakon analize svih faktora i formiranja matematičkog modela simulatora.

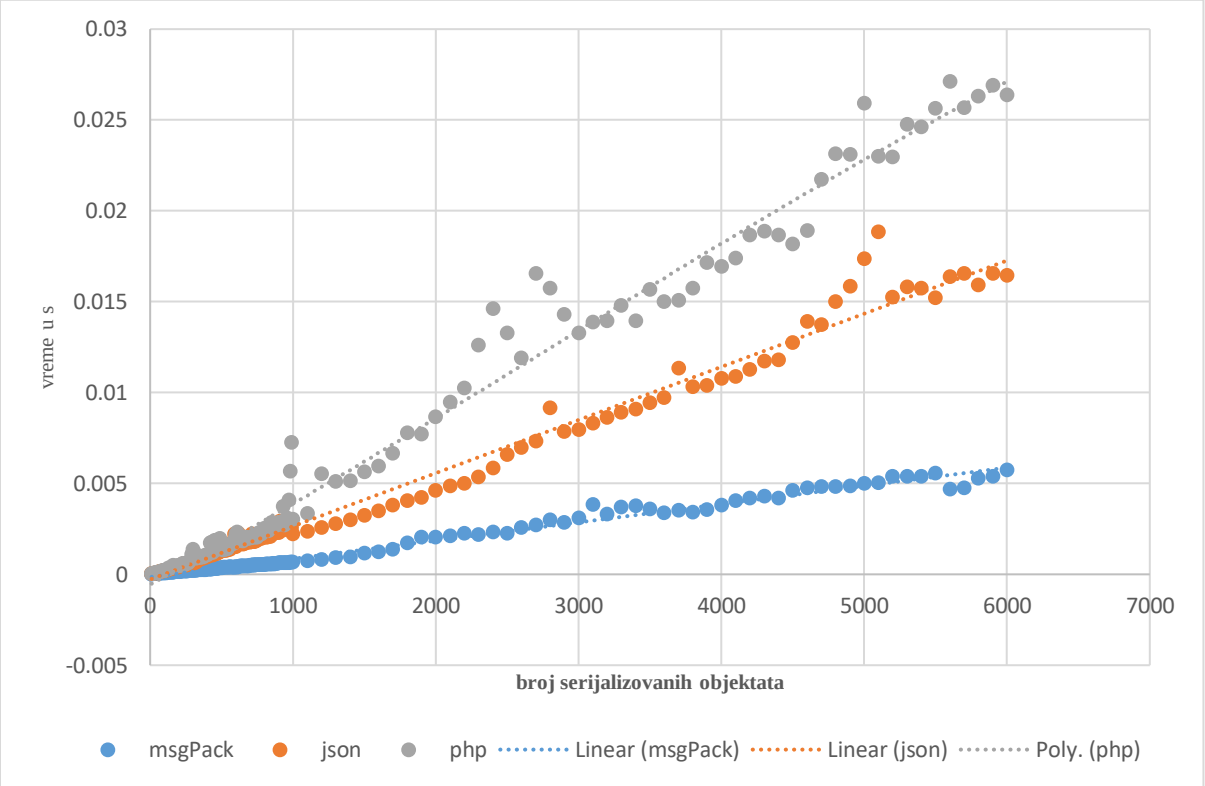
Drugi segment, ovog uporednog testa merenja performansi sistema, je vreme koje je potrebno veb serveru da izvrši proces serijalizacije. U prethodnom poglavlju smo videli da vreme koje se provede od trenutka kada server krene u obradu primljenog zahteva pa do početka slanja podatka klijentu (mobilni uređaj) ima značajan uticaj na utrošak energije klijenta.

Ovaj segment obrade rezultata merenja imao je za cilj da pokaže utrošeno vreme u milisekundima za svaki set ulaznih podatka. Na sledećem grafiku Slika 33 se vidi vreme ali i odnos brzina rada sistema za serijalizaciju u zavisnosti od veličine ulaznih podataka.



Slika 33 Odnos brzina sistema za serijalizaciju

Na ovim grafiku su prikazani svi testirani sistemi za serijalizaciju, odmah se zaključuje je da YAML, koji po pitanju stepena odnosa veličine ulaza i izlaza ima veoma slične performanse kao JSON, u pogledu vremena potrebnog za izvršavanja serijalizacije (brzine) ima jako loše performanse. Dok sa druge strane PHP, JSON i msgPack u odnosu na YAML i XML prednjače u brzini.

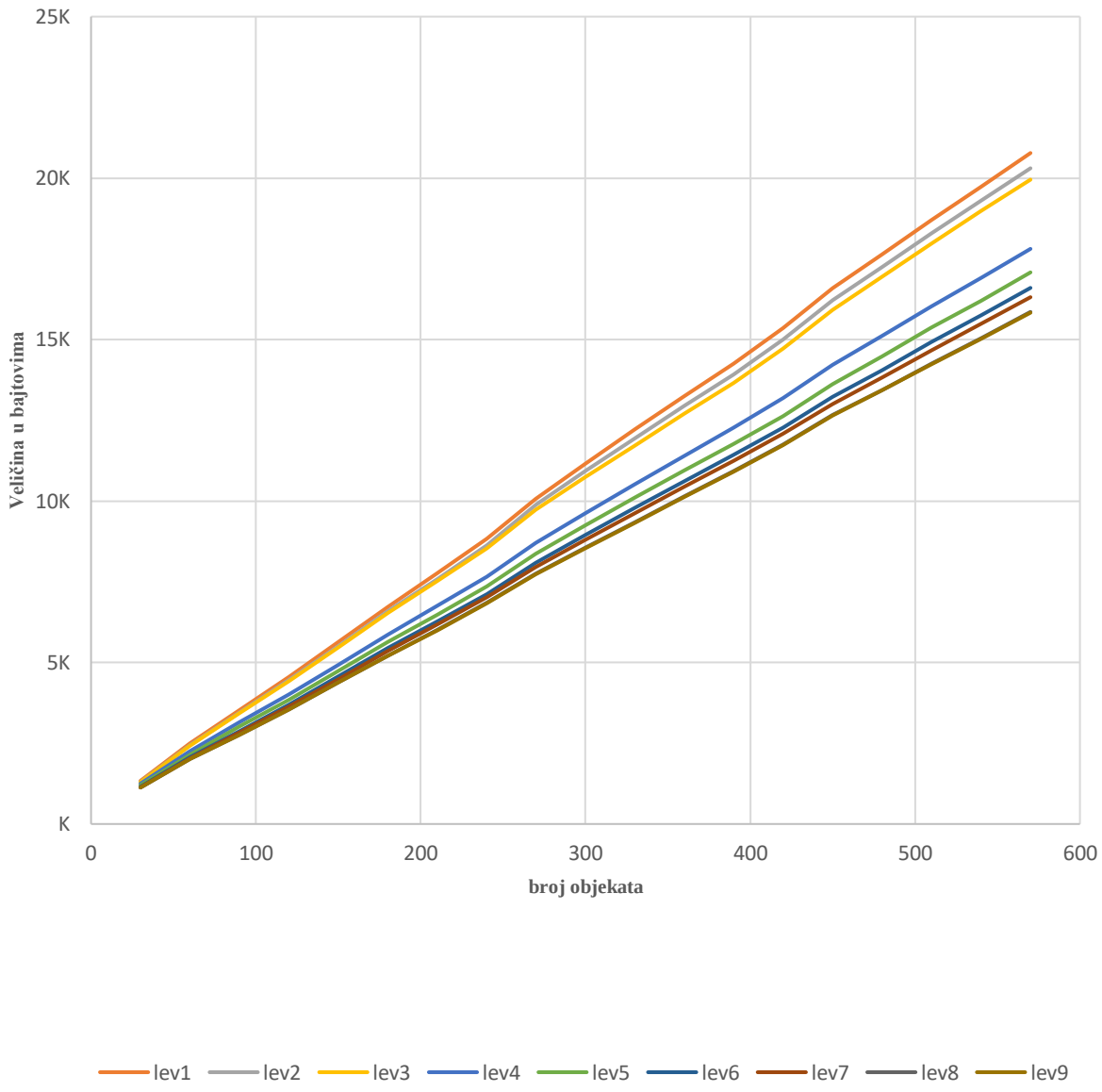


Slika 34 Odnos brzina tri najbrža sistema

Dubljom analizom se primećuje da msgPack ima najbolje performanse. Kao i u analizi performansi veličina podataka pokazalo se da je binarni oblik serijalizacije superiorniji u odnosu na tekstualne. Kod performanse brzine pokazatelja najveću ulogu ima implementacija algoritma sistema. Konkretno kod JSON-a i YAML-a struktura pakovanja podatka je jako slična, što se vizuelno vidi pregledom struktura ali i u analizi veličina podatka. Međutim, vreme koje je potrebno bibliotekama YAML-a da serijalizuju podatke je drastično veće nego kod JSON biblioteka. Ovo ukazuje na lošije implementirani algoritam kod YAML biblioteka.

5.5.3 Kompresija podataka

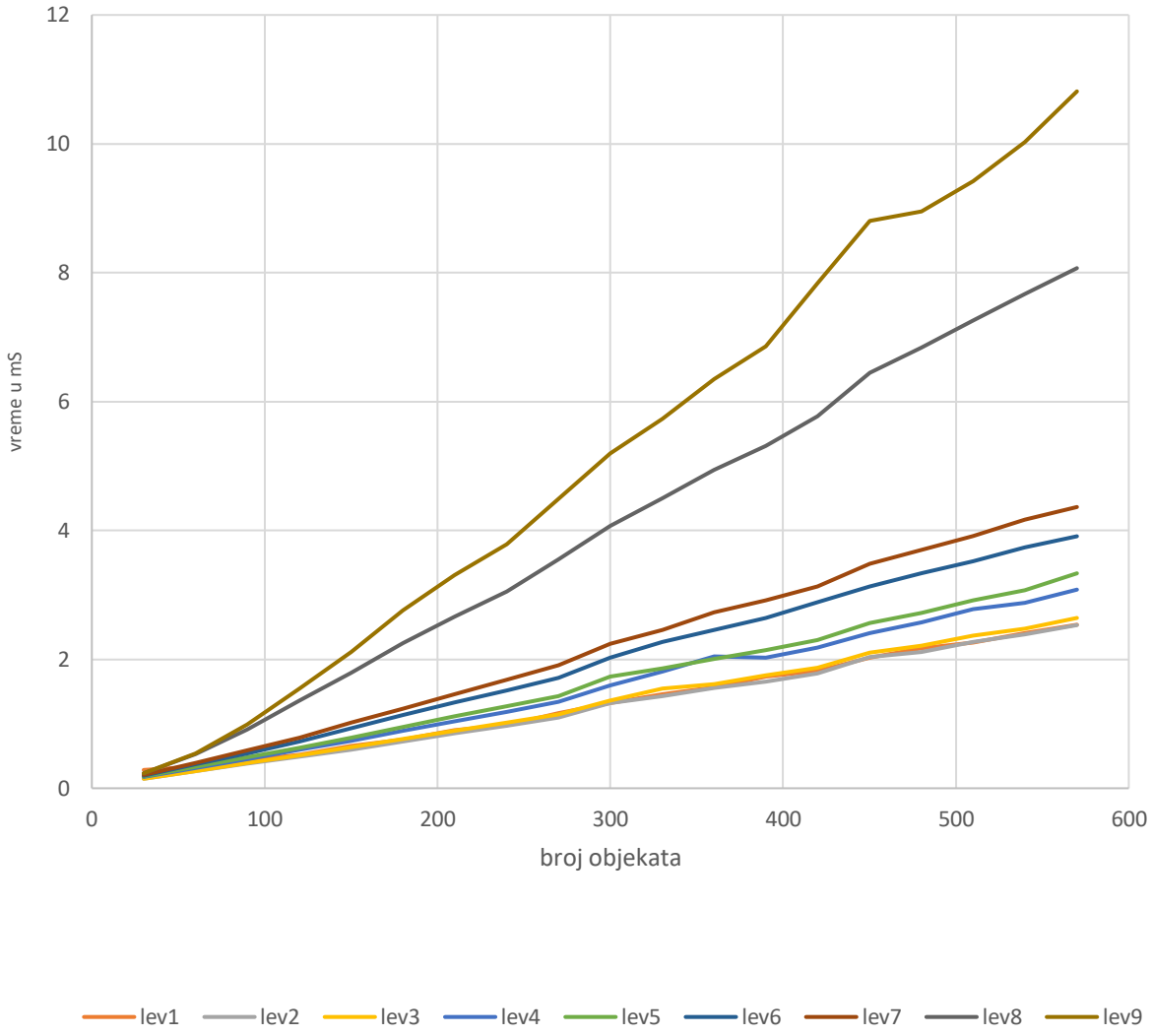
Nakon eksperimenta sa sistemima za serijalizaciju nastavljeni su eksperimenti sa sistemima za kompresiju. Kao i u prethodnom testu veličina ulaznih podataka je povećavana počev od 10 kB pa do 140kB. Za razliku od testa gde su menjani tipovi serijalizacije, ovde su menjani stepeni kompresije podataka od 1 od 9. Kao rezultati merenja dobijene su veličina kompresovanih podataka i vreme utrošeno na kompresiji. Na sledećim graficima su prikazani rezultati merenja.



Slika 35 Odnos nivoa kompresije

Sa grafika može se zaključiti da nivo kompresije 4 pravi jako veliku razliku u odnosu na nivoe 1,2,3 dok se kod viših nivoa kompresije 6, 7, 8, 9 veće razlike primećuju samo na većim setovima podatka.

Na sledećem grafiku Slika 36 prikazano je vreme potrebno za kompresuju podatak.



Slika 36 Odnos brzina nivoa kompresija u odnosu na veličinu ulaznih podataka

Posmatrajući prethodni grafik možemo zaključiti da vreme potrebno da se kompresija izvrši nelinearno raste sa povećavanjem stepena. Naizgled laičkim ocenjivanjem grafika nastalih

eksperimentom sistema za kompresiju podatka moglo bi se zaključiti da su srednji stepeni najoptimalniji. Međutim, da bi ovo naučno i egzaktno utvrdili moramo za svaku od kompresija kreirati matematičke modele. Kao i kod serijalizacije kreirani su modeli za veličinu podatka i vreme utrošeno na kompresiji.

5.6 Združeni matematički model

Analizom serijalizacije je dobijen matematički model koji je dalje ugrađen u prethodni model potrošnje, tako da, ukoliko u jednačini (2) iz prethodnog poglavlja prostom zamenom uvrstimo dobijene modele u uopštenom obliku, dobićemo sledeće:

$$E[Ws] = (0.123A * t_c + 0.163A * (t_p + F_d(D_s, V_s)) + 0.140A * F_l(D_s) * 3.7V + 2.875Ws \quad (3)$$

Gde je:

$F_d(D_s, V_s)$ Vreme potrebno da zahtevana količina podatka stigne do klijenta, $F_l(D_s)$ Vreme potrebno da sever pripremi podatke za slanje klijentu:

$$F_l(D_s) = F_T^{gzi}\left(G_z, F_S^{gzi}(G_z, F_S^{ser}(S_t, D_s)\right) + F_T^{ser}(S_t, D_s) \quad (4)$$

$$F_d(D_s, V_s) = \frac{F_S^{gzi}(G_z, F_S^{ser}(S_t, D_s)}{V_s} \quad (5)$$

Gde je:

F_T^{gzi} - Funkcija koja računa vreme potrebno da se kompresuju podaci u zavisnosti od tipa kompresije i veličine podataka

F_T^{ser} - Funkcija koja računa vreme potrebno da se serializuju podaci u zavisnosti od veličine podatka i tipa serijalizacije,

F_S^{gzi} - Funkcija koja računa veličinu podataka nakon kompresije u zavisnosti od tipa kompresije i veličine podataka

F_S^{ser} Funkcija koja računa veličinu podataka nakon serijalizacije u zavisnosti od veličine podataka i tipa serijalizacije,

G_z- nivo kopresije

S_t- tip serializacije

$$F_T^{gzi}(G_z, D_s) = \begin{cases} D_s * 8 * 10^{-9} - 10^{-6} & , G_z = 1 \\ Ds * 1 * 10^{-8} - 3 * 10^{-5} & , Gz = 1 \\ Ds * 1.02 * 10^{-8} - 2.7 * 10^{-5} & , Gz = 3 \\ Ds * 1.08 * 10^{-8} - 2.3 * 10^{-5} & , Gz = 4 \\ Ds * 1.12 * 10^{-8} - 2.9 * 10^{-5} & , Gz = 5 \\ Ds * 2.1 * 10^{-8} - 2.3 * 10^{-5} & , Gz = 6 \\ Ds * 3.1 * 10^{-8} - 5.3 * 10^{-3} & , Gz = 7 \\ Ds * 5.7 * 10^{-8} - 4.3 * 10^{-3} & , Gz = 8 \\ Ds * 6.98 * 10^{-8} - 5.1 * 10^{-3} & , Gz = 9 \end{cases} \quad (6)$$

$$F_S^{gzi}(Gz, Ds) = \begin{cases} Ds * 0.1564 + 374.7 & , Gz = 1 \\ Ds * 0.1524 + 349.2 & , Gz = 1 \\ Ds * 0.1499 + 380.3 & , Gz = 3 \\ Ds * 0.1332 + 422.2 & , Gz = 4 \\ Ds * 0.1276 + 409.7 & , Gz = 5 \\ Ds * 0.1244 + 353.2 & , Gz = 6 \\ Ds * 0.1221 + 359.4 & , Gz = 7 \\ Ds * 0.1184 + 369.3 & , Gz = 8 \\ Ds * 0.1183 + 349.2 & , Gz = 9 \end{cases} \quad (7)$$

$$F_S^{ser}(St, Ds) = \begin{cases} Ds * 229 - 179 & , St = 1 \text{ json} \\ Ds * 234.11 - 183.2 & , St = 2 \text{ php} \\ Ds * 234 + 179.2 & , St = 3 \text{ xml} \\ Ds * 0.1499 + 349.2 & , St = 4 \text{ yaml} \\ Ds * 176.1 + 184.2 & , St = 5 \text{ msgPack} \end{cases} \quad (8)$$

$$F_T^{ser}(St, Ds) = \begin{cases} Ds * 3 * 10^{-6} + 4 * 10^{-6} & , St = 1 \text{ json} \\ Ds * 4 * 10^{-6} + 3 * 10^{-3} & , St = 2 \text{ php} \\ Ds * 2 * 10^{-5} + 4 * 10^{-6} & , St = 3 \text{ xml} \\ Ds * 3 * 10^{-5} + 2 * 10^{-2} & , St = 4 \text{ yaml} \\ Ds * 7 * 10^{-7} + 4 * 10^{-6} & , St = 5 \text{ msgPack} \end{cases} \quad (9)$$

Ovako dobijenim matematičkim modelima je moguće izvršiti simulaciju i komparaciju energetske efikasnosti analiziranih sistema za serijalizaciju i kompresiju.

Na sledećem primeru nalazi se konkretan primer matematičke funkcije u slučaju JSON serijalizacije 10 objekta iz testne baze bez kompresije .

$t_c = 0.1s$

$t_p = 0.3s$

$Vd = 0{,}2\ Mb/s$

$$E_{(W \cdot s)} = (123 * 0.1_s + 163 * (0.3_s + \frac{229 * 2300_B - 197_B}{25 * 10^3{}_{B/s}}) + 140 * (3 * 10^{-6}{}_{s/B} * 2300_B)) * 0.005_W + 2{,}875_{W \cdot s}$$

$$E_{(W \cdot s)} = (17{,}47 + 2{,}875)_{W \cdot s}$$

Dok sa istim parametrima ali ako se koristi xml dobijamo sledeće rezultate

$$E_{(W \cdot s)} = (123 * 0.1_s + 163 * (0.3_s + \frac{234 * 2300_B - 197_B}{25 * 10^3{}_{B/s}}) + 140 * (2 * 10^{-5}{}_{s/B} * 2300_B)) * 0.005_W + 2{,}875_{W \cdot s}$$

$$E_{(W \cdot s)} = (30{,}11 + 2{,}875)_{W \cdot s}$$

Na ovom primeru se vidi da na osnovu proračuna razlika u samo jednom zahtevu iznosi $12{,}6_{W \cdot s}$. Na ako bi se ovi zahtevi tokom rada aplikacije slali na savkih 30 sekundi u toku jednog sata rada aplikacije bi razlika iznosila $1512_{W \cdot s}$ a kada se watt sekunda konvertuje u mAh pod pretpostavkom da je baterija 5v dobija se 84 mAh.

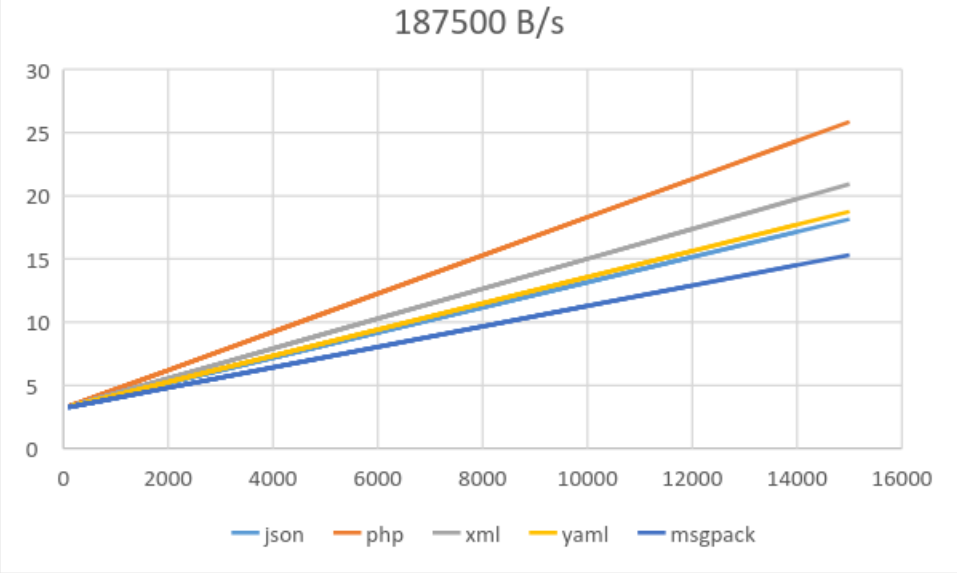
5.7 Upotreba matematičkog modela

5.7.1 Komparacija tipova serijalizacije na utrošku energije

U prethodnom poglavlju je razrađen matematički model potrošnje energije tokom REST komunikacije kombinovanjem sa različitim tipovima serijalizacije. Na kraju ovog poglavlja dat je primer odnosa potrošnje JSONA i XML-a tokom prenosa 10 objekata iz baze do klijenta i matematičkim putem prikazana razlika. Na bazi matematičkih modela napravljen je program koji analizira i računa potrošnju modema. Program jednostavno zahteva od korisnika da unese tip serijalizacije, veličinu zahtevanih podatka i brzinu transfera podatka. Korišćenjem ovog simulatora dobijeni su sledeći rezultati.

Slika 34 i 35 predstavlja grafik na kome se vide krive potrošnje električne energije u zavisnosti od tipa serijalizacije i veličine podatka kada je brzina interneta konstantna iznosi 25kB/s.

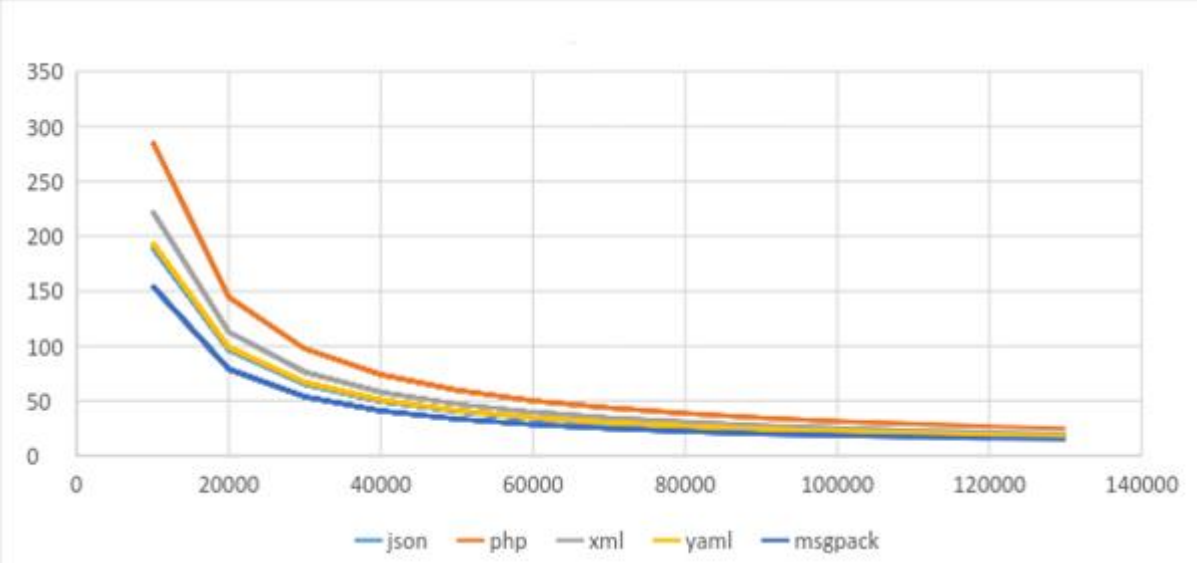
Slika 37 Odnos potrošnje energije u zavisnosti od odabira tipa serijalizacije i obima podataka na osnovu matematičkih modela



Slika 38 Odnos potrošnje energije u zavisnosti od odabira tipa serijalizacije i obima podataka na osnovu matematičkih modela

Na graficima (Slika 39 Slika 40) se vide krive potrošnje električne energije u zavisnosti od tipa serializacije i brzine interneta podatka za konstantnu količinu podatka .

Slika 39 Odnos potrošnje energije u zavisnosti od odabira tipa serijalizacije na osnovu matematičkih modela



Slika 40 Odnos potrošnje energije u zavisnosti od odabira tipa serijalizacije na osnovu matematičkih modela

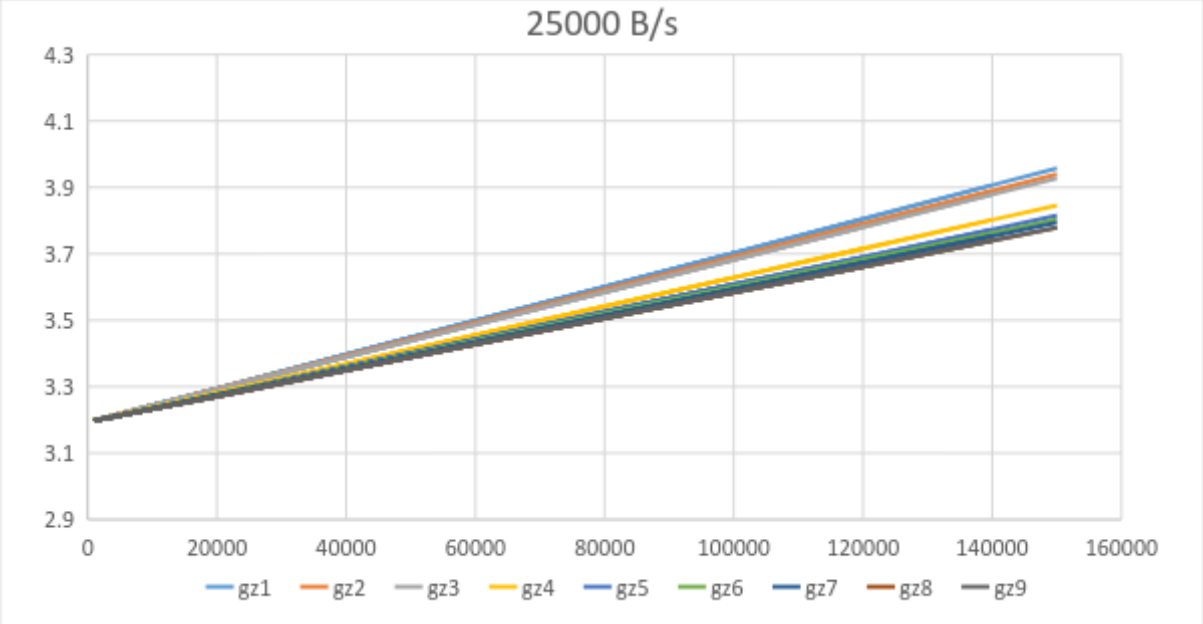
Sa grafika se može jednoznačno videti uticaj odabira sistema za serijalizaciju na utrošak energije kod klijentske strane. Zaključak koje se ovde može izvesti je što je veći obim podatka koji se transportuje to je i veća razlika između tipova, ali i da ta razlika opada sa povećanjem brzine transporta kao i potrošnja. Dakle, uticaj formata serijalizacije nije jednoznačan.

5.7.2 Upotreba kompresije i njen uticaj na energetsko stanje

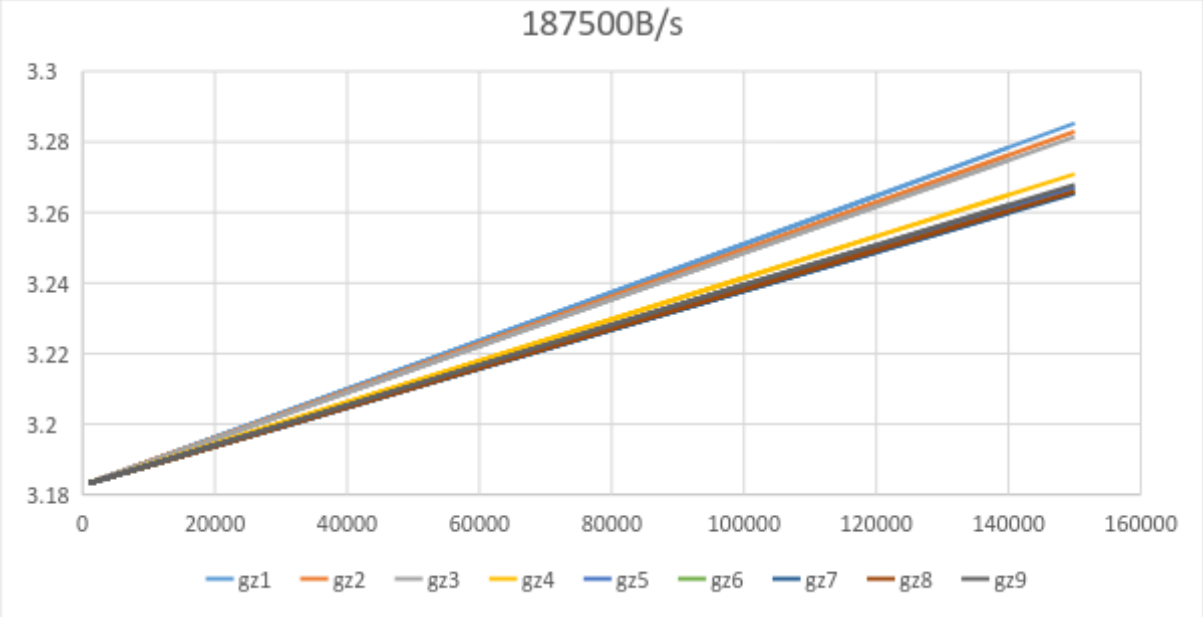
Veliki broj aplikacija kombinuje neki od vidova serijalizacije sa kompresijom serijalizovanog stringa u nadi da će povećati efikasnosti. Posmatrajući rezultate istraživanja i krive procesa serijalizacija i kompresije u sprezi sa procesom REST komunikacije, može se doći od zaključka da energetska efikasnost nije tako linearna. Drugačije rečeno: u zavisnosti od stanja mreže količine podatka i stepena kompresije koji koristimo energetski efekat može biti podjednako i pozitivan ili negativan.

Dakle matematički model (3) ima dve veličine na koje može uticati stepen kompresije a to su veličina podatka koju transportujemo od servera i vreme koje čekamo na odgovor. Zahvaljujući matematičkim modelima koje posedujemo moguće je simulirati razne slučajeve i kombinacije i analizirati energetsku efikasnost.

Na slikama Slika 41 Slika 42 je prikazana potrošnja za svaki od stepena kompresije kada je brzina fiksirana a veličina podatka se menja.

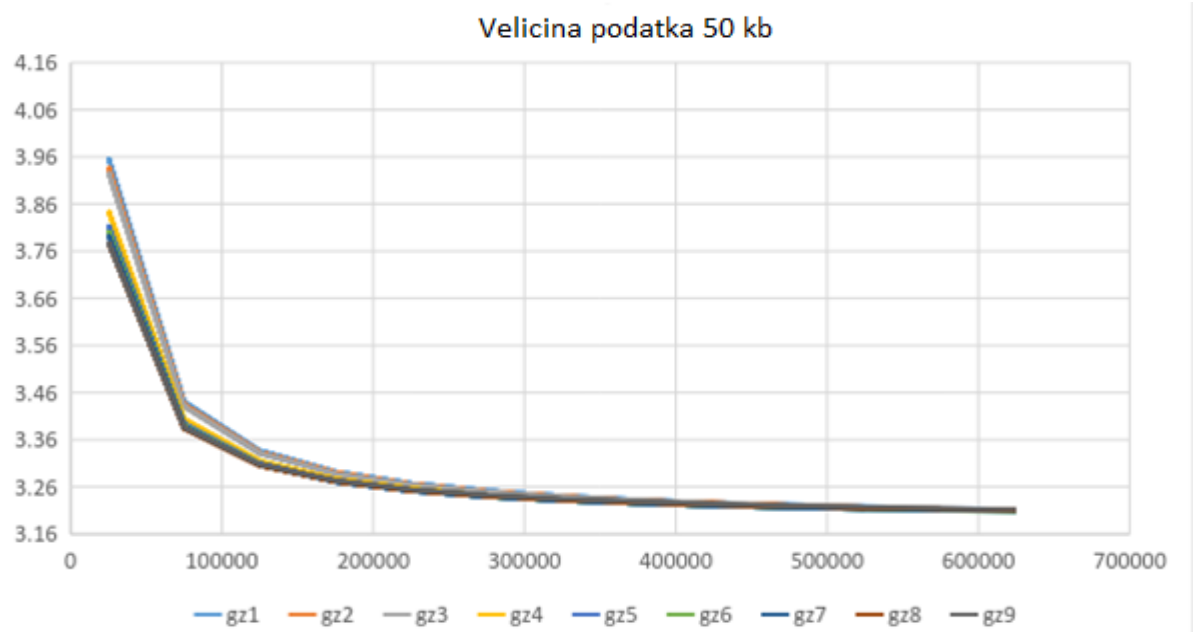


Slika 41



Slika 42

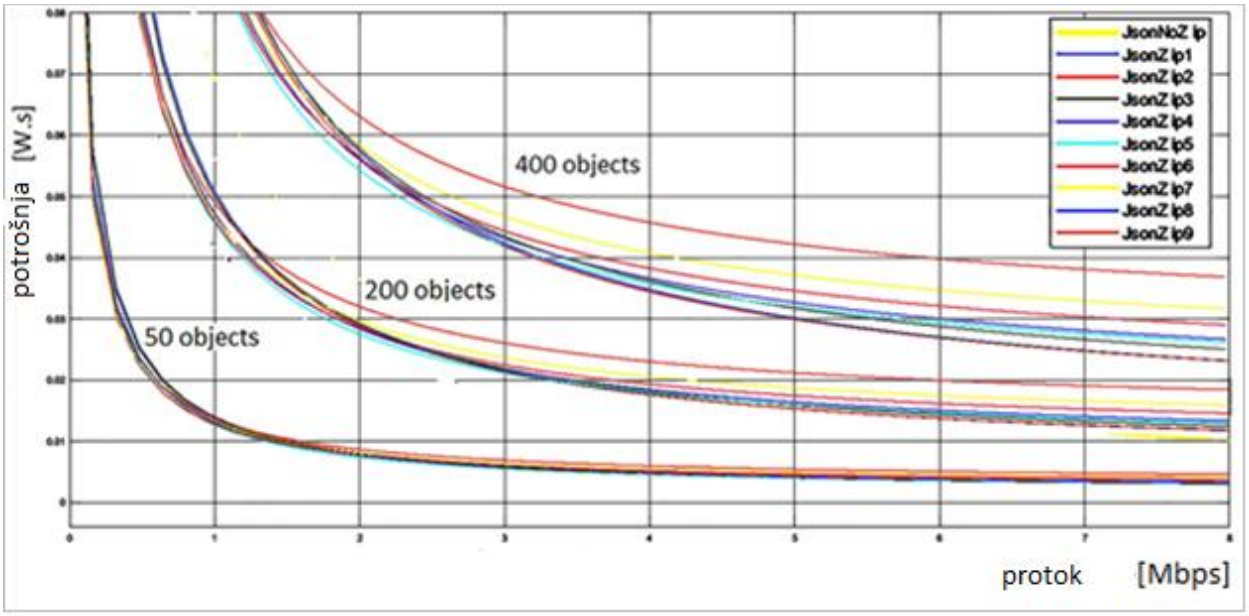
Sa ova dva grafika se vidi da veličina podataka koji se zahtevaju linerano kao i kod serijalizacije povećava potrošnju. Pored toga na manjim brzinama najveći stepen kompresije je energetski najoptimalniji. Sa drugog grafika gde je brzina veća primećuje se da je stepen kompresije 7 najoptimalniji. Zbog ovoga je urađeno i simulacija gde je brzina primenljiva a podaci fiksne veličine Slika 43



Slika 43 Simulacija uticaja promene brzine na potrošnju kod kompresije

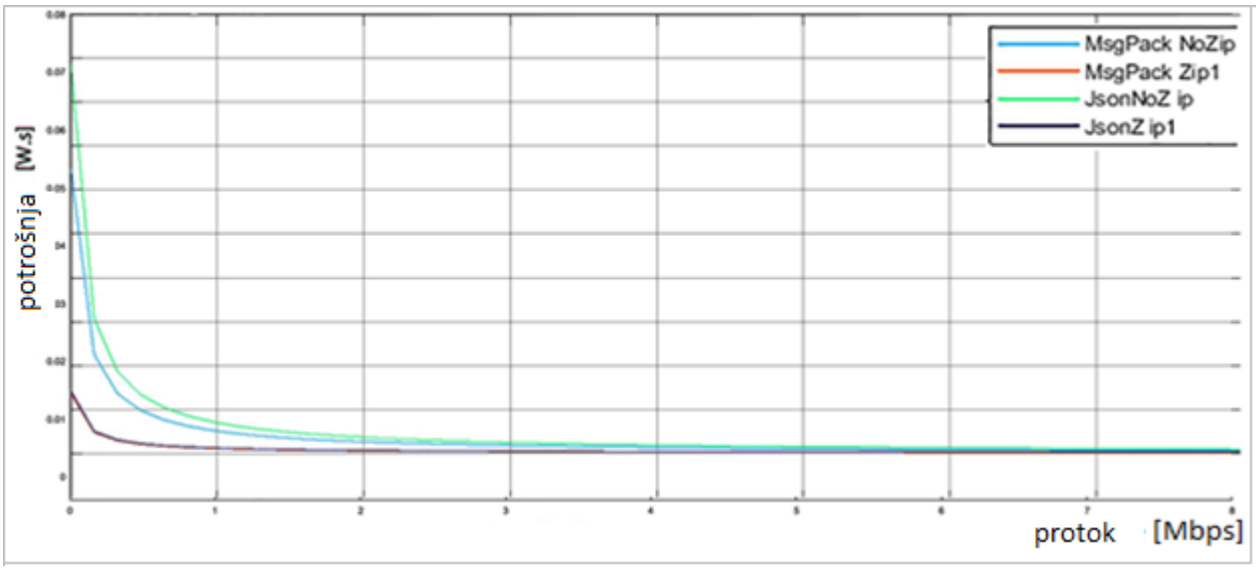
Sa povećanjem brzine transporta podataka vreme potrebno za skidanje podatka t_d postaje sve manje dominantni faktor u jednačini 1 pa onda na potrošnju sve više utice vreme koje se potroši tokom rada servera na pakovanju podatka t_l . Isto važi i za prethodnu analizu serijalizacije.

Sumiranjem ovih rezultata dolazi se na ideju da se kombinuju kompresija i serijalizacija i da se za zadane parametre dođe od trenutno energetski najoptimalnijeg rešenja Zahvaljujući dobijenim modelima moguća je analiza uticaja šeme serijalizacije i stepena kompresije na energetsku efikasnost klijentskog uređaja u vrlo ranim fazama projektovanja aplikacije. Pomoću *Matlab simulnik* programskog paketa simulirani su uporedni scenariji za sve pomenute slučajeve serijalizacije i *gzcompress* kompresije. Shodno tome na slici Slika 44 prikazani su uporedni dijagrami potrošnje u zavisnosti od brzine protoka podataka i količine objekata koji se zahtevaju od servera. Korišćeni obim podatka tokom simulacije varira od 10 do 400 objekata po zahtevu, dok je za brzinu simuliran opseg od 0.1 od 8 Mbps.



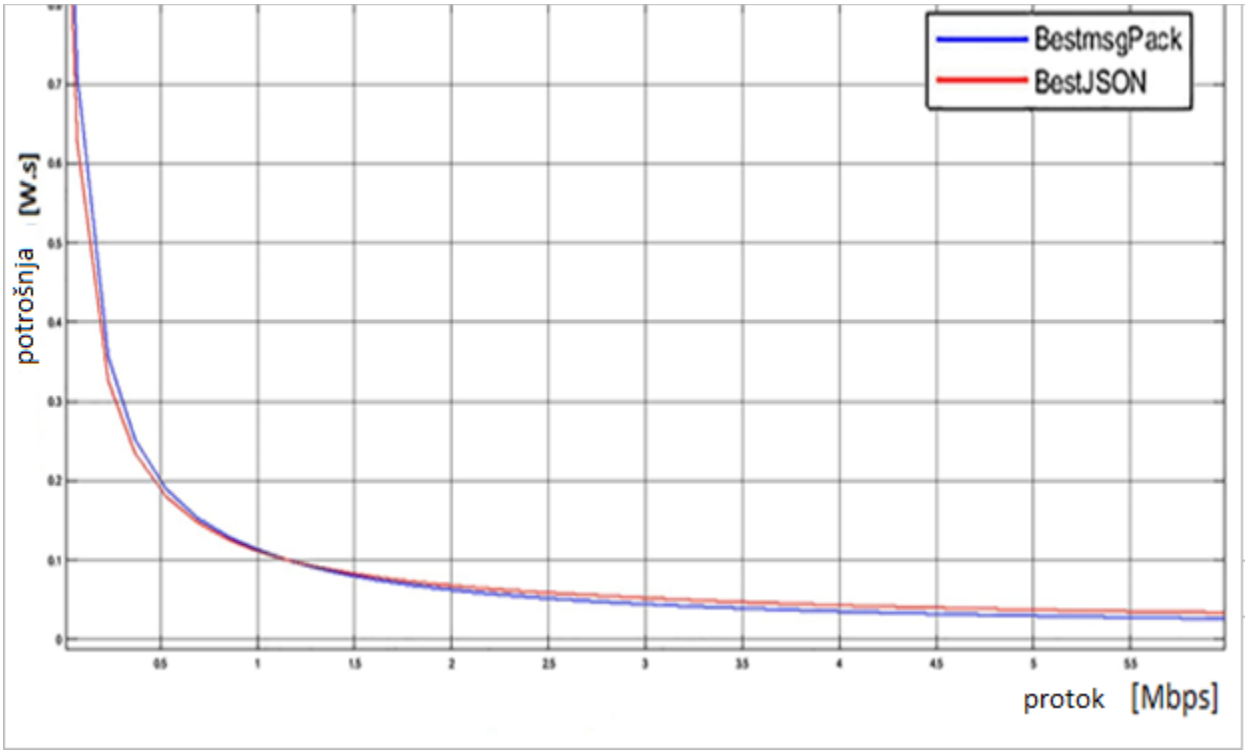
Slika 44 Združeni prikaz odnosa potrošnje u zavisnosti od obima podataka brzine i stepena kompresije kod JSON formata

Iz ovih simulacija nameće se zaključak da su pojedine kombinacije serijalizacije i *Gzip* kompresije efikasnije u pojedinim zonama od drugih. Zone su definisane brzinom i brojem objekata koji se zahtevaju. Na slici Slika 45 prikazan je odnos JSON serijalizacije i msgPack sa svim gzcompress nivoima kompresije ali bez kompresije.



Slika 45 Odnos potrošnje JSON i msgPack šema sa različitim nivoima kompresije

Na osnovu zapažanja da ne postoji literarnost i optimalni sistem postavlja se sledeće pitanje: „Može li se za trenutne uslove pronaći optimalno energetski efikasno rešenje?“ Odgovor na ovo pitanje daje nova simulacija koje ima za cilj da prikaže trenutno najbolje rešenje koje ima JSON i msgPack serijalizacija. Rezultat je prikazan na slici Slika 46, gde se vidi da optimalna rešenja JSON-a (sa kompresijom) i msgPack (sa kompresijom) nisu energetski optimalna. Konkretno optimalno rešenje JSON-a je optimalno u zonama niže brzine dok msgPack u zonama veće brzine. Ovome treba još i dodati da povećanje broja objekata koji se zahtevaju ovu tačku preseka pomeraju u levo. Dakle ovo otkriće je i prelomna tačka ovog rada. Ovo saznanje je otvorilo vrata za novi napredni sistem za energetski optimalni prenos podataka.



Slika 46 Prikaz potrošnje energija kod najoptimalnijih scenarija JSON kompresovanog i msgPack kompresovanog odgovora u zavisnosti od brzine protoka

5.8 Završne napomene poglavlja

Matematički model u ovom poglavlju je u sprezi sa programom za računanje potrošnje poslužio da na lakši način (u odnosu na eksperimentalno merenje na uređaju) uvidimo uticaje serijalizacije i kompresije direktno na potrošnju električne energije na klijentskom uređaju. Pored toga

omogućio je pregled uticaja svake od komponenti na potrošnju i analizu u cilju optimizacije. Jedan od ciljeva ovog rada je istraživanje mogućnosti da programer na veoma visokim aplikativnom nivou tokom projektovanja programa pravilnim odabirom komponenti (programskih instrukcija) može povećati energetsku efikasnost sistema. Setimo se, postupak koji je prethodio dobijanju matematičkog modela, započeo je merenjem i analizom ponašanja modema tokom REST komunikacije. Pomoću projektovane opreme i specijalizovanog programskog test okruženja uspešno je izmeren profil REST komunikacije na finom aplikativnom nivou. Drugim rečima, snimljene su električne karakteristike za svaku od programskih instrukcija tokom životnog ciklusa jedne REST komunikacije. Zahvaljujući tim podacima formiran je energetski profil REST komunikacije. Matematičkom analizom dobijen je matematički model. Nakon toga, na serveru su izvršena dalja merenja u cilju dobijanja formule koje bi se zamenile u jednačini (1) uslovljene procesima serijalizacije i kompresije. Za svaki od tipova serijalizacije je formiran po par funkcija koje definišu vreme potrebno da se serijalizacija obavi i količina sirovih podatka koja se tom prilikom dobije u zavisnosti od količine podatka koja ulazi zahtev. Na isti način su formirani i parovi kod procesa kompresije za svaki od nivoa kompresije ponaosob. Tako dobijene formule su ugrađene u matematički model (2). Dalji rad biće posvećen spajanju dobijenih znanja iz ovog ali i prethodnih poglavlja u eksperimentalni sistem i eksperimentalnu aplikaciju koja ima za cilj da ostvari znatnu uštedu u električnoj energiji u odnosu na klasične metode koje se koriste.

6 VI DEO

6.1 Sistem za optimizaciju i analizu energetske efikasnosti baziran na matematičkom modelu *CBSaver*

Nakon sveobuhvatne analize mehanizama prenosa objektno orijentisanih podatka između servera i mobilnih aplikacija usledila je ideja o autonomnom omotaču prenosa podataka koji bi kod mobilnih uređaja smanjio potrošnju električne energije. Prethodno pomenutom analizom CBSaver (*Client Battery Saver*) je dokazano da se optimizovanim softverskim metodama u pojedinim uslovima eksploatacije može postići nezanemarljiva ušteda električne energije mobilnih uređaja. Na osnovu laboratorijskih merenja i analiza prvenstveno REST komunikacije zatim serijalizacije podatka i kompresije podatka, javila se ideja o sistemu koji na osnovu grupe parametara samostalno odlučuje o načinu *pakovanja podataka* koji je u datom trenutku energetski najoptimalniji. Dakle, ovakav koncept ne bi predstavljao tehnološki novi vid komunikacije već bi koristio postojeće sisteme i oko njih gradio okruženje koje samostalno odlučuje o izboru načina komunikacije.

Na osnovu pomenutih sprovedenih istraživanja u ovom radu došlo se do ideje o energetski efikasnom sistemu prenosa objektno orijentisanih podataka između servera i mobilnog uređaja, baziranom na matematičkim modelima dobijenim u prethodnom istraživanju. Sistem za rad tokom komunikacije koristi trenutne parametre i na osnovu njih bira način koji je energetski najefikasniji. U prethodnim istraživanjima matematički modeli koje smo dobili zavise od platforme na kojoj je izvršeno istraživanje i od tipa podataka koji se transportuju. Prema tome, ako želimo da sistem bude precizan pre implementacije celog sistema potrebno je uraditi korekciju univerzalnog matematičkog modela. Inicijalizacijom sistema se formira korigovani matematički model. Ovaj matematički model bi predstavljao prvi sloj sistema za energetsku efikasnost prenosa podataka

Drugi sloj sistema za energetsku efikasnost predstavlja sloj koji radi u realnom vremenu. On na osnovu podatka koji dobija od klijenta o trenutnoj brzini protoka podatka i parametra o veličini podatka koje treba transportovati određuje načine serijalizacije i stepen kompresije. Zaključno s

tim, ovaj sloj se sastoji od klijentske i serverske strane. Uloga klijentske strane je da pored slanja zahteva za prijem podataka pošalje i informacije serveru o brzini protoka informacija mobilnog uređaja.

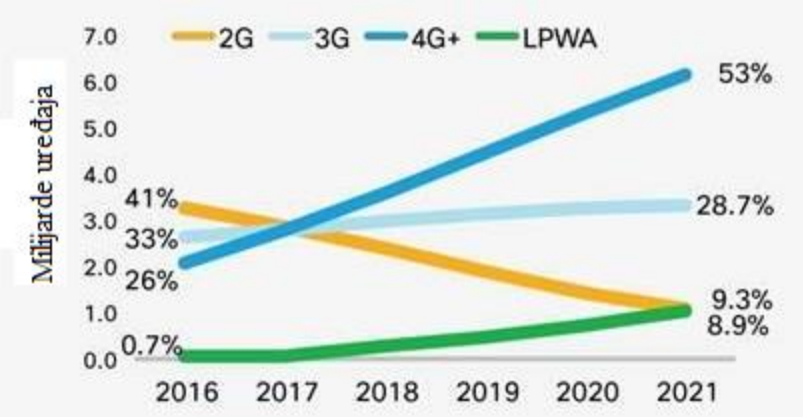
6.1.1 Opis problema na realnoj platformi

Zamislimo da je pred nama postavljen zadatak da se postavi energetski najefikasniji sistem prenosa objektno orijentisanih podataka za određenu mobilnu aplikaciju. Kao primer uzećemo zahtev za razvojem aplikacije koja na osnovu geografske lokacije prikuplja podatke o objektima od interesa *POI* iz okoline mobilnog uređaja koji se dalje prikazuju na mapi. Učestalost zahteva kojima mobilna aplikacija dovlači podatke sa centralizovanog servera uslovljena je promenom položaja samog uređaja i maksimalnim vremenskim intervalom ukoliko ne dođe do promene položaja. Aplikacije ovog tipa najčešće rade ovakvim metodom.

Bilo da se radi o aplikaciji koja prikazuje prepreke na putu, parking mesta, informacije o gužvi u saobraćaju, lokaciju drugih korisnika ili atrakcije u okolini; gotovo da je kod svih mehanizam i tehnologija svedena na veoma sličan princip rada. Skoro svi postojeći sistemi koji se nalaze na tržištu koriste REST komunikaciju sa nekim od vidova serijalizacije, najčešće JSON. Kod nekih proizvođača se nakon serijalizacije podatka dobijen string kompresuje i šalje do klijenta. Na ovakvim platformama se podaci koji se zahtevaju od servera smatraju objektno orijentisanim. Konkretno, ako se radi o najbližem parkingu koji se treba prikazati korisniku, od servera se zahteva set podatka vezan sa svaki objekat u blizini. Svaki od objekata poseduje svoju geografsku lokaciju izraženu kroz geografsku širinu i dužinu, naziv, adresu i ostale informacije od interesa.

Za ove sisteme je karakteristično da se koriste u pokretu. To znači da se obim podatka koje uređaj zahteva od servera menja i nije konstantan. Na primer u centru nekog većeg grada u blizini se nalazi do 10 puta više objekata od interesa nego u nekom manjem gradu ili ruralnom delu. Takođe, internet konekcija i brzina interneta može da varira u zavisnosti od lokacije, broja korisnika na baznoj stanici i vremenskih uslova. Ovi promenljivi uslovi su veoma ključni za optimalnu energetsku efikasnost uređaja. Treba uzeti u obzir da među korisnicima aplikacije (ako se radi o aplikaciji za široke narodne mase) imamo određeni broj korisnika koji imaju uređaje koji mogu

raditi na velikim brzinama (3G 4G) interneta ali i one koji imaju čak i do dve generacije starije uređaje. Na osnovu predviđanja kompanije CISCO [37]



*Slika 47 Predikcija broja generacija mobinih uređaja **preuzeto od** [37]*

možemo videti udeo svake od tehnologija mobilnih uređaja u upotrebi kod korisnika. Trenutno gledano kod korisnika su podjednako zastupljene sve tri tehnologije. To znači da na istoj geografskoj lokaciji, istoj baznoj stanici možemo imati, procentualno gledano, isti broj korisnika koji poseduju 2G, 3G i 4G uređaje. Ovo implicira da čak i u istim uslovima njihove aplikacije ne mogu da budu energetski podjednako efikasne.

Sva ova prethodna razmatranja i opservacije u cilju kreiranja eksperimentalne energetski efikasne aplikacije nas dovode do sledećih karakteristika .

- Promenljivi obim podatka u zavisnosti od geografske lokacije koje server prosleđuje klijentu
- Promenljiva brzina prenosa podataka u zavisnosti od geografske lokacije između servera i mobilnog uređaja
- Nejednaka maksimalna brzina prenosa podatka u zavisnosti od tipa klijentskog uređaja

Ako se osvrnemo na prethodna istraživanja iz poglavlja o energetskoj efikasnosti i pogledamo matematičke modele i testove koji su obavljeni u V poglavlju, jasno se vidi da bez obzira kako podesili stepen kompresije, odabrali tip serijalizacije i način slanja, ne možemo formirati sistem za prenos podatka koji u datom trenutku garantovano energetski najoptimalnije radi Slika 46. Kada

se uzme u obzir i raznorodnost uređaja jedino rešenje je napraviti hibridni sistem koji, na osnovu obima podatka koji se šalju korisniku i brzine protoka podatka, bira određeni režim slanja podatka. Drugim rečima: ako posedujemo matematičke modele servera, obim podatka koji se u datom trenutku šalje ali i brzinu kojom klijent komunicira sa serverom možemo odrediti energetski najoptimalniji režim slanja podatka. I tako za svaki od zahteva. Ova pretpostavka je bazirana na simulacionom modelu iz prethodnog poglavlja.

Zaključak koji se može izvesti na osnovu razmatranja prethodne aplikacije i njene energetske efikasnosti može se generalizovati i na druge tipove aplikacija jer kod 50% aplikacija koje razmenjuju podatke sa serverima važi neki od tri razmatrana uslova rada.

6.1.2 Razmatranje zahteva realizacije

Pored benefita koje donosi ovakav koncept energetski efikasne komunikacije ima i svoje nedostatke. Prvenstveno, ceo koncept se zasniva na matematičkom modelu koji je izveden na testnom serveru. Matematički modeli dobijeni tokom istraživanja se razlikuju od modela servera na kojima će se instalirati ovakav sistem.

6.1.2.1 Preciznost matematičkog modela na serverima različitim od testiranog

Konstatacija da se serveri razlikuju i da će matematički modeli do kojih smo došli biti neprecizni, ako se koriste na drugim računarima, realno je opravdana. Ovo je posledica različite hardverske konstrukcije servera i uslova u kojima radi. Kao rešenje ovog potencijalnog problema nameću su se dve solucije. Prvo rešenje podrazumeva kreiranje univerzalnog (opšteg) matematičkog modela nastalog na analizi matematičkih modela što većeg broja operativnih servera (*Pristup A*). Drugo rešenje predstavlja pojednostavljenu i automatizovanu proceduru inicijalizacije sistema koja bi samotestiranjem i samoevaluacijom kreirala matematički model računara na kome će se nalaziti server (*Pristup B*).

6.1.2.1.1 Pristup A

Prvi pristup, pored aproksimacija koje mogu da dovedu do nepreciznosti a u krajnjem ishodu i do pogoršanja energetske efikasnosti, imaju i problem koji je vezan za razvoj servera u hardverskom

smislu. Dakle ukoliko bi u ovaj API smestili matematičke modele koji su aproksimovani došli bismo u situaciju da u nekim slučajevim API ne radi kako je očekivano. Jedno od podrešenja koje se može uzeti u razmatranje je korekcija modela na samom serveru u zavisnosti od parametra servera.

6.1.2.1.2 Pristup B

Drugo rešenje predstavlja automatizaciju postupka koji je u ovom radu veće opisan. Kada je analizirana opšta energetska efikasnost u eksperimentalnim ulovima dobijeni su matematički modeli operacija na serveru. Međutim, ne može se očekivati od budućih korisnika ovog sistema da svaki put kada rade novu implementaciju prolaze kroz istraživanje i kreiranje matematičkih modela ponaosob koja su bila deo ovog rada. Ovaj pristup bi se svodio na automatizaciju procesa koji je već opisan u cilju dolaska do egzaktnog matematičkog modela. Korisnik bi morao prilikom implementacije sistema nakon instalacije paketa na serveru, da pokrene niz test aplikacija koje bi automatski pomoću unapred zadatih setova podataka formirali matematičke modele za svaki od predloženih stepena kompresije i serijalizacije. Takvi modeli bi se čuvali i ugrađivali u lokalnu bazu na serveru i uvrstili u celoviti matematički model. Kasnije bi se koristili za rad sistema transporta objektno orjentisanih podatka sa maksimalnom energetskom efikasnošću.

U oba ponuđena rešenja može se javiti odstupanje u tačnosti modela usled preopterećenja servera. To može značiti da vreme koja se procenjuje matematičkim modelom nije egzaktno tačno u datom trenutku tj. iznosi manje nego što je u realnosti. Rešenje koje se nameće je da se kao parametar u celoviti matematički model uvrste opterećenje CPU-a servera i zauzeće RAM memorije. Pretpostavka je da bi se time ublažila odstupanja nestala preopterećenjem servera. Ovo ostavlja prostor za dalja istraživanja koja neće biti obavljena u ovoj disertaciji.

6.1.2.2 Detekcija brzine protoka informacija

Pored veličine podataka koje klijent zahteva od servera za što optimalniji rad API-ja potrebna je i procena brzine prenosa podatka između servera i klijenta (mobilne aplikacije). Kako se model i

proračun odabira režima komunikacije nalazi na serveru potrebno je da server za svaki zahtev i svakog klijenta posebno proceni brzinu kojim klijent razmenjuje podatke.

Sam server a priori ne može da zna u kakvom je stanju mobilni uređaj ako mu sam mobilni uređaj ne neki način tu informaciju na dostavi ili pak ne dostavi potrebne podatke na osnovu kojih se procenjuje ta vrednost. Sa druge stane kod mobilnih uređaja koji su povezani na mobilni internet moguće je da sam uređaj prvenstveno zna kojim tipom komunikacije raspolaže u svakom trenutku (da li je tip EDGE, HSPA, 3G..). U tabeli 9 mogu se videti teorijske maksimalne brzine prijema podataka. Treba napomenuti da maksimalna teorijska brzina i realna brzina nisu iste i da je realna u mnogim slučajevima znatno niža.

Ono što se može uraditi je da se na veoma niskom nivou, kod modema na samom mobilnim uređaju, detektuje tip ali ne i realna brzina. Tokom zahteva za prijem podataka pored specifičnog zahteva šalje se i tip komunikacije. Na osnovu toga API bi na serveru mogao da zna u kom opsegu brzina se nalazi aplikacija na klijentskom uređaju. Međutim, ako pogledamo tabelu u njoj su date samo teorijske maksimalne brzine i realna može da odstupa i varira što u našem slučaju dovodi do narušavanja energetske efikasnosti. Kao što se vidi razlika u brzini između dva standarda konekcije može iznositi 80Mbits/s. Ukoliko se ovo primeni na matematički model dovodi do greške.

Kao i kod prethodnih problema koje smo razmatrali ponovo se postavlja pitanje o preciznosti detekcije brzine. *Da li se mogu preciznije odrediti brzine komunikacije sa serverom i koliko to utiče na energetsku uštedu?*

Simbol	Standard	Verzija	Teorijska maksimalna brzina prijema podatka
G	GPRS		53,6 Kbits/s
E	EDGE		217,6 Kbits/s
3G	UMTS		384 Kbits/s
H	HSPA		7,2 Mbits/s
H+	HSPA+	Verzija 6	14,4 Mbits/s
H+	HSPA+	Verzija 7	21,1 Mbits/s
H+	HSPA+	Verzija 8	42,2 Mbits/s
H+	HSPA+	Verzija 9	84,4 Mbits/s
H+	HSPA+	Verzija 10	168 Mbits/s
4G	LTE		100 Mbits/s
4G	LTE-A	Napredna	1 Gbits/s

Tabela 7 Brzine prenosa podataka mobilnog interneta

Jedna od mogućnosti tačnijeg ispitivanje brzine je da mobilni uređaj povremeno sam evaluira brzinu komunikacije sa serverom. Drugim rečima on bi slao probne poruke do servera i nazad i na osnovu toga računao brzinu. Međutim ako se modem optereti sa ovim probnim porukama povećaće se znatno i potrošnja električne energije, sve ovo može da poništi efekat energetske efikasnosti koji se pokušava dosegnuti.

Drugo rešenje koje se može upotrebiti, tako da ne naruši postojeću efikasnost i ne optereti dodatno uređaj, je da klijent na osnovu prethodno primljenih podatka evaluira realnu brzinu. U nastavku ova brzina će biti upotrebljena da se u sledećem zahtevu pošalje serveru. Naravno ova brzina se već posle nekoliko sekundi može promeniti pa neće biti egzaktna ali se može kombinovati sa tipom prenosa. Mobilni uređaj će na osnovu tipa konekcije kojim trenutno raspolaže i prethodno

izmerene realne brzine protoka u sledećoj iteraciji obaveštavati server. Ukoliko se između dva zahteva promeni tip konekcije API na serveru neće koristiti prethodnu već poslednju izmerenu brzinu iz trenutne kategorije konekcije. Ovakvim rešenjem se dolazi do neinvazivnog metoda evaluacije realne brzine.

6.1.2.3 Algoritam za odlučivanje

API u sebi pored matematički modela koji su specifični za server na kome se izvršava ima i svoj izvršni deo koji ima algoritam za evaluaciju i odabir režima rada. Sam API se može smatrati softverskim omotačem već postojećih metoda koje se koriste. API na osnovu algoritma za procenu određuje nivoe kompresije podataka i tip serijalizacije.

Algoritam za procenu se sastoji iz 3 koraka :

- Merenje veličine podataka koju treba poslati
- Računanje vremena transporta podatka prema klijentu uz pomoć matematičkih modela za svaki scenario korišćenjem brzine prenosa koju je dobio od klijenta i veličine podatka koji treba poslati klijentu
- Odabir i izvršavanje scenarija koji ima najmanje vreme transporta dobijeno u tački 2

6.1.2.3.1 Opis rada algoritma za odlučivanje

Kada segment za odlučivanje primi informaciju od klijenta za zahtevanim podacima proces odlučivanja se započinje upitom u bazi podataka. Na osnovu rezultata upita računa se količina podataka koja se transportuje do klijenta. U ovom slučaju u objektima količina podataka se ogleda u broju objekata od interesa iz razloga jer je matematički model tako koncipiran. Dakle podešen je za konkretan slučaj baze podataka koja je kreirana za potrebe testa. Drugi parametar je procenjena vrednost intenziteta protoka podataka od servera do klijenta ova procenjena vrednost se dobija od klijenta u sklopu njegovog zahteva. Algoritam za procenu počinje sa preračunavanjem energetske efikasnosti svakog od pomenutih tipova serijalizacije u kombinaciji sa svim nivoima gzip kompresije. Procena se bazira na prethodno dobijenim matematičkim modelima uz pomoć podataka o broju objekata i vrednosti intenziteta protoka. Od svih kombinacija odabira se ona za koju je dobijena minimalna vrednost i prema scenariju te kombinacije se podaci serijalizuju i zatim kompresuju i kao takvi prosleđuju nazad do klijenta. Kako je radom na simulatoru utvrđeno da od

svih sistema za serijalizaciju u kombinaciji sa kompresijom najbolje rezultate daju isključivo JSON i MsgPack, onda se broj kombinacija svodi na 20.

6.1.2.4 Klijentska strana API

Iako većinski teret obrade zahteva pada na server on ne može bez saradnje sa klijentom funkcionisati uspešno. Već smo naglasili da klijent ima ulogu da obavesti API na serveru o brzini protoka. On ne poseduje matematičke modela niti vrši odlučivanje. Drugim rečima on će obmotati klasični REST zahtev gde će u zaglavljima poslati serveru potrebne informacije. Po prijemu zahteva i slanju kreiraće lokalnu bazu sa poslednjim izračunatim brzinama prenosa. Kako se radi o klijentskom delu API koji bi se koristio za mobilne uređaje njegova implementacija bi bila u obliku klase u Java ili C# programskom jeziku.

6.1.2.5 Celina API modela

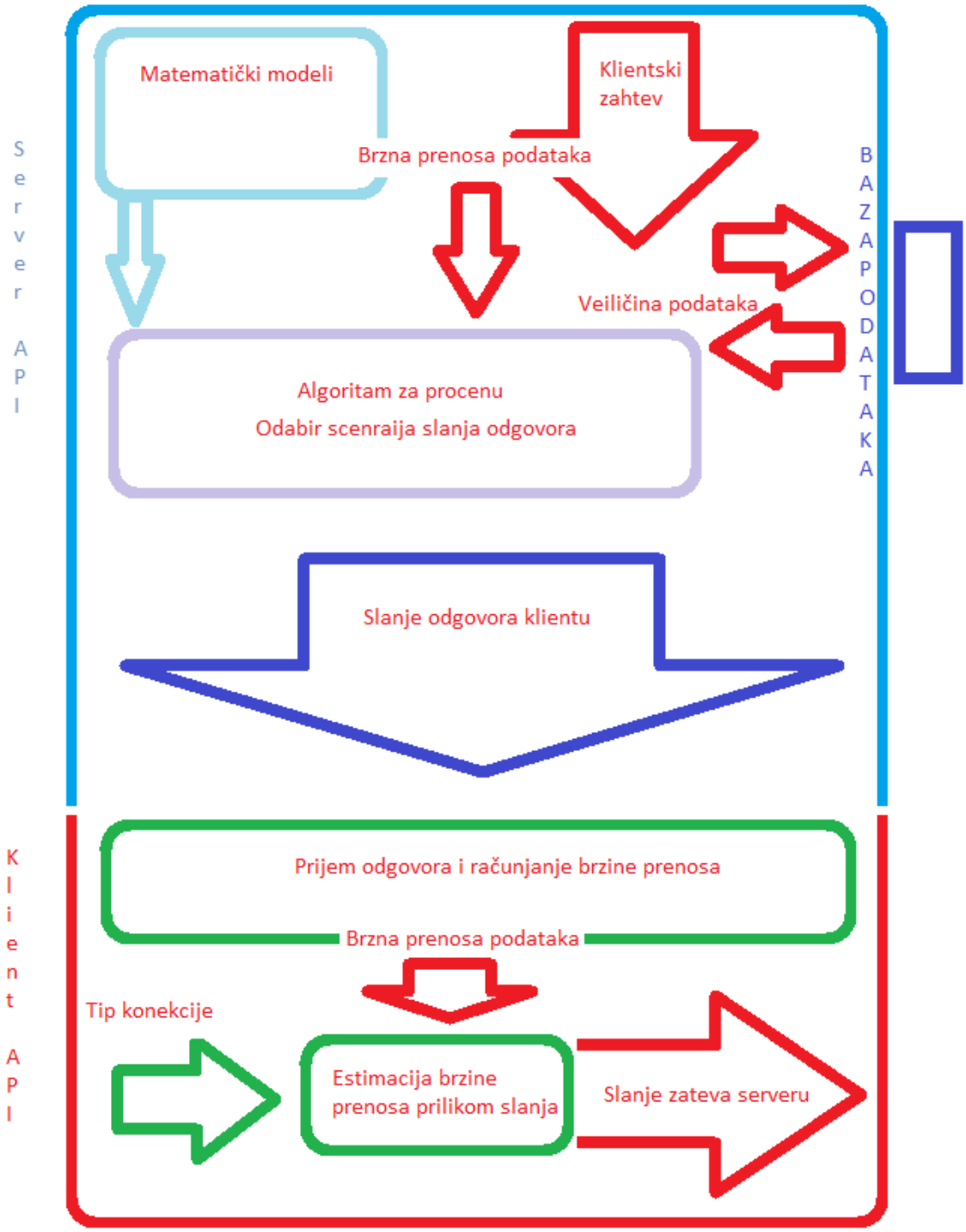
Nakon svih prethodnih razmatranja može se predstaviti API u celini. Na slici 42 prikazan je dijagram koji opisuje sve elemente API-ja i njihove relacije. Klijent na osnovu procenjene brzine kojom raspolaže šalje potrebne podatke serveru tokom zahteva za podacima. Server prima zahtev odvajajući API podatke (tj. brzinu klijenta) i meri veličinu podatka koju treba da transportuje nazad. Na osnovu matematičkih modela koji su specifični za sam hardver na kome se izvršava upit, pomoću ova dva parametra on računa najbolji scenario za transport podatka i šalje ga nazad. Sa druge strane klijent nakon prijema određuje brzinu transporta podatka i čuva je u lokalnoj bazi. Tu brzinu će upotrebiti za neki od sledećih zahteva.

6.2 Implementacija rešenja

Kako bi se u realnom okruženju testirao novi sistem komunikacije za potrebe ovog rada je formirana test aplikacija i server. Formiran je test sistem koji se u pogledu komunikacije transporta podatka i korisničke eksploatacije ne bi razlikovao od realne komercijalne aplikacije koja bi koristila pomenuti API. Razlike koje su evidentne u odnosu na realnu komercijalnu aplikaciju su

verodostojnos podatka, grafički interfej i druge opcije koje su samom korisniku bitne da bi aplikacija bila laka za upotrebu.

Ceo test je osmišljen tako da se formiraju dva test softverska paketa potpuno istih namena i funkcionalnosti. Razlika je u tome što se u jednom rešenju koristi CBSaver API, a u drugom klasičan REST sistem sa JSON serijalizacijom. kao uobičajeni način kod većine savremenih aplikacija. Aplikacija ima ulogu da na osnovu svoje geografske pozicije, koju dobija od svog GPS prijemnika, zahteva od servera podatke koji su vezani za tu lokaciju. Aplikacija zahteva podatke na osnovu pomeranja mobilnog uređaja ali i vremenskog intervala. Ako se mobilni uređaj kreće na svakih 50 metara se zahtevaju novi podaci od servera sa novom pozicijom. Ukoliko uređaj miruje onda se podaci zahtevaju u intervalu od 90 sekundi. Ovo predstavlja osnovu za



Slika 48- Dijagram strukture CBSaver API-ja

aplikacije koje se korite kao navigacioni sistemi, pomoćni sistemi prilikom pretrage prostora, sistemi za upozoravanje, pa cak i sistemi za komunikaciju. Pored toga ovakav način komunikacije je okosnica budućih autonomnih sistema u javnom transportu ali i kod specijalizovanih industrijskih sistema. Naredna tabela prikazuje aplikacije koje u osnovi komuniciraju po sličnom principu kao i naša test aplikacija. Koriste geografsku lokaciju korisnika i na osnovu informacije o toj lokaciji koju prosleđuju svom serveru dovlače podatke potrebne za rad. Podaci o broju korisnika su dobijeni sa *GooglePlay* prodavnice. Treba naglasiti da pored Android operativnog sistema ove aplikacije postoji i za IOS operativni sistem tako da je broj korisnika znatno veći.

Aplikacija	Broj korisnika	Opis
Gmaps	1,000,000,000 - 5,000,000,000	Mapa, navigacija , saobraćaj.
Nike +	10,000,000 - 50,000,000	Aplikacija za rekreativno i profesionalno trčanje i fitnes
Waze	100,000,000 - 500,000,000	Navigacija, obaveštenja o saobraćaju, opasnostima i policiji.
Pokemon go	100,000,000 - 500,000,000	Video igra koja na osnovu geo lokacije postavlja virtualne predmete koje korisnici traže i sakupljaju.
Run Kepper	10,000,000 - 50,000,000	Aplikacija za rekreativno i profesionalno trčanje i fitnes
Uber	100,000,000 - 500,000,000	Aplikacija za javi prevoz i kolaboraciju slobodnih taksi vozila
Escort Radar	100,000 - 500,000	Aplikacija za prikazivanje policijskih patrola, radara i kamera

Smaz	10,000 - 50,000	Dostava slatkiša i obaveštenje u koliko se korisnik nađe u lokaciji blizu kamiona.
Nearyby	50,000 - 100,000	Objekt od interesa koji se nalaze u blizini
Friend Loacator	10,000,000 - 50,000,000	Prikaz osoba na mapi i njihova trenutna lokacija, komunikacija

Tabela 8 Aplikacije koje se koriste na Android mobilnim uređajima koje su u pogledu prenosa podatka na osnovu lokacije uređaja slične test aplikaciji

6.2.1 Koncept aplikacije

Koncept aplikacije se sastoji iz android mobilne aplikacije koja ima za cilj da na osnovu položaja mobilnog uređaja od servera potražuje podatke koji su specifični za tu oblast. S obzirom da se radi o test aplikaciju koja je kreirana za potrebe ovog rada i istraživanja, grafički interfejs se sastoji od samo jedne aktivnosti koja prikazuje status aplikacije i broj uspešno primljenih zahteva. U realnoj komercijalnoj aplikaciju na ovom mestu bi postojala neka od mapa na kojoj bi se prikazivali podaci.

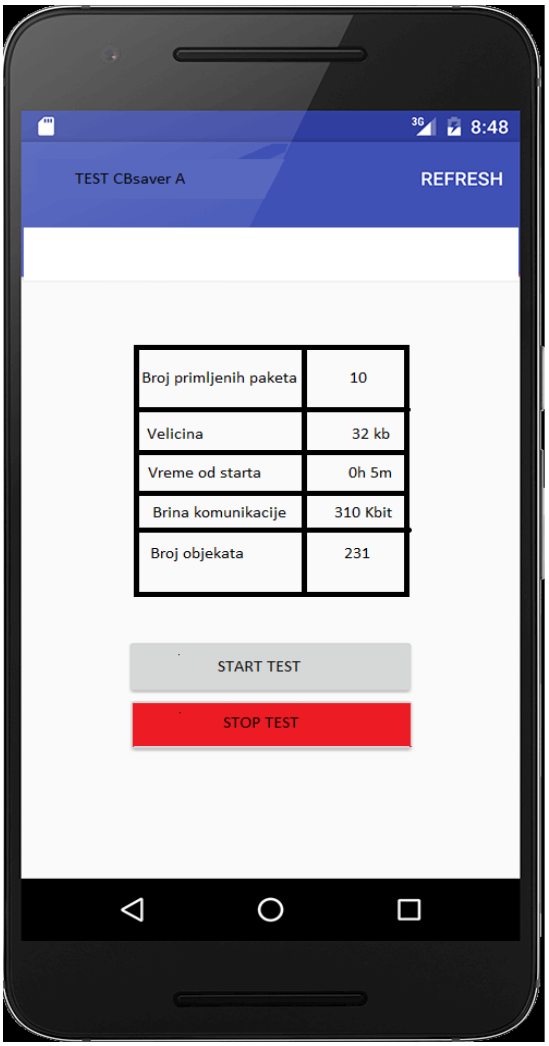
U oba slučaja test aplikacija ima isti mehanizam a razlikuje se samo slanje zahteva. U aplikaciji A se nalazi posebna klasa sa CBSaver API dok je u drugoj nema. Podaci koji se zahtevaju od servera su koncipirani tako da predstavlja objekat koji se sastoji od pozicije, naziva, identifikacionog broja, statusa i opisa.

Analizom aplikacija slične prirode iz tabele 10 uviđa se da su proizvođači većine ovih aplikacija otišli i korak dalje. Kreirali su svoje API-je da i drugi proizvođači mogu koristiti njihove servere i baze. Za *Uber, RunKeeper, Nike* ili *Escort* drugi korisnici mogu upisivati podatke i skidati ali kod svih njih je kombinacija RESTfull.i oni isključivo pakuju i serijalizuju podatke kao JSON [38] [39] [40] [41] [42].

Na sledećim slikama se može videti izgled aplikacije. Postoji dugme za startovanje aplikacije i tabela koja prikazuje statuse rada. U statusima rada aplikacije vide se:

- vreme od poslednjeg starta aplikacije
- broj zahteva prosleđenih serveru
- trenutna procenjena brzina
- ukupan broj objekat primljenih od servera
- ukupna količina podata priljena od servera u kB

Ovi trenutni podaci služe za nadgledanje rada sistema i za zapažanje kakvih nepravilnosti u radu koji mogu da odvedu do pogrešnih rezultata.



Slika 49 Izgled aktivnosti test aplikacije

Podaci potrebni za testiranje kreirani su od postojećih baza podataka dostupnih na internetu. Baza podatka je kreirana od postojećih objekata za evropsku teritoriju tako što su geografska dužina i širina tranširale na geografsku poziciju Rasinskog okruga gde je aplikacija testirana. Sa veb portala *HERE* [43] su preuzeti svi objekti za grad Minhen i koordinate geografske širine i dužine translirane tako da se za centar Kruševca iz baze dobijaju objekti koji se nalaze u centru Minhena. Na ovaj način je dobijena baza sa fiktivnim podacima tako da zadovolji evropske standarde u vidu količine podatka koji se nalaze u blizini korisnika. Takođe, sa udaljavanjem od centra grada smanjuje se i broj objekata u blizini.


```
{
"title": "Chrysler Camera Repair",
"highlightedTitle": "<b>Chrysler</b> Camera Repair",
"vicinity": "367 W 34th St\nNew York, NY 10001-1647",
"position": [ 40.753098, -73.995239 ],
"category": "business-services",
"href": "https://...",
"type": "urn:nlp-types:place"
}
```


Slika 50 Prikaz strukture objekta od interesa mape sa objektima

6.2.1.1 Server

Server se u programskom smislu sastoji od baze podataka i interfejsa za obradu zahteva. U jednom slučaju se sastoji od CBSavera API preko koga se vrši komunikacija dok je u drugom klasično slanje podatka serijalizovanih JSON-on formatom. Server na osnovu podatka koje primi od mobilnog uređaja izvrši selekciju podataka, nakon toga u zavisnosti od tipa testa radi zahtevanu obradu i slanje. CBSaver je već razmatran u prethodnom poglavlju. Za potrebe ovog testa razvijena je beta verzija u PHP programskom jeziku koja u sebi sadrži matematičke modele koji su dobijeni

na serveru na kome će se vršiti testiranje. Prema tome, radi se o uprošćenom API-ju koji nema opciju samo korekcije već koristi unapred sračunati matematički model.

Pored funkcije razmene podataka, server ima ulogu i da beleži sve zahteve koje su klijenti uputili. Za svaki test uređaj server u bazi podatka beleži poziciju, veličinu podataka koju mu je poslao stepen kompresije koji je izabrao i brzinu koju je uređaj poslao. Ovi podaci su bitni za dublju analizu rada i optimizaciju sistema za prenos objektno orjentisanih podatka.

6.2.2 Metodologija testa

Test se sastoji iz što realnije upotrebe aplikacije u svakodnevnom životu. Aplikacija je korišćena svakodnevno u periodu od 7 dana na više različitih uređaja koji su posebno pripremljeni za ovo testiranje. Odstranjene su sve nepotrebne aplikacije koje koriste internet i rade kao servisi u pozadini. Svakog dana kada je vršeno testiranje, uređaji su pre početka testa punjeni do maksimalnog kapaciteta baterije. Nakon završetka dnevnog testa upisivana je vrednost kapaciteta baterije Beležena je i količina prenetih podataka. Test je vršen na ukupno šest fizičkih uređaja: po dva identična iz svake klase 2G, 3G i 4G. Na svakom paru uređaja, na jednom je bila instalirana test aplikacija sa CBSaver API-jem, a na drugom test aplikacija sa JSON serijalizacijom podataka.

Nakon 7 dana testiranja usledilo je identično testiranje na ovim uređajima ali tako što aplikacije zamenjene. Na telefonima na kojima je bila aplikacija sa JSON-om instalirana je test aplikacija sa CBSaver-API-jem i obratno. Na ovaj način je eliminisan uticaj samog uređaja na test. Iako su uređaji istog modela ne može se sa sigurnošću tvrditi da baterije ova dva uređaja imaju identične kapacitete zbog samog stanja u kome se uređaji nalaze usled prethodne upotrebe. Za dalju analizu kao rezultat testa je uzeta prosečna vrednost sa oba uređaja za svaki od dana u nedelji.

Vođeno je računa da se kretanje uređaja prvog dana prve nedelje testa, poklapa sa kretanjem prvog dana druge nedelje i tako za svaki od narednih dana u testu. Ovo je bino jer od šeme kretanja zavisi i količina podataka koja se zahteva od servera i intenzitet protoka.

6.2.2.1 Test uređaji

Vođeni podatkom o trenutnoj zastupljenosti mobilnih [37] uređaja tokom testa korišćena su tri modela mobilnih uređaja različitih generacija: *Samsung galaxy S i9000-DevC, Samsung galaxy S II,-BevB i Samsung galaxy J5-DevA.* Pored toga vođeno je računa da se na svih šest uređaja koriste SIM kartice istog operatera a u našem slučaju je to bio VIP operater.

6.2.3 Rezultati dobijeni testom

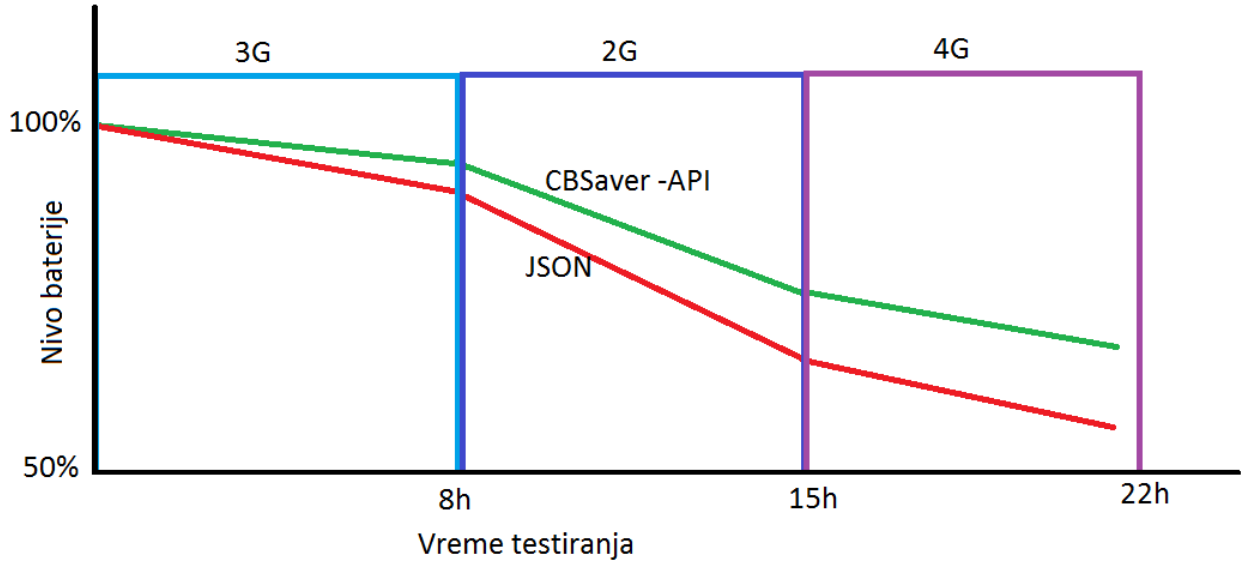
Nakon dvonedeljnog testiranja dobijeni su sledeći rezultati. U prikazanim tabelama sumirane su vrednost za svaki dan i svaki uređaj ponaosob.

Dan	Dev A CB	Dev A JSON	Dev B CB	Dev B JSON	Dev C CB	Dev C JSON
1	65	52	60	55	40	10
2	70	62	78	72	52	15
3	45	20	40	36	63	35
4	80	87	70	68	76	43
5	66	70	57	52	49	44
6	19	2	12	0	12	8
7	20	18	10	1	3	0

Tabela 9Rezultati testa

Pored gore pomenutih testova koji su pokazali da je CBSaver itekako ima ulogu u energetskoj efikasnosti klijentskog uređaja, urađen je i strogo kontrolisani test na jednom uređaju koji je testirao oba sistema tako da uređaj bude u istovetnim uslovima. Dva dana za redom u određeno doba startovana je aplikacija tako da narednih 10 sati radi u 3G režimu, zatim je sledećih 7 sati na uređaju omoguć da radi samo u 2G režimu i na kraju poslednjih 8 sati u 4G .Testovi su ciklično

ponavljani u isto doba dana. Na slici Slika 51 su prikazani profili i tok potrošnje baterija na dve uporedne aplikacije.



Slika 51 Prikaz merenja dobijenih testom

6.2.4 Analiza rezultata eksperimenta

Površnim pregledom tabele Tabela 9 nedvosmisleno se može zaključiti da CBSaver API ia impozantni efekat na uštedu baterije . Pre same analize rezultat treba podsetiti da uređaji A,B i C nad kojima su vršeni testovi su 4G 3G i 2G uređaji. Tako da su teorijske maksimalne brzine ovih uređaja različite i ograničene samim hardverom. Ovo je bitno naglasiti jer mogućnosti protoka podataka uređaja bitno utiču na energetsku efikasnost..

Krenimo prvo od Dev C uređaja, ovde se jasno vidi da je aplikacija koja je koristila CBSavr API u svim danima imala bolji učinak od klasičnog REST zahteva sa JSON tipom serijalizacije. Takođe razlike u uštedi baterije su prilično velike ovo je i bilo očekivano jer ako se vratimo na prethodno poglavlje i pogledamo slike (Slika 43 Slika 44 Slika 45) možemo vide da su u područjima nižih vrednosti protoka podataka razlike u potrošnji električne energije kod različitih tipova izraženije. Naročito ako je razlika u veličini podataka koje transportujemo izraženija. Kod ovih opsega protoka podataka shodno matematičkoj formuli (3) vreme provedeno u transportu podataka je znatno veće nego vreme kašnjenja servera tako i dominantnije utiče na potrošnju u celosti. Potvrda

ovih teza se može pronaći i u analizi rezultata strog kontrolisanog testa koji je naknadno izveden i čije rezultate možemo videti na slici Slika 51. Dakle sa slike se vidi da je odnos potrošnje najizraženiji kada je uređaj bio ograničen isključivo na 2G područje prenosa podataka.

Nastavimo dalje sa analizom učinak uređaja Dev B ,kod nega je gotovo slična situacija kako od dev C sa time da je razlika u uštedi na kraju tesnog dana nešto manja. Kao i kod prethodnog slučaja ovo je očekivano i u skladu sa matematičkim modelima ali i simulacijama prethodno izvedenim. Smanjenje učinak CBSaver algoritma se u odnosu na prethodni slučaj vidimo u povećanju opsega protoka prenosa podataka. Ovde treba biti obazriv jer to što je uređaju dozvoljeno da komunicira 3G opsezima ne znači da on u realnim ulovima eksploatacije i raspolaže tim brzinama.

Kod Dev A imamo na izgled bolje rezultate CBSaver algoritma , ali su oni prilično lošiji nego kod B i C Konkretno u 7 dana testa tokom dva dana imamo lošije rezultate nego kada je uređaj radio sa klasičnim prenosom podataka. Ovo se može objasniti time da je opseg protoka podataka jako širok za 4G uređaje (Tabela 7). Može se zaključiti da algoritam za procenu brzine ima nedostatke i njegova nepreciznost dovodi do greške u odlučivanju. Ova greška u proceni intenziteta protoka dovodi do netačnosti matematičkog modela i odabira neadekvatnog režima serijalizacije i kompresije podataka. Neadekvatna procena dakle dovodi od toga odabrani režim bude energetski neefikasniji od klasičnog tako da se može desiti da se naruši energetska efikasnost uređaja. Kao što je slučaji u dva testirana dana. Kada se radi o režimima rada gde su viši intenziteti protoka razlika u uštedi je znatno manja, što pokazuje kriva odnosa potrošnje u strogo kontrolisanom testu (Slika 51).

6.2.5 Završne napomene poglavlja

Testovima (u kojima je CBSaver API korišćen kao vid komunikacije u aplikaciji koja na osnovu položaja zahteva specifične informacije od interesa za taj rejon) je pokazano da je pod određenim uslovima ovakav API energetski efikasniji u odnosu na klasične metode koje se trenutno koriste. Na osnovu ovog istraživanja može se ukazati i na rešenja u samom softverskom inženjerstvu koja nude veću energetsku efikasnost od klasičnih.

7 VII DEO

7.1 Zaključak

Savremeni svet se više ne može zamisliti bez interneta i mobilnih uređaja. Oni se dalje ne mogu zamisliti bez povezanosti na globalnu mrežu i razmenu podataka. Tehnologija koja se ustalila poslednjih pet godina omogućila je da se što više priđe korisnicima i nametne im se stil života koji zahteva od njih da imaju aplikaciju vezane za svaki segment života. Aplikacije su široko dostupne iz oblasti zdravlja, sporta, lepote, zabave i kupovine. Programerske kompanije su uvidele mogućnosti velikog tržišta i krenula je hiper produkcija. Nije se mnogo vodilo računa o energetskoj efikasnosti aplikacija prilikom ekspanzije aplikacija. Energetska efikasnost je prepuštena proizvođačima čvrstih komponenti a sav teret je pao na proizvođače baterija. Kao rezultat toga danas imamo mobilne uređaje koje moramo puniti na kraju svakog radnog dana osim ukoliko ih ne koristimo samo za razgovor.

Većina aplikacija je slične arhitekture. Dakle svaka aplikacija u zavisnosti od svoje namene ima svoj dodeljeni server i bazu podatka. Oni međusobno razmenjuju podatke. Svi ti podaci se mogu posmatrati kao objektno orijentisani podaci. Prenos objektno orijentisanih podatka je poslednjih petnaest godina omogućen procesom serijalizacije. Pored toga kao način za transport se ustalio RESTfull oblik i na taj način posao standard za gotovo svaki API dostupan na tržištu. Što se serijalizacije tiče JSON je preuzeo primat. Zamenio je preveliki XML i nedovoljno funkcionalni CSV. Sadašnjica pripada objektno orijentisanim programskim jezicima. Ovaj rad prvenstveno ima za cilj da analizira prenos objektno orijentisanih podataka i uticaj prenosa na potrošnju električne energije na mobilnom uređaju. Analizirani su savremeni sistemi, njihova arhitektura potrebe i funkcionalnost. Sagledani su aktuelni naučni radovi koji obrađuju energetsku efikasnost. Analizirane su eksperimentalne metode za merenje potrošnje energije u realnom vremenu. Na osnovu dostupnih naučnih radova drugih autora, njihovih zaključaka i rezultata formirana je oprema za merenje. Tokom rada analiziran je način prenosa podatka, tačnije REST zahtev. U programskom smislu svaki REST zahtev je podeljen u programske celine na koje sam programer može uticati u višim nivoima API-ja .Eksperimentalno je merena potrošnja električne energija na

3G modemu, ali tako da se prethodno pomenuti segmenti mogu izdvojiti na energetskom profilu potrošnje tokom REST zahteva. Tako dobijenim presekom potrošnje po stanjima omogućeno je da se u daljem nastavku analizira uticaj serijalizacije na potrošnju. Profil potrošnje je omogućio kreiranje matematičkog modela REST zahteva. Takav model u kome figurišu parametri na koji indirektno utiče i proces serijalizacije dao je odličnu polaznu osnovu da se utvrdi uticaj procesa serijalizacije koji se odvija na serveru na potrošnju električne energije klijenta koji zahteva te podatke. Nastavak istraživanja ogledao se u analizama postojećih sistema za serijalizaciju, merenjem njihovih performansi i kreiranjem matematičkih modela koji su se kasnije ugradili u prethodno formirani matematički model REST profila potrošnje. Ovakvo celovita matematička funkcija u kojoj figurišu parametri o intenzitetu protoka podataka na mreži, količini podataka i tipu serijalizacije, omogućila je simulaciju i analizu energetske efikasnosti. Analiza je pokazala kako se, pravilnim odabirom tipa serijalizacije i nivoa kompresije, mogu postići značajne uštede u električnoj energiji klijentskog uređaja. Simulacijama i analizom je pokazano koliko aktuelne aplikacije koje koriste JSON kao sistem za serijalizaciju imaju veću potrošnju energije u odnosu na neke druge sisteme. Tokom analiza javila se ideja o hibridnom sistemu za prenos podatka koji bi na pametan način kombinovao postojeće metode tako da minimizira potrošnju energije.

Na bazi svih istraživanja u radu, razmatran je energetsko efikasni API za prenos objektno orijentisanih podataka CVSaver. On je korišćenjem matematičkog modela dobijenog u istraživanju i parametara o brzini prenosa podatka između klijenta i servera i parametara o obimu podatka koje treba transportovati odbirao energetski najefikasnije rešenje. Kao ključ celog rada je kreiran opitni CBSaver API i kreirana test aplikacija. Paralelno na tri različita uređaja je vršen test sedam dana a rezultati koji su dobijeni su potvrdili sva razmatranja i istraživanja sproveden tokom rada. Pravilan odabir serijalizacije i kompresije može da u pojedinim slučajevima i od 20% bude efikasnije nego kasični metodi koji su u masovnoj primeni. Rad na ovoj disertaciji otvara još mogućnosti za dalje istraživanje unapređivanje prenosa objektno orjentisanih podataka u cilju energetske optimizacije. Tokom rada na CBSaver API-ju ostavljeno je prostora za zamenu matematičkih modela fazi logikom ili neuralnim mrežama. Takođe postoji prostora za unapređivanje računanja brine protka podataka.klienta.

8 Literatura

1. *FACTORS INFLUENCING QUALITY OF MOBILE APPS: ROLE OF MOBILE APP DEVELOPMENT.* **Inukollu, Venkata N.** s.l. : International Journal of Software Engineering & Applications (IJSEA), September 2014 , Tom. Vol.5, No.5, . DOI : 10.5121/ijsea.2014.5502 .

2. *A Programming Interface for Application-Aware Adaptation in Mobile Computing.* **Noble, Brian and Satyanarayanan, M. and Price, Morgan.** Berkeley, CA, USA: USENIX Association : Symposium on Mobile and LocationIndependent Computing. , 1995.

3. *Every joule is precious: the case for revisiting operating system design for energy efficiency.* **A. Vahdat, A. Lebeck, and C. S. Ellis.** New York, NY, USA: ACM, 2000, pp. 31–36 : 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system, ser, 2000.

4. *Energy management in mobile devices with the cinder operating system.* **A. Roy, S. M. Rumble, R. Stutsman, P. Levis, D. Mazi`eres, and N. Zeldovich.** s.l. : Proceedings of the sixth conference on Computer systems, ser. EuroSys ’11. New York, NY, USA: ACM, 2011, pp. 139–152. , 2011.

5. *Mobile apps: it’s time to move up to CondOS.* **D. Chu, A. Kansal, J. Liu, and F. Zhao.** s.l. : Proceedings of the 13th USENIX conference on Hot topics in operating systems, ser. HotOS’13. Berkeley, CA, USA: USENIX Association, 2011, pp. 16–16.

6. *Energy Is Just Another Resource: Energy Accounting and Energy Pricing in the Nemesis OS.* **McAuley, R. Neugebauer and D.** s.l. : Proceedings of the Eighth Workshop on Hot Topics in Operating Systems. Washington, DC, USA: IEEE Computer Society, 2001.

7. *ErdOS: achieving energy savings in mobile OS.* **Crowcroft, N. Vallina-Rodriguez and J.** s.l. : Proceedings of the sixth international workshop on MobiArch, ser. MobiArch ’11. New York, NY, USA, 2011.

8. *AnAnalysisofPowerConsumptioninaSmartphone.* **Carroll, Aaron.** Boston : USENIX conference on USENIX annual technical conference, 2010.

9. *On the impact of 2G and 3G network usage for mobile phones’ battery life.* **G. P. Perrucci, F. H. P. Fitzek, G. Sasso, W. Kellerer, and J. Widmer.** s.l. : in Wireless Conference, 2009. EW 2009. European, 2009, pp. 255–259. , 2009.

10. *Impacts of inactivity timer values on umts system capacity.* **X. Chuah, M., Luo, Wei / Zhang.** s.l. : In Wireless Communications and Networking Conference (2002), volume 2, pages 897–903. IEEE, 2002. .

11. *“Energy consumption in mobile phones: a measurement study and implications for network applications.* **N. Balasubramanian, A. Balasubramanian, and A. Venkataramani.** s.l. : in Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference, ser. IMC ’09. New York, NY, USA: ACM, 2009, pp. 280–293, 2009.

12. **Xia, Feng.** The Power of Smartphones. https://arxiv.org/ftp/arxiv/papers/1312/1312.6740.pdf.

13. *A survey of energy efficient MAC protocols for IEEE 802.11 WLAN.* **Huang, S.-L. Tsao and C.-H.** s.l. : Comput. Commun., vol. 34, pp. 54–67, January 2011. .

14. *CoolSpots: Reducing the Power Consumption of.* **all, Trevor Pering at.** s.l. : MobiSys'06, June 19–22, 2006, Uppsala, Sweden.

15. ***https://www.rootusers.com/gzip-vs-bzip2-vs-xz-performance-comparison/.*** **[Na mreži]**

16. *Efficient wire formats for high performance computing.* F. E. Bustamante, G. Eisenhauer, K. Schwan, and P. Widener..

17. *Latency performance of SOAP implementations.* Parashar, D. Davis and M.

18. *On the Energy Efficiency of Lossless DataCompression in Wireless Sensor Networks.* Andreas Reinhardt Delphine Christin, Matthias Hollick, Ralf Steinmetz. s.l. : Fourth IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2009), October 2009, 2009.

19. *U. Kremer,"Compiler-directed remote task execution for power management,".* U. Kremer, J. Hicks, and J. M. Rehg. s.l. : in Workshop on Compilers and Operating Systems for Low Power, 2000.

20. *Tacticsbased remote execution for mobile computing,.* R. K. Balan, M. Satyanarayanan, S. Y. Park, and T. Okoshi. s.l. : Proceedings of the 1st international conference on Mobile systems, applications and services, ser. MobiSys '03. New York, NY, USA: ACM, 200.

21. *Cloud Computing for Mobile Devices - Reducing Energy Consumption1 .* Veronika Strokova2, Sergey Sapegin3, Andreas Winter4. s.l. : Proceedings of the 28th EnviroInfo 2014 Conference, Oldenburg, Germany September 10-12, 2014 .

22. *Fine Grained Energy Accounting on Smartphones with Eprof.* Abhinav Pathak, Ming Zang. s.l. : April 10–13, 2012, Bern, Switzerland.

23. *Energy Consumption in Mobile Devices Why Future Systems Need Requirements-Aware Energy Scale-Down.* Labs, Robert N. Mayo and Parthasarathy Ranganathan Hewlett Packard.

24. *Energy consumption and conservation in wifi based phones: A measurement-based study. .* Mohapatra., A. Gupta and P. s.l. : In Sensor and Ad Hoc Communications and Networks (SECON), pages 121–131, Washington, DC, USA, 2007. IEEE Computer Society .

25. *An_Empirical_LTE_Smartphone_Power_Model_with_a_View_to_Energy_Efficiency_Evolution.* Laurdsen Mads, Noel Laurent,. s.l. : Intel tehnology Jurnal, 2014, T. 18.

26. http://www.littleeye.co/. [Na mreži]

27. *Decomposing power measurements for mobile devices.* Andrew Rice, Simon Hay. s.l. : Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on. DOI: 10.1109/PERCOM.2010.5466991 .

28. Bray, T. RFC 7159 – the javascript object notation (json) data interchange format. http: //tools.ietf.org/html/rfc7159, 2014. [Na mreži]

29. ***JSON on Mobile: is there an Efficient Parser ?*** **Queirós, Ricardo. DOI 10.4230/OASIcs.SLATE.2014.93 .**

30. Komanov, Dmitry. scala-serialization. [Na mreži] https://medium.com/@dkomanov/scala-serialization-419d175c888a.

31. Technologies, AVG®. AndroidTM App Performance & Trend Report By AVG® Technologies H1 2016. [Na mreži] 2016.

32. ***https://xmpp.org/about/history.html.*** **[Na mreži]**

33. ***RPC and REST: dilemma, disruption, and displacement. IEEE Internet Computing 12,. 2008, pp. 92–95.*** **Vinoski, S.,.**

34. ***RESTful service composition at a glance: A survey., Journal of Network and Computer Applications 60, 2016., pp. 32–53.*** **M. Gariga, et al.,.**

35. ***, Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm, Ad Hoc Networks, Vol. 56, Pages 122–140. March 2017.*** **L.Atzori, A. Iera, G. Morabito.**

36. [Na mreži] http://www.ti.com/lit/ds/symlink/ina219.pdf.

37. Cisco. ***Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper.***

38. [Na mreži] https://www.programmableweb.com/api/fitbit.

39. [Na mreži] https://www.programmableweb.com/api/runkeeper-health-graph.

40. [Na mreži] http://oauth.withings.com/api.

41. [Na mreži] https://developers.google.com/fit/rest/v1/data-types.

42. [Na mreži] https://developer.uber.com/docs/riders/references/api.

43. [Na mreži] https://here.navigation.com/europe/.